Abstract

The PROC SUMMARY procedure allows the user to obtain statistical analyses on data obtained from a permanent, or working storage, SAS data set. This tutorial presents the basic concepts of using the procedure through examples. Using sample code followed by output produced through the procedure (printed using PROC PRINT), the reader may use the examples to see the cause and effect of most options of the procedure.

Introduction

PROC SUMMARY is one of the procedures that should be included in every SAS programmer's toolset. It offers many features for obtaining and manipulating simple statistical output for both the neophyte and experienced programmer. The purpose of the procedure may be summarized as follows:

* To compute summary statistics for specified levels of subgrouped observations. The resulting statistics will be assigned to new variables, while the old variables are dropped.
* To produce a SAS data set for use with subsequent data steps or procedures.

The power of the statistics is in the packaged options SAS gives the procedure. The statistics available are listed in Figure 11.

Of the many options which the SAS Institute, Inc. makes available with the procedure, aside from the statistical gathering capabilities, PROC SUMMARY has the option of allowing the output to be sent to a working or permanent SAS data set. This allows the programmer to apply the output in many ways. The following is a list of the uses often used:

* Data steps following PROC SUMMARY
* PROC PRINT
* PROC TABULATE

The PROC SUMMARY examples that follow are based on sample data as might be used in the transportation industry. The listing of the measured data, although not listed in the paper, may be obtained from the author. The program code used to create the original SAS data set used as a basis for the example may also be obtained from the author.

The following variables are used by this paper in demonstrating the manipulative powers of the procedure:

<table>
<thead>
<tr>
<th>SAS Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REVENUE</td>
<td>A billed dollar amount (A numeric in dollars &amp; cents)</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>Weight of a shipment (A numeric in pounds)</td>
</tr>
<tr>
<td>CITY</td>
<td>Originating City of shipment (A character value)</td>
</tr>
<tr>
<td>ST</td>
<td>Originating City's State (A character variable)</td>
</tr>
<tr>
<td>PIECES</td>
<td>The number of boxes, pallets, or cartons in a single shipment (A numeric variable)</td>
</tr>
<tr>
<td>DATE</td>
<td>The date of delivery (A character variable)</td>
</tr>
<tr>
<td>FACTOR1</td>
<td>Weighting factor (A numeric variable = .1)</td>
</tr>
</tbody>
</table>

Case 1: The 'Slightly Simple Summary' example:

Syntax:

```
PROC SUMMARY;
VAR REVENUE;
OUTPUT;
```

The output may be seen in Figure 1.

![Figure 1](image)

Comments:

* The VAR statement lists the numeric variables for which summary statistics are desired. This is a required statement for PROC SUMMARY.
* The output statement is required. Specifying OUTPUT without options will create a new data set specified as _DATA_. The _DATA_ data set may then be used as if the programmer had used the OUT = option.

169
* The **DATA =** option is not used in this example, as it is not required. When this option is not used, SAS uses the data set created last, prior to the procedure.

* The SAS variable **_TYPE_** is created containing the level of subgroup specified. In this example **_TYPE_ = 0**, specifying the 'total' of the summary levels available.

* The SAS variable **_FREQ_** is created by the procedure and contains the number of observations in the subgroup defined. All observations, including missing observations within the subgroup are included.

**Case 2**: The 'Let's get OUT and have fun' example:

Syntax:

```sas
PROC SUMMARY DATA = DSN1;
VAR REVENUE WEIGHT;
OUTPUT OUT = DSN2 SUM = ;
```

The output may be seen in Figure 2.

![Figure 2](image)

**Figure 2**

Comments:

* The **DATA =** option is specified for clarity of coding. As mentioned above, this option is not required.

* The **VAR** statement lists the numeric variables possibly of interest for summary statistics. Extra variables may be listed here (eg. for testing), but are not recommended.

* The **VAR** statement must precede the **OUTPUT** statement.

* Listing more variables than are listed in the **VAR** statement will result in a warning that the **SUM** variable listing will be truncated to match the **VAR** listing.

