CHAPTER
24

The MEANS Procedure

Overview

The MEANS procedure provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations. For example, PROC MEANS

- □ calculates descriptive statistics based on moments
- □ estimates quantiles, which includes the median
- □ calculates confidence limits for the mean
- □ identifies extreme values
- □ performs a *t* test.

By default, PROC MEANS displays output. You can also use the OUTPUT statement to store the statistics in a SAS data set.

PROC MEANS and PROC SUMMARY are very similar; see Chapter 36, "The SUMMARY Procedure," on page 1149 for an explanation of the differences.

Output 24.1 on page 624 shows the default output that PROC MEANS displays. The data set that PROC MEANS analyzes contains the integers 1 through 10. The output reports the number of observations, the mean, the standard deviation, the minimum value, and the maximum value. The statements that produce the output follow:

```
proc means data=OnetoTen;
run;
```

Output 24.1 The Default Descriptive Statistics

```
                              The SAS System                              1

                           The MEANS Procedure

                      Analysis Variable : Integer

  N            Mean         Std Dev         Minimum         Maximum
 ------------------------------------------------------------------
 10       5.5000000       3.0276504       1.0000000      10.0000000
 ------------------------------------------------------------------
```

Output 24.2 on page 624 shows the results of a more extensive analysis of two variables, MoneyRaised and HoursVolunteered. The analysis data set contains information about the amount of money raised and the number of hours volunteered by high-school students for a local charity. PROC MEANS uses six combinations of two categorical variables to compute the number of observations, the mean, and the range. The first variable, School, has two values and the other variable, Year, has three values. For an explanation of the program that produces the output, see Example 11 on page 677.

Output 24.2 Specified Statistics for Class Levels and Identification of Maximum Values

```
                  Summary of Volunteer Work by School and Year                 1

                              The MEANS Procedure

                         N
School           Year  Obs  Variable            N          Mean         Range
-----------------------------------------------------------------------------
47.33              26    1  MoneyRaised         0             .             .
                            HoursVolunteered    0             .             .

Kennedy          1992   15  MoneyRaised        15    29.0800000    39.7500000
                            HoursVolunteered   15    22.1333333    30.0000000

                 1993   20  MoneyRaised        20    28.5660000    23.5600000
                            HoursVolunteered   20    19.2000000    20.0000000

                 1994   18  MoneyRaised        18    31.5794444    65.4400000
                            HoursVolunteered   18    24.2777778    15.0000000

Monroe           1992   16  MoneyRaised        16    28.5450000    48.2700000
                            HoursVolunteered   16    18.8125000    38.0000000

                 1993   12  MoneyRaised        11    26.2972727    52.4600000
                            HoursVolunteered   11    14.9090909    18.0000000

                 1994   28  MoneyRaised        28    29.4100000    73.5300000
                            HoursVolunteered   28    19.1428571    26.0000000
-----------------------------------------------------------------------------
```

```
             Best Results: Most Money Raised and Most Hours Worked            2

                                         Most     Most      Money     Hours
  Obs  School   Year  _TYPE_  _FREQ_  Cash     Time     Raised  Volunteered

    1              .     0      110   Willard  Tonya     78.65       40
    2             26     1        1                        .          .
    3           1992     1       31   Tonya    Tonya     55.16       40
    4           1993     1       32   Cameron  Amy       65.44       31
    5           1994     1       46   Willard  L.T.      78.65       33
    6  47.33       .     2        1                        .          .
    7  Kennedy     .     2       53   Luther   Jay       72.22       35
    8  Monroe      .     2       56   Willard  Tonya     78.65       40
    9  47.33      26     3        1                        .          .
   10  Kennedy  1992     3       15   Thelma   Jay       52.63       35
   11  Kennedy  1993     3       20   Bill     Amy       42.23       31
   12  Kennedy  1994     3       18   Luther   Che-Min   72.22       33
   13  Monroe   1992     3       16   Tonya    Tonya     55.16       40
   14  Monroe   1993     3       12   Cameron  Tyra      65.44       23
   15  Monroe   1994     3       28   Willard  L.T.      78.65       33
```

In addition to the report, the program also creates an output data set (located on page 2 of the output) that identifies the students who raised the most money and who volunteered the most time over all the combinations of School and Year and within the combinations of School and Year:

- □ The first observation in the data set shows the students with the maximum values overall for MoneyRaised and HoursVolunteered.
- □ Observations 2 through 4 show the students with the maximum values for each year, regardless of school.
- □ Observations 5 and 6 show the students with the maximum values for each school, regardless of year.

- Observations 7 through 12 show the students with the maximum values for each school-year combination.

Procedure Syntax

Tip: Supports the Output Delivery System, see Chapter 2, “Fundamental Concepts for Using Base SAS Procedures,” on page 15

Reminder: You can use the ATTRIB, FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," for details. You can also use any global statements as well. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

PROC MEANS <*option(s)*> <*statistic-keyword(s)*>;
  BY <DESCENDING> *variable-1* <... <DESCENDING> *variable-n*><NOTSORTED>;
  CLASS *variable(s)* </ *option(s)*>;
  FREQ *variable*;
  ID *variable(s)*;
  OUTPUT <OUT=*SAS-data-set*> <*output-statistic-specification(s)*>
    <*id-group-specification(s)*> <*maximum-id-specification(s)*>
    <*minimum-id-specification(s)*> </ *option(s)*> ;
  TYPES *request(s)*;
  VAR *variable(s)* < / WEIGHT=*weight-variable*>;
  WAYS *list*;
  WEIGHT *variable*;

To do this	Use this statement
Calculate separate statistics for each BY group	BY
Identify variables whose values define subgroups for the analysis	CLASS
Identify a variable whose values represent the frequency of each observation	FREQ
Include additional identification variables in the output data set	ID
Create an output data set that contains specified statistics and identification variables	OUTPUT
Identify specific combinations of class variables to use to subdivide the data	TYPES
Identify the analysis variables and their order in the results	VAR

To do this	Use this statement
Specify the number of ways to make unique combinations of class variables	WAYS
Identify a variable whose values weight each observation in the statistical calculations	WEIGHT

PROC MEANS Statement

See also: Chapter 36, "The SUMMARY Procedure," on page 1149

PROC MEANS <*option(s)*> <*statistic-keyword(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Disable floating point exception recovery	NOTRAP
Specify the amount of memory to use for data summarization with class variables	SUMSIZE=
Control the classification levels	
Specify a secondary data set that contains the combinations of class variables to analyze	CLASSDATA=
Create all possible combinations of class variable values	COMPLETETYPES
Exclude from the analysis all combinations of class variable values that are not in the CLASSDATA= data set	EXCLUSIVE
Use missing values as valid values to create combinations of class variables	MISSING
Control the statistical analysis	
Specify the confidence level for the confidence limits	ALPHA=
Exclude observations with nonpositive weights from the analysis	EXCLNPWGTS
Specify the sample size to use for the P2 quantile estimation method	QMARKERS=
Specify the quantile estimation method	QMETHOD=
Specify the mathematical definition used to compute quantiles	QNTLDEF=
Select the statistics	*statistic-keyword*
Specify the variance divisor	VARDEF=
Control the output	
Specify the field width for the statistics	FW=
Specify the number of decimal places for the statistics	MAXDEC=

To do this	Use this option
Suppress reporting the total number of observations for each unique combination of the class variables	NONOBS
Suppress all displayed output	NOPRINT
Order the values of the class variables according to the specified order	ORDER=
Display the output	PRINT
Display the analysis for all requested combinations of class variables	PRINTALLTYPES
Display the values of the ID variables	PRINTIDVARS
Control the output data set	
Specify that the _TYPE_ variable contain character values.	CHARTYPE
Order the output data set by descending _TYPE_ value	DESCENDTYPES
Select ID variables based on minimum values	IDMIN
Limit the output statistics to the observations with the highest _TYPE_ value	NWAY

Options

ALPHA=*value*
specifies the confidence level to compute the confidence limits for the mean. The percentage for the confidence limits is (1–*value*)×100. For example, ALPHA=.05 results in a 95% confidence limit.

Default: .05

Range: between 0 and 1

Interaction: To compute confidence limits specify the *statistic-keyword* CLM, LCLM, or UCLM.

See also: “Confidence Limits” on page 652

Featured in: Example 7 on page 671

CHARTYPE
specifies that the _TYPE_ variable in the output data set is a character representation of the binary value of _TYPE_. The length of the variable equals the number of class variables.

Main discussion: “Output Data Set” on page 655

Interaction When you specify more than 32 class variables, _TYPE_ automatically becomes a character variable.

Featured in: Example 10 on page 676

CLASSDATA=*SAS-data-set*
specifies a data set that contains the combinations of values of the class variables that must be present in the output. Any combinations of values of the class variables that occur in the CLASSDATA= data set but not in the input data set appear in the output and have a frequency of zero.

Restriction: The CLASSDATA= data set must contain all class variables. Their data type and format must match the corresponding class variables in the input data set.

Interaction: If you use the EXCLUSIVE option, PROC MEANS excludes any observation in the input data set whose combination of class variables is not in the CLASSDATA= data set.

Tip: Use the CLASSDATA= data set to filter or to supplement the input data set.

Featured in: Example 4 on page 662

COMPLETETYPES
creates all possible combinations of class variables even if the combination does not occur in the input data set.

Interaction: The PRELOADFMT option in the CLASS statement ensures that PROC MEANS ouputs all user-defined format ranges or values for the combinations of class variables, even when a frequency is zero.

Tip: Using COMPLETETYPES does not increase the memory requirements.

Featured in: Example 6 on page 668

DATA=*SAS-data-set*
identifies the input SAS data set.

Main discussion: "Input Data Sets" on page 18

DESCENDTYPES
orders observations in the output data set by descending _TYPE_ value.

Alias: DESCENDING | DESCEND

Interaction: Descending has no effect if you specify NWAY.

Tip: Use DESCENDTYPES to make the overall total (_TYPE_=0) the last observation in each BY group.

See also: "Output Data Set" on page 655

Featured in: Example 9 on page 674

EXCLNPWGTS
excludes observations with nonpositive weight values (zero or negative) from the analysis. By default, PROC MEANS treats observations with negative weights like those with zero weights and counts them in the total number of observations.

Alias: EXCLNPWGT

See also: WEIGHT= on page 647 and "WEIGHT Statement" on page 648

EXCLUSIVE
excludes from the analysis all combinations of the class variables that are not found in the CLASSDATA= data set.

Requirement: If a CLASSDATA= data set is not specified, this option is ignored.

Featured in: Example 4 on page 662

FW=*field-width*
specifies the field width to display the statistics in the output.

Default: 12

Tip: If PROC MEANS truncates column labels in the output, increase the field width.

Featured in: Example 1 on page 657, Example 4 on page 662, and Example 5 on page 665

IDMIN
specifies that the output data set contain the minimum value of the ID variables.

Interaction: Specify PRINTIDVARS to display the value of the ID variables in the output.

See: "ID Statement" on page 639

MAXDEC=*number*
specifies the maximum number of decimal places to display the statistics in the output.

Default: BEST. width for columnar format, typically about 7. (This does not apply to the PROBT statistic. The SAS system option PROBSIG= determines its format. See SAS system options in *SAS Language Reference: Concepts* for details.)

