Paper 240-29

Steps to Success with PROC MEANS

Andrew H. Karp
SIERRA INFORMATION SERVICES, INC.
SONOMA, CALIFORNIA USA

Introduction
One of the most powerful and flexible Procedures you'll find in the SAS System is PROC MEANS. You can use it to rapidly and efficiently analyze the values of numeric variables and place those analyses either in the Output Window or in a SAS Data Set (or both). Mastering the basic syntax and features of this procedure will enable you to easily create many of the analyses you need from your data sets. Taking the time to master advanced PROC MEANS features, many of which were added (or enhanced) in Version 8 of the SAS System, will further expand your ability to create powerful and effective analyses of your data. This paper presents a series of "Steps to Success" that show you some of the core features of PROC MEANS and how you can apply them to the observations/variables in a SAS data set. Starting from the basics, and moving through more advanced capabilities in PROC MEANS, the "Steps to Success" presented here offer an in-depth understanding of many tools in this BASE SAS Procedure. Even more tools and capabilities are available than can be presented here; so, after reading this paper, you should take the time to look at the detailed PROC MEANS documentation in the BASE SAS Procedures Guide to learn about even more features, options and capabilities available to you.

Overview
PROC MEANS is used in a variety of analytic, business intelligence, reporting and data management situations. Data warehousing experts may use it during the ETL (Extract-Transform-Load) process to create "lightly summarized" data sets from very large, transaction-level data sets. These 'lightly summarized" data sets are then stored in "data warehouses" or "data marts" for use in other analytic and reporting tasks. On the other hand, business analysts and programmers may use its capabilities to generate analyses of data, and group those analyses "by" the values of specified in the "by" or "classification" variables placed in the PROC MEANS task. These analyses may be sent to the Output Window, "delivered" to other Output Delivery System (ODS) destinations (such as a PDF or HTML file) , or stored in SAS data sets which are then used by other SAS Procedures or exported to CSV or spreadsheet files. Lastly, PROC MEANS capabilities may be employed in "data cleansing" or "exploratory data analysis" tasks to determine if incorrect or "bad" values of analysis variables are contained in the data set that must be transformed or removed prior to further analysis.

Key Terms and Concepts
PROC MEANS is included the BASE Module of SAS System Software. When using it, the term "analysis variable" refers to the numeric variable (or variables) whose values you want to have the procedure analyze. "Classification" or "By" variables are those (numeric or character) variables whose values will be used to "classify" the analyses of the values of the analysis variables. For example, if you want to *analyze* the numeric variable SALARY *classified by* GENDER, the *analysis variable* is SALARY and the *classification variable* is GENDER. The analyses you can perform using PROC MEANS are referred to as "statistics" or "analyses" and are implemented by specifying the "statistics keyword" for the desired analysis within the PROC MEANS "task" or "unit of work." So, if you wanted the MEAN and MEDIAN values of SALARY classified by GENDER, the MEAN and MEDIAN *statistics keywords* would be specified in your PROC MEANS task. (A full list of statistics keywords and their proper placement in the PROC MEANS syntax is discussed later in this paper.) To round out the terms used in this paper, "input data set" refers to the "source" data set on whose observations and variables will be used by PROC MEANS and "output data set" describes a SAS data set created by PROC MEANS which "stores" or "holds" the desired analyses.

Data Sets and SAS Software Version Used in the Examples
Two SAS data sets are used to generate the examples you'll see in this tutorial. An Early Adopter Release of SAS 9 Software was used to create the code and output, but *everything* presented in this paper is available in Release 8.0 and higher of the SAS System.

The first data set, ELEC_ANNUAL, contains about 16,300 customer-level observations (rows) with information about how much electricity they consumed in a year, the rate schedule on which they were billed for the electricity, the total revenue billed for that energy and the geographic region in which they live. The variables in the data set are:

• PREMISE	Premise Number	[Unique identifier for customer meter]
• TOTKWH	Total Kilowatt Hours	[KwH is the basic unit of electricity consumption]
• TOTREV	Total Revenue	[Amount billed for the KwH consumed]

• TOTHRS	Total Hours	[Total Hours Service in Calendar Year]
• RATE_SCHEDULE	Rate Schedule	[Table of Rates for Electric Consumption Usage]
• REGION	Geographic Region	[Area in which customer lives]