* The **OUTPUT** statement has two parts:
  1) **OUT = data set name**

  This option creates a SAS data set which may be used in later data steps or PROC's.

  A two level data set name may be specified here to send output to a permanent SAS data set.

  2) **SUM = summary variable name**

  **SUM** is only one of many statistics that may be used against variables listed in the **VAR** statement. See Figure 10 for other statistics available.

  Using **SUM = ** without specifying any new variable names will result in SAS re-using the names listed on the **VAR** statement in the output data set as the summary variables.

  **DO NOT** use multiple keyword's syntax:

  \[\text{Statistic1} = \text{Statistic2} =\]

  without specifying the output variable names. Doing so for more than one statistic will cause the PROC to attempt to create the output data set with duplicated names.

**Case 3**: The 'CLASS, may I have your attention?' example:

Syntax:

```sas
PROC SUMMARY DATA = DSN1 MISSING;
CLASS CITY;
VAR REVENUE WEIGHT;
OUTPUT OUT = DSN2 SUM = REVENUE
     MEAN(SHIP_WT)=MEAN_WT;
```

The output may be seen in Figure 3.

![Figure 3](image)

**Figure 3**

Comments:

* The **CLASS** statement allows for subgrouping of summary observation variables based upon the change of value of the variables listed.

  * The maximum number of **CLASS** variables is 24.

  * Using the **CLASS** statement limits the combination of **CLASS** levels to 32767.

  * The **MISSING** option is specified to allow for any **CLASS** variable's observation to be included in the statistics, if the variable has a missing value.
• The _TYPE_ variable now identifies subgrouping of observation as specified. In this example _TYPE_ = 0, specifies the 'total' of all summary levels available, with each subgroup level increasing the value of _TYPE_ by one.

• _FREQ_ now takes on added meaning. At _TYPE_ -= 0, the _FREQ_ of a given subgroup will total all observations. At each separate subgroup level, _FREQ_ will total the number of observations at that level.

• The new variable names given to summary statistics may be specified as shown in the example. The variable name listed first following the first statistic will match the corresponding variable as ordered on the VAR statement; the variable name listed next corresponds to the variable as ordered next on the VAR statement; etc.

• The ordering defaults of the procedure's variable to statistics correspondence (in the statistic's specification of the OUTPUT statement) may be eliminated by encapsulating the VAR statement variable in parentheses prior to assigning the new summary variable name.

• The original VAR statement variables will be dropped from the output data set once the summary variables have been created.

Case 4: The 'DESCENDING into the abyss' example:

Syntax:
```sql
PROC SUMMARY DATA = DSN1 DESCENDING;
CLASS CITY;
VAR REVENUE WEIGHT;
OUTPUT OUT = DSN2 SUM = REVENUE MEAN(WEIGHT) = MEAN_WT;
```

The output may be seen in Figure 4.

<table>
<thead>
<tr>
<th>OBS</th>
<th>CITY</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>REVENUE</th>
<th>MEAN_WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Anchorage</td>
<td>1</td>
<td>15</td>
<td>12948.9</td>
<td>2892.2</td>
</tr>
<tr>
<td>2</td>
<td>Juneau</td>
<td>1</td>
<td>2</td>
<td>1075.0</td>
<td>4235.0</td>
</tr>
<tr>
<td>3</td>
<td>Memphis</td>
<td>1</td>
<td>7</td>
<td>4143.7</td>
<td>3902.6</td>
</tr>
<tr>
<td>4</td>
<td>Miami</td>
<td>1</td>
<td>4</td>
<td>2581.3</td>
<td>8583.8</td>
</tr>
<tr>
<td>5</td>
<td>Nashville</td>
<td>1</td>
<td>13</td>
<td>8978.6</td>
<td>4711.3</td>
</tr>
<tr>
<td>6</td>
<td>Orlando</td>
<td>1</td>
<td>17</td>
<td>9001.6</td>
<td>3985.1</td>
</tr>
<tr>
<td>7</td>
<td>San Jose</td>
<td>1</td>
<td>10</td>
<td>8576.2</td>
<td>11202.7</td>
</tr>
</tbody>
</table>

Figure 4

Comments:

* Everything stays the same in this example with the exception of using the DESCENDING option.

