The MEANS/SUMMARY Procedure: Getting Started and Doing More

Arthur L. Carpenter
California Occidental Consultants

ABSTRACT
The MEANS/SUMMARY procedure is a workhorse for most data analysts. It is used to create tables of summary statistics as well as complex summary data sets. The user has a great many options which can be used to customize what the procedure is to produce. Unfortunately most analysts rely on only a few of the simpler basic ways of setting up the PROC step, never realizing that a number of less commonly used options and statements exist that can greatly simplify the procedure code, the analysis steps, and the resulting output.

This tutorial begins with the basic statements of the MEANS/SUMMARY procedure and follows up with introductions to a number of important and useful options and statements that can provide the analyst with much needed tools. With this practical knowledge, you can greatly enhance the usability of the procedure and then you too will be doing more with MEANS/SUMMARY.

KEY WORDS
OUTPUT, MEANS, SUMMARY, AUTONAME, _TYPE_, WAYS, LEVELS, MAXID, GROUPID, preloaded formats

INTRODUCTION
PROC MEANS is one of SAS’s original procedures, and it’s initial mandate was to create printed tables of summary statistics. Later PROC SUMMARY was introduced to create summary data sets. Although these two procedures grew up on the opposite side of the tracks, over time both has evolved so that under the current version of SAS they actually both use the same software behind the scenes.

These two procedures completely share capabilities. In fact neither can do anything that the other cannot do. Only some of the defaults are different (as they reflect the procedures’ original roots).

For the analyst faced with creating statistical summaries, the MEANS/SUMMARY procedure is indispensable. While it is fairly simple to generate a straightforward statistical summary, these procedures allow a complex list of options and statements that give the analyst a great deal of control.

Because of the similarity of these two procedures, examples will tend to show one or the other but not both. When I use MEANS or SUMMARY, I tend to select the procedure based on it primary objective of the step (SUMMARY for a summary data set and MEANS for a printed table). Even that ‘rule’, however is rather lax as MEANS has the further advantage of only having 5 letters in the procedure name.
BASIC STATEMENTS
The MEANS/SUMMARY procedure is so powerful that just a few simple statements and options can produce fairly complex and useful summary tables.

Differences Between MEANS and SUMMARY
Originally MEANS was used to generate printed tables and SUMMARY a summary data set. While both procedures can now create either type of output, the defaults for both tend to reflect the original roots of the procedure.

One of the primary differences in defaults is seen by looking at the way each procedure creates printed tables. Printed tables are routed through the Output Delivery System to a destination such as LISTING or HTML. By default MEANS always creates a table to be printed. If you do not want a printed table you must explicitly turn it off (NOPRINT option). On the other hand, the SUMMARY procedure never creates a printed table unless it is specifically requested (PRINT option).

There are a few other differences between MEANS and SUMMARY. In each case the difference reflects default behaviors, and these will be pointed out in the appropriate sections of this paper.

Creating a Basic Summary Table
Very little needs to be done to create a simple summary table. The DATA= option in the PROC statement identifies the data set to be summarized and the VAR statement lists one or more numeric variables to be analyzed.

```plaintext
proc means data=sashelp.class;
var weight;
run;
```

We can see that the mean weight of the 19 students in the CLASS data set is something over 100 pounds. Because we left the selection of the statistics to the defaults, the table contains N, mean, standard deviation, minimum and the maximum.

Selecting Statistics
Generally we want more control over which statistics are to be selected. When you want to specifically select statistics, they are listed as options on the PROC statement.

```plaintext
title1 'The First Two Statistical Moments';
proc means data=sashelp.class
   n mean var std stderr;
var weight;
run;
```

```plaintext
The First Two Statistical Moments

The MEANS Procedure

Analysis Variable : Weight

<table>
<thead>
<tr>
<th>N</th>
<th>Mean</th>
<th>Variance</th>
<th>Std Dev</th>
<th>Std Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>100.0263158</td>
<td>518.6520468</td>
<td>22.7739335</td>
<td>5.2246987</td>
</tr>
</tbody>
</table>
```
The list of available statistics is fairly comprehensive. A subset of which includes:

- **n** number of observations used to calculate the statistics
- **nmiss** number of observations with missing values
- **min** minimum value taken on by the data
- **max** maximum value taken on by the data
- **range** difference between the min and the max
- **sum** total of the data
- **mean** arithmetic mean
- **std** standard deviation
- **stderr** standard error
- **var** variance
- **skewness** symmetry of the data's distribution
- **kurtosis** peakedness of the data's distribution

A number of statistics having to do with percentiles and quantiles are also available, including:

- **median** 50th percentile
- **p50** 50th percentile (or second quartile)
- **p25 | q1** 25th percentile (or first quartile)
- **p75 | q3** 75th percentile (or third quartile)
- **p1 p5 p10** other percentiles
- **p90 p95 p99** other percentiles

Starting in SAS9.2 the MODE statistic is also available.