Range: 0-8

Featured in: Example 2 on page 658 and Example 4 on page 662

MISSING
considers missing values as valid values to create the combinations of class variables. Special missing values that represent numeric values (the letters A through Z and the underscore (_) character) are each considered as a separate value.

Default: If you omit MISSING, PROC MEANS excludes the observations with a missing class variable value from the analysis.

See also: *SAS Language Reference: Concepts* for a discussion of missing values that have special meaning.

Featured in: Example 6 on page 668

NONOBS
suppresses the column that displays the total number of observations for each unique combination of the values of the class variables. This column corresponds to the _FREQ_ variable in the output data set.

See also: "The N Obs Statistic" on page 655

Featured in: Example 5 on page 665 and Example 6 on page 668

NOPRINT
See PRINT | NOPRINT.

NOTRAP
disables floating point exception (FPE) recovery during data processing. By default, PROC MEANS traps these errors and sets the statistic to missing.

In operating environments where the overhead of FPE recovery is significant, NOTRAP can improve performance. Note that normal SAS System FPE handling is still in effect so that PROC MEANS terminates in the case of math exceptions.

NWAY
specifies that the output data set contain only statistics for the observations with the highest _TYPE_ and _WAY_ values. When you specify class variables, this corresponds to the combination of all class variables.

Interaction: If you specify a TYPES statement or a WAYS statements, PROC MEANS ignores this option.

See also: "Output Data Set" on page 655

Featured in: Example 10 on page 676

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
specifies the sort order to create the unique combinations for the values of the class variables in the output, where

DATA
orders values according to their order in the input data set.

Interaction: If you use PRELOADFMT in the CLASS statement, the order for the values of each class variable matches the order that PROC FORMAT uses to store the values of the associated user-defined format. If you use the CLASSDATA= option, PROC MEANS uses the order of the unique values of each class variable in the CLASSDATA= data set to order the output levels. If you use both options, PROC MEANS first uses the user-defined formats to order the output. If you omit EXCLUSIVE, PROC MEANS appends after the user-defined format and the CLASSDATA= values the unique values of the class variables in the input data set based on the order that they are encountered.

Tip: By default, PROC FORMAT stores a format definition in sorted order. Use the NOTSORTED option to store the values or ranges of a user defined format in the order that you define them.

FORMATTED
orders values by their ascending formatted values. This order depends on your operating environment.

Alias: FMT | EXTERNAL

FREQ
orders values by descending frequency count so that levels with the most observations are listed first.

Interaction: For multiway combinations of the class variables, PROC MEANS determines the order of a class variable combination from the individual class variable frequencies.

Interaction: Use the ASCENDING option in the CLASS statement to order values by ascending frequency count.

UNFORMATTED
orders values by their unformatted values, which yields the same order as PROC SORT. This order depends on your operating environment.

Alias: UNFMT | INTERNAL

Default: UNFORMATTED

See also: "Ordering the Class Values" on page 649

PRINT | NOPRINT
specifies whether PROC MEANS displays the statistical analysis. NOPRINT suppresses all the output.

Default: PRINT

Tip: Use NOPRINT when you want to create only an OUT= output data set.

Featured in: For an example of NOPRINT, see Example 8 on page 672 and Example 12 on page 680

PRINTALLTYPES
displays all requested combinations of class variables (all _TYPE_ values) in the output. Normally, PROC MEANS shows only the NWAY type.

Alias: PRINTALL

Interaction: If you use the NWAY option, the TYPES statement, or the WAYS statement, PROC MEANS ignores this option.

Featured in: Example 4 on page 662

PRINTIDVARS
displays the values of the ID variables in output.

Alias: PRINTIDS

Interaction: Specify IDMIN to display the minimum value of the ID variables.

See: "ID Statement" on page 639

QMARKERS=*number*
specifies the default number of markers to use for the P^2 quantile estimation method. The number of markers controls the size of fixed memory space.

Default: The default value depends on which quantiles you request. For the median (P50), *number* is 7. For the quartiles (P25 and P50), *number* is 25. For the quantiles P1, P5, P10, P90, P95, or P99, *number* is 105. If you request several quantiles, PROC MEANS uses the largest value of *number*.

Range: an odd integer greater than 3

Tip: Increase the number of markers above the defaults settings to improve the accuracy of the estimate; reduce the number of markers to conserve memory and computing time.

Main Discussion "Quantiles" on page 653

QMETHOD=OS|P2

specifies the method PROC MEANS uses to process the input data when it computes quantiles. If the number of observations is less than or equal to the QMARKERS= value and QNTLDEF=5, both methods produce the same results.

OS

uses order statistics. This is the same method that PROC UNIVARIATE uses.

Note: This technique can be very memory-intensive. △

P2

uses the P^2 method to approximate the quantile.

Default: OS

Restriction: When QMETHOD=P2, PROC MEANS will not compute weighted quantiles.

Tip: When QMETHOD=P2, reliable estimations of some quantiles (P1,P5,P95,P99) may not be possible for some data sets.

Main Discussion: "Quantiles" on page 653

QNTLDEF=1|2|3|4|5

specifies the mathematical definition that PROC MEANS uses to calculate quantiles when QMETHOD=OS. To use QMETHOD=P2, you must use QNTLDEF=5.

Default: 5

Alias: PCTLDEF=

Main discussion: "Calculating Percentiles" on page 1404

statistic-keyword(s)

specifies which statistics to compute and the order to display them in the output. The available keywords in the PROC statement are

Descriptive statistic keywords

CLM	RANGE
CSS	SKEWNESS|SKEW
CV	STDDEV|STD
KURTOSIS|KURT	STDERR
LCLM	SUM
MAX	SUMWGT
MEAN	UCLM
MIN	USS
N	VAR
NMISS	

Quantile statistic keywords

MEDIAN | P50	Q3 | P75
P1	P90
P5	P95
P10	P99
Q1 | P25	QRANGE

Hypothesis testing keyword

PROBT	T

Default: N, MEAN, STD, MIN, and MAX

Requirement: To compute standard error, confidence limits for the mean, and the Student's *t* test you must use the default value of VARDEF= which is DF. To compute skewness or kurtosis you must use VARDEF=N or VARDEF=DF.

Tip: Use CLM or both LCLM and UCLM to compute a two-sided confidence limit for the mean. Use only LCLM or UCLM, to compute a one-sided confidence limit.

Main discussion: The definitions of the keywords and the formulas for the associated statistics are listed in "Keywords and Formulas" on page 1458.

Featured in: Example 1 on page 657 and Example 3 on page 660

SUMSIZE=*value*
specifies the amount of memory that is available for data summarization when you use class variables. *value* may be one of the following:

n | *n*K | *n*M | *n*G
specifies the amount of memory available in bytes, kilobytes, megabytes, or gigabytes, respectively. If *n* is 0, PROC MEANS use the value of the SAS system option SUMSIZE=.

MAXIMUM | MAX
specifies the maximum amount of memory that is available.

Default: The value of the SUMSIZE= system option.

Tip: For best results, do not make SUMSIZE= larger than the amount of physical memory that is available for the PROC step. If additional space is needed, PROC MEANS uses utility files.

See also: The SAS system option SUMSIZE= in *SAS Language Reference: Dictionary*.

Main discussion: "Computational Resources" on page 650

VARDEF=*divisor*
specifies the divisor to use in the calculation of the variance and standard deviation. Table 24.1 on page 633 shows the possible values for *divisor* and associated divisors.

Table 24.1 Possible Values for VARDEF=

Value	Divisor	Formula for Divisor
DF	degrees of freedom	$n - 1$
N	number of observations	n

Value	Divisor	Formula for Divisor
WDF	sum of weights minus one	$(\Sigma_i \, w_i) - 1$
WEIGHT |WGT	sum of weights	$\Sigma_i \, w_i$

The procedure computes the variance as $CSS/divisor$, where CSS is the corrected sums of squares and equals $\sum (x_i - \overline{x})^2$. When you weight the analysis variables, CSS equals $\sum w_i (x_i - \overline{x}_w)^2$, where $\overline{x}_w$ is the weighted mean.

Default: DF

Requirement: To compute the standard error of the mean, confidence limits for the mean, or the Student's t-test, use the default value of VARDEF=.

Tip: When you use the WEIGHT statement and VARDEF=DF, the variance is an estimate of σ^2, where the variance of the ith observation is $var(x_i) = \sigma^2/w_i$ and w_i is the weight for the ith observation. This yields an estimate of the variance of an observation with unit weight.

Tip: When you use the WEIGHT statement and VARDEF=WGT, the computed variance is asymptotically (for large n) an estimate of $\sigma^2/\overline{w}$, where $\overline{w}$ is the average weight. This yields an asymptotic estimate of the variance of an observation with average weight.

See also: the example of weighted statistics“Example” on page 74

Main discussion: “Keywords and Formulas” on page 1458

BY Statement

Produces separate statistics for each BY group.

Main discussion: “BY” on page 68

See also: “Comparison of the BY and CLASS Statements” on page 638

Featured in: Example 3 on page 660

BY <DESCENDING> *variable-1* <...<DESCENDING> *variable-n*> <NOTSORTED>;

Required Arguments

variable

specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you omit the NOTSORTED option in the BY statement, the observations in the data set must either be sorted by all the variables that you specify, or they must be indexed appropriately. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The observations are sorted in another way, for example, chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

Using the BY Statement with the SAS System Option NOBYLINE

If you use the BY statement with the SAS system option NOBYLINE, which suppresses the BY line that normally appears in output that is produced with BY-group processing, PROC MEANS always starts a new page for each BY group. This behavior ensures that if you create customized BY lines by putting BY-group information in the title and suppressing the default BY lines with NOBYLINE, the information in the titles matches the report on the pages. (See "Creating Titles That Contain BY-Group Information" on page 54"Suppressing the Default BY Line" on page 54.)

CLASS Statement

Specifies the variables whose values define the subgroup combinations for the analysis.

Tip: You can use multiple CLASS statements.

Tip: Some CLASS statement options are also available in the PROC MEANS statement. They affect all CLASS variables rather than just to the one(s) you specify in a CLASS statement.

See also: For information about how the CLASS statement groups formatted values, see "Formatted Values" on page 59.

Featured in: Example 2 on page 658, Example 4 on page 662, Example 5 on page 665, Example 6 on page 668, and Example 10 on page 676

CLASS *variable(s)* <*/ options*>;

Required Arguments

variable(s)
specifies one or more variables that the procedure uses to group the data. Variables in a CLASS statement are referred to as *class variables*. Class variables are numeric or character. Class variables can have continuous values, but they typically have a few discrete values that define levels of the variable. You do not have to sort the data by class variables.

Interaction: Use the TYPES statement and the WAYS statement to control which class variables that PROC MEANS uses to group the data.

Tip: To reduce the number of class variable levels, use a FORMAT statement to combine variable values. When a format combines several internal values into one formatted value, PROC MEANS outputs the lowest internal value.

See also: "Using Class Variables" on page 649

Options

ASCENDING

specifies to sort the class variable levels in ascending order.

Alias: ASCEND

Interaction: PROC MEANS issues a warning message if you specify both ASCENDING and DESCENDING and ignores both options.

Featured in: Example 10 on page 676

DESCENDING

specifies to sort the class variable levels in descending order.

Alias: DESCEND

Interaction: PROC MEANS issues a warning message if you specify both ASCENDING and DESCENDING and ignores both options.

