Paper CT-14

A PROC MEANS Primer

David Kerman, Bank of America, Charlotte, NC

ABSTRACT

A PROC MEANS Primer gives an introduction to the PROC MEANS procedure (included in Base SAS), describing the syntax and key options, and providing examples on how and when to use this procedure. Special focus is given to a couple of important options, NWAY and COMPLETETYPES, which are very powerful but can lead to confusion and errors if not used properly. The paper concludes by giving an example of these types of errors, and provides recommendations on how to avoid the errors when using PROC MEANS.

WHAT IS PROC MEANS?

PROC MEANS is a procedure (included in Base SAS) that provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations. PROC MEANS can be used to calculate descriptive statistics based on moments, estimate quantiles (like median or 25th percentile, for example), calculate confidence limits for the mean, identify extreme values, and perform a t test.

PROC MEANS is similar to PROC SUMMARY, with a main difference being that PROC MEANS produces display output, where PROC SUMMARY does not.

WAYS TO USE PROC MEANS

PROC MEANS can be used to calculate various statistics (sum, mean, standard deviation, percentiles) on data grouped by categories. It is also often used to prepare data for export into standard reports, or as a basis for Excel pivots. The basic syntax of PROC MEANS is as follows:

```
PROC MEANS DATA=score_data NOPRINT;
    CLASS state city;
    VAR score;
    OUTPUT OUT=scor_stats SUM=;
    RUN;
```

The CLASS statement is used to list the categories that you want to count by, and the VAR statement is used for the variable that you are counting/calculating (the variable that you want to calculate the sum, mean or median of, for example). As noted above, PROC MEANS will normally produce display output – to inhibit this display we are using the NOPRINT option here. In Table 1 below we see a sample of the score_data data set. Note that the CLASS statement in the syntax shown above is used for the categorical variables (City and State), while the VAR statement is used for the numerical variable (Score).

emp_id	City	State	Score
1	Berkeley	CA	51
2	Berkeley	CA	98
3	Berkeley	CA	94
4	Berkeley	CA	64
5	Berkeley	CA	2
6	Fresno	CA	72
7	Fresno	CA	49
8	Atlanta	GA	35
9	Atlanta	GA	74

Table 1. Sample of Score_data Data set

When the PROC MEANS procedure (shown above) runs, the resulting data set produced (scor_stats) contains statistics summarizing score by city and state. A sample of the scor_stat data set is shown in Table 2.

State	City	_TYPE_	_FREQ_	Score
		0	58	3520
	Annandale	1	3	226
	Durham	1	6	318
	Richmond	1	5	339
	Yellow Springs	1	4	263
CA		2	7	430
GA		2	6	375
OH		2	10	726
VA		2	8	565
CA	Berkeley	3	5	309
CA	Fresno	3	2	121
GA	Atlanta	3	6	375
MA	Boston	3	4	249

Table 2. Sample of Score_stats Data set

In addition to the state, city and score variables, the output data set also contains two SAS-generated variables, _TYPE_ and _FREQ_. The _TYPE_ variable denotes the level of summarization used, starting at zero and ending at a value equal to (2**n)-1, where n equals the number of categorical variables in the class statement. In this example, _TYPE_ = 0 is used for the overall summary, _TYPE_=1 is summarized by city, _TYPE_=2 is summarized by state, and _TYPE_=3 denotes the state by city summary.

If you are only interested in the highest level summary, you can use the NWAY option. This option will limit the output data set to only the highest value of _TYPE_ - in this example, where _TYPE_=3.

The syntax becomes:

```
PROC MEANS DATA=score_data NOPRINT NWAY;
    CLASS state city;
    VAR score;
    OUTPUT OUT=scor_stats SUM=;
    RUN;
```

Note the output in Table 3 is now limited to the state by city summary

State	City	_TYPE_	_FREQ_	Score
CA	Berkeley	3	5	309
GA	Atlanta	3	6	375
MA	Boston	3	4	249
MN	Duluth	3	2	73
MN	Minneapolis	3	6	191
NC	Charlotte	3	4	324
NC	Durham	3	6	318

Table 3. Sample of Score_stats Data set (using NWAY option)

In the examples shown so far, the PROC MEANS procedure has been used to calculate the sum of the score variable, and by using the syntax SUM= , the value of the sum is listed with the variable name (Score) as the label.

For the next example, PROC MEANS is used to generate the mean, median, and percentiles.

Syntax:

```
PROC MEANS DATA=score_data NOPRINT NWAY;
    CLASS state city;
    VAR score;
    OUTPUT OUT=scor_stats_p (drop=_TYPE_ _FREQ_)
    MEAN=mean MEDIAN=median P25=Q1 P75=Q3;
    RUN;
```

There are two types of modifications to the syntax above (see **bold type**). First, we use the drop= option to omit the unneeded _TYPE_ and _FREQ_ variables. Also, the sum= code has been replaced with four different statistics. For each of these (MEAN, MEDIAN, P25 and P75) we have provided labels (mean, median, Q1, Q3). A sample of the resulting output data set is shown in Table 4.