Statistics listed on the PROC statement are only applied to the printed table and have **NOTHING** to do with and summary data sets that are also created.

**Creating a Summary Data Set**
Both procedures can also be used to create a summary data set through the use of the OUTPUT statement. Without using ODS, a summary data set will not be created unless the OUTPUT statement is present. This is true for both the MEANS and SUMMARY procedures.

```sas
title1 'A Simple Summary Data Set';
proc means data=sashelp.class noprint;
var weight;
output out= summrydat;
run;
```

The NOPRINT option is used with MEANS, because a printed table is not wanted. A PROC PRINT of the summary data set (WORK.SUMMRYDAT) shows the following:

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>STAT</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>19</td>
<td>N</td>
<td>19.000</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>19</td>
<td>MIN</td>
<td>50.500</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>19</td>
<td>MAX</td>
<td>150.000</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>19</td>
<td>MEAN</td>
<td>100.026</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>19</td>
<td>STD</td>
<td>22.774</td>
</tr>
</tbody>
</table>

Again since statistics were not specified the same default list of statistics as was used in the MEANS’s printed table appears here.

**Selecting the Statistics and Naming the Variables in the Summary Data Set**
Usually when you create a summary data set, you will want to specifically select the statistics. These are specified on the OUTPUT statement. Remember statistics listed on the PROC statement only apply to printed tables and have nothing to do with the statistics that you want in the summary data set.

The techniques shown below can be combined - experiment.

**Selecting Statistics**
Statistics are selected by using their names as options in the OUTPUT statement. The name of each statistic is followed by an equal sign. The following OUTPUT statement requests that the mean weight be calculated and saved in the data set SUMMRYDAT.
title1 'Selected Statistics';
proc summary data=sashelp.class;
  var weight;
  output out=summrydat mean=;
run;

The mean weight will be stored in a variable named WEIGHT. This technique allows you to only pick a single statistic, and as such it is limited, however when combined with the techniques shown below, it can be very flexible.

**Explicate Naming**

By following the equal sign with a name, you can provide names for the new variables. This allows you to name more than one statistic on the OUTPUT statement.

```
title1 'Selecting Multiple Statistics';
proc summary data=sashelp.class;
  var weight;
  output out=summrydat n=number mean=average std=std_deviation;
run;
```

You can also name multiple analysis variables. Here both HEIGHT and WEIGHT are specified.

```
title1 'Multiple Analysis Variables';
proc summary data=sashelp.class;
  var height weight;
  output out =summrydat
    n    = ht_n    wt_n
    mean = mean_ht mean_wt
    std  = sd_ht   sd_wt;
run;
```

Be sure to be careful here as the order of the variables in the VAR statement determines which variable is for height and which is for weight. You should also be smart about naming conventions. In the previous example the statistics for N are not consistently named relative to those for the MEAN and STD.

This technique does not allow you to ‘skip’ statistics. If you did not want the mean for HEIGHT, but only the mean for WEIGHT, this would not be possible, because HEIGHT is first on the VAR statement. To get around this you can use the techniques on naming the statistics shown in the next section.
Selected Naming
When there is more than one variable in the VAR statement, but you do not want every statistic calculated for every analysis variable, you can selectively associate statistics with analysis variables.

```
title1 'Selective Associations';
proc summary data=sashelp.class;
  var height weight;
  output out =summrydat
    n =ht_n  wt_n
    mean(weight)= wt_mean
    std(height) = ht_std;
run;
```

Alternate forms of the statistic selections (in this case for the MEAN) could have included the following:

```
mean(weight height)=wt_mean ht_mean
mean(weight)=wt_mean
mean(height)=ht_mean
```

Automatic Naming of Summary Variables
When you do not NEED to control the naming of the new summary variables, the AUTONAME and AUTOLABEL options can be used on the OUTPUT statement.

The AUTONAME option allows you to select statistics without picking a name for the resulting variable in the OUTPUT table. This eliminates naming conflicts. The AUTOLABEL option creates a label for variables added to the OUT= data set.

```
title1 'Using AUTONAME';
proc summary data=sashelp.class;
  var height weight;
  output out =summrydat
    n =
    mean= /autoname;
run;
```

```
Using AUTONAME

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>Height_N</th>
<th>Weight_N</th>
<th>Height_Mean</th>
<th>Weight_Mean</th>
<th>Height_StdDev</th>
<th>Weight_StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>62.3636</td>
<td>100.026</td>
<td>5.12708</td>
<td>22.7739</td>
</tr>
</tbody>
</table>
```

Notice that the names are in the form of `variable_statistic`. This is a nicely consistent, dependable, and usable naming convention.
Using the CLASS Statement

The CLASS statement can be used to create subgroups. Unlike the BY statement the data do not have to be sorted prior to its use. Like in most other procedures that utilize the CLASS statement, there can be one or more classification variables.

