SUM Enchanted Procedure:
The Advanced Power of PROC SUMMARY

David Beam
Systems Seminar Consultants
Madison, WI 53716 (608) 222-7081

Abstract

Many SAS users are uncomfortable or unsure about the inherent power of PROC SUMMARY. The creation of summarized data sets, with emphasis on the CLASS statement will be demonstrated in this paper.

Explanations of how SAS organizes the output file will help the user take advantage of this powerful SAS tool. There are many techniques of preparing new variables before going into PROC SUMMARY that can allow the procedure to generate counters and statistics, as well as the traditional sums.

The use of PROC FORMAT together with PROC SUMMARY can also allow much flexibility in the use of this procedure.

Introduction

PROC SUMMARY is a basic ‘statistical’ procedure, in that numeric variables can be used to compute a variety of statistics, including averages, sums, standard deviations, etc. Often overlooked is how this procedure can save programming time when used to full potential.

This paper is not meant to be a statistics course, but will look at the procedure for very common business analysis needs.

In the simple sense, PROC SUMMARY will read in a data set and create an output data set with whatever statistics are needed. The statistics coming out can be grouped together by values of the CLASS variables, or only a single total can be created.

The two procedures, MEANS and SUMMARY, are now (with release 6 or higher of The SAS System) virtually identical. The main difference is that SUMMARY does not produce a printed report as a default, but PROC MEANS assumes a report is needed.

The basic syntax of PROC SUMMARY is shown here. Many of the optional statements and keywords will be discussed later. This is NOT meant to replace the SAS documentation, refer to the SAS Language: Reference manual for complete description of the procedure.

```
PROC SUMMARY <option list> <statistic-keyword-list>;
    VAR variable-list;
    CLASS variable-list;
    FREQ variable;
    WEIGHT variable;
    BY variable-list;
    OUTPUT <OUT=SAS-data-set>
            <output-statistic-list>
            <statistic(variable)=new-var-name>
            <statistic(variable)=new-var-name>
    ;
RUN;
```

Partial List of Keywords on the PROC Statement

- DATA= data set name (input)
- DESCENDING orders the output data set in order of the _TYPE_ variable, default is ascending
- MISSING treats missing values for the CLASS variable as a valid subgroup
- NWAY only the highest value for the _TYPE_ variable will be output
- ORDER= DATA | EXTERNAL | FORMATTED | FREQ | INTERNAL specifies sorting order of the CLASS variables for output of the data set. Default is INTERNAL, which is sequential unformatted values.

Partial List of Keyword Statistics Available

- N the count of observations having non-missing values for the selected variable
- NMISS the count of missing values for the variable
- MIN the minimum value
- MAX the maximum value
- SUM the total of the specified variable
- MEAN the average (mean)
- STD the standard deviation
Advanced Tutorials

If no keyword statistics are present, PROC SUMMARY produces the following default statistics:

N, MEAN, STD, MIN, MAX

Statements used with PROC SUMMARY

The following is a partial list of the statements that can be used to control the procedure:

BY variables;
   Names variable(s) that will be used to obtain separate statistics when the value changes. The data set must be sorted or indexed by these variables.

CLASS variables;
   Used to form sub-groups of output data. Similar to the BY statement, but does not require sorting the data first.
With multiple variables on the CLASS statement, the procedure will generate statistics for each unique value of the CLASS variable, as well as combination statistics for all combinations of all CLASS values.

OUTPUT statement
   This "drives" the procedure. The OUTPUT statement can name the new data set being created, select the variables and statistics to generate, and name the new variables being created.
   There is no limit to the number of OUTPUT statements in a single PROC SUMMARY.