EXCLUSIVE

excludes from the analysis all combinations of the class variables that are not found in the preloaded range of user-defined formats.

Requirement: You must specify PRELOADFMT to preload the class variable formats.

Featured in: Example 6 on page 668

GROUPINTERNAL

specifies not to apply formats to the class variables when PROC MEANS groups the values to create combinations of class variables.

Interaction: If you specify the PRELOADFMT option, PROC MEANS ignores this option and uses the formatted values.

Tip: This option saves computer resources when the numeric class variables contain discrete values.

See also: "Computer Resources" on page 638

MISSING

considers missing values as valid values for the class variable levels. Special missing values that represent numeric values (the letters A through Z and the underscore (_) character) are each considered as a separate value.

Default: If you omit MISSING, PROC MEANS excludes the observations with a missing class variable value from the analysis.

See also: *SAS Language Reference: Concepts* for a discussion of missing values with special meanings.

Featured in: Example 10 on page 676

MLF

enables PROC MEANS to use the primary and secondary format labels for a given range or overlapping ranges to create subgroup combinations when a multilabel format is assigned to a class variable.

Requirement: You must use PROC FORMAT and the MULTILABEL option in the VALUE statement to create a multilabel format.

Interaction: If you use the OUTPUT statement with MLF, the class variable contains a character string that corresponds to the formatted value. Because the formatted value becomes the internal value, the length of this variable is the number of characters in the longest format label.

Interaction: Using MLF with ORDER=FREQ may not produce the order that you expect for the formatted values.

Tip: If you omit MLF, PROC MEANS uses the primary format labels, which corresponds to using the first external format value, to determine the subgroup combinations.

See also: The MULTILABEL on page 442 option in the VALUE statement of the FORMAT procedure.

Featured in: Example 5 on page 665

Note: When the formatted values overlap, one internal class variable value maps to more than one class variable subgroup combination. Therefore, the sum of the N statistics for all subgroups is greater the number of observations in the data set (the overall N statistic). △

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED

specifies the order to group the levels of the class variables in the output, where

DATA

orders values according to their order in the input data set.

Interaction: If you use PRELOADFMT, the order for the values of each class variable matches the order that PROC FORMAT uses to store the values of the associated user-defined format. If you use the CLASSDATA= option in the PROC statement, PROC MEANS uses the order of the unique values of each class variable in the CLASSDATA= data set to order the output levels. If you use both options, PROC MEANS first uses the user-defined formats to order the output. If you omit EXCLUSIVE in the PROC statement, PROC MEANS appends after the user-defined format and the CLASSDATA= values the unique values of the class variables in the input data set based on the order that they are encountered.

Tip: By default, PROC FORMAT stores a format definition in sorted order. Use the NOTSORTED option to store the values or ranges of a user defined format in the order that you define them.

Featured in: Example 10 on page 676

FORMATTED

orders values by their ascending formatted values. This order depends on your operating environment.

Alias: FMT | EXTERNAL

Featured in: Example 5 on page 665

FREQ

orders values by descending frequency count so that levels with the most observations are listed first.

Interaction: For multiway combinations of the class variables, PROC MEANS determines the order of a level from the individual class variable frequencies.

Interaction: Use the ASCENDING option to order values by ascending frequency count.

Featured in: Example 5 on page 665

UNFORMATTED
orders values by their unformatted values, which yields the same order as PROC SORT. This order depends on your operating environment. This sort sequence is particularly useful for displaying dates chronologically.

Alias: UNFMT | INTERNAL

Default: UNFORMATTED

Tip: By default, all orders except FREQ are ascending. For descending orders, use the DESCENDING option.

See also: "Ordering the Class Values" on page 649

PRELOADFMT
specifies that all formats are preloaded for the class variables.

Requirement: PRELOADFMT has no effect unless you specify either COMPLETETYPES, EXCLUSIVE, or ORDER=DATA and you assign formats to the class variables.

Interaction: To limit PROC MEANS output to the combinations of formatted class variable values present in the input data set, use the EXCLUSIVE option in the CLASS statement.

Interaction: To include all ranges and values of the user-defined formats in the output, even when the frequency is zero, use COMPLETETYPES in the PROC statement.

Featured in: Example 6 on page 668

Comparison of the BY and CLASS Statements

Using the BY statement is similar to using the CLASS statement and the NWAY option in that PROC MEANS summarizes each BY group as an independent subset of the input data. Therefore, no overall summarization of the input data is available. However, unlike the CLASS statement, the BY statement requires that you previously sort BY variables.

When you use the NWAY option, PROC MEANS may encounter insufficient memory to the summarization all the class variables. You can move some class variables to the BY statement. For maximum benefit, move class variables to the BY statement that are already sorted or that have the greatest number of unique values.

You can use the CLASS and BY statements together to analyze the data by the levels of class variables within BY groups. See Example 3 on page 660.

How PROC MEANS Handles Missing Values for Class Variables

By default, if an observation contains a missing value for any class variable, PROC MEANS excludes that observation from the analysis. If you specify the MISSING option in the PROC statement, the procedure considers missing values as valid levels for the combination of class variables.

Specifying the MISSING option in the CLASS statement allows you to control the acceptance of missing values for individual class variables.

Computer Resources

The total of unique class values that PROC MEANS allows depends on the amount of computer memory that is available. See "Computational Resources" on page 650 for more information.

The GROUPINTERNAL option can improve computer performance because the grouping process is based on the internal values of the class variables. If a numeric class variable is not assigned a format and you do not specify GROUPINTERNAL, PROC MEANS uses the default format to format numeric values as character strings. Then PROC MEAN groups these numeric variables by their character values, which takes additional time and computer memory.

FREQ Statement

Specifies a numeric variable that contains the frequency of each observation.

Main discussion: "FREQ" on page 70

FREQ *variable*;

Required Arguments

variable

specifies a numeric variable whose value represents the frequency of the observation. If you use the FREQ statement, the procedure assumes that each observation represents *n* observations, where *n* is the value of *variable*. If *n* is not an integer, the SAS System truncates it. If *n* is less than 1 or is missing, the procedure does not use that observation to calculate statistics.

The sum of the frequency variable represents the total number of observations.

Note: The FREQ variable does not affect how PROC MEANS identifies multiple extremes when you use the IDGROUP syntax in the OUTPUT statement. △

ID Statement

Includes additional variables in the output data set.

ID *variable(s)*;

Required Arguments

variable(s)

identifies one or more variables from the input data set whose maximum values for groups of observations PROC MEANS includes in the output data set.

Interaction: Use IDMIN in the PROC statement to include the minimum value of the ID variables in the output data set.

Tip: Use the PRINTIDVARS option in the PROC statement to include the value of the ID variable in the displayed output.

Selecting the Values of the ID Variables

When you specify only one variable in the ID statement, the value of the ID variable for a given observation is the maximum (minimum) value found in the corresponding group of observations in the input data set. When you specify multiple variables in the ID statement, PROC MEANS selects the maximum value by processing the variables in the ID statement in the order that you list them. PROC MEANS determines which observation to use from all the ID variables by comparing the values of the first ID variable. If more than one observation contains the same maximum (minimum) ID value, PROC MEANS uses the second and subsequent ID variable values as "tie breakers". In any case, all ID values are taken from the same observation for any given BY group or classification level within a type.

See “Sorting Orders for Character Variables” on page 1012 for information on how PROC MEANS compares character values to determine the maximum value.

OUTPUT Statement

Outputs statistics to a new SAS data set.

Tip: You can use multiple OUTPUT statements to create several OUT= data sets.

Featured in: Example 8 on page 672, Example 9 on page 674, Example 10 on page 676, Example 11 on page 677, and Example 12 on page 680

OUTPUT <OUT=*SAS-data-set*> <*output-statistic-specification(s)*>
<*id-group-specification(s)*> <*maximum-id-specification(s)*>
<*minimum-id-specification(s)*> </ *option(s)*>;

Options

OUT=*SAS-data-set*
names the new output data set. If *SAS-data-set* does not exist, PROC MEANS creates it. If you omit OUT=, the data set is named DATA*n*, where *n* is the smallest integer that makes the name unique.

Default: DATA*n*

Tip: You can use data set options with OUT=.

output-statistic-specification(s)
specifies the statistics to store in the OUT= data set and names one or more variables that contain the statistics. The form of the *output-statistic-specification* is

statistic-keyword<(*variable-list*)>=<*name(s)*>

where

statistic-keyword
specifies which statistic to store in the output data set. The available statistic keywords are

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

By default the statistics in the output data set automatically inherit the analysis variable's format, informat, and label. However, statistics computed for N, NMISS, SUMWGT, USS, CSS, VAR, CV, T, PROBT, SKEWNESS, and KURTOSIS will not inherit the analysis variable's format because this format may be invalid for these statistics (for example, dollar or datetime formats).

Restriction: If you omit *variable* and *name(s)* then PROC MEANS allows the *statistic-keyword* only once in a single OUTPUT statement, unless you also use the AUTONAME option.

Featured in: Example 8 on page 672, Example 9 on page 674, Example 11 on page 677, and Example 12 on page 680

variable-list

specifies the names of one or more numeric analysis variables whose statistics you want to store in the output data set.

Default: all numeric analysis variables

name(s)

specifies one or more names for the variables in output data set that will contain the analysis variable statistics. The first name contains the statistic for the first analysis variable; the second name contains the statistic for the second analysis variable; and so on.

Default: the analysis variable name. If you specify AUTONAME, the default is the combination of the analysis variable name and the *statistic-keyword*.

Interaction: If you specify *variable-list*, PROC MEANS uses the order that you specify the analysis variables to store the statistics in the output data set variables.

Featured in: Example 8 on page 672

Default: If you use the CLASS statement and an OUTPUT statement without an *output-statistic-specification*, the output data set contains five observations for each combination of class variables: the value of N, MIN, MAX, MEAN, and STD. If you use the WEIGHT statement or the WEIGHT option in the VAR statement, the output data set also contains an observation with the sum of weights (SUMWGT) for each combination of class variables.

Tip: Use the AUTONAME option to have PROC MEANS generate unique names for multiple variables and statistics.

id-group-specification

combines the features and extends the ID statement, the IDMIN option in the PROC statement, and the MAXID and MINID options in the OUTPUT statement to create an OUT= data set that identifies multiple extreme values. The form of the *id-group-specification* is

IDGROUP (<MIN | MAX (*variable–list-1*) <…MIN | MAX (*variable–list-n*)>>
<<MISSING> <OBS> <LAST>> OUT <[*n*]>
(*id-variable–list*)=<*name(s)*>)

MIN | MAX(*variable-list*)

specifies the selection criteria to determine the extreme values of one or more input data set variables specified in *variable-list*. Use MIN to determine the minimum extreme value and MAX to determine the maximum extreme value.

When you specify multiple selection variables, the ordering of observations for the selection of *n* extremes is done the same way that PROC SORT sorts data with multiple BY variables. PROC MEANS concatenates the variable values into a single key. The MAX(*variable-list*) selection criterion is similar to using PROC SORT and the DESCENDING option in the BY statement.

Default: If you do not specify MIN or MAX, PROC MEANS uses the observation number as the selection criterion to output observations.

Restriction: If you specify criteria that are contradictory, PROC MEANS only uses the first selection criterion.