In a Printed Table

When the resulting table is to be printed, CLASS creates one summary for each combination of classification variables.

```plaintext
proc means data=sashelp.class(where=(age in(12,13,14)))
   class age sex;
   var height;
run;
```

In a Summary Data Set

When creating a summary data set, one can get not only the classification variable interaction statistics, but the main factor statistics as well. This can be very helpful to the statistician.

```plaintext
proc summary data=sashelp.class(where=(age in(12,13,14)));
   class age sex;
   var height;
   output out=clsummry n=ht_n mean=ht_mean std=ht_sd;
run
```

A PROC PRINT of the data set CLSUMMRY shows:

```
Obs Age Sex _TYPE_ _FREQ_ ht_n ht_mean ht_sd
 1 . . 0 12 12 61.7583 3.97868
 2 . F 1 6 6 60.8333 3.90470
 3 . M 1 6 6 62.6833 4.18637
 4 12 . 2 5 5 59.4400 3.29742
 5 13 . 2 3 3 61.4333 4.49592
 6 14 . 2 4 4 64.9000 2.80119
 7 12 F 3 2 2 58.0500 2.47487
 8 12 M 3 3 3 60.3667 3.93245
 9 13 F 3 2 2 60.9000 3.29742
10 13 M 3 1 1 62.5000 .
11 14 F 3 2 2 63.5500 1.06066
12 14 M 3 2 2 66.2500 3.88907
```

Two additional variables have been added to the summary data set: _TYPE_ (which is described below in more detail), and _FREQ_ (which counts observations). Although not apparent in this example, _FREQ_ counts all observations, while the N
statistic only counts observations with non-missing values.

If you only want the statistics for the highest order interaction, you can use the NWAY option on the PROC statement.

```plaintext
proc summary data=sashelp.class(where=(age in(12,13,14)))
nway;
```

**Understanding _TYPE_**
The _TYPE_ variable in the output data set helps us track the level of summarization, and can be used to distinguish the sets of statistics. Notice in the previous example that _TYPE_ changes for each level of summarization.

- **_TYPE_ = 0** Summarize across all classification variables
- **_TYPE_ = 1** Summarize as if the right most classification variable (SEX) was the only one
- **_TYPE_ = 2** Summarize as if the next to the right most classification variable (AGE) was the only one
- **_TYPE_ = 3** Interaction of the two classification variables.

In the following example there are three CLASS variables and _TYPE_ ranges from 0 to 7.

```plaintext
title1 'Understanding _TYPE_';
proc summary data=advrpt.demog(where=(race in('1','4')
   & 12 le edu le 15
   & symp in('01','02','03')));
  class race edu symp;
  var ht;
  output out=stats mean= meanHT;
run;
```

<table>
<thead>
<tr>
<th>Obs</th>
<th>RACE</th>
<th>EDU</th>
<th>SYMP</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>HT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.</td>
<td></td>
<td></td>
<td>0</td>
<td>8</td>
<td>66.25</td>
</tr>
<tr>
<td>2</td>
<td>.</td>
<td>01</td>
<td>1</td>
<td>2</td>
<td>64.00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.</td>
<td>02</td>
<td>1</td>
<td>4</td>
<td>66.50</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>.</td>
<td>03</td>
<td>1</td>
<td>2</td>
<td>68.00</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td></td>
<td>2</td>
<td>4</td>
<td>67.50</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td></td>
<td>2</td>
<td>2</td>
<td>64.00</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td></td>
<td>2</td>
<td>2</td>
<td>66.00</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>02</td>
<td>3</td>
<td>2</td>
<td>67.00</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>03</td>
<td>3</td>
<td>2</td>
<td>68.00</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>01</td>
<td>3</td>
<td>2</td>
<td>64.00</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>15</td>
<td>02</td>
<td>3</td>
<td>2</td>
<td>66.00</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td></td>
<td>4</td>
<td>6</td>
<td>67.00</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td></td>
<td>4</td>
<td>2</td>
<td>64.00</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>02</td>
<td>5</td>
<td>4</td>
<td>66.50</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>03</td>
<td>5</td>
<td>2</td>
<td>68.00</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>01</td>
<td>5</td>
<td>2</td>
<td>64.00</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>12</td>
<td>6</td>
<td>4</td>
<td>67.50</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>15</td>
<td>6</td>
<td>2</td>
<td>66.00</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>4</td>
<td>14</td>
<td>6</td>
<td>2</td>
<td>64.00</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>12</td>
<td>02</td>
<td>7</td>
<td>67.00</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>12</td>
<td>03</td>
<td>7</td>
<td>68.00</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>15</td>
<td>02</td>
<td>7</td>
<td>66.00</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td>14</td>
<td>01</td>
<td>7</td>
<td>64.00</td>
<td></td>
</tr>
</tbody>
</table>
When calculating the value of \_TYPE\_, assign a zero (0) when summarizing over a CLASS variable and assign a one (1) when summarizing for the CLASS variable. In the table below the zeros and ones associated with the class variables form a binary value. This binary value can be converted to decimal to obtain \_TYPE\_.