Sample Data Set

The examples that follow will use the following set of data as input. The CONTENTS of the data set is:

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STATE</td>
<td>Char</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>SALESDEP</td>
<td>Char</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>MONTH</td>
<td>Char</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>SALES</td>
<td>Num</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>COST</td>
<td>Num</td>
<td>8</td>
<td>17</td>
</tr>
</tbody>
</table>

PROC PRINT of the sample data set:

<table>
<thead>
<tr>
<th>STATE</th>
<th>SALESDEP</th>
<th>MONTH</th>
<th>SALES</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>DONNA</td>
<td>01</td>
<td>231</td>
<td>180</td>
</tr>
<tr>
<td>CA</td>
<td>FRED</td>
<td>02</td>
<td>74</td>
<td>83</td>
</tr>
<tr>
<td>CA</td>
<td>MARY</td>
<td>01</td>
<td>156</td>
<td>148</td>
</tr>
<tr>
<td>CA</td>
<td>MARY</td>
<td>02</td>
<td>143</td>
<td>97</td>
</tr>
<tr>
<td>CA</td>
<td>KARY</td>
<td>02</td>
<td>95</td>
<td>65</td>
</tr>
<tr>
<td>NY</td>
<td>DONNA</td>
<td>01</td>
<td>210</td>
<td>155</td>
</tr>
<tr>
<td>NY</td>
<td>FRED</td>
<td>02</td>
<td>104</td>
<td>96</td>
</tr>
<tr>
<td>OK</td>
<td>FRED</td>
<td>02</td>
<td>64</td>
<td>72</td>
</tr>
<tr>
<td>OK</td>
<td>SAM</td>
<td>01</td>
<td>237</td>
<td>184</td>
</tr>
<tr>
<td>OK</td>
<td>SAM</td>
<td>01</td>
<td>44</td>
<td>72</td>
</tr>
<tr>
<td>WI</td>
<td>FRED</td>
<td>02</td>
<td>197</td>
<td>142</td>
</tr>
<tr>
<td>WI</td>
<td>JERRY</td>
<td>02</td>
<td>215</td>
<td>184</td>
</tr>
<tr>
<td>WI</td>
<td>JERRY</td>
<td>02</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>WV</td>
<td>DONNA</td>
<td>01</td>
<td>315</td>
<td>207</td>
</tr>
<tr>
<td>WV</td>
<td>DONNA</td>
<td>02</td>
<td>96</td>
<td>78</td>
</tr>
<tr>
<td>WV</td>
<td>SAM</td>
<td>02</td>
<td>174</td>
<td>169</td>
</tr>
<tr>
<td>WV</td>
<td>SAM</td>
<td>02</td>
<td>97</td>
<td>53</td>
</tr>
</tbody>
</table>

2572 2046

Case 1 - Summary by State

For our first example, we will create a summary file for each state. We need to know total sales and the count of missing values for sales in each state, as well as the highest (MAX) sales in each state.

PROC SUMMARY DATA = sugi.sales;
   CLASS state;
   VAR sales;
   OUTPUT OUT = example1
      SUM(sales) = totsales
      MAX(sales) = maxsales
      NMISS(sales) = badsales;
RUN;

PROC PRINT DATA = example1;
RUN;

<table>
<thead>
<tr>
<th>STATE</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>TOTSALES</th>
<th>MAXSALES</th>
<th>BADSALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>21</td>
<td>2572</td>
<td>315</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>CA</td>
<td>5</td>
<td>702</td>
<td>231</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>NY</td>
<td>3</td>
<td>314</td>
<td>210</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>OK</td>
<td>4</td>
<td>395</td>
<td>237</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>WI</td>
<td>5</td>
<td>479</td>
<td>215</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>WV</td>
<td>4</td>
<td>682</td>
<td>315</td>
<td>0</td>
</tr>
</tbody>
</table>

The output data set contains one observation with total statistics for the complete incoming data set, as well as one observation for each value of the CLASS variable.
The _TYPE_ variable created by SAS shows the level of the totals on that observation:

- _TYPE_ = 0 observation are the stats for the complete data set.
- _TYPE_ = 1 level are the stats for the values of STATE, or the class variable.

When more variables are present on the CLASS statement, then SAS produces even more output observations. The value of the _TYPE_ variable will increase as well. A summary observation is produced for each possible combination of values of the CLASS variables.