State	City	mean	median	Q1	Q3
CA	Berkeley	61.8	64	51	94
CA	Fresno	60.5	60.5	49	72
GA	Atlanta	62.5	73	35	84
MA	Boston	62.25	61.5	36	88.5
MN	Duluth	36.5	36.5	25	48
MN	Minneapolis	31.8333	30.5	17	51
NC	Charlotte	81	86	68.5	93.5
NC	Durham	53	57.5	16	84
NC	Raleigh	53.8	51	38	67
OH	Cleveland	77.1667	77.5	68	95
VA	Annandale	75.3333	75	61	90
VA	Richmond	67.8	73	64	83

Table 4. Sample of Score_stats_p Data set

In this output data set, instead of seeing the sum of score for each of the state by city combinations, we have used PROC MEANS to generate some different statistics (mean, median, 25th percentile and 75th percentile). The statistics are shown with the labels indicated in the syntax above (i.e. P25=Q1).

COMPLETETYPES OPTION

If you are interested in producing statistics using all possible permutations of the class variables, you will want to consider the COMPLETETYPES option. In the syntax below, we have added this option to the code from the prior example:

```
PROC MEANS DATA=score_data NOPRINT NWAY COMPLETETYPES;
    CLASS state city;
    VAR score;
    OUTPUT OUT=scor_stats_p (drop=_TYPE_ _FREQ_)
    MEAN=mean MEDIAN=median P25=Q1 P75=Q3;
    RUN;
```

A sample of the resulting data set displays some interesting results (Table 5).

State	City	mean	median	Q1	Q3
CA	Atlanta	.	.	.	.
CA	Berkeley	61.8	64	51	94
GA	Atlanta	62.5	73	35	84
MN	Cleveland	.	.	.	.

Table 5. Sample of Score_stats_p Data set with COMPLETETYPES option

You are probably wondering, “Why would I want to use the COMPLETETYPES option?” In our example, a lot of the resulting data records are meaningless. For example, there is no Atlanta, CA or Cleveland, MN in our data, so the statistical values are all missing.

Let’s consider a different example. In the data set samples shown below, we have monthly data on new hires by state and category. We are looking to set up a reporting program where we can export from SAS to drop data into an existing excel report which uses cell reference links to populate the report.

State	City	Category	new_hires
CA	Modesto	Strategy	2
CA	Modesto	Operations	8
CA	San Jose	Strategy	3
CA	San Jose	Policy	9
GA	Atlanta	Analysis	4
GA	Atlanta	Policy	6
GA	Atlanta	Strategy	11

Table 6. Sample of march_new_hires data set

State	City	Category	new_hires
CA	Modesto	Operations	6
CA	Modesto	Policy	11
CA	San Jose	Policy	1
CA	San Jose	Strategy	8
GA	Atlanta	Analysis	5
GA	Atlanta	Strategy	17
GA	Augusta	Strategy	10

Table 7. Sample of april_new_hires data set

For the march_new_hires data set, we will summarize by state and category. Our PROC MEANS syntax will be very similar to what we have shown previously, and the resulting march_sum data set is shown in Table 8 below:

```
PROC MEANS DATA=march_new_hires NOPRINT NWAY;
    CLASS state category;
    VAR new_hires;
    OUTPUT OUT=march_sum (drop=_TYPE_ _FREQ_) SUM=;
    RUN;
```

state	category	new_hires
CA	Analysis	1
CA	Operations	10
CA	Policy	9
CA	Strategy	8
GA	Analysis	4
GA	Operations	1
GA	Policy	7
GA	Strategy	15
NY	Analysis	2
NY	Operations	6
NY	Policy	4
NY	Strategy	8
VA	Analysis	3
VA	Operations	11
VA	Policy	4
VA	Strategy	5

Table 8. march_sum data set

Note that the city variable is not included on the class statement, so the summary is just at the state and category level. As you might expect, since our data contains four state values and four category values, the resulting march_sum data set contains 16 records.