The second data set, CARD_TRANS2, contains about 1.35 million observations (rows), each representing one (simulated) credit card transaction. The variables in the data set are:

• CARDNUMBER	Credit Card Number	
• CARDTYPE	Credit Card Type	[Visa, MasterCard, etc.]
• CHARGE_AMOUNT	Transaction Amount (in dollars/cents)	
• TRANS_DATE	Transaction Date	[SAS Date Variable]
• TRANS_TYPE	Transaction Type	[1=Electronic 2=Manual]

Step 1: Basics and Defaults
By default, PROC MEANS will analyze all numeric variables in your data set and "deliver' those analyses to your Output Window. Five default statistical measures are calculated:

• **N**	Number of observations with a non-missing value of the analysis variable
• **MEAN**	Mean (Average) of the analysis variable's non-missing values
• **STD**	Standard Deviation
• **MAX**	Largest (Maximum) Value
• **MIN**	Smallest (Minimum) Value

Using the ELEC_ANNUAL Data Set and PROC MEANS, we can see how the default actions of PROC MEANS are carried out by submitting the following code:

```
* Step 1: Basics and Defaults;
  PROC MEANS DATA=SUGI.ELEC_ANNUAL;
  title 'SUGI 29 in Montreal';
  title2 'Steps to Success with PROC MEANS';
  title3 'Step 1: The Basics and Defaults';
  run;
```

The results displayed in the Output Window are:

```
SUGI 29 in Montreal
Steps to Success with PROC MEANS
Step 1: The Basics and Defaults

The MEANS Procedure

Variable          N            Mean         Std Dev         Minimum         Maximum
------------------------------------------------------------------------------------

totkwh        16329         6237.48         8963.74               0       361920.00
totrev        16382     753.8247088         1046.65       1.6500000        40665.50
tothrs        16378         8648.72     610.7884985     240.0000000         9120.00
------------------------------------------------------------------------------------
```

Since TOTKWH, TOTREV and TOTHRS are all numeric variables, PROC MEANS calculated the five default statistical measures on them and placed the results in the Output Window.

Step 2: Taking Control: Selecting Analysis Variables, Analyses to be Performed by PROC MEANS , and Rounding of Results

In most situations, your data sets will probably have many more numeric variables you want PROC MEANS to analyze. This particularly true if some of your numeric variables don't "admit" of a "meaningful arithmetic operation," which is a fancy way of saying that the results of calculating a statistic on them results in meaningless "information." For example, the sum of ZIPCODE or the MEAN of telephone number is unlikely to be useful. So, we don't want to waste time having these values calculated or clutter up our output with meaningless "information." Also, we may not need all of the five statistical analyses that PROC MEANS will perform automatically. And, we may want to round the values to a more useful number of decimal places than what PROC MEANS will do for us automatically.

Again using the ELEC_ANNUAL data set, here is how we can take more control over what PROC MEANS will do for us. Suppose we just want the SUM and MEAN of TOTREV, rounded to two decimal places. The following PROC MEANS task gets us just what we want.

```
* Step 2: Taking Control;
  PROC MEANS DATA=SUGI.ELEC_ANNUAL MEAN SUM MAXDEC=2;
  VAR TOTREV;
  Title3 'Step 2: Taking Control';
  run;
```

A box has been drawn around the important features presented iin Step 2. First, the SUM and MEAN *statistics keywords* were specified, which instructs PROC MEANS to just perform those analyses. Second, the MAXDEC option was used to round the results in the Output Window to just two decimal places. (If we had wanted the analyses rounded to the nearest whole number, then MAXDEC = 0 would have been specified.) Finally, the VAR Statement was added, giving the name of the variable for which the analyses were desired. You can put as many (numeric) variables as you need/want in to one VAR Statement in your PROC MEANS task.