Case 2 - Summary by State and Salesrep

Now we will run the same PROC SUMMARY as above, changing the CLASS statement to include two variables:

```sas
PROC SUMMARY DATA= sugi.sales;
CLASS state salesrep;
VAR sales;
OUTPUT OUT=example2
SUM(sales) = totsales
MAX(sales) = maxsales 
MISS(sales) = badsales 
RUN;
PROC PRINT DATA=example2 NOOBS WIDTH=MIN HEADING=V;
TITLE "State Totals Only";
WHERE _type_ = 2;
RUN;
```

As can be seen, the output now contains values of the _TYPE_ variable that range from 0 to 3:

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>totals for complete data set</td>
</tr>
<tr>
<td>1</td>
<td>totals for each SALESREP</td>
</tr>
<tr>
<td>2</td>
<td>totals for each STATE</td>
</tr>
<tr>
<td>3</td>
<td>totals for SALESREP within STATE</td>
</tr>
</tbody>
</table>

The data set produced by PROC SUMMARY can now be used to generate a variety of reports, by using the WHERE statement to pull the required data:

```sas
PROC PRINT DATA=example2(DROP=salesrep) NOOBS WIDTH=MIN HEADING=V;
TITLE "State Totals Only";
WHERE _typ_ = 2;
RUN;
```

Case 3 - Three (or more) variables on the CLASS statement

Change the above program so the CLASS statement contains three variables:

```sas
PROC SUMMARY DATA= sugi.sales;
CLASS state salesrep month;
VAR sales;
OUTPUT OUT=example3
SUM(sales) = totsales
MAX(sales) = maxsales 
MISS(sales) = badsales 
RUN;
```

Advanced Tutorials
This program works because the data set produced by PROC SUMMARY is output in the order of the _TYPE_ values:

/* Produce data set with one observation per value */
/* of the _TYPE_ variable that is created from */
/* PROC SUMMARY earlier. */
DATA small;
SET example3;
BY _type_;
IF FIRST._type_; RUN;

PROC PRINT DATA = small;
TITLE "One per value of _TYPE_";
VAR _type_ state sales rep month;
RUN;

One per value of _TYPE_

<table>
<thead>
<tr>
<th>OBS</th>
<th><em>TYPE</em></th>
<th>STATE</th>
<th>SALESREP</th>
<th>MONTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>DONNA</td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>DONNA</td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>CA</td>
<td>DONNA</td>
<td>01</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>CA</td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>CA</td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>CA</td>
<td>DONNA</td>
<td>01</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>CA</td>
<td>DONNA</td>
<td>01</td>
</tr>
</tbody>
</table>

This report can now be used to easily see what the _TYPE_ values represent:

_TYPE_  Description
0  totals for complete data set
1  totals for each MONTH alone
2  totals for each SALESREP alone
3  totals for each combination of MONTH and SALESREP
4  totals for STATE alone
5  each MONTH and STATE combo
6  each SALESREP and STATE combo
7  final detail, totals for each combination of MONTH, SALESREP and STATE

Using the NWAY Option

Adding the keyword NWAY to the PROC statement tells SAS that the output data set should only contain the highest value of the _TYPE_ variable. So:

PROC SUMMARY DATA=sugi.sales NWAY;

will create a data set with only _TYPE_ = 7 in the above example.

Case 5 - No output Statistics Specified

If you use a CLASS statement and an OUTPUT statement but do not specify the desired statistic keywords, you will get a very different type of data set.

SAS produces five observations for each value of the CLASS variables, one observation for each of the five default statistics: N, MEAN, STD, MIN and MAX.

The name of the statistic generated is stored in the variable _STAT_ in the new data set. The following program and print show the results:

PROC SUMMARY DATA= sugi.sales;
CLASS state;
VAR sales;
OUTPUT OUT= example4;
RUN;

PROC PRINT DATA= example4;
RUN;

<table>
<thead>
<tr>
<th>OBS</th>
<th>STATE</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th><em>STAT</em></th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NY</td>
<td>5</td>
<td>N</td>
<td>18.000</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>OK</td>
<td>4</td>
<td>N</td>
<td>19.000</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>WI</td>
<td>3</td>
<td>N</td>
<td>20.000</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>WI</td>
<td>1</td>
<td>MIN</td>
<td>21.000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>WI</td>
<td>1</td>
<td>MAX</td>
<td>22.000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>WI</td>
<td>1</td>
<td>MEAN</td>
<td>23.000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>WI</td>
<td>1</td>
<td>STD</td>
<td>24.000</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>WI</td>
<td>1</td>
<td>1</td>
<td>25.000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>WI</td>
<td>1</td>
<td>2</td>
<td>26.000</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>WI</td>
<td>1</td>
<td>3</td>
<td>27.000</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>WI</td>
<td>1</td>
<td>4</td>
<td>28.000</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>WI</td>
<td>1</td>
<td>5</td>
<td>29.000</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>WI</td>
<td>1</td>
<td>6</td>
<td>30.000</td>
<td></td>
</tr>
</tbody>
</table>