<table>
<thead>
<tr>
<th>Observations</th>
<th>RACE</th>
<th>EDU</th>
<th>SYMP</th>
<th>Binary Value</th>
<th>_TYPE_</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2 - 4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5 - 7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>8 - 11</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>12 - 13</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>14 - 16</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>17 - 19</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>20 - 23</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>111</td>
<td>7</td>
</tr>
</tbody>
</table>

A binary value of 110 = 1*2^2 + 1*2^1 + 0*2^0 = 1*4 + 1*2 + 0*1 = 6 = \_TYPE\_

Some SAS programmers find converting binary values to decimal values a bit tedious. Fortunately the developers at SAS Institute have provided us with alternatives.

**Using CHARTYPE**

The CHARTYPE option causes \_TYPE\_ to be displayed as a character variable in binary form rather than as a decimal value.

```
title1 'Understanding \_TYPE\_ Using CHARTYPE';
proc summary data=advrpt.demog(where=(race in('1','4')
& 12 le edu le 15
& symp in('01','02','03'))) chartype;
   class race edu symp;
   var ht;
   output out=stats mean= meanHT;
run;
```
UNDERSTANDING _TYPE_ USING CHARTYPE

<table>
<thead>
<tr>
<th>Obs</th>
<th>RACE</th>
<th>EDU</th>
<th>SYMP</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>HT</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.</td>
<td></td>
<td>000</td>
<td>8</td>
<td>66.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>.</td>
<td>01</td>
<td>001</td>
<td>2</td>
<td>64.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.</td>
<td>02</td>
<td>001</td>
<td>4</td>
<td>66.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>.</td>
<td>03</td>
<td>001</td>
<td>2</td>
<td>68.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td></td>
<td>010</td>
<td>4</td>
<td>67.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td></td>
<td>010</td>
<td>2</td>
<td>64.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td></td>
<td>010</td>
<td>2</td>
<td>66.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>02</td>
<td>011</td>
<td>2</td>
<td>67.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>03</td>
<td>011</td>
<td>2</td>
<td>68.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... portions of the table not shown ... 

CREATING SUMMARY DATA SUBSETS

Once you have started to create summary data sets with MEANS/SUMMARY, you will soon discover how very useful they can be. Of course you will often find that you do not need all the information contained in the summary data set and that you need to create a data subset. As with most things in SAS there are multiple ways to do this. We have already seen the use of the NWAY option to subset for only the highest order interaction. This is fine but not very flexible. Let’s look at some techniques that are a bit more useful.

Select Rows Using _TYPE_

Once you understand and can predict the value of _TYPE_, it can be used to provide subsetting information in a followup DATA step. Suppose that in the previous example we would like to have only those rows for which EDU is a factor. Our DATA step might be written something like:

```sas
data edufactor;
set stats;
where _type_ in(2,3,6,7);
run;
```
Using the WAYS and LEVELS Options

The _TYPE_ variable is only one of several ways to identify levels of summarizations in the summary data set. The WAYS and LEVELS options on the OUTPUT statement provide additional discrimination capabilities. These options add the variables _LEVEL_ and _WAY_ to the summary data table.

```
title1 'Using LEVELS and WAYS Options';
proc summary data=advrpt.demog;
class race edu;
var ht;
output out=stats
   mean= meanHT
   /levels ways;
run;
```

**LEVELS option**

Adds the variable _LEVEL_ to the OUT= data table. This numeric variable counts the observations within _TYPE_. This means that when FIRST_TYPE is true _LEVEL_ will equal 1.

**WAYS option**

Adds the variable _WAY_ to the OUT= data table. This numeric variable equals the number of classification variables that were used to calculate each observation e.g. for a three way interaction _WAY_ will equal 3.

---

Using the WAYS and TYPE Statements

In addition to the WAYS and LEVELS options on the OUTPUT statement there are also the WAYS and TYPE statements than can also be used to control what information is written to the summary data set. These have the further advantage of controlling what is actually calculated and can therefore also save computer resources when there are a large number of classification variables.
Controlling Summary Subsets Using WAYS

The WAYS statement can be used to specify a list of combinations of class variables, which are to be displayed. Combinations of the WAYS statement for three classification variables include the following summarizations:

ways 0; across all class variables
ways 1; each classification variable (no cross products)
ways 2; each two way combination of the classification variables
ways 3; three way combination for three classification variables this is the same as using the NWAY option when there are three classification variables.
ways 0,3; lists of numbers are acceptable

When the number of classification variables becomes large the WAYS statement can utilize an incremental list.

ways 0 to 9 by 3;

In the following example, the main effect summaries (_TYPE_ = 1, 2) are not even calculated.

title1 'Using the WAYS Statement';
proc summary data=advrpt.demog;
class race edu symp;
var ht;
ways 0,2;
output out=stats
    mean= meanHT;
run;

