FF07 - 2010

The MEANS/SUMMARY Procedure: Getting Started

Arthur L. Carpenter
California Occidental Consultants, Anchorage, AK

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.

```
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.

A Simple Printed Table

The MEANS Procedure

Analysis Variable : Weight

N	Mean	Std Dev	Minimum	Maximum
19	100.0263158	22.7739335	50.5000000	150.0000000

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.

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

The First Two Statistical Moments

The MEANS Procedure

Analysis Variable : Weight

N	Mean	Variance	Std Dev	Std Error
19	100.0263158	518.6520468	22.7739335	5.2246987

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 50^{th} percentile
- p50 50^{th} percentile (or second quartile)
- p25 | q1 25^{th} percentile (or first quartile)
- p75 | q3 75^{th} 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.

```
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:

A Simple Summary Data Set

Obs	_TYPE_	_FREQ_	_STAT_	Weight
1	0	19	N	19.000
2	0	19	MIN	50.500
3	0	19	MAX	150.000
4	0	19	MEAN	100.026
5	0	19	STD	22.774

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.

Selecting Multiple Statistics

Obs	_TYPE_	_FREQ_	number	average	std_ deviation
1	0	19	19	100.026	22.7739

```
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;
```

Selective Associations

Obs	_TYPE_	_FREQ_	ht_n	wt_n	wt_mean	ht_std
1	0	19	19	19	100.026	5.12708

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=
       std = / autoname;
run;
```

Using AUTONAME

Obs	_TYPE_	_FREQ_	Height_N	Weight_N	Height_ Mean	Weight_ Mean	Height_ StdDev	Weight_ StdDev
1	0	19	19	19	62.3368	100.026	5.12708	22.7739

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.

```
title1 'CLASS and a Printed
Table';
proc means
data=sashelp.class(where=(age
in(12,13,14)))
            n mean std;
class age sex;
var height;
run;
```

CLASS and a Printed Table

The MEANS Procedure

Analysis Variable : Height

Age	Sex	N Obs	N	Mean	Std Dev
12	F	2	2	58.0500000	2.4748737
	M	3	3	60.3666667	3.9323445
13	F	2	2	60.9000000	6.2225397
	M	1	1	62.5000000	.
14	F	2	2	63.5500000	1.0606602
	M	2	2	66.2500000	3.8890873

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.

```
title1 'CLASS and a Summary Data Set';
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:

CLASS and a Summary Data Set

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	3.93234
9	13	F	3	2	2	60.9000	6.22254
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	3.88909

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.

```
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.

```
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;
```

Understanding _TYPE_

Obs	RACE	EDU	SYMP	_TYPE_	_FREQ_	mean HT
1		.		0	8	66.25
2		.	01	1	2	64.00
3		.	02	1	4	66.50
4		.	03	1	2	68.00
5		12		2	4	67.50
6		14		2	2	64.00
7		15		2	2	66.00
8		12	02	3	2	67.00
9		12	03	3	2	68.00
10		14	01	3	2	64.00
11		15	02	3	2	66.00
12	1	.		4	6	67.00
13	4	.		4	2	64.00
14	1	.	02	5	4	66.50
15	1	.	03	5	2	68.00
16	4	.	01	5	2	64.00
17	1	12		6	4	67.50
18	1	15		6	2	66.00
19	4	14		6	2	64.00
20	1	12	02	7	2	67.00
21	1	12	03	7	2	68.00
22	1	15	02	7	2	66.00
23	4	14	01	7	2	64.00

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_.

	CLASS VARIABLES				
Observations	**RACE**	**EDU**	**SYMP**	**Binary Value**	**_TYPE_**
1	0	0	0	000	0
2 - 4	0	0	1	001	1
5 - 7	0	1	0	010	2
8 - 11	0	1	1	011	3
12 - 13	1	0	0	100	4
14 - 16	1	0	1	101	5
17 - 19	1	1	0	110	6
20 - 23	1	1	1	111	7
	$2^2=4$	$2^1=2$	$2^0=1$		

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

Obs	RACE	EDU	SYMP	_TYPE_	_FREQ_	mean HT
1		.		000	8	66.25
2		.	01	001	2	64.00
3		.	02	001	4	66.50
4		.	03	001	2	68.00
5		12		010	4	67.50
6		14		010	2	64.00
7		15		010	2	66.00
8		12	02	011	2	67.00
9		12	03	011	2	68.00

. . . . 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:

```
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;

data edufactor;
   set stats;
   where _type_ in(2,3,6,7);
   run;
```

Better yet why not avoid the DATA step and build the subset directly when the data are created.

```
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=edufactor(where=(_type_ in(2,3,6,7)))
          mean= meanHT;
   run;
```

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