This data set can be used for specific kinds of reports, where each statistic is a separate observation. A common use for this data would be a PROC PLOT, with each statistic as a separate plot.
Case 7 - Counting a Variety of Situations

Frequently we need counts of activity happening with the data, but don’t have the specific information needed as separate variables in the data set.

For example, in the sample data set, suppose we need to know the number or count of observations that had a cost/sales ratio of greater than 75%. We also need a count of SALES under $150.00.

One method would be to pass the data set through a DATA step, create new variables with either a value 1 or a missing value to signify the desired “true or false” conditions. These can then be counted with PROC SUMMARY, at the same time other statistics are generated.

```sas
DATA changed;
SET sugi.sales;
**************assign 1 to counter variable for profit percent over 75% and sales under $150**************/
profit = ROUND(cost/sales, .01); IF profit> .75 THEN over75 = 1;
IF sales < 150 THEN und150 = 1;
RUN;
PROC SUMMARY DATA= changed NWAY;
CLASS state;
VAR sales cost over75 und150;
OUTPUT OUT=example5 CDROP= FREQ ) SUM(sales) = totsales SUM(cost) = totcost N{Over75) :: over75 N{und150) ; und150
RUN;
PROC PRINT DATA=example5 NOOBS LABEL SPLIT='*' WIDTH=MIN;
TITLE "Counting Computed events in data";
LABEL over75 = "Count*Ratio*> 75"
und150 = "Count*Sales< $150"
RUN;
Counting Computed events in data

<table>
<thead>
<tr>
<th>STATE</th>
<th><em>TYPE</em></th>
<th>TOTSALES</th>
<th>TOTCOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>702</td>
<td>573</td>
<td>3</td>
</tr>
<tr>
<td>NY</td>
<td>714</td>
<td>251</td>
<td>1</td>
</tr>
<tr>
<td>OK</td>
<td>395</td>
<td>328</td>
<td>3</td>
</tr>
<tr>
<td>WI</td>
<td>479</td>
<td>387</td>
<td>2</td>
</tr>
<tr>
<td>W1</td>
<td>682</td>
<td>507</td>
<td>2</td>
</tr>
</tbody>
</table>
```

Case 8 - Using Formatted CLASS Variables

Suppose you really want to group the STATE values together into regions for the summary totals. You could create a new variable containing the region assignments in a DATA step.

A better approach is to use PROC FORMAT to assign the region codes based on STATE values, and tell PROC SUMMARY to use the formatted region assignments.

```sas
PROC FORMAT;
VALUE $regions
'WI' = 'Midwest'
'CA', 'OK' = 'West'
'NY', 'W1' = 'East'
RUN;
PROC SUMMARY DATA= sugi.sales NWAY;
CLASS state;
FORMAT state $regions.;
VAR sales cost;
OUTPUT OUT=example6
SUM(sales) = totsales SUM(cost) = totcost
RUN;
PROC PRINT DATA=example6 NOOBS WIDTH=MIN;
TITLE "Proc SUMMARY used formatted States";
RUN;
Proc Summary used formatted States

<table>
<thead>
<tr>
<th>STATE</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>TOTSALES</th>
<th>TOTCOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>West</td>
<td>1</td>
<td>9</td>
<td>1097</td>
<td>901</td>
</tr>
<tr>
<td>East</td>
<td>1</td>
<td>7</td>
<td>996</td>
<td>758</td>
</tr>
<tr>
<td>Midwest</td>
<td>1</td>
<td>5</td>
<td>479</td>
<td>387</td>
</tr>
</tbody>
</table>
```

NOTE - the data set created above has a variable STATE, that contains the unformatted value of state. The $REGIONS Format was carried with the data set and used in PROC PRINT. The value of STATE came out, as a default, in ascending order ('CA' first).