Controlling Summary Subsets Using TYPES

The TYPES statement can be used to select and limit the data roll up summaries. The TYPES statement eliminates much of your need to understand the automatic variable _TYPE_. The TYPES statement is used to list those combinations of the classification variable that are desired. Like the WAYS statement this also can be used to limit the number of calculations that need to be performed.

title1 'Using the TYPES Statement';
proc summary data=advrpt.demog;
class race edu symp;
var ht;

types edu race*symp;
output out=stats mean= meanHT;
run;

Using the WAYS Statement

<table>
<thead>
<tr>
<th>Obs</th>
<th>RACE</th>
<th>edu</th>
<th>SYMP</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>meanHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.</td>
<td>0</td>
<td>64</td>
<td>67.1875</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>04</td>
<td>3</td>
<td>74.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>10</td>
<td>3</td>
<td>69.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>02</td>
<td>3</td>
<td>67.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>1</td>
<td>5</td>
<td>91.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>2</td>
<td>4</td>
<td>66.5000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>65.2857</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>15</td>
<td>6</td>
<td>65.2000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>01</td>
<td>5</td>
<td>71.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>02</td>
<td>5</td>
<td>66.5000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>03</td>
<td>5</td>
<td>68.0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... portions of the table not shown ...

Using the TYPES Statement

<table>
<thead>
<tr>
<th>Obs</th>
<th>RACE</th>
<th>edu</th>
<th>SYMP</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>meanHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>2</td>
<td>9</td>
<td>72.3333</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>2</td>
<td>15</td>
<td>66.2667</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>2</td>
<td>6</td>
<td>68.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>2</td>
<td>9</td>
<td>64.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>2</td>
<td>7</td>
<td>65.2857</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>2</td>
<td>10</td>
<td>70.4000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>17</td>
<td>2</td>
<td>10</td>
<td>65.2000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>2</td>
<td>2</td>
<td>65.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>01</td>
<td>5</td>
<td>71.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>02</td>
<td>5</td>
<td>66.5000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>03</td>
<td>5</td>
<td>68.0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... portions of the table not shown ...
For the following CLASS statement

   class race edu symp;

variations of the TYPES statement could include:

   types ();
   types race*edu edu*symp;
   types race*(edu symp);

Using the CLASSDATA= and EXCLUSIVE Options
You can specify which combinations of levels of the classification variables are to appear in the report by creating a data set that contains the combinations of interest. These can include levels that do not exist in the data itself, but that are to none-the-less appear in the data set or report. The EXCLUSIVE option forces only those levels in the CLASSDATA= data set to appear in the report.

The following example builds the data set that is to be used with the CLASSDATA= option. It also adds a level for each classification variable that does not exist in the data.

   title1 'Using the CLASSDATA and EXCLUSIVE Options';
   data selectlevels(keep=race edu symp);
   set advrpt.demog(where=(race in('1','4')
   & 12 le edu le 15
   & symp in('01','02','03')));
   output;
   * For fun add some nonexistent levels;
   if _n_=1 then do;
      edu=0;
      race='0';
      symp='00';
   output;
   end;
   run;
proc summary data=advrpt.demog
   classdata=selectlevels exclusive;
   class race edu symp;
   var ht;
   output out=stats mean=
      meanHT;
   run;

The summary lines for observations 2 and 6 represent levels of the classification variables that do not appear in the data. They were generated thru a combination of the CLASSDATA= data set and the EXCLUSIVE option.

Using the COMPLETETYPES Option
All combinations of the classification variables may not exist in the data and therefore those combinations will not appear in the summary table. If all possible combinations are desired, regardless as to whether or not they exist in the data, use the COMPLETETYPES option on the PROC statement.

   title1 'Using the COMPLETETYPES Option';
   proc summary data=advrpt.demog(where=(race in('1','4')
   & 12 le edu le 15
   & symp in('01','02','03')))
Using the COMPLETETYPES Option

<table>
<thead>
<tr>
<th>Obs</th>
<th>RACE</th>
<th>EDU</th>
<th>SYMP</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>HT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.</td>
<td>0</td>
<td>8</td>
<td>66.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>.</td>
<td>01</td>
<td>1</td>
<td>2</td>
<td>64.00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.</td>
<td>02</td>
<td>1</td>
<td>4</td>
<td>66.50</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>.</td>
<td>03</td>
<td>1</td>
<td>2</td>
<td>68.00</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>2</td>
<td>4</td>
<td>67.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>2</td>
<td>2</td>
<td>64.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>2</td>
<td>2</td>
<td>66.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>01</td>
<td>3</td>
<td>0</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>02</td>
<td>3</td>
<td>2</td>
<td>67.00</td>
<td></td>
</tr>
</tbody>
</table>

. . . portions of the table not shown . . .

FINDING THE EXTREME VALUES

When working with data, it is not at all unusual to want to be able to identify the observations that contain the highest or lowest values of the analysis variables. These extreme values are automatically displayed in PROC UNIVARIATE output, but must be requested in MEANS/SUMMARY.