Interaction: When multiple observations contains the same extreme values in all the MIN or MAX variables, PROC MEANS uses the observation number to resolve which observation to output. By default, PROC MEANS outputs the first observation to resolve any ties. However, if you specify the LAST option then PROC MEANS outputs the last observation to resolve any ties.

LAST

specifies that the OUT= data set contains values from the last observation. The OUT= data set may contain several observations because in addition to the value of the last observation, PROC MEANS outputs values from the last observation of each subgroup level that is defined by combinations of class variable values.

Interaction: When you specify MIN or MAX and when multiple observations contain the same extreme values, PROC MEANS use the observation number to resolve which observation to output. If you specify LAST, PROC MEANS outputs the last observation to resolve any ties.

MISSING

specifies that missing values be used in selection criteria.

Alias: MISS

OBS

includes an _OBS_ variable in the OUT= data set that contains the number of the observation in the input data set where the extreme value was found.

Interaction: If you use WHERE processing, the value of _OBS_ may not correspond to the location of the observation in the input data set.

Interaction: If you use [*n*] to output multiple extreme values, PROC MEANS creates *n* _OBS_ variables and uses the suffix *n* to create the variable names, where *n* is a sequential integer from 1 to *n*.

[*n*]

specifies the number of extreme values for each variable in *id-variable-list* to include in the OUT= data set. PROC MEANS creates *n* new variables and uses the suffix _*n* to create the variable names, where *n* is a sequential integer from 1 to *n*.

By default, PROC MEANS determines one extreme value for each level of each requested type. If *n* is greater than one, then *n* extremes are output for each level of each type. When *n* is greater than one and you request extreme value selection, the time complexity is $\Theta\ (\ T * N \log_2 n)$ where T is the number of types requested and N is the number of observations in the input data set. By comparison, to group the entire data set, the time complexity is $\Theta\ \ (N \log_2 N)$.

Default: 1

Range: an integer between 1 and 100

Example: To output two minimum extreme values for each variable, use

```
idgroup(min(x) out[2](x y z)=MinX MinY MinZ);
```

The OUT= data set contains the variables MinX_1, MinX_2, MinY_1, MinY_2, MinZ_1, and MinZ_2.

(*id-variable-list*)

identifies one or more input data set variables whose values PROC MEANS includes in the OUT= data set. PROC MEANS determines which observations to output by the selection criteria that you specify (MIN, MAX, and LAST).

name(s)

specifies one or more names for variables in the OUT= data set.

Default: If you omit *name*, PROC MEANS uses the names of variables in the *id-variable-list*.

Tip: Use the AUTONAME option to automatically resolve naming conflicts.

Alias: IDGRP

Requirement: You must specify the MIN|MAX selection criteria first and OUT(*id-variable-list*)= after the suboptions MISSING, OBS, and LAST.

Tip: You can use *id-group-specification* to mimic the behavior of the ID statement and a *maximum-id-specification* or *mimimum-id-specification* in the OUTPUT statement.

Tip: When you want the output data set to contain extreme values along with other id variables, it is more efficient to include them in the *id-variable-list* than to request separate statistics. For example, the statement

```
output idgrp(max(x) out(x a b)= );
```

is more efficient than the statement

```
output idgrp(max(x) out(a b)= ) max(x)=;
```

Featured in: Example 8 on page 672 and Example 12 on page 680

CAUTION:

The IDGROUP syntax allows you to create output variables with the same name. When this happens, only the first variable appears in the output data set. Use the AUTONAME option to automatically resolve these naming conflicts. △

Note: If you specify fewer new variable names than the combination of analysis variables and identification variables then the remaining output variables use the corresponding names of the ID variables as soon as PROC MEANS exhausts the list of new variable names. △

maximum-id-specification(s)

specifies that one or more identification variables be associated with the maximum values of the analysis variables. The form of the *maximum-id-specification* is

MAXID <(*variable-1* <(*id-variable-list-1*)> <*...variable-n* <(*id-variable-list–n*)>>)> = *name(s)*

variable

identifies the numeric analysis variable whose maximum values PROC MEANS determines. PROC MEANS may determine several maximum values for a variable because, in addition to the overall maximum value, subgroup levels, which are defined by combinations of class variables values, also have maximum values.

Tip: If you use an ID statement and omit *variable*, PROC MEANS uses all analysis variables.

id-variable-list

identifies one or more variables whose values identify the observations with the maximum values of the analysis variable.

Default: the ID statement variables

name(s)

specifies the names for new variables that contain the values of the identification variable associated with the maximum value of each analysis variable.

Tip: If you use an ID statement, and omit *variable* and *id-variable*, PROC MEANS associates all ID statement variables with each analysis variable. Thus, for each analysis variable, the number of variables that are created in the output data set equals the number of variables that you specify in the ID statement.

Tip: Use the AUTONAME option to automatically resolve naming conflicts.

Limitation: If multiple observations contain the maximum value within a class level, PROC MEANS saves the value of the ID variable for only the first of those observations in the output data set.

Featured in: Example 11 on page 677

CAUTION:

The MAXID syntax allows you to create output variables with the same name. When this happens, only the first variable appears in the output data set. Use the AUTONAME option to automatically resolve these naming conflicts. △

Note: If you specify fewer new variable names than the combination of analysis variables and identification variables then the remaining output variables use the corresponding names of the ID variables as soon as PROC MEANS exhausts the list of new variable names. △

minid-specification

See the description of maximum-id-specification on page 644. This option behaves in exactly the same way, except that PROC MEANS determines the minimum values instead of the maximum values. The form of the *minid-specification* is

MINID<(*variable-1* <(*id-variable-list-1*)> <*...variable-n* <(*id-variable-list–n*)>>)> = *name(s)*

AUTOLABEL

specifies that PROC MEANS appends the statistic name to the end of the variable label. If an analysis variable has no label, PROC MEANS creates a label by appending the statistic name to the analysis variable name.

Featured in: Example 12 on page 680

AUTONAME
specifies that PROC MEANS creates a unique variable name for an output statistic when you do not explicitly assign the variable name in the OUTPUT statement. This is accomplished by appending the *statistic-keyword* to the end of the input variable name from which the statistic was derived. For example, the statement

```
output min(x)=/autoname;
```

produces the x_Min variable in the output data set.
AUTONAME activates the SAS internal mechanism to automatically resolve conflicts in the variable names in the output data set. Duplicate variables will not generate errors. As a result, the statement

```
output min(x)= min(x)=/autoname;
```

produces two variables, x_Min and x_Min2, in the output data set.

Featured in: Example 12 on page 680

KEEPLEN
specifies that statistics in the output data set inherit the length of the analysis variable that PROC MEANS uses to derive them.

CAUTION:
You permanently lose numeric precision when the length of the analysis variable causes PROC MEANS to truncate or round the value of the statistic. However, the precision of the statistic will match that of the input. △

LEVELS
includes a variable named _LEVEL_ in the output data set. This variable contains a value from 1 to *n* that indicates a unique combination of the values of class variables (the values of _TYPE_ variable).

Main discussion: "Output Data Set" on page 655

Featured in: Example 8 on page 672

NOINHERIT
specifies that the variables in the output data set that contain statistics do not inherit the attributes (label and format) of the analysis variables which are used to derive them.

Tip: By default, the output data set includes an output variable for each analysis variable and for five observations that contain N, MIN, MAX, MEAN, and STDDEV. Unless you specify NOINHERIT, this variable inherits the format of the analysis variable, which may be invalid for the N statistic (for example, datetime formats).

WAYS
includes a variable named _WAY_ in the output data set. This variable contains a value from 1 to the maximum number of class variables that indicates how many class variables PROC MEANS combines to create the TYPE value.

Main discussion: "Output Data Set" on page 655

See also: "WAYS Statement" on page 647

Featured in: Example 8 on page 672

TYPES Statement

Identifies which of the possible combinations of class variables to generate.

Main discussion: "Output Data Set" on page 655

Requirement: CLASS statement

Featured in: Example 2 on page 658, Example 5 on page 665, and Example 12 on page 680

TYPES *request(s)*;

Required Arguments

request(s)

specifies which of the 2^k combinations of class variables PROC MEANS uses to create the types, where k is the number of class variables. A request is composed of one class variable name, several class variable names separated by asterisks, or ().

To request class variable combinations quickly, use a grouping syntax by placing parentheses around several variables and joining other variables or variable combinations. For example, the following statements illustrate grouping syntax:

Request	Equivalent to
`types A*(B C);`	`types A*B A*C;`
`types (A  B)*(C D);`	`types A*C A*D B*C B*D;`
`types (A  B C)*D;`	`types A*D B*D C*D;`

Interaction The CLASSDATA= option places constraints on the NWAY type. PROC MEANS generates all other types as if derived from the resulting NWAY type.

Tip: Use ( )to request the overall total (_TYPE_=0).

Tip: If you do not need all types in the output data set, use the TYPES statement to specify specific subtypes rather than applying a WHERE clause to the data set. This saves time and space.

VAR Statement

Identifies the analysis variables and their order in the output.

Default: If you omit the VAR statement, PROC MEANS analyzes all numeric variables that are not listed in the other statements. When all variables are character variables, PROC MEANS produces a simple count of observations.

Tip: You can use multiple VAR statements.

See also: Chapter 36, "The SUMMARY Procedure," on page 1149

Featured in: Example 1 on page 657

VAR *variable(s)* <**/** WEIGHT=*weight-variable*>;

Required Arguments

variable(s)
identifies the analysis variables and specifies their order in the results.

Option

WEIGHT=*weight-variable*
specifies a numeric variable whose values weight the values of the variables that are specified in the VAR statement. The variable does not have to be an integer. If the value of the weight variable is

Weight value...	PROC MEANS...
0	counts the observation in the total number of observations
less than 0	converts the value to zero and counts the observation in the total number of observations
missing	excludes the observation

To exclude observations that contain negative and zero weights from the analysis, use EXCLNPWGT. Note that most SAS/STAT procedures, such as PROC GLM, exclude negative and zero weights by default.

The weight variable does not change how the procedure determines the range, extreme values, or number of missing values.

Restriction: To compute weighted quantiles, use QMETHOD=OS in the PROC statement.

Restriction: Skewness and kurtosis are not available with the WEIGHT option.

Tip: When you use the WEIGHT option, consider which value of the VARDEF= option is appropriate. See the discussion of VARDEF= on page 633.

Tip: Use the WEIGHT option in multiple VAR statements to specify different weights for the analysis variables.

Note: Prior to Version 7 of the SAS System, the procedure did not exclude the observations with missing weights from the count of observations. △

WAYS Statement

Specifies the number of ways to make unique combinations of class variables.

Tip: Use the TYPES statement to specify additional combinations of class variables.

Featured in: Example 6 on page 668

WAYS *list*;

Required Arguments

list

specifies one or more integers that define the number of class variables to combine to form all the unique combinations of class variables. For example, you can specify 2 for all possible pairs and 3 for all possible triples. The *list* can be specified in the following ways:

m

m1 m2 … mn

m1,m2,…,mn

m TO *n* <BY *increment*>

m1,m2, TO *m3* <BY *increment*>, *m4*

Range: 0 to maximum number of class variables

Example: To create the two way types for the classification variables A, C, and C use

```
class A B C ;
ways 2;
```

This is equilavent to specifying a*b, a*c, and b*c in the TYPES statement.

See also: WAYS option on page 645

WEIGHT Statement

Specifies weights for observations in the statistical calculations.