The Output Window displays:

```
SUGI 29 in Montreal
Steps to Success with PROC MEANS
Step 2: Taking Control

The MEANS Procedure

 Analysis Variable : totrev

        Mean              Sum
____________________________

      753.82      12349156.38
____________________________
```

Step 3: Selecting Other Analyses

So far we've worked some of the (five) default statistical analyses available from PROC MEANS. There are many other statistical analyses you can obtain from the procedure! Here is a complete list:

Descriptive Statistics Keywords		***Quantile Statistlcs Keywords***	
CLM	**RANGE**	**MEDIAN|P50**	**Q3|P75**
CSS	**SKEWNESS|SKEW**	**P1**	**P90**
CV	**STDDEV|STD**	**P5**	**P95**
KURTOSIS|KURT	**STDERR**	**P10**	**P99**
LCLM	**SUM**	**Q1|P25**	**QRANGE**
MAX	**SUMWGT**	***Hypothesis testing keywords***	
MEAN	**UCLM**	**PROBT**	**T**
MIN	**USS**		
N	**VAR**		
NMISS	**---**		

Suppose the observations in ELEC_DATA are a random sample from a larger population of utility customers. We might therefore want to obtain, say, a 95 percent confidence interval around the mean total KwH consumption and around the mean billed revenue, along with the mean and median. From the above table, you can see that the MEAN, MEDIAN and CLM statistics keywords will generate the desired analyses. The PROC MEANS task below generates the desired analyses. The task also includes a LABEL Statement, which add additional information about the variables in the Output Window.

```
* Step 3: Selecting Statistics;
  PROC MEANS DATA=SUGI.ELEC_ANNUAL
                        MEDIAN MEAN CLM MAXDEC=0;
  Label TOTREV = 'Total Billed Revenue'
        TOTKWH = 'Total KwH Consumption';
  VAR TOTREV TOTKWH;
  title3 'Step 3: Selecting Statistics';
  run;
```

The output generated is:

```
SUGI 29 in Montreal
Steps to Success with PROC MEANS
Step 3: Selecting Statistics

The MEANS Procedure

                                                              Lower 95%      Upper 95%
Variable   Label                       Median          Mean   CL for Mean    CL for Mean
ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
totrev     Total Billed Revenue           606           754           738            770
totkwh     Total KwH Consumption         5082          6237          6100           6375
ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
```

Step 4: Analysis with CLASS (variables)

So far we've analyzed the values of variables from ELEC_ANNUAL without regard to the values of potentially interesting and useful *classification variables.* PROC MEANS can do this for you with a minimum of additional coding. First, we need to understand what the CLASS and BY Statements "do" when included in a PROC MEANS task. The CLASS statement does not require that the input (source) data set be sorted by the values of the classification variables. On the other hand, using the BY Statement requires that the input data set be sorted by the values of the classification variables.

In most situations, it does not matter if you use the CLASS or BY statement to request analyses classified by the values of a classification variable. If you are working with a very large file, however, with many classification variables (and/or classification variables with many distinct values), you may obtain significant processing time reductions if you first use PROC SORT to sort the data by the values of the classification variable and then use PROC MEANS with a BY Statement. Unfortunately, I cannot give you a "magic number" of observations or variables at which it become more efficient to first sort and then use a BY statement versus using the CLASS statement on a unsorted data set. Factors such as the actual number of observations, the number of unique values of the CLASS variables, memory allocation/availability, CPU power, etc. all come in to play and can't really be estimated in advance. You'll have to use some trial and error to figure out which approach is best for your unique data structures and computing capabilities.

Having said all of this, let's take a look at how we can obtain the MEAN and SUM of TOTREV classified by REGION in the ELEC_ANNUAL data set. All we need to do is add the CLASS statement (with REGION as the classification variable) to the PROC MEANS task, as shown below.

```
* Step 4: Analysis with CLASS (Variables);
   PROC MEANS DATA=SUGI.ELEC_ANNUAL
                                SUM MEAN MAXDEC=0 ;
  CLASS REGION;
  VAR TOTREV TOTKWH;
  title3 'Step 4: Analysis with CLASS (Variables)';
  run;
```

The Output Window displays:

```
SUGI 29 in Montreal
Steps to Success with PROC MEANS
Step 4: Analysis with CLASS (Variables)

The MEANS Procedure

REGION           N Obs    Variable             Sum           Mean
-----------------------------------------------------------------
EASTERN           5100    totrev           3315024            674
                          totkwh          28053594           5517

NORTHERN          5447    totrev           4383227            834
                          totkwh          37582600           6917

SOUTHERN           718    totrev            557616            810
                          totkwh           4873954           6788

WESTERN           5061    totrev           3324052            680
                          totkwh          27949333           5549
-----------------------------------------------------------------
```

By specifying REGION in the CLASS Statement, we now have the MEAN and SUM of TOTREV and TOTKWH for each unique value of region. We also have a column called "N Obs," which is worthy of further discussion. By default, PROC MEANS shows the number of observations for each value of the classification variable. So, we can see that there are, for example, 5,061 observations in the data set from the WESTERN Region.