If you do a PROC PRINT with the following FORMAT assigned to STATE, you will see the original STATE value is passed to the data set:

```sas
FORMAT STATE $2.;
```

Using the ORDER= option on PROC SUMMARY

To change to order of the output observations to reflect the formatted values, the above PROC SUMMARY should be changed to request SAS use the formatted value.

```
ORDER= FREQ
```

```sas
```
values instead of the unformatted values for output
(the results are shown below):

```sas
PROC SUMMARY DATA= sugi.sales NWAY ORDER=FORMATTED;
STATE _TYPE_ _FREQ_ TOTSales TOTCOST
East 1 7 996 758
Midwest 1 5 479 387
West 1 9 1097 901
```

Remember that the variable STATE still contains the
unformatted two character state code for the first
observation in the formatted CLASS grouping.

Other values for the ORDER= option include:

- DATA in original order of the incoming data set
- EXTERNAL or FORMATTED
  As shown above, using the assigned formats
- FREQ output is in descending frequency counts
  (levels with most observations first)
- INTERNAL
  unformatted value (default)

Using the MAXID or MINID options

Suppose you want to compute the total sales by
STATE, and find out the name of the SALESREP that
had the highest and lowest sales in each STATE.

Using the MAXID and MINID options on the OUTPUT
statement, name the variable that contains the value
you want to see to identify the highest/lowest value for
that value of the CLASS variable:

```sas
PROC SUMMARY DATA= sugi.sales;
CLASS state;
VAR sales;
OUTPUT WT=example7
  SUM(sales) = totsales
  MAXID(sales(salesrep)) = best
  MINID(sales(salesrep)) = worst
RUN;
```

NOTE - it is important to understand that the above
shows which salesrep had the highest/lowest individual
observation in the data set, not the highest/lowest total
sales. In 'CA', Mary had total sales higher than Donna,
but Donna had the single highest sales amount in that
state.

Missing Values were ignored in the statistics,
including the selection of the lowest sales by salesrep.

CONCLUSION

PROC SUMMARY is one of my favorite tools to reduce
a large set of data to meaningful and manageable
totals. With some creative use of new numeric
variables, formats and options in the procedure, this
tool can perform very complex summarization tasks.

I suggest reviewing the SAS documentation for further
use of this PROC, specifically the more statistically
oriented features.

SAS is a registered trademark of SAS Institute Inc.

References

SAS Institute Inc., SAS Procedures Guide, Version 6,

Author

The author will be glad to answer questions and accept
suggestions at the following address:

David Beam
Systems Seminar Consultants
1220 Femrite Drive
Suite 206
Madison, WI 53716
voice: (608) 222-7081
Internet: train@sys-seminar.com

Best/Worst Salesreps in Each State
and in Total U.S.

<table>
<thead>
<tr>
<th>STATE</th>
<th><em>TYPE</em></th>
<th><em>FREQ</em></th>
<th>TOTSales</th>
<th>BEST</th>
<th>WORST</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>1</td>
<td>5</td>
<td>702</td>
<td>DONNA</td>
<td>JERRY</td>
</tr>
<tr>
<td>NY</td>
<td>1</td>
<td>3</td>
<td>314</td>
<td>FRED</td>
<td>SAM</td>
</tr>
<tr>
<td>OK</td>
<td>1</td>
<td>4</td>
<td>395</td>
<td>SAM</td>
<td>FRED</td>
</tr>
<tr>
<td>WI</td>
<td>1</td>
<td>5</td>
<td>479</td>
<td>JERRY</td>
<td>JERRY</td>
</tr>
<tr>
<td>WV</td>
<td>1</td>
<td>4</td>
<td>682</td>
<td>DONNA</td>
<td>DONNA</td>
</tr>
</tbody>
</table>

NOTE - it is important to understand that the above
shows which salesrep had the highest/lowest individual
observation in the data set, not the highest/lowest total
sales. In 'CA', Mary had total sales higher than Donna,
but Donna had the single highest sales amount in that
state.

Missing Values were ignored in the statistics,
including the selection of the lowest sales by salesrep.