See also: For information on how to calculate weighted statistics and for an example that uses the WEIGHT statement, see “WEIGHT” on page 73

WEIGHT *variable*;

Required Arguments

variable

specifies a numeric variable whose values weight the values of the analysis variables. The values of the variable do not have to be integers. If the value of the weight variable is

Weight value…	PROC MEANS…
0	counts the observation in the total number of observations
less than 0	converts the value to zero and counts the observation in the total number of observations
missing	excludes the observation

To exclude observations that contain negative and zero weights from the analysis, use EXCLNPWGT. Note that most SAS/STAT procedures, such as PROC GLM, exclude negative and zero weights by default.

Restriction: To compute weighted quantiles, use QMETHOD=OS in the PROC statement.

Restriction: Skewness and kurtosis are not available with the WEIGHT statement.

Interaction: If you use the WEIGHT= option in a VAR statement to specify a weight variable, PROC MEANS uses this variable instead to weight those VAR statement variables.

Tip: When you use the WEIGHT statement, consider which value of the VARDEF= option is appropriate. See the discussion of VARDEF= on page 633 and the calculation of weighted statistics in "Keywords and Formulas" on page 1458 for more information.

Note: Prior to Version 7 of the SAS System, the procedure did not exclude the observations with missing weights from the count of observations. △

Concepts

Using Class Variables

The TYPES statement controls which of the available class variables PROC MEANS uses to subgroup the data. The unique combinations of these active class variable values that occur together in any single observation of the input data set determine the data subgroups. Each subgroup that PROC MEANS generates for a given type is called a *level* of that type. Note, for all types the inactive class variables can still affect the total observation count of the rejection of observations with missing values.

When you use a WAYS statement, PROC MEANS generates types that correspond to every possible unique combination of *n* class variables chosen from the complete set of class variables. For example

```
proc means;
 class a b c d e;
 ways 2 3;
 run;
```

is equivalent to

```
proc means;
 class a b c d e;
 types a*b a*c a*d a*e b*c b*d b*e c*d c*e d*e
       a*b*c a*b*d a*b*e a*c*d a*c*e a*d*e
       b*c*d b*c*e c*d*e;
 run;
```

If you omit the TYPES statement and the WAYS statement, PROC MEANS uses all class variables to subgroup the data (the NWAY type) for displayed output and computes all types (2^k) for the output data set.

Ordering the Class Values

PROC MEANS determines the order of each class variable in any type by examining the order of that class variable in the corresponding one-way type. You see the effect of this behavior in the options ORDER=DATA or ORDER=FREQ. When PROC MEANS subdivides the input data set into subsets, the classification process does not apply the options ORDER=DATA or ORDER=FREQ independently for each subgroup. Instead, one frequency and data order is established for all output based on an nonsubdivided view of the entire data set. For example, consider the following statements:

```
data pets;
 input Pet $ Gender $;
 datalines;
dog  m
dog  f
dog  f
dog  f
cat  m
cat  m
cat  f
;

proc means data=pets order=freq;
   class pet gender;
run;
```

The statements produce this output.

```
                        The SAS System                              1

                    The MEANS Procedure

                                       N
               Pet         Gender      Obs
               ---------------------------
               dog         f             3

                           m             1

               cat         f             1

                           m             2
               ---------------------------
```

In the example, PROC MEANS does not list male cats before female cats. Instead, it determines the order of gender for all types over the entire data set. PROC MEANS found more observations for female pets (f=4, m=3).

Computational Resources

PROC MEANS employs the same memory allocation scheme across all host environments. When class variables are involved, PROC MEANS must keep a copy of each unique value of each class variable in memory. You estimate the memory requirements to group the class variable by calculating

$$Nc_1\left(Lc_1 + K\right) + Nc_2\left(Lc_2 + K\right) + \ldots + Nc_n\left(Lc_n + K\right)$$

where

Nc_i	is the number of unique values for the class variable
Lc_i	is the combined unformatted and formatted length of c_i

K	is some constant on the order of 32 bytes (64 for 64-bit architectures).

When you use the GROUPINTERNAL option in the CLASS statement, Lc_i is simply the unformatted length of c_i.

Each unique combination of class variables, $c1_i \ \ c2_j$, for a given type forms a level in that type (see "TYPES Statement" on page 646). You can estimate the maximum potential space requirements for all levels of a given type, when all combinations actually exist in the data (a complete type), by calculating

$$W * Nc_1 * Nc_2 * \ ... \ * Nc_n$$

where

W	is a constant based on the number of variables analyzed and the number of statistics calculated (unless you request QMETHOD=OS to compute the quantiles).
$Nc_1...Nc_n$	are the number of unique levels for the active class variables of the given type.

Clearly, the memory requirements of the levels overwhelm those of the class variables. For this reason, PROC MEANS may open one or more utility files and write the levels of one or more types to disk. These types are either the primary types that PROC MEANS built during the input data scan or the derived types.

If PROC MEANS must write partially complete primary types to disk while it processes input data, then one or more merge passes may be required to combine type levels in memory with those on disk. In addition, if you use an order other than DATA for any class variable, PROC MEANS groups the completed type on disk. For this reason, the peak disk space requirements can be more than twice the memory requirements for a given type.

When PROC MEANS uses a temporary work file, you will receive the following note in the SAS log:

```
Processing on disk occurred during summarization.
Peak disk usage was approximately nnn Mbytes.
Adjusting SUMSIZE may improve performance.
```

In most cases processing ends normally.

When you specify class variables in a CLASS statement, the amount of data-dependent memory that PROC MEANS uses before it writes to a utility file is controlled by the SAS system option and PROC option SUMSIZE=. Like the system option SORTSIZE=, SUMSIZE= sets the memory threshold where disk-based operations begin. For best results, set SUMSIZE= to less than the amount of real memory that is likely to be available for the task. For efficiency reasons, PROC MEANS may internally round up the value of SUMSIZE=. SUMSIZE= has no effect unless you specify class variables.

If PROC MEANS reports that there is insufficient memory, increase SUMSIZE=. A SUMSIZE= value greater than MEMSIZE= will have no effect. Therefore, you may also need to increase MEMSIZE=. If PROC MEANS reports insufficient disk space, increase the WORK space allocation. See the SAS documentation for your operating environment for more information on how to adjust your computation resource parameters.

Statistical Computations

PROC MEANS uses single-pass algorithms to compute the moment statistics (such as mean, variance, skewness, and kurtosis). See “Keywords and Formulas” on page 1458 for the statistical formulas.

The computational details for confidence limits, hypothesis test statistics, and quantile statistics follow.

Confidence Limits

With the keywords CLM, LCLM, and UCLM, you can compute confidence limits for the mean. A *confidence limit* is a range, constructed around the value of a sample statistic, that contains the corresponding true population value with given probability (ALPHA=) in repeated sampling.

A two-sided $100\,(1-\alpha)\%$ confidence interval for the mean has upper and lower limits

$$\overline{x} \pm t_{(1-\alpha/2;n-1)}\frac{s}{\sqrt{n}}$$

where s is $\sqrt{\frac{1}{n-1}\sum\left(x_i-\overline{x}\right)^2}$ and $t_{(1-\alpha/2;n-1)}$ is the $(1-\alpha/2)$ critical value of the Student’s *t* statistics with $n-1$ degrees of freedom.

A one-sided $100\,(1-\alpha)\%$ confidence interval is computed as

$$\begin{array}{lc} \overline{x} + t_{(1-\alpha;n-1)}\frac{s}{\sqrt{n}} & \text{(upper)} \\ \overline{x} - t_{(1-\alpha;n-1)}\frac{s}{\sqrt{n}} & \text{(lower)} \end{array}$$

A two-sided $100\,(1-\alpha)\%$ confidence interval for the standard deviation has lower and upper limits

$$s\sqrt{\frac{n-1}{\chi^2_{(1-\alpha/2;n-1)}}},\ s\sqrt{\frac{n-1}{\chi^2_{(\alpha/2;n-1)}}}$$

where $\chi^2_{(1-\alpha/2;n-1)}$ and $\chi^2_{(\alpha/2;n-1)}$ are the $(1-\alpha/2)$ and $\alpha/2$ critical values of the chi-square statistic with $n-1$ degrees of freedom. A one-sided $100\,(1-\alpha)\%$ confidence interval is computed by replacing $\alpha/2$ with α.

A $100\,(1-\alpha)\%$ confidence interval for the variance has upper and lower limits that are equal to the squares of the corresponding upper and lower limits for the standard deviation.

When you use the WEIGHT statement or WEIGHT= in a VAR statement and the default value of VARDEF=, which is DF, the $100\,(1-\alpha)\%$ confidence interval for the weighted mean has upper and lower limits

$$\overline{y}_w \pm t_{(1-\alpha/2)} \frac{s_w}{\sqrt{\sum_{i=1}^{n} w_i}}$$

where $\overline{y}_w$ is the weighted mean, s_w is the weighted standard deviation, w_i is the weight for ith observation, and $t_{(1-\alpha/2)}$ is the $(1-\alpha/2)$ critical value for the Student's *t* distribution with $n-1$ degrees of freedom.

Student's *t* Test

PROC MEANS calculates the *t* statistic as

$$t = \frac{\overline{x} - \mu_0}{s/\sqrt{n}}$$

where $\overline{x}$ is the sample mean, n is the number of nonmissing values for a variable, and s is the sample standard deviation. Under the null hypothesis, the population mean equals μ_0. When the data values are approximately normally distributed, the probability under the null hypothesis of a *t* statistic as extreme, or more extreme, than the observed value (the *p*–value) is obtained from the *t* distribution with $n-1$ degrees of freedom. For large n, the *t* statistic is asymptotically equivalent to a *z* test.

When you use the WEIGHT statement or WEIGHT= in a VAR statement and the default value of VARDEF=, which is DF, the Student's *t* statistic is calculated as

$$t_w = \frac{\overline{y}_w - \mu_0}{s_w/\sqrt{\sum_{i=1}^{n} w_i}}$$

where $\overline{y}_w$ is the weighted mean, s_w is the weighted standard deviation, and w_i is the weight for ith observation. The t_w statistic is treated as having a Student's *t* distribution with $n-1$ degrees of freedom. If you specify the EXCLNPWGT option in the PROC statement, n is the number of nonmissing observations when the value of the WEIGHT variable is positive. By default, n is the number of nonmissing observations for the WEIGHT variable.

Quantiles

The options QMETHOD=, QNTLDEF=, and QMARKERS= determine how PROC MEANS calculates quantiles. QNTLDEF= deals with the mathematical definition of a quantile. See "Calculating Percentiles" on page 1404. QMETHOD= deals with the mechanics of how PROC MEANS handles the input data. The two methods are

OS
: reads all data into memory and sorts it by unique value.

P2
: accumulates all data into a fixed sample size that is used to approximate the quantile.

If data set A has 100 unique values for a numeric variable X and data set B has 1000 unique values for numeric variable X then OMETHOD=OS for data set B will take 10 times as much memory as it does for data set A. If QMETHOD=P2, both data sets A and B will require the same memory space to generate quantiles.

The QMETHOD=P2 technique is based on the piecewise-parabolic (P^2) algorithm invented by Jain and Chlamtac (1985). P^2 is a one-pass algorithm to determine quantiles for a large data set. It requires a fixed amount of memory for each variable for each level within the type. However, using simulation studies, reliable estimations of some quantiles (P1, P5, P95, P99) may not be possible for some data sets such as those with heavily tailed or skewed distributions.