How does PROC MEANS handle missing values of classification variables? Suppose there were some observations in ELEC_ANNUAL with missing values for REGION. By default, those observations *would not be included in the analyses* generated by PROC MEANS…but, we have an option in PROC MEANS that we can use to include observations with missing values of the classification variables in our analysis. This option is shown in Step 5.

Step 5: Don't Miss the Missings!

As we saw in Step 4, PROC MEANS automatically creates a column called "N Obs" when a classification variable is placed in a CLASS or BY Statement. But, observations with a missing value are, by default, excluded (not portrayed) in the output analysis. There are certainly many instances where it would be useful to know: a) how many observations have a missing value for the classification variable and b) what the analyses of the analysis variables are for observations that have a missing value for the given classification variable. We can easily obtain this information by specifying the MISSING option in the PROC MEANS statement. Here's how to do it:

```
* Step 5: Don't Miss the Missings;
  PROC MEANS DATA=SUGI.ELEC_ANNUAL SUM MEAN MAXDEC=0 MISSING;
  CLASS REGION;
  VAR TOTREV TOTKWH;
  title3 "Step 5: Don't Miss the Missings!";
  run;
```

The output this task created is:

```
SUGI 29 in Montreal
Steps to Success with PROC MEANS
Step 5: Don't Miss the Missings!

The MEANS Procedure

REGION            N Obs    Variable             Sum             Mean
--------------------------------------------------------------------
                     38    totrev             55671             1505
                           totkwh            450702            11861

EASTERN            5100    totrev           3315024              674
                           totkwh          28053594             5517

NORTHERN           5447    totrev           4383227              834
                           totkwh          37582600             6917

SOUTHERN            718    totrev            557616              810
                           totkwh           4873954             6788

WESTERN            5061    totrev           3324052              680
                           totkwh          27949333             5549
--------------------------------------------------------------------
```

Now it's clear that our data set has some (38, to be exact) observations with a missing value of the classification variable REGION. Maybe we need to examine our data more carefully to find out why some observations don't have a valid value for REGION?

Step 6: Don't Miss the Missings (Part II)

In Step 5 we used the MISSING option to see how many *observations* had a missing value for the classification variable (REGION). In this Step, we'll look at the NMISS and N *statistics keywords* to assess how many observations have a missing (or non-missing) value of the *analysis variables.* This is often an important exploratory/data cleaning step in an analytic project…after all, the presence of missing values of an analysis variable can point to errors in past programming steps, inaccurate data collection methods, or other problems that can have negative impacts on your analyses. Let's take a look at what the NMISS and N *statistics keywords* can do for us.

```
* Step 6: Don't Miss the Missings (PART II);
  PROC MEANS DATA=SUGI.ELEC_ANNUAL
                              MAXDEC=0 MISSING N NMISS;
  CLASS REGION;
  VAR TOTREV TOTKWH;
  title3 "Step 6: Don't Miss the Missings! (Part II)";
  run;
```

The output is:

Continued on Next Page…

```
SUGI 29 in Montreal
Steps to Success with PROC MEANS
Step 6: Don't Miss the Missings! (Part II)

The MEANS Procedure
                                                      N
REGION           N Obs    Variable         N       Miss
_______________________________________________________

                    38    totrev          37          1
                          totkwh          38          0

EASTERN           5100    totrev        4921        179
                          totkwh        5085         15

NORTHERN          5447    totrev        5257        190
                          totkwh        5433         14

SOUTHERN           718    totrev         688         30
                          totkwh         718          0

WESTERN           5061    totrev        4889        172
                          totkwh        5037         24
_______________________________________________________
```

The output from Step 6 shows the interrelationship between the “N Obs” Column and the N and NMISS Statistics keywords. For example, look at the Eastern Region. There are 5,100 observations with a value of REGION equal to “EASTERN.” Of those, 4,921 have a non-missing value of TOTREV and 179 have a missing value.