• The DESCENDING option merely reverses the order of the _TYPE_ observations in the output data set. Note that the order of the observations within a subgroup do not change.

• This option may be useful to some presentations.

Case 5: The 'How much does an NWAY?' example:

Syntax:
```sql
PROC SUMMARY DATA = DSN1 NWAY;
CLASS CITY;
VAR REVENUE WEIGHT;
OUTPUT OUT = DSN2 SUM = REVENUE MIN(WEIGHT) = MIN_WT;
```

The output may be seen in Figure 5.

<table>
<thead>
<tr>
<th>OBS</th>
<th>CITY</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>REVENUE</th>
<th>MEAN_WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Anchorage</td>
<td>1</td>
<td>15</td>
<td>12948.9</td>
<td>2892.2</td>
</tr>
<tr>
<td>2</td>
<td>Juneau</td>
<td>1</td>
<td>2</td>
<td>1075.0</td>
<td>4235.0</td>
</tr>
<tr>
<td>3</td>
<td>Memphis</td>
<td>1</td>
<td>7</td>
<td>4143.7</td>
<td>3902.6</td>
</tr>
<tr>
<td>4</td>
<td>Miami</td>
<td>1</td>
<td>4</td>
<td>2581.3</td>
<td>8583.8</td>
</tr>
<tr>
<td>5</td>
<td>Nashville</td>
<td>1</td>
<td>13</td>
<td>8978.6</td>
<td>4711.3</td>
</tr>
<tr>
<td>6</td>
<td>Orlando</td>
<td>1</td>
<td>17</td>
<td>9001.6</td>
<td>3985.1</td>
</tr>
<tr>
<td>7</td>
<td>San Jose</td>
<td>1</td>
<td>10</td>
<td>8576.2</td>
<td>11202.7</td>
</tr>
</tbody>
</table>

Figure 5

Comments:

* Everything stays the same in this example with the exception of using the NWAY option.

* The NWAY option allows only observations with the highest value of the _TYPE_ variable to be output to the procedure's resulting data set.

Case 6: The 'Let's have some ORDER here!' example:

Syntax:
```sql
PROC SUMMARY DATA = DSN1 ORDER=FREQ;
CLASS CITY;
VAR REVENUE WEIGHT;
OUTPUT OUT = DSN2 SUM = REVENUE MIN(WEIGHT) = MIN_WT;
```

The output may be seen in Figure 6.

Comments:

* The ORDER = option allows the programmer to sort the values of the CLASS variables.

* ORDER = FREQ orders the output observations in descending order of frequency.

* ORDER = DATA orders the summary observations as found in the input data set.
* ORDER = EXTERNAL orders the summary observations in their external format.

* The default order scheme is ORDER = INTERNAL, or internal representation sorted order.

Case 7: The 'Are you BYing all this?' example

Syntax:

```plaintext
PROC SUMMARY DATA=DSN1;
    BY STATE;
    CLASS CITY;
    VAR REVENUE WEIGHT;
    OUTPUT OUT=DSN2 SUM=REVENUE MEAN(WEIGHT)=MEAN_WT;
```

The output may be seen in Figure 7.