If the number of observations is less than the QMARKERS= value, QMETHOD=P2 produces the same results as QMETHOD=OS when QNTLDEF=5. To compute weighted quantiles, you must use QMETHOD=OS.

Results

Missing Values

PROC MEANS excludes missing values for the analysis variables before calculating statistics. Each analysis variable is treated individually; a missing value for an observation in one variable does not affect the calculations for other variables. The statements handle missing values as follows:

- If a class variable has a missing value for an observation, PROC MEANS excludes that observation from the analysis unless you use the MISSING option in the PROC statement or the CLASS statement.
- If a BY or an ID variable value is missing, PROC MEANS treats it like any other BY or ID variable value. The missing values form a separate BY group.
- If a FREQ variable value is missing or nonpositive, PROC MEANS excludes the observation from the analysis.
- If a WEIGHT variable value is missing, PROC MEANS excludes the observation from the analysis.

PROC MEANS tabulates the number of the missing values. Before the number of missing values are tabulated, PROC MEANS excludes observations with frequencies that are nonpositive when you use the FREQ statement and observations with weights that are missing or nonpositive (when you use the EXCLNPWGT option) when you use the WEIGHT statement. To report this information in the procedure output use the NMISS statistical keyword in the PROC statement.

Column Width for the Output

You control the column width for the displayed statistics with the FW= option in the PROC statement. Unless you assign a format to a numeric class or an ID variable, PROC MEANS uses the value of the FW= option. When you assign a format to a numeric class or an ID variable, PROC MEANS determines the column width directly from the format. If you use the PRELOADFMT option in the CLASS statement, PROC MEANS determines the column width for a class variable from the assigned format.

The N Obs Statistic

By default when you use a CLASS statement, PROC MEANS displays an additional statistic called N Obs. This statistic reports the total number of observations or the sum of the observations of the FREQ variable that PROC MEANS processes for each class level. PROC MEANS may omit observations from this total due to missing values in one or more class variables or due to the effect of the EXCLUSIVE option when you use it with the PRELOADFMT option or the CLASSDATA= option. Because of this and the exclusion of observations when the WEIGHT variable contains missing values, there is not always a direct relationship between NObs, N, and NMISS.

In the output data set, the value of N Obs is stored in the _FREQ_ variable. Use the NONOBS option in the PROC statement to suppress this information in the displayed output.

Output Data Set

PROC MEANS can create one or more output data sets. The procedure does not print the output data set. Use PROC PRINT, PROC REPORT, or another SAS reporting tool to display the output data set.

Note: By default the statistics in the output data set automatically inherit the analysis variable's format and label. However, statistics computed for N, NMISS, SUMWGT, USS, CSS, VAR, CV, T, PROBT, SKEWNESS, and KURTOSIS do not inherit the analysis variable's format because this format may be invalid for these statistics. Use the NOINHERIT option in the OUTPUT statement to prevent the other statistics from inheriting the format and label attributes. △

The output data set can contain these variables:

- the variables specified in the BY statement.
- the variables specified in the ID statement.
- the variables specified in the CLASS statement.
- the variable _TYPE_ that contains information about the class variables. By default _TYPE_ is a numeric variable. If you specify CHARTYPE in the PROC statement, _TYPE_ is a character variable. When you use more than 32 class variables, _TYPE_ is automatically a character variable.
- the variable _FREQ_ that contains the number of observations that a given output level represents.
- the variables requested in the OUTPUT statement that contain the output statistics and extreme values.
- the variable _STAT_ that contains the names of the default statistics if you omit statistic keywords.
- the variable _LEVEL_ if you specify the LEVEL option.
- the variable _WAY_ if you specify the WAYS option.

The value of _TYPE_ indicates which combination of the class variables PROC MEANS uses to compute the statistics. The character value of _TYPE_ is a series of zeros and ones, where each value of one indicates an active class variable in the type. For example, with three class variables, PROC MEANS represents type 1 as 001, type 5 as 101, and so on.

Usually, the output data set contains one observation per level per type. However, if you omit statistical keywords in the OUTPUT statement, the output data set contains five observations per level (six if you specify a WEIGHT variable). Therefore, the total number of observations in the output data set is equal to the sum of the levels for all the types you request multiplied by 1, 5, or 6, whichever is applicable.

If you omit the CLASS statement (_TYPE_ = 0), there is always exactly one level of output per BY-group. If you use a CLASS statement, then the number of levels for each type you request has an upper bound equal to the number of observations in the input data set. By default, PROC MEANS generates all possible types. In this case the total number of levels for each BY-group has an upper bound equal to

$$m \cdot \left(2^k - 1\right) \cdot n + 1$$

where k is the number of class variables and n is the number of observations for the given BY group in the input data set and $m$is 1, 5, or 6.

PROC MEANS determines the actual number of levels for a given type from the number of unique combinations of each active class variable. A single level is composed of all input observations whose formatted class values match.

Figure 24.1 on page 656 shows the values of _TYPE_ and the number of observations in the data set when you specify one, two, and three class variables.

Figure 24.1 The Effect of Class Variables on the OUTPUT Data Set

three CLASS variables
two CLASS variables
one CLASS variable

C	B	A	_WAY_	_TYPE_	Subgroup defined by	Number of observations of this _TYPE_ and _WAY_ in the data set	Total number of observations in the data set
0	0	0	0	0	Total	1	
0	0	1	1	1	A	a	1+a
0	1	0	1	2	B	b	
0	1	1	2	3	A*B	a*b	1+a+b+a*b
1	0	0	1	4	C	c	
1	0	1	2	5	A*C	a*c	
1	1	0	2	6	B*C	b*c	1+a+b+a*b+c
1	1	1	3	7	A*B*C	a*b*c	+a*c+b*c+a*b*c
Character binary equivalent of _TYPE_ (CHARTYPE option)					A ,B ,C=CLASS variables	a, b, c,=number of levels of A, B, C, respectively	

Examples

Example 1: Computing Specific Descriptive Statistics

Procedure features:

PROC MEANS statement options:
- statistic keywords
- FW=

VAR statement

This example
- □ specifies the analysis variables
- □ computes the statistics for the specified keywords and displays them in order
- □ specifies the field width of the statistics.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

The data set CAKE contains each participant's last name and age, score for presentation, score for taste, cake flavor, and number of cake layers for a cake-baking contest. The number of cake layers is missing for two observations. The cake flavor is missing for another observation.

```
data cake;
   input LastName $ 1-12 Age 13-14 PresentScore 16-17
         TasteScore 19-20 Flavor $ 23-32 Layers 34 ;
   datalines;
Orlando     27 93 80  Vanilla    1
Ramey       32 84 72  Rum        2
Goldston    46 68 75  Vanilla    1
Roe         38 79 73  Vanilla    2
Larsen      23 77 84  Chocolate  .
Davis       51 86 91  Spice      3
Strickland  19 82 79  Chocolate  1
Nguyen      57 77 84  Vanilla    .
Hildenbrand 33 81 83  Chocolate  1
Byron       62 72 87  Vanilla    2
Sanders     26 56 79  Chocolate  1
Jaeger      43 66 74             1
Davis       28 69 75  Chocolate  2
Conrad      69 85 94  Vanilla    1
Walters     55 67 72  Chocolate  2
Rossburger  28 78 81  Spice      2
```

```
Matthew     42 81 92  Chocolate  2
Becker      36 62 83  Spice      2
Anderson    27 87 85  Chocolate  1
Merritt     62 73 84  Chocolate  1
;
```

The statistic keywords specify the statistics and their order in the output. FW= uses a field width of eight to display the statistics.

```
proc means data=cake n mean max min range std fw=8;
```

The VAR statement specifies the analysis variables and their order in the output.

```
   var PresentScore TasteScore;
   title 'Summary of Presentation and Taste Scores';
run;
```

Output

PROC MEANS lists PresentScore first because this is the first variable specified in the VAR statement. A field width of eight truncates the statistics to four decimal places.

```
                   Summary of Presentation and Taste Scores                  1

                            The MEANS Procedure

Variable         N        Mean     Maximum     Minimum       Range     Std Dev
------------------------------------------------------------------------------
PresentScore    20     76.1500     93.0000     56.0000     37.0000      9.3768
TasteScore      20     81.3500     94.0000     72.0000     22.0000      6.6116
------------------------------------------------------------------------------
```

Example 2: Computing Descriptive Statistics with Class Variables

Procedure features:

PROC MEANS statement option:

MAXDEC=

CLASS statement

TYPES statement

This example

- □ analyzes the data for the two-way combination of class variables and across all observations

- limits the number of decimal places for the displayed statistics.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

The data set GRADE contains each student's last name, gender, status of either undergraduate (1) or graduate (2), expected year of graduation, class section (A or B), final exam score, and final grade for the course.

```
data grade;
   input Name $ 1-8 Gender $ 11 Status $13 Year $ 15-16
         Section $ 18 Score 20-21 FinalGrade 23-24;
   datalines;
Abbott    F 2 97 A 90 87
Branford  M 1 98 A 92 97
Crandell  M 2 98 B 81 71
Dennison  M 1 97 A 85 72
Edgar     F 1 98 B 89 80
Faust     M 1 97 B 78 73
Greeley   F 2 97 A 82 91
Hart      F 1 98 B 84 80
Isley     M 2 97 A 88 86
Jasper    M 1 97 B 91 93
;
```

MAXDEC= limits the displayed statistics to three decimal places.

```
proc means data=grade maxdec=3;
```

The CLASS statement separates the analysis by values of Status and Year.

```
   class Status Year;
```

The TYPES statement requests the analysis across all the observations and the two-way combination of Status and Year, which results in four levels.

```
   types () status*year;
```

The VAR statement specifies the analysis variable.

```
   var Score;
title 'Final Exam Grades for Student Status and Year of Graduation';
run;
```

Output

PROC MEANS displays the default statistics for all the observations (_TYPE_=0) and the four class levels of the Status and Year combination (Status=1, Year=97; Status=1, Year=98; Status=2, Year=97; Status=2, Year=98).

```
        Final Exam Grades for Student Status and Year of Graduation         1

                          The MEANS Procedure

                       Analysis Variable : Score

   N
  Obs     N            Mean         Std Dev         Minimum         Maximum
  -------------------------------------------------------------------------
   10    10          86.000           4.714          78.000          92.000
  -------------------------------------------------------------------------


                       Analysis Variable : Score

                 N
Status  Year  Obs   N          Mean       Std Dev       Minimum       Maximum
-----------------------------------------------------------------------------
1       97      3   3        84.667         6.506        78.000        91.000

        98      3   3        88.333         4.041        84.000        92.000

2       97      3   3        86.667         4.163        82.000        90.000

        98      1   1        81.000          .           81.000        81.000
-----------------------------------------------------------------------------
```

Example 3: Using the BY Statement with Class Variables

Procedure features:

PROC MEANS statement option:

statistic keywords

BY statement

CLASS statement

Other features:

SORT procedure

Data set: GRADE on page 659

This example

- separates the analysis for the combination of class variables within BY values
- shows the sort order requirement for the BY statement
- calculates the minimum, maximum, and median.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