Let’s review the output data set april_sum (Table 9), produced when we run the same PROC MEANS code, substituting the april_new_hires data set

state	category	new_hires
CA	Operations	8
CA	Policy	12
CA	Strategy	13
GA	Analysis	10
GA	Operations	3
GA	Policy	11
GA	Strategy	27
NY	Analysis	11
NY	Policy	5
NY	Strategy	12
VA	Analysis	13
VA	Operations	7
VA	Policy	10
VA	Strategy	7

Table 9. april_sum data set

At first glance, the april_sum data set (Table 9) looks very similar to the march_sum data set (Table 8), but a careful review reveals an important difference. While we noted 16 records in march_sum, the april_sum data set only contains 14. Let’s see what changes when we add the COMPLETETYPES option to our code.

```
PROC MEANS DATA=april_new_hires NOPRINT NWAY COMPLETETYPES;
    CLASS state category;
    VAR new_hires;
    OUTPUT OUT=april_sum_cpt (drop=_TYPE_ _FREQ_) SUM=;
    RUN;
```

state	category	new_hires
CA	Analysis	.
CA	Operations	8
CA	Policy	12
CA	Strategy	13
GA	Analysis	10
GA	Operations	3
GA	Policy	11
GA	Strategy	27
NY	Analysis	11
NY	Operations	.
NY	Policy	5
NY	Strategy	12
VA	Analysis	13
VA	Operations	7
VA	Policy	10
VA	Strategy	7

Table 10. april_sum_cpt data set

As highlighted by the arrows above, we now have a data set with 16 rows, including missing new hire values for CA Analysis and NY Operations. For the month of April, our CA sites did not add any analysis staff, and our NY sites did not hire any new operations personnel. If we were using excel for monthly reporting, this april_sum_cpt data set could be pasted directly into excel without cell reference errors (since the number of rows will now match with 16).

The COMPLETETYPES option is not appropriate for all situations, but it is a valuable PROC MEANS option to keep in your SAS toolkit.

A PUZZLER

Using the same april_new_hires data set from the last example, we run two different PROC MEANS steps – one using state as the only class variable, and the other using category as the only class variable (see below).

```
PROC MEANS DATA=april_new_hires NOPRINT;
    CLASS state;
    VAR new_hires;
    OUTPUT OUT=april_st (drop=_FREQ_) SUM=;
    RUN;


PROC MEANS DATA=april_new_hires NOPRINT;
    CLASS category;
    VAR new_hires;
    OUTPUT OUT=april_cat (drop=_FREQ_) SUM=;
    RUN;
```

Samples of the output data sets april_st and april_cat are shown below (Tables 11 and 12):

state	_TYPE_	new_hires
	0	150
CA	1	33
GA	1	51
NY	1	29
VA	1	37

Table 11. Sample of april_st data set

category	_TYPE_	new_hires
	0	149
Analysis	1	34
Operations	1	18
Policy	1	38
Strategy	1	59

Table 12. Sample of april_cat data set

In the two data sets above, we see that summing by state produces 150 new hires, but summing by category produces 149. What is happening here?

This sample of the actual april_new_hires data set will shed some light on the mystery:

state	City	category	new_hires
GA	Macon	Operations	2
GA	Macon	Policy	7
NY	Brooklyn	Strategy	6
NY	Brooklyn	Analysis	10
NY	Brooklyn		1
NY	Rochester	Analysis	1
NY	Rochester	Policy	5

Table 13. Sample of april_new_hires data set

A record from the april_new_hires data set contains state, city and new_hires but is blank for category. In this case, using category under the class statement misses this record and the numeric value of new_hires (1, in this example) is bypassed in the sum.

In order to prevent the confusion and potential errors from the last example, there a few steps you can take to improve your results when using PROC MEANS.

1. Prior to running the code, review the input data and contact the data provider to fill in the blank category for the Brooklyn new hire in April. This may not be a realistic option.
2. Prior to running PROC MEANS, run a datastep like this:

```
DATA april_new_hires;
    SET april_new_hires;
    if category=’’ then category=’missing’;
    RUN;
```

3. Add the “missing” option on the CLASS statement line as shown here:

```
PROC MEANS DATA=april_new_hires NOPRINT;
    CLASS category / missing;
    VAR new_hires;
    OUTPUT OUT=april_cat_m (drop=_FREQ_) SUM=;
    RUN;
```

A sample of the output data set april_cat_m is shown in Table 14.

category	_TYPE_	new_hires
	0	150
	1	1
Analysis	1	34
Operations	1	18
Policy	1	38
Strategy	1	59

Table 14. Sample of april_cat_m data set

By using the “missing” option on the CLASS statement, a blank category row is shown with 1 new hire listed, bringing the overall new_hires total up to 150.

CONCLUSION

Over the course of my career using SAS, I have found PROC MEANS to be one of my most important tools for data summarization and statistical analysis. In this brief paper, I have detailed the basic function and syntax, and highlighted a couple of options (NWAY, COMPLETETYPES) that can be very helpful to beginning or experienced SAS users. I have really only scratched the surface of all that PROC MEANS can do for you as a SAS programmer.

RECOMMENDED READING

- Base SAS® Procedures Guide

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David Kerman
Bank of America
NC1-028-12-01
150 N. College St.
Charlotte, NC 28255
980-386-2992
david.kerman@bankofamerica.com