**Figure 7**

<table>
<thead>
<tr>
<th>ST</th>
<th>CITY</th>
<th>TYPE</th>
<th>FREQ</th>
<th>REVENUE</th>
<th>MEAN_WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AK</td>
<td>Anchorage</td>
<td>0</td>
<td>17</td>
<td>14023.9</td>
<td>3050.2</td>
</tr>
<tr>
<td>AK</td>
<td>Anchorage</td>
<td>1</td>
<td>15</td>
<td>12548.9</td>
<td>2992.2</td>
</tr>
<tr>
<td>CA</td>
<td>Juneau</td>
<td>0</td>
<td>2</td>
<td>1076.0</td>
<td>2350.0</td>
</tr>
<tr>
<td>CA</td>
<td>Juneau</td>
<td>1</td>
<td>10</td>
<td>8076.2</td>
<td>11202.7</td>
</tr>
<tr>
<td>CA</td>
<td>San Jose</td>
<td>0</td>
<td>10</td>
<td>8576.2</td>
<td>11202.7</td>
</tr>
<tr>
<td>FL</td>
<td>Miami</td>
<td>0</td>
<td>21</td>
<td>11613.9</td>
<td>4661.0</td>
</tr>
<tr>
<td>FL</td>
<td>Miami</td>
<td>1</td>
<td>4</td>
<td>2812.3</td>
<td>3863.8</td>
</tr>
<tr>
<td>FL</td>
<td>Orlando</td>
<td>0</td>
<td>17</td>
<td>9001.6</td>
<td>3197.7</td>
</tr>
<tr>
<td>TN</td>
<td>Memphis</td>
<td>0</td>
<td>20</td>
<td>13122.3</td>
<td>3493.3</td>
</tr>
<tr>
<td>TN</td>
<td>Nashville</td>
<td>1</td>
<td>7</td>
<td>4143.7</td>
<td>3862.6</td>
</tr>
<tr>
<td>TN</td>
<td>Nashville</td>
<td>1</td>
<td>13</td>
<td>8978.6</td>
<td>4711.3</td>
</tr>
</tbody>
</table>

**Figure 6**

* Using the BY statement may be considered as an expanded CLASS statement. Separate _TYPE_ analyses (_TYPE_ = 0, 1, etc.) will be created.

* Using the BY statement will increase the length of CPU time taken, but the memory requirements will be less than when using the CLASS statement. The BY statement releases memory resources each time the BY group changes.

* This example's output was printed with PROC PRINT, using the ID option of the procedure, thus eliminating the OBS variable in the printout.

Case 8: The 'Is this some kind of FREQ ?' example

Syntax:

```plaintext
PROC SUMMARY DATA=DSN1 NWAY;
    CLASS ORIGIN;
    VAR REVENUE WEIGHT;
    FREQ PIECES;
    OUTPUT OUT=DSN2 SUM=REVENUE MEAN(WEIGHT)=MEAN_WT;
```

The output may be seen in Figure 8.

**Figure 8**

<table>
<thead>
<tr>
<th>OBS</th>
<th>ORIGIN</th>
<th>TYPE</th>
<th>FREQ</th>
<th>REVENUE</th>
<th>MEAN_WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Anchorage</td>
<td>1</td>
<td>17</td>
<td>899</td>
<td>1417734.0</td>
</tr>
<tr>
<td>3</td>
<td>Juneau</td>
<td>1</td>
<td>169</td>
<td>90836.1</td>
<td>4611.3</td>
</tr>
<tr>
<td>4</td>
<td>Memphis</td>
<td>1</td>
<td>576</td>
<td>496861.6</td>
<td>3865.1</td>
</tr>
<tr>
<td>5</td>
<td>Miami</td>
<td>1</td>
<td>280</td>
<td>227515.4</td>
<td>6219.3</td>
</tr>
<tr>
<td>6</td>
<td>Nashville</td>
<td>1</td>
<td>992</td>
<td>492003.4</td>
<td>3197.7</td>
</tr>
<tr>
<td>7</td>
<td>Orlando</td>
<td>1</td>
<td>378</td>
<td>157607.2</td>
<td>2802.7</td>
</tr>
<tr>
<td>8</td>
<td>San Jose</td>
<td>1</td>
<td>222</td>
<td>206291.3</td>
<td>13089.1</td>
</tr>
</tbody>
</table>

**Figure 7**

Comments:

* The BY statement is used to derive separate analyses on subgroups defined by the BY variable(s).

* The SUMMARY procedure requires that the input data set be previously sorted in the order specified with the BY.

* Using the BY statement may be considered as an expanded CLASS statement. Separate _TYPE_ analyses (_TYPE_ = 0, 1, etc.) will be created.