PROC SORT sorts the observations by the variable Section. This is required to use Section as a BY variable in the PROC MEANS step.

```
proc sort data=Grade out=GradeBySection;
   by section;
run;
```

The statistic keywords specify the statistics and their order in the output.

```
proc means data=GradeBySection min max median;
```

The BY statement produces a separate analysis for each value of Section.

```
   by section;
```

The CLASS statement separates the analysis by the values of Status and Year.

```
   class Status Year;
```

The VAR statement specifies the analysis variable.

```
   var Score;
title1 'Final Exam Scores for Student Status and Year of Graduation';
title2 ' Within Each Section';
run;
```

Output

```
         Final Exam Scores for Student Status and Year of Graduation         1
                              Within Each Section

---------------------------------- Section=A -----------------------------------

                              The MEANS Procedure

                          Analysis Variable : Score

                        N
     Status    Year    Obs         Minimum         Maximum          Median
     ---------------------------------------------------------------------
     1         97        1      85.0000000      85.0000000      85.0000000

               98        1      92.0000000      92.0000000      92.0000000

     2         97        3      82.0000000      90.0000000      88.0000000
     ---------------------------------------------------------------------


---------------------------------- Section=B -----------------------------------

                          Analysis Variable : Score

                        N
     Status    Year    Obs         Minimum         Maximum          Median
     ---------------------------------------------------------------------
     1         97        2      78.0000000      91.0000000      84.5000000

               98        2      84.0000000      89.0000000      86.5000000

     2         98        1      81.0000000      81.0000000      81.0000000
     ---------------------------------------------------------------------
```

Example 4: Using a CLASSDATA= Data Set with Class Variables

Procedure features:

PROC MEANS statement options:

- CLASSDATA=
- EXCLUSIVE
- FW=
- MAXDEC=
- PRINTALLTYPES

CLASS statement

Data set: CAKE on page 657

This example

- □ specifies the field width and decimal places of the displayed statistics
- □ uses only the values in CLASSDATA= data set as the levels of the combinations of class variables
- □ calculates the range, median, minimum, and maximum
- □ displays all combinations of the class variables in the analysis.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

The data set CAKETYPE contains the cake flavors and number of layers that must occur in the PROC MEANS output.

```
data caketype;
   input Flavor $ 1-10  Layers 12;
   datalines;
Vanilla    1
Vanilla    2
Vanilla    3
Chocolate  1
Chocolate  2
Chocolate  3
;
```

FW= uses a field width of seven and MAXDEC= use zero decimal places to display the statistics. CLASSDATA= and EXCLUSIVE restrict the class levels to the values in the CAKETYPE data set. PRINTALLTYPES displays all combinations of class variables in the output.

```
proc means data=cake range median min max fw=7 maxdec=0
           classdata=caketype exclusive printalltypes ;
```

The CLASS statement separates the analysis by the values of Flavor and Layers.

```
   class flavor layers;
```

The VAR statement specifies the analysis variable.

```
   var TasteScore;
   Title 'Taste Score For Number of Layers and Cake Flavor';
run;
```

Output

PROC MEANS calculates statistics for the 13 chocolate and vanilla cakes. Because the CLASSDATA= data set contains 3 as the value of Layers, PROC MEANS uses 3 as a class value even though the frequency is zero.

```
                 Taste Score For Number of Layers and Cake Flavor               1

                              The MEANS Procedure

                        Analysis Variable : TasteScore

                  N
                 Obs      Range     Median    Minimum    Maximum
                 -----------------------------------------------
                  13         22         80         72         94
                 -----------------------------------------------


                         Analysis Variable : TasteScore

                       N
            Layers    Obs      Range     Median    Minimum    Maximum
           ----------------------------------------------------------
                 1      8         19         82         75         94

                 2      5         20         75         72         92

                 3      0          .          .          .          .
           ----------------------------------------------------------


                        Analysis Variable : TasteScore

                        N
         Flavor        Obs      Range     Median    Minimum    Maximum
         -------------------------------------------------------------
         Chocolate       8         20         81         72         92

         Vanilla         5         21         80         73         94
         -------------------------------------------------------------


                        Analysis Variable : TasteScore

                              N
    Flavor         Layers    Obs      Range     Median    Minimum    Maximum
    ------------------------------------------------------------------------
    Chocolate           1      5          6         83         79         85

                        2      3         20         75         72         92

                        3      0          .          .          .          .

    Vanilla             1      3         19         80         75         94

                        2      2         14         80         73         87

                        3      0          .          .          .          .
    ------------------------------------------------------------------------
```

Example 5: Using Multi-label Value Formats with Class Variables

Procedure features:

PROC MEANS statement options:

- statistic keywords
- FW=
- NONOBS

CLASS statement options:

- MLF
- ORDER=

TYPES statement

Other features

FORMAT procedure

FORMAT statement

Data set: CAKE on page 657

This example

- □ computes the statistics for the specified keywords and displays them in order
- □ specifies the field width of the statistics
- □ suppresses the column with the total number of observations
- □ analyzes the data for the one-way combination of cake flavor and the two-way combination of cake flavor and participant's age
- □ assigns user-defined formats to the class variables
- □ uses multi-label formats as the levels of class variables
- □ orders the levels of the cake flavors by the descending frequency count and orders the levels of age by the ascending formatted values.

Program

```
options nodate pageno=1 linesize=80 pagesize=64;
```

PROC FORMAT creates user-defined formats to categorize the cake flavors and age of the participants. MULTILABEL allows overlapping ranges for age.

```
proc format;
   value $flvrfmt
                'Chocolate'='Chocolate'
                'Vanilla'='Vanilla'
                'Rum','Spice'='Other Flavor';
   value agefmt (multilabel)
                  15 - 29='below 30 years'
                  30 - 50='between 30 and 50'
                  51 - high='over 50 years'
                  15 - 19='15 to 19'
                  20 - 25='20 to 25'
                  25 - 39='25 to 39'
                  40 - 55='40 to 55'
                  56 - high='56 and above';
run;
```

FW= uses a field width of six to display the statistics. The statistic keywords specify the statistics and their order in the output. NONOBS suppresses the N Obs column.

```
proc means data=cake fw=6 n min max median nonobs;
```

The CLASS statements separate the analysis by values of Flavor and Age. ORDER=FREQ orders the levels of Flavor by descending frequency count. ORDER=FMT orders the levels of Age by ascending formatted values. MLF specifies that multi-label value formats be used for Age.

```
   class flavor/order=freq;
   class  age /mlf order=fmt;
```

The TYPES statement requests the analysis for the one-way combination of Flavor and the two-way combination of Flavor and Age.

```
    types flavor flavor*age;
```

The VAR statement specifies the analysis variable.

```
   var TasteScore;
```

The FORMAT statement assigns user-defined formats to Age and Flavor for this analysis.

```
   format age agefmt. flavor $flvrfmt.;
   title 'Taste Score for Cake Flavors and Participant''s Age';
run;
```

Output

The one-way combination of class variables appears before the two-way combination. A field width of six truncates the statistics to four decimal places. For the two-way combination of Age and Flavor, the total number of observations is greater than the one-way combination of Flavor. This is because of the multi-label format for age, which maps one internal value to more than one formatted value.

The order of the levels of Flavor is based on the frequency count for each level. The order of the levels of Age is based on the order of the user-defined formats.

```
                Taste Score for Cake Flavors and Participant's Age              1

                               The MEANS Procedure

                         Analysis Variable : TasteScore

                 Flavor           N       Min       Max    Median
                 ------------------------------------------------
                 Chocolate        9     72.00     92.00     83.00

                 Vanilla          6     73.00     94.00     82.00

                 Other Flavor     4     72.00     91.00     82.00
                 ------------------------------------------------


                         Analysis Variable : TasteScore

      Flavor          Age                   N       Min       Max    Median
      ---------------------------------------------------------------------
      Chocolate       15 to 19              1     79.00     79.00     79.00

                      20 to 25              1     84.00     84.00     84.00

                      25 to 39              4     75.00     85.00     81.00

                      40 to 55              2     72.00     92.00     82.00

                      56 and above          1     84.00     84.00     84.00

                      below 30 years        5     75.00     85.00     79.00

                      between 30 and 50     2     83.00     92.00     87.50

                      over 50 years         2     72.00     84.00     78.00

      Vanilla         25 to 39              2     73.00     80.00     76.50

                      40 to 55              1     75.00     75.00     75.00

                      56 and above          3     84.00     94.00     87.00

                      below 30 years        1     80.00     80.00     80.00

                      between 30 and 50     2     73.00     75.00     74.00

                      over 50 years         3     84.00     94.00     87.00

      Other Flavor    25 to 39              3     72.00     83.00     81.00

                      40 to 55              1     91.00     91.00     91.00

                      below 30 years        1     81.00     81.00     81.00

                      between 30 and 50     2     72.00     83.00     77.50

                      over 50 years         1     91.00     91.00     91.00
      ---------------------------------------------------------------------
```

Example 6: Using Preloaded Formats with Class Variables

Procedure features:

PROC MEANS statement options:

- COMPLETETYPES
- FW=
- MISSING
- NONOBS

CLASS statement options:

- EXCLUSIVE
- ORDER=
- PRELOADFMT

WAYS statement

Other features

FORMAT procedure

FORMAT statement

Data set: CAKE on page 657

This example

- □ specifies the field width of the statistics
- □ suppresses the column with the total number of observations
- □ includes all possible combinations of class variables values in the analysis even if the frequency is zero
- □ considers missing values as valid class levels
- □ analyzes the one-way and two-way combinations of class variables
- □ assigns user-defined formats to the class variables
- □ uses only the preloaded range of user-defined formats as the levels of class variables
- □ orders the results by the value of the formatted data.

Program

```
options nodate pageno=1 linesize=80 pagesize=64;
```

PROC FORMAT creates user-defined formats to categorize the number of cake layers and the cake flavors. NOTSORTED keeps $FLVRFMT unsorted to preserve the original order of the format values.

```
proc format;
   value layerfmt 1='single layer'
                  2-3='multi-layer'
                  .='unknown';
   value $flvrfmt (notsorted)
                  'Vanilla'='Vanilla'
                  'Orange','Lemon'='Citrus'
                  'Spice'='Spice'
                  'Rum','Mint','Almond'='Other Flavor';
run;
```

FW= uses a field width of seven to display the statistics. COMPLETETYPES includes class levels with a frequency of zero. MISSING considers missing values valid values for all class variables. NONOBS suppresses the N Obs column.

```
proc means data=cake fw=7 completetypes missing nonobs;
```

The CLASS statement separates the analysis by values of Flavor and Layers. PRELOADFMT and EXCLUSIVE restrict the levels to the preloaded values of the user-defined formats. ORDER=DATA orders the levels of Flavor and Layer by formatted data values.

```
   class flavor layers/preloadfmt exclusive order=data;
```

The WAYS statement requests one-way and two–way combinations of class variables.

```
   ways 1 2;
```

The VAR statement specifies the analysis variable.

```
   var TasteScore;
```

The FORMAT statement assigns user-defined formats to Flavor and Layers for this analysis.

```
   format layers layerfmt. flavor $flvrfmt.;
   title 'Taste Score For Number of Layers and Cake Flavors';
run;
```

Output

The one-way combination of class variables appears before the two-way combination. PROC MEANS just reports the level values that are listed in the preloaded range of user-defined formats even when the frequency of observations is zero (i.e., citrus). PROC MEANS rejects entire observations based on the exclusion of any single class value in a given observation. Therefore, when the number of layers is unknown, statistics are calculated for only one observation. The other observation is excluded because the flavor chocolate was not included in the preloaded user-defined format for Flavor.