As was shown earlier the MIN and MAX statistics show the extreme value, unfortunately they do not identify the observation that contains the extreme. Fortunately there are a couple of ways to do this.

Using MAXID and MINID

The MAXID and MINID options in the OUTPUT statement can be used to identify the observations with the maximum and minimum values. The general form of the statement is:

```
MAXID(analysis var(ID var))=PDV var
```

A new variable is added to the OUTPUT data set which takes on the value of the ID variable for the maximum observation.

```
title1 'Using MAXID';
proc summary data=advrpt.demog;
class race edu;
var ht wt;
output out=stats mean= meanHT MeanWT
      max=maxHT maxWT
      maxid(ht(subject) wt(subject))=maxHtSubject MaxWtSubject
      ;
run;
```
Using MAXID

<table>
<thead>
<tr>
<th>Obs</th>
<th>RACE</th>
<th>EDU</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>meanHT</th>
<th>MeanWT</th>
<th>maxHt</th>
<th>maxWT</th>
<th>Subject</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.</td>
<td>0</td>
<td>75</td>
<td>67.5200</td>
<td>160.267</td>
<td>74</td>
<td>240</td>
<td>110</td>
<td>137</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>11</td>
<td>71.3636</td>
<td>194.091</td>
<td>74</td>
<td>215</td>
<td>110</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>1</td>
<td>18</td>
<td>66.8889</td>
<td>167.722</td>
<td>70</td>
<td>240</td>
<td>106</td>
<td>137</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>1</td>
<td>4</td>
<td>70.0000</td>
<td>197.000</td>
<td>72</td>
<td>215</td>
<td>148</td>
<td>117</td>
<td></td>
</tr>
</tbody>
</table>

. . . . portions of the table not shown . . . .

The OUTPUT statement could also have been written as:

```plaintext
 output out=stats
   mean= meanHT MeanWT
   max=maxHt maxWT
   maxid(ht(subject))=maxHtSubject
   maxid(wt(subject))=maxWtSubject
 ;
```

When more than one variable is needed to identify the observation with the extreme value, the MAXID supports a list. As before when specifying lists, there is a one-to-one correspondence between the two lists. In the following OUTPUT statement both the SUBJECT and SSN are used in the list of identification variables. Consequently a new variable is created for each in the summary data set.

```plaintext
 output out=stats
   mean= meanHT MeanWT
   max=maxHt maxWT
   maxid(ht(subject ssn))=MaxHtSubject MaxHtSSN
   maxid(wt(subject ssn))=MaxWtSubject MaxWtSSN
 ;
```

The MAXID and MINID options allow you to only capture a single extreme. It is also possible to display a group of the extreme values using the GROUPID option.

**Using the GROUPID Option**

Like the MAXID and MINID options, this option allows you to capture the maximum or minimum value and associated ID variable. More importantly, however, you may select more than just the single extreme value.

```plaintext
 title1 'Using GROUPID';
 proc summary data=advrpt.demog;
  class race edu;
  var ht wt;
  output out=stats
    mean= MeanHT MeanWT
    max=maxWT
    idgroup(max(wt)out[2](wt subject race)=maxval)
       ;
 run;
```
Using GROUPID

<table>
<thead>
<tr>
<th>Obs</th>
<th>RACE</th>
<th>EDUC</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>MeanHT</th>
<th>MeanWT</th>
<th>WT</th>
<th>maxval_1</th>
<th>maxval_2</th>
<th>1</th>
<th>2</th>
<th>RACE_1</th>
<th>RACE_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.</td>
<td>0</td>
<td></td>
<td>75</td>
<td>67.5200</td>
<td>160.267</td>
<td>240</td>
<td>215</td>
<td>137</td>
<td>109</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>11</td>
<td>11</td>
<td>71.3636</td>
<td>194.091</td>
<td>215</td>
<td>215</td>
<td>215</td>
<td>109</td>
<td>143</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>1</td>
<td>18</td>
<td>18</td>
<td>66.8889</td>
<td>167.722</td>
<td>240</td>
<td>240</td>
<td>185</td>
<td>137</td>
<td>119</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>70.0000</td>
<td>197.000</td>
<td>215</td>
<td>215</td>
<td>215</td>
<td>117</td>
<td>163</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>1</td>
<td>11</td>
<td>11</td>
<td>64.1818</td>
<td>108.091</td>
<td>115</td>
<td>115</td>
<td>115</td>
<td>131</td>
<td>141</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

... portions of the table not shown ...

1 MAX statistic is superfluous in this example, and is included only for your reference.
2 We are asking for the maximum of WT. GROUPID also is available for MIN, therefore in this example we could have also specified:
   idgroup(min(ht) out[3](ht subject race)=minht minsub minrace)
3 The top 2 values are to be shown
4 This is a list of variables that will be shown as observation identifiers. The analysis variable is usually included. The MAX statistic has also been requested for comparison purposes 1, however it will only provide one value and not the next highest.
5 You can choose the prefix of the ID variable or you can let the procedure do it for you 6. In either case, a number is appended to the variable name. In this example we can see that the second heaviest subject in the study was subject 137 with a weight of 215 pounds and a RACE of 1.