Step 7: Take What You Need and Leave the Rest.

This Step shows how your can apply the WHERE Clause Data Step Option to limit PROC MEANS’ “work” to just the observations in your data set you’re really interested in analyzing without first having to create a subset SAS data set. By using the WHERE Clause as a SAS Data Set Option, you can tell PROC MEANS which observations you want to analyze and which ones you, in effect, want the procedure to “ignore” while creating the analyses you want.

In addition, this Step shows you what happens when you specify two variables in a CLASS statement. What we want to do is calculate the mean total kilowatt hours classified by REGION and RATE_SCHEDULE, but only for the WESTERN and SOUTHERN REGIONS, and only for customers in those two regions whose values of RATE_SCHEDULE start with the string “E1.” We can do all of this in one PROC MEANS task, with the WHERE Clause Data Set Option playing a large role in this example. Also shown in Step 7 is the NONOBS Option, which eliminates the (default) presentation of the number of observations column in the Output Window. The code is:

```
* Step 7: Take What You Want and Leave the Rest;
PROC MEANS
 DATA=SUGI.ELEC_ANNUAL(WHERE=(REGION IN('WESTERN','SOUTHERN')
            and  SUBSTR(RATE_SCHEDULE,1,2) = 'E1'))
              MAXDEC = 0 MEAN SUM NONOBS;
VAR TOTKWH;
CLASS REGION RATE_SCHEDULE;
title3 'Step 7: Take What You Need and Leave the Rest';
run;
```

The output is:

Continued on Next Page…

```
SUGI 29 in Montreal
Steps to Success with PROC MEANS
Step 7: Take What You Need and Leave the Rest

The MEANS Procedure

               Analysis Variable : totkwh

                 rate_
REGION           schedule            Mean              Sum
__________________________________________________________

SOUTHERN         E1                  6781          3770511
                 E1C                11144           100299
                 E1L                 4582           559014
                 E1M                14550            43650

WESTERN          E1                  5255         23961761
                 E1C                 5615            39305
                 E1L                 4635          1529594
                 E1M                22206           999292
__________________________________________________________
```

What Have We Learned So Far?

The first seven “Steps to Success” have shown various options and approaches to generate analyses and have those analyses displayed in the Output Window. We’ve seen various options to control which statistics are generated, how many decimal places the analyses are rounded, etc. We’ve also seen the role of the CLASS Statement, when both one and two classification variables. Now we’ll take a look at how to place the analyses PROC MEANS creates in to SAS data sets.

Creating SAS Data Sets with PROC MEANS: Some General Considerations

By default, PROC MEANS presents its results in your Output Window. The OUTPUT Statement is used in the PROC MEANS task to create a SAS Data Set with the analyses created by the Procedure. Options within the OUTPUT Statement will control which analyses are created, the names of the variables in the data set which contain or “hold” these analyses, and other aspects of the data sets you can create with the Procedure. In some situations, the structure and/or content of the data set will also be governed by statements and options elsewhere in the PROC MEANS task.

Step 8: Creating a Simple Data Set with PROC MEANS

Looking back to Step 2, suppose that we wanted to store the PROC MEANS-generated analyses in a SAS data set instead of placing them in the Output Window. We can do this by including an OUTPUT Statement as part of the PROC MEANS task, which will control what variables and observations are placed in to the data set. Also, by adding the NOPRINT option to the PROC MEANS Statement, we will suppress presentation of the results in the Output Window. Here is the PROC MEANS task we need:

```
* Step 8: Create a Simple Data Set;
PROC MEANS DATA=SUGI.ELEC_ANNUAL NOPRINT;
VAR TOTREV;
OUTPUT OUT=SUGI1 MEAN=MeanRev SUM=TotalRev;
RUN;

PROC PRINT DATA=SUGI1;
TITLE3 'Step 8: Create a Simple Data Set'; run;
```

Before looking at the output, let’s go over the PROC MEANS task. First, the NOPRINT Option was used to “tell” PROC MEANS to not put its work in the Output Window. The Output Statement’s syntax included the name of the (temporary) Data Set (SUGI1) in which PROC MEANS will place its results. The MEAN and SUM Statistics