* Using the BY statement will increase the length of CPU time taken, but the memory requirements will be less than when using the CLASS statement. The BY statement releases memory resources each time the BY group changes.

* This example's output was printed with PROC PRINT, using the ID option of the procedure, thus eliminating the OBS variable in the printout.

Case 8: The 'Is this some kind of FREQ ?' example

Syntax:

```plaintext
PROC SUMMARY DATA=DSN1 NWAY;
    CLASS ORIGIN;
    VAR REVENUE WEIGHT;
    FREQ PIECES;
    OUTPUT OUT=DSN2 SUM=REVENUE MEAN(WEIGHT)=MEAN_WT;
```

The output may be seen in Figure 8.

<table>
<thead>
<tr>
<th>OBS</th>
<th>ORIGIN</th>
<th>TYPE</th>
<th>FREQ</th>
<th>REVENUE</th>
<th>MEAN_WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Anchorage</td>
<td>1</td>
<td>17</td>
<td>899</td>
<td>1417734.0</td>
</tr>
<tr>
<td>3</td>
<td>Juneau</td>
<td>1</td>
<td>169</td>
<td>90836.1</td>
<td>4611.3</td>
</tr>
<tr>
<td>4</td>
<td>Memphis</td>
<td>1</td>
<td>576</td>
<td>496861.6</td>
<td>3865.1</td>
</tr>
<tr>
<td>5</td>
<td>Miami</td>
<td>1</td>
<td>280</td>
<td>227515.4</td>
<td>6219.3</td>
</tr>
<tr>
<td>6</td>
<td>Nashville</td>
<td>1</td>
<td>992</td>
<td>492003.4</td>
<td>3197.7</td>
</tr>
<tr>
<td>7</td>
<td>Orlando</td>
<td>1</td>
<td>378</td>
<td>157607.2</td>
<td>2802.7</td>
</tr>
<tr>
<td>8</td>
<td>San Jose</td>
<td>1</td>
<td>222</td>
<td>206291.3</td>
<td>13089.1</td>
</tr>
</tbody>
</table>

**Figure 7**

Comments:

* The FREQ statement is used to specify that each observation is representing \( n \) observations of the input data set. The value \( n \) is the value of the variable listed. Derive separate analyses on subgroups defined by the BY variable(s).

* Only one variable is allowed with the FREQ statement. Attempting to use more than one will result in an Error 1: Syntax Error: Expecting Semicolon.

* Only numeric variables are allowed in the FREQ statement. If the value of the FREQ variable is not an integer, the value will be truncated to its integer position.
* If the value of the specified variable is missing or less than one, the procedure will skip that observation. This is the case, even if the MISSING option is included in the PROC SUMMARY statement.

Case 9: The 'WEIGHT'y issue isn't it ?" example

Syntax:

```
PROC SUMMARY DATA = DSN1 NWAY;
CLASS CITY ;
VAR REVENUE WEIGHT;
WEIGHT FACTOR1;
OUTPUT OUT = DSN2 SUM = REVENUE MEAN(WEIGHT)= MEAN_WT;
```

The output may be seen in Figure 9.

<table>
<thead>
<tr>
<th>OBS</th>
<th>ORIGIN</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>REVENUE</th>
<th>MEAN_WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Anchorage</td>
<td>1</td>
<td>15</td>
<td>1294.89</td>
<td>2892.2</td>
</tr>
<tr>
<td>3</td>
<td>Juneau</td>
<td>1</td>
<td>2</td>
<td>107.50</td>
<td>4235.0</td>
</tr>
<tr>
<td>4</td>
<td>Memphis</td>
<td>1</td>
<td>7</td>
<td>414.37</td>
<td>3802.6</td>
</tr>
<tr>
<td>5</td>
<td>Miami</td>
<td>1</td>
<td>4</td>
<td>2812.3</td>
<td>8583.8</td>
</tr>
<tr>
<td>6</td>
<td>Nashville</td>
<td>1</td>
<td>13</td>
<td>8978.6</td>
<td>4711.3</td>
</tr>
<tr>
<td>7</td>
<td>Orlando</td>
<td>1</td>
<td>17</td>
<td>9001.6</td>
<td>3985.1</td>
</tr>
<tr>
<td>8</td>
<td>San Jose</td>
<td>1</td>
<td>10</td>
<td>8576.2</td>
<td>11202.7</td>
</tr>
</tbody>
</table>

Figure 9

Comments:

* The WEIGHT statement is used to assign a variable to be used in weighting each observation.