DOING MORE WITH MEANS/SUMMARY
Using Options on CLASS Statements
The CLASS statement can now accept options. These include:

- ASCENDING / DESCENDING
- GROUPINTERNAL
- MISSING
- MLF
- ORDER
- preloaded format options (discussed below)

Most of the following discussion applies to virtually all SAS procedures that accept the CLASS statement.

ASCENDING / DESCENDING
Normally output (in tables or a summary data set) is placed in ascending order for each classification variable. You can change this by using the DESCENDING option on the CLASS statement.

```sas
   title1 'Using the DESCENDING CLASS Option';
   proc summary
     data=advrpt.demog;
     class race / descending;
     var ht wt;
     output out=stats
       mean= MeanHT MeanWT
       ;
   run;
```

Using the DESCENDING CLASS Option

<table>
<thead>
<tr>
<th>Obs</th>
<th>RACE</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>MeanHT</th>
<th>MeanWT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>76</td>
<td>67.5526</td>
<td>160.461</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>66.5000</td>
<td>147.000</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>64.5000</td>
<td>113.500</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>64.8889</td>
<td>111.222</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>17</td>
<td>67.6471</td>
<td>162.000</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>42</td>
<td>68.4762</td>
<td>176.143</td>
</tr>
</tbody>
</table>
GROUPINTERNAL
When a classification variable is associated with a format, the format is used when forming groups.

```latex
proc format;
value edulevel
0-12 = 'High School'
13-16= 'College'
17-high='Post Graduate';
run;

title1 'Without Using the GROUPINTERNAL CLASS Option';
proc summary data=advrpt.demog;
    class edu;
    var ht wt;
    output out=stats
        mean= MeanHT MeanWT
    ;
    format edu edulevel.;
run;
```

The resulting table will show at most three levels for EDU. To use the original data values (internal values), the GROUPINTERNAL option is added to the CLASS statement.

```latex
    class edu/groupinternal;
```

MISSING
When a classification variable takes on a missing value that observation is eliminated from the analysis. If a missing value is OK or if the analyst needs to have it included in the summary, the MISSING option can be used. Most procedures that have either an implicit or explicit CLASS statement also have a MISSING option. However when the MISSING option is used on the PROC statement it is applied to all the classification variables and this may not be acceptable. By using the MISSING option on the CLASS statement you can control which classification variables are to be handled differently.

In the following example there are three classification variables. However the MISSING option has only been applied to two of them.

```latex
title1 'Using the MISSING CLASS Option';
proc means data=advrpt.demog n mean std;
    class race ;
    class edu symp/ missing;
    var ht wt;
run;
```

ORDER
When classification variables are displayed or written to a table the values are ordered according to one of several possible schemes. These include:

- **data** order is based on the order of the incoming data
- **formatted** values are formatted and then ordered (default when the variable is formatted)
- **freq** the order is based on the frequency of the class level
- **unformatted** same as INTERNAL or GROUPINTERNAL

Using the `order=freq` option on the CLASS statement causes the table to be ordered according to the most common levels of education.

```latex
    class edu/order=freq;
```
Using the ORDER CLASS Option

The MEANS Procedure

<table>
<thead>
<tr>
<th>years of education</th>
<th>N</th>
<th>Obs</th>
<th>Variable</th>
<th>Label</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>19</td>
<td></td>
<td>HT</td>
<td>height in inches</td>
<td>19</td>
<td>66.9473684</td>
<td>2.7582942</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WT</td>
<td>weight in pounds</td>
<td>19</td>
<td>171.5263158</td>
<td>32.2703311</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td></td>
<td>HT</td>
<td>height in inches</td>
<td>11</td>
<td>64.1818182</td>
<td>0.4045199</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WT</td>
<td>weight in pounds</td>
<td>11</td>
<td>108.0909091</td>
<td>4.3921417</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td></td>
<td>HT</td>
<td>height in inches</td>
<td>11</td>
<td>71.3636364</td>
<td>3.2022719</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WT</td>
<td>weight in pounds</td>
<td>11</td>
<td>194.0909091</td>
<td>19.0811663</td>
</tr>
<tr>
<td>17</td>
<td>10</td>
<td></td>
<td>HT</td>
<td>height in inches</td>
<td>10</td>
<td>65.2000000</td>
<td>2.3475756</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WT</td>
<td>weight in pounds</td>
<td>10</td>
<td>145.2000000</td>
<td>25.0900600</td>
</tr>
</tbody>
</table>

... portions of the table not shown ...

Using Multiple CLASS Statements

Because CLASS statements now accept options, and because those options may not apply to all the classification variables, it is often necessary to specify multiple CLASS statements - each with its own set of options.