Keywords were given, and to the right of the equals sign is the name of the variable in output data set SUGI1 that will hold the analyses generated by those keywords. Now, let's look at PROC PRINT output:

```
SUGI 29 in Montreal
Steps to Success with PROC MEANS
Step 8: Create a Simple Data Set

Obs    _TYPE_    _FREQ_    MeanRev      TotalRev

 1        0       16364    736.803    11635590.67
```

There's one observation (row) in the data set. The variables specified in the OUTPUT Statement, MeanRev and TotalRev, hold the results of applying the MEAN and SUM statistics keywords to the analysis variable TOTREV. By default, PROC MEANS generated two other variables: _TYPE_ and _FREQ_. We'll look at _TYPE_ shortly. _FREQ_ shows the number of observations in the input data set that were used to calculate the analyses stored in data set SUGI1.

Step 9: Creating a More Complex Data Set with PROC MEANS

What happens when we have two or more analysis variables? What we'll need to do is add some more instructions to the OUTPUT Statement to make sure we get exactly what we want in the output data set. In the next example, we will ask PROC MEANS to calculate the SUMs of TOTWKH and TOTREV, and the MEAN of TOTREV and the MEDIAN of TOTHRS. Notice, please, how the OUTPUT Statement is written to obtain the desired results.

```
* Step 9: A More Complex Data Set;
PROC MEANS DATA=SUGI.ELEC_ANNUAL NOPRINT;
VAR TOTREV TOTKWH TOTHRS;
OUTPUT OUT=SUGI2 sum(TOTREV TOTKWH) = sum_rev sum_kwh
                 mean(TOTREV) = mean_rev
                 median(TOTHRS) = median_hrs;
run;
```

You'll see that there are three analysis variables in the VAR statement. In the OUTPUT statement creating temporary data set SUGI2, the name of the variable (or variables) for which the desired analysis is to be carried out are enclosed in parentheses, and the variable names are again given to the right of the equals sign. This example demonstrates how you can select different analyses to be performed on different analysis variables. So, we can easily obtain the SUMs, MEAN and MEDIAN of different analysis variables in one PROC MEANS task. The data set is:

```
SUGI 29 in Montreal
Steps to Success with PROC MEANS
Step 10: A More Complex Data Set

                                                                 median_
Obs    _TYPE_    _FREQ_       sum_rev        sum_kwh    mean_rev      hrs

 1        0       16364    11635590.67    98910183    736.803      8760
```

Step 10: Using the AUTONAME Option

In Steps 8 and 9 we've supplied variable names in the Output Statement to use when creating the output data set. Using the AUTONAME Option, which was added to PROC MEANS in Version 8, instructs the procedure to automatically name the variables it creates and stores in your output data sets. Let's repeat Step 9, but this time use the AUTONAME Option to have PROC MEANS "do the work for us." The PROC MEANS task is:

```
* Step 19: Using the AUTONAME Option;
PROC MEANS DATA=SUGI.ELEC_ANNUAL NOPRINT;
VAR TOTREV TOTKWH TOTHRS;
OUTPUT OUT=SUGI2    sum(TOTREV TOTKWH) =
                    mean(TOTREV) =
                    median(TOTHRS) =  / AUTONAME;
run;
```

When the AUTONAME option is specified, PROC MEANS creates variable names in the output data set by taking the name of the analysis variable, appending an underscore to it, and then "attaching" the name of the statistics keyword to right of the hyphen. Here is the output data set:

```
SUGI 29 in Montreal
Steps to Success with PROC MEANS
Step 10: Using the AUTONAME Option

                                          totkwh_    totrev_    tothrs_
Obs    _TYPE_    _FREQ_    totrev_Sum       Sum        Mean      Median

 1        0       16364    11635590.67    98910183    736.803     8760
```