* Only one variable is allowed with the WEIGHT statement. Attempting to use more than one will result in an Error 1: Syntax Error: Expecting Semicolon.

* Only a numeric variable is allowed in the WEIGHT statement. The variable's value may be a non-integer and will be used in the calculation.

* If the value of the specified variable is missing or less than one, the procedure will give the weighted variable a value of zero.

Case 10: The 'May I see to your ID ?" example

Syntax:

```
PROC SUMMARY DATA = DSN1 NWAY;
CLASS CITY ;
VAR REVENUE WEIGHT;
IODATE;
OUTPUT OUT = DSN2 SUM = REVENUE MEAN(WEIGHT)= MEAN_WT;
```

The output may be seen in Figure 10.

<table>
<thead>
<tr>
<th>DATE</th>
<th>ORIGIN</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>REVENUE</th>
<th>MEAN_WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>03/19</td>
<td>Anchorage</td>
<td>1</td>
<td>15</td>
<td>1294.89</td>
<td>2892.2</td>
</tr>
<tr>
<td>03/22</td>
<td>Juneau</td>
<td>1</td>
<td>2</td>
<td>107.50</td>
<td>4235.0</td>
</tr>
<tr>
<td>03/09</td>
<td>Memphis</td>
<td>1</td>
<td>7</td>
<td>414.37</td>
<td>3802.6</td>
</tr>
<tr>
<td>03/11</td>
<td>Miami</td>
<td>1</td>
<td>4</td>
<td>2812.3</td>
<td>8583.8</td>
</tr>
<tr>
<td>03/26</td>
<td>Nashville</td>
<td>1</td>
<td>13</td>
<td>8978.6</td>
<td>4711.3</td>
</tr>
<tr>
<td>03/18</td>
<td>Orlando</td>
<td>1</td>
<td>17</td>
<td>9001.6</td>
<td>3985.1</td>
</tr>
<tr>
<td>03/14</td>
<td>San Jose</td>
<td>1</td>
<td>10</td>
<td>8576.2</td>
<td>11202.7</td>
</tr>
</tbody>
</table>

Figure 10

Comments:

* The ID statement is used to include additional variables in summary observations.

* If the ID statement contains one variable, the variable's value for the output summary observation is the maximum value of the subgrouped observations.

* If the ID statement has two or more variables, then the maximum value is chosen as if the variables were concatenated together, and then the maximum value chosen. The order of listing on the ID statement will affect the value output.

In Conclusion

The PROC SUMMARY procedure is a versatile tool for obtaining summary statistics. Usable at all levels of programming, the procedure's strength lies in its ability to create summary statistics without lengthy coding, coupled with its ability to place the resulting observation(s) in either a working, or permanent, SAS data set. The re-usability of the output gives the Base SAS product line a depth not easily achieved by other methods.

N
The number of subgroup observations, excluding missing values.

NMISS The number of subgroup observations, including missing values.

MEAN The mean

STD The standard deviation

MIN The minimum value

MAX The maximum value

RANGE The range of value

SUM The sum of the observation

VAR The variance

CSS The corrected sum of the squares

USS The uncorrected sum of the squares

CV The coefficient of variation

STDERR The standard error of the mean

T The t value used in testing whether a population's mean = 0

PRT The probability of a greater absolute value for the Student's t value above

SUMWGT sum of the WEIGHT variable values

Figure 11

173
Author

John Boyden
Viking Freight, Inc.
411 E. Plumeria Dr., Suite 15
San Jose, CA 95134
(408) 922-7200 x2525

References


SAS is a registered trademark of SAS Institute, Inc., Cary, N.C., USA