With or without options, when multiple CLASS statements are specified, the order of the statements themselves becomes important. The following CLASS statement

```
class race edu;
```

could be rewritten as

```
class race;
class edu;
```

PRELOADED FORMATS

Several options and techniques are available to control which levels of classification variables are to appear in the summary. Those that were discussed earlier in this paper include the CLASSDATA and COMPLETETYPES options. Also discussed were the WAYS and TYPES statements, as well as the WAYS and LEVELS options on the OUTPUT statement.

A related set of options come under the general topic of Preloaded Formats. Variations of these options are available for most of the procedures that utilize classification variables. Like the others listed above these techniques/options are used to control the relationship of levels of classification variables that may not appear in the data and how those levels are to appear (or not appear) in the summary.

Generally speaking when a level of a classification variable is not included in the data, the associated row will not appear in the table. This behavior relative to the missing levels can be controlled through the use of preloaded formats.

For the MEANS/SUMMARY procedures, options used to preload formats include:

```
PRELOADFMT  Loads the format levels prior to execution. This option will always be present when you want to use a preloaded format.
EXCLUSIVE    Only data levels that are included in the format definition are to appear in summary table
COMPLETETYPES All levels representing format levels are to appear in the summary
```

17
It is the interaction of these three options that gives us a wide range of possible outcomes. In each case the option PRELOADFMT will be present.

As the name of the technique implies, the control is maintained through the use of user defined formats. For the examples that follow, the format $SYMPX has been created, and it contains one level, '00', that is not in the data. In the data the values of SYMP range from '01' to '10'.

```
proc format;
  value $sympx
    '01' = 'Sleepiness'
    '02' = 'Coughing'
    '00' = 'Bad Code'
run;
```

**PRELOADFMT with EXCLUSIVE**

Preloading with the CLASS statement options PRELOADFMT and EXCLUSIVE limits the levels of the classification variable to those that are both on the format and in the data. Essentially the format acts as a filter without resorting to either a subsetting IF or a WHERE clause.

```
title1 'Preloading and the EXCLUSIVE Option';
proc summary data=advrpt.demog;
  class symp / preloadfmt exclusive;
  var ht;
  output out=stats mean= meanHT;
  format symp $sympx.;
run;
```

Symptoms that are not both on the format $SYMPX. and in the data, are not included on the summary table.

**PRELOADFMT with the COMPLETETYPES Option**

The COMPLETETYPES option requests that all combinations of levels appear in the summary. When it is used with preloaded formats, the complete list of levels comes from the format rather than from the data itself. In this example the format %SYMPX. is again preloaded, however rather than using the EXCLUSIVE CLASS statement option, the COMPLETETYPES option appears on the PROC statement.

```
title1 'Preloading and the COMPLETETYPES Option';
proc summary data=advrpt.demog completetypes;
  class symp / preloadfmt;
  var ht;
  output out=stats mean= meanHT;
  format symp $sympx.;
run;
```

The summary now contains an observation for each SYMP in the data as well as each in the format $SYMPX.
PRELOADFMT with the COMPLETETYPES and the EXCLUSIVE Options

When a preloaded format is used with both the COMPLETETYPES and the EXCLUSIVE options, the summary includes all levels of the format, but not necessarily all levels in the data.

```sas
title1 'Preloading With Both';
title2 'the COMPLETETYPES and EXCLUSIVE Options';
proc summary data=advrpt.demog
   completetypes;
   class symp / preloadfmt exclusive;
   var ht;
   output out=stats mean= meanHT;
   format symp $sympx.;
run;
```

<table>
<thead>
<tr>
<th>Obs</th>
<th>SYMP</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>HT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>14</td>
<td>67.0</td>
</tr>
<tr>
<td>2</td>
<td>Bad Code</td>
<td>1</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>Sleepiness</td>
<td>1</td>
<td>4</td>
<td>67.5</td>
</tr>
<tr>
<td>4</td>
<td>Coughing</td>
<td>1</td>
<td>10</td>
<td>66.8</td>
</tr>
</tbody>
</table>

SUMMARY

The MEANS /SUMMARY procedure produces a wide variety of summary reports and summary data tables. It is very flexible and, while it can be quite complex, a few basic statements allow the user to create useful summaries.

As you develop a deeper knowledge of the MEANS/SUMMARY procedure, you will find that the generation of highly sophisticated summarizations is possible from within a single step.

ABOUT THE AUTHOR

Art Carpenter’s publications list includes four books, and numerous papers and posters presented at SUGI, SAS Global Forum, 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
10606 Ketch Circle
Anchorage, AK 99515

(907) 865-9167
art@caloxy.com
www.caloxy.com

TRADEMARK INFORMATION

SAS, SAS Certified Professional, SAS Certified Advanced Programmer, and all other SAS Institute Inc. product or service names are registered trademarks of SAS Institute, Inc. in the USA and other countries.
® indicates USA registration.