Step 11: Understanding _TYPE_ and _FREQ_

In Steps 10 and 11 we've seen the variables _TYPE_ and _FREQ_, which are automatically generated by PROC MEANS when you use the OUTPUT statement to create a SAS data set. Let's take a look at what _TYPE_ is, and what it can do for us. To gain an appreciation of what this variable is, and what it represents, let's re-run the PROC MEANS task shown in Step 7, but this time we'll store the analyses in a SAS data set. Here's the revised code:

```
* Step 11: Understanding _TYPE_ and _FREQ_;
 PROC MEANS
 DATA=SUGI.ELEC_ANNUAL(WHERE=(REGION IN('WESTERN','SOUTHERN')
           and  SUBSTR(RATE_SCHEDULE,1,2) = 'E1')) ;
VAR TOTKWH;
CLASS REGION RATE_SCHEDULE;
OUTPUT OUT=SUGI3 MEAN= SUM= MEDIAN=/AUTONAME;
```

The output data set looks like this:

```
Step 11: Understanding _TYPE_ and _FREQ_

                           rate_                           totkwh_      totkwh_     totkwh_
Obs     REGION           schedule    _TYPE_    _FREQ_       Mean          Sum       Median

  1                                     0       5645       5504.87     31003426     4731.0
  2                         E1          1       5129       5420.69     27732272     4760.0
  3                         E1C         1         16       8725.25       139604     7704.0
  4                         E1L         1        452       4620.81      2088608     4209.5
  5                         E1M         1         48      21727.96      1042942     8301.0
  6     SOUTHERN                        2        690       6483.30      4473474     5825.5
  7     WESTERN                         2       4955       5368.26     26529952     4633.0
  8     SOUTHERN            E1          3        556       6781.49      3770511     6175.0
  9     SOUTHERN            E1C         3          9      11144.33       100299     9789.0
 10     SOUTHERN            E1L         3        122       4582.08       559014     4424.0
 11     SOUTHERN            E1M         3          3      14550.00        43650    14550.0
 12     WESTERN             E1          3       4573       5254.77     23961761     4648.5
 13     WESTERN             E1C         3          7       5615.00        39305     5615.0
 14     WESTERN             E1L         3        330       4635.13      1529594     4177.0
 15     WESTERN             E1M         3         45      22206.49       999292     8002.0
```

The data set at the bottom of the previous page gives us a very clear idea of what information is contain in the "automatic" variables _TYPE_ and _FREQ_. First, _FREQ_ shows us the number of observations in the input, or source data set, that were used to calculate the analyses in that observation (row) in the output data set. For example, there were 45 observations from ELEC_ANNUAL that had a value of REGION equal to WESTERN and a value of RATE_SCHEDULE equal to E1H. These 45 observations were used to calculate the SUM, MEAN and MEDIAN of TOTKWH shown in the last row of the PROC PRINT output.

The variable _TYPE_ identifies the combination of classification variables whose values were used to create the analyses in that observation in the output data set. By default, _TYPE_ is a numeric variable (unless you have more than 32 classification variables, in which case it is a character variable, as discussed below). The first observation, with _TYPE_ = 0, is the "grand" or "overall" analysis without regard to the values of the classification variables. Observations with _TYPE_ = 1 give the analysis by RATE_SCHEDULE without regard to REGION, _TYPE_ = 2 are the analyses by REGION without regard to RATE_SCHEDULE, and _TYPE_ = 3 are the analyses at all possible combinations of the values of both REGION *and* RATE_SCHEDULE.

With just one classification variable you will have two values of _TYPE_ (zero and one). When you have two classification variables you get four values, and with three you will have eight values. By now you've probably figured out that the number of values of _TYPE_ in your output data set is equal to 2^N, where N is the number of classification variables.

Step 12: Use the NWAY Option

In many analytic situations you'll only want observations in the output data set that have the highest possible value of _TYPE_ given the number of classification variables. For example, suppose in the analysis shown in Step 11 you just want observations with _TYPE_ = 3 (those at all possible combinations of the values of REGION and RATE_SCHEDULE). Adding the NWAY Option to your PROC MEANS Statement limits the placement of observations in your output data set to just those where, in this example, _TYPE_ = 3. [To make room for other output, an example of implementing the NWAY Option is not presented in this paper.]

Step 13: Use the CHARTYPE Option to Facilitate Creating Multiple Output Data Sets in a Single PROC MEANS Task

You can use multiple OUTPUT Statements in a single PROC MEANS task to create several output datasets with different analyses at different combinations of the classification variables. This is clearly more efficient than writing (and running) multiple PROC MEANS tasks against the same data set. Instead, what we can do is write multiple OUTPUT statements, and use the WHERE clause SAS data set option to direct placement of the PROC MEANS-generated observations in to the data sets you need. For this example we'll switch over to the CARD_TRANS data set. The classification variables are CARDTYPE, TRANS_DATE and TRANS_TYPE and the analysis variable is CHARGE_AMOUNT.

With three classification variables we are going to have eight values of _TYPE_, numbered zero (0) to seven (7). Suppose we want to create a data set with the analysis just by CARD_TYPE, one by CARD_TYPE and TRANS_TYPE, and another one by TRANS_DATE and CARDTYPE and, finally one with the analysis by CARD_TYPE, TRANS_DATE a TRANS_TYPE. We can create all four in one PROC MEANS task (rather than in four separate tasks) and by using the CHARTYPE option (added in SAS Version 8) in the PROC MEANS statement, facilitate output of observations from memory in to the four desired data sets *without having to know the numeric value of _TYPE_ corresponding to the observations* we want output to the different data sets.

To fix ideas, let's first look at the PROC MEANS task:

```
PROC MEANS NOPRINT DATA=SUGI.CARDTRANS2;
CLASS TRANS_DATE TRANS_TYPE CARD_TYPE;
VAR CHARGE_AMOUNT;
OUTPUT OUT=SUGI4 MEAN=/AUTONAME;
run;
```

By running this task, PROC MEANS will output all observations it creates to temporary data set SUGI4. But, what we really want are four separate SAS data sets, each with the MEAN of CHARGE_AMOUNT at different combinations of the variables in the CLASS statement. How do we do this? Well, the easiest way would be to create four separate OUTPUT statements, and use the WHERE clause data set option to direct the output of the observations PROC MEANS creates to the four data sets. The CHARTYPE option converts the (default) numeric value of _TYPE_ in to a character variable whose values are a series of zeros and ones corresponding to the position of the variables in the CLASS statement. Using the CHARTYPE option makes it much easier to take advantage of PROC MEANS' ability to create multiple output data sets than if you had to figure out the default numeric value. Here's how it works:

```
PROC MEANS NOPRINT DATA=SUGI.CARDTRANS2 CHARTYPE;
CLASS TRANS_DATE TRANS_TYPE CARD_TYPE;
VAR CHARGE_AMOUNT;
OUTPUT OUT=A(WHERE=(_TYPE_ = '001')) MEAN=/AUTONAME;
OUTPUT OUT=B(WHERE=(_TYPE_ = '011')) MEAN=/AUTONAME;
OUTPUT OUT=C(WHERE=(_TYPE_ = '101')) MEAN=/AUTONAME;
OUTPUT OUT=D(WHERE=(_TYPE_ = '111')) MEAN=/AUTONAME;
RUN;
```

Since we have three classification variables, the value of _TYPE_ is a three-byte character variable (again, because the CHARTYPE option was supplied in the PROC MEANS statement). The first position corresponds to TRANS_DATE, the first (left-most) variable in the CLASS statement. The second position corresponds to TRANS_DATE and the third position corresponds to CARD_TYPE. For example, when _TYPE_ = '001' we are identifying observations where the variable CHARGE_AMOUNT is being analyzed by the values of the classification variable CARD_TYPE.

Other Steps

There are many other Steps to Success with PROC MEANS that you can learn about from the Procedure's documentation.

Conclusion

In this paper I have identified what I think are twelve key Steps to Success that you can use to have PROC MEANS create the analyses and data sets you need from your data sets. They offer what I hope is a comprehensive insight in to how the take advantage of the procedure's functionalities.

Acknowledgements

Thanks to Robert Ray of SAS Institute's BASE Information Technology group for his insights in to PROC MEANS and many of the enhancements added to it in Version 8. Also, thanks to the many people who have attended my "Summarizing and Reporting Data Using the SAS System" seminar who have made comments or asked questions that challenged me to learn more about PROC MEANS.

Author contact

Andrew H. Karp
President
Sierra Information Services, Inc.
19229 Sonoma Highway PMB 264
Sonoma, California 94115 USA
707 996 7380
SierraInfo@AOL.COM
www.SierraInformation.com

Copyright