The order of the levels is based on the order of the user-defined formats. PROC FORMAT automatically sorted the Layers format and did not sort the Flavor format.

```
                 Taste Score For Number of Layers and Cake Flavors              1

                                 The MEANS Procedure

                           Analysis Variable : TasteScore

          Layers           N       Mean    Std Dev    Minimum    Maximum
          -------------------------------------------------------------
          unknown          1     84.000          .     84.000     84.000

          single layer     3     83.000      9.849     75.000     94.000

          multi-layer      6     81.167      7.548     72.000     91.000
          -------------------------------------------------------------


                           Analysis Variable : TasteScore

          Flavor           N       Mean    Std Dev    Minimum    Maximum
          -------------------------------------------------------------
          Vanilla          6     82.167      7.834     73.000     94.000

          Citrus           0          .          .          .          .

          Spice            3     85.000      5.292     81.000     91.000

          Other Flavor     1     72.000          .     72.000     72.000
          -------------------------------------------------------------


                           Analysis Variable : TasteScore

  Flavor           Layers           N       Mean    Std Dev    Minimum    Maximum
  -----------------------------------------------------------------------------
  Vanilla          unknown          1     84.000          .     84.000     84.000

                   single layer     3     83.000      9.849     75.000     94.000

                   multi-layer      2     80.000      9.899     73.000     87.000

  Citrus           unknown          0          .          .          .          .

                   single layer     0          .          .          .          .

                   multi-layer      0          .          .          .          .

  Spice            unknown          0          .          .          .          .

                   single layer     0          .          .          .          .

                   multi-layer      3     85.000      5.292     81.000     91.000

  Other Flavor     unknown          0          .          .          .          .

                   single layer     0          .          .          .          .

                   multi-layer      1     72.000          .     72.000     72.000
  -----------------------------------------------------------------------------
```

Example 7: Computing a Confidence Limit for the Mean

Procedure features:

PROC MEANS statement options:

ALPHA=
FW=
MAXDEC=

CLASS statement

This example

- specifies the field width and number of decimal places of the statistics
- computes a two-sided 90 percent confidence limit for the mean values of MoneyRaised and HoursVolunteered for the three years of data.

If these data are representative of a larger population of volunteers, the confidence limits provide ranges of likely values for the true population means.

Program

The data set CHARITY contains information about high-school students' volunteer work for a charity. The variables give the name of the high school, the year of the fundraiser, the first name of each student, the amount of money each student raised, and the number of hours each student volunteered. A DATA step on page 1494 creates this data set.

```
data charity;
   input School $ 1-7 Year 9-12 Name $ 14-20 MoneyRaised 22-26
         HoursVolunteered 28-29;
   datalines;
Monroe  1992 Allison 31.65 19
Monroe  1992 Barry   23.76 16
Monroe  1992 Candace 21.11  5
     .
     .  more lines of data
     .
Kennedy 1994 Sid     27.45 25
Kennedy 1994 Will    28.88 21
Kennedy 1994 Morty   34.44 25
;
```

FW= uses a field width of eight and MAXDEC= uses two decimal places to display the statistics. ALPHA=.1 specifies a 90% confidence limit, and the CLM keyword requests two-sided confidence limits. MEAN and STD request the mean and the standard deviation, respectively.

```
proc means data=charity fw=8 maxdec=2 alpha=.1 clm mean std;
```

The CLASS statement separates the analysis by values of Year.

```
      class Year;
```

The VAR statement specifies the analysis variables and their order in the output.

```
      var MoneyRaised HoursVolunteered;
      title 'Confidence Limits for Fund Raising Statistics';
      title2 '1992-94';
   run;
```

Output

PROC MEANS displays the lower and upper confidence limits for both variables for each year.

```
                 Confidence Limits for Fund Raising Statistics                1
                                    1992-94

                              The MEANS Procedure

              N                          Lower 90%    Upper 90%
      Year  Obs  Variable             CL for Mean  CL for Mean      Mean   Std Dev
  ------------------------------------------------------------------------------
      1992   31  MoneyRaised                25.21        32.40     28.80     11.79
                 HoursVolunteered           17.67        23.17     20.42      9.01

      1993   32  MoneyRaised                25.17        31.58     28.37     10.69
                 HoursVolunteered           15.86        20.02     17.94      6.94

      1994   46  MoneyRaised                26.73        33.78     30.26     14.23
                 HoursVolunteered           19.68        22.63     21.15      5.96
  ------------------------------------------------------------------------------
```

Example 8: Computing Output Statistics

Procedure features:

PROC MEANS statement option:

NOPRINT

CLASS statement

OUTPUT statement options

statistic keywords
IDGROUP
LEVELS
WAYS

Other features:

PRINT procedure

Data set: GRADE on page 659

This example

- □ suppresses the display of default statistics
- □ outputs the average final grade to a new variable
- □ output the name of the student with the two final exam score to a new variable
- □ outputs how many class variables are combined to the _WAY_ variable
- □ outputs the value of the class level to the _LEVEL_ variable
- □ displays the output data set.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

NOPRINT suppresses the display of default statistics.

```
proc means data=Grade noprint;
```

The CLASS statement separates the analysis by values of Status and Year.

```
   class Status Year;
```

The VAR statement specifies the analysis variable.

```
   var finalgrade;
```

The OUTPUT statement creates the SUMSTAT data set and outputs the mean value for the final grade to the new variable AverageGrade. IDGROUP outputs the name of the student with the top exam score to the variable BestScore and the observation number that contained the top score. WAYS and LEVELS output information on how the class variables are combined.

```
   output out=sumstat mean=AverageGrade
          idgroup (max(score) obs out (name)=BestScore)
          /ways levels;
run;
```

PROC PRINT displays the SUMSTAT data set without the observation numbers.

```
proc print data=sumstat noobs;
   title1 'Average Undergraduate and Graduate Course Grades';
   title2 'For Two Years';
run;
```

Output

The first observation contains the average course grade and the name of the student with the highest exam score over the two-year period. The next four observations contain values for each class variable value. The remaining four observations contain values for the Year and Status combination. The variables _WAY_, _TYPE_, and _LEVEL_ show how PROC MEANS created the class variable combinations. The variable _OBS_ contains the observation number in the GRADE data set that contained the highest exam score.

```
                Average Undergraduate and Graduate Course Grades              1
                                 For Two Years

                                                       Average     Best
 Status    Year   _WAY_    _TYPE_    _LEVEL_    _FREQ_   Grade     Score     _OBS_

                    0        0          1         10    83.0000   Branford     2
            97      1        1          1          6    83.6667   Jasper      10
            98      1        1          2          4    82.0000   Branford     2
   1                1        2          1          6    82.5000   Branford     2
   2                1        2          2          4    83.7500   Abbott       1
   1        97      2        3          1          3    79.3333   Jasper      10
   1        98      2        3          2          3    85.6667   Branford     2
   2        97      2        3          3          3    88.0000   Abbott       1
   2        98      2        3          4          1    71.0000   Crandell     3
```

Example 9: Computing Different Output Statistics for Several Variables

Procedure features:

PROC MEANS statement options:
- DESCEND
- NOPRINT

CLASS statement

OUTPUT statement options:
- statistic keywords

Other features:

PRINT procedure

WHERE= data set option

Data set: GRADE on page 659

This example
- □ suppresses the display of default statistics
- □ outputs the statistics for the class level and combinations of class variables specified by WHERE=
- □ orders observations in the output data set by descending _TYPE_ value
- □ outputs the mean exam scores and mean final grades without assigning new variables names
- □ outputs the median final grade to a new variable
- □ displays the output data set.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

NOPRINT suppresses the display of default statistics. DESCEND orders the observations in the OUT= data set by descending _TYPE_ value.

```
proc means data=Grade noprint descend;
```

The CLASS statement separates the analysis by values of Status and Year.

```
   class Status Year;
```

The VAR statement specifies the analysis variables.

```
   var Score FinalGrade;
```

The OUTPUT statement outputs the mean for Score and FinalGrade to variables of the same name. The median final grade is output to the variable MedianGrade. The WHERE= data set option restricts the observations in SUMDATA. One observation contains overall statistics (_type_=0). The remainder must have a status of 1.

```
   output out=Sumdata (where=(status='1' or _type_=0))
          mean= median(finalgrade)=MedianGrade;
run;
```

PROC PRINT displays the SUMDATA data set.

```
proc print data=Sumdata;
   title 'Exam and Course Grades for Undergraduates Only';
   title2 'and for All Students';
run;
```

Output

The first three observations contain statistics for the class variable levels with a status of 1. The last observation contains the statistics for all the observations (no subgroup). Score contains the mean test score anf FinalGrade contains the mean final grade.

```
                  Exam and Course Grades for Undergraduates Only                1
                               and for All Students

                                                             Final     Median
  Obs    Status     Year    _TYPE_    _FREQ_     Score       Grade      Grade

   1       1         97        3         3      84.6667     79.3333       73
   2       1         98        3         3      88.3333     85.6667       80
   3       1                   2         6      86.5000     82.5000       80
   4                           0        10      86.0000     83.0000       83
```

Example 10: Computing Output Statistics with Missing Class Variable Values

Procedure features:

PROC MEANS statement options:

CHARTYPE
NOPRINT
NWAY

CLASS statement options:

ASCENDING
MISSING
ORDER=

OUTPUT statement

Other features:

PRINT procedure

Data set: CAKE on page 657

This example

- □ suppresses the display of default statistics
- □ considers missing values as valid level values for only one class variable
- □ orders observations in the output data set by the ascending frequency for a single class variable
- □ outputs observations for only the highest _TYPE_ value
- □ outputs _TYPE_ as binary character values
- □ outputs the maximum taste score to a new variable
- □ displays the output data set.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

CHARTYPE outputs the _TYPE_ values as binary characters. NWAY outputs observations with the highest _TYPE_ value. NOPRINT suppresses the display of default statistics.

```
proc means data=cake chartype nway noprint;
```

The CLASS statements separates the analysis by Flavor and Layers. ORDER=FREQ and ASCENDING order the levels of Flavor by ascending frequency. MISSING uses missing values of Layers as a valid class level value.

```
   class flavor /order=freq ascending;
   class layers /missing;
```

The VAR statement specifies the analysis variable.

```
   var TasteScore;
```

The OUTPUT statement creates the CAKESTAT data set and outputs the maximum value for the taste score to the new variable HighScore.

```
   output out=cakestat max=HighScore;
run;
```

PROC PRINT displays the CAKESTAT data set.

```
proc print data=cakestat;
   title 'Maximum Taste Score for Flavor and Cake Layers';
run;
```

Output

The OUT= output data set contains only observations for the combination of both class variables, Flavor and Layers. Therefore, _TYPE_ contains the binary character string 11. The observations are ordered by ascending frequency of Flavor. The missing value in Layers is a valid value for this class variable. PROC MEANS excludes the observation with the missing flavor because it an invalid value for Flavor.

```
              Maximum Taste Score for Flavor and Cake Layers                1

                                                            High
          Obs    Flavor       Layers    _TYPE_    _FREQ_    Score

           1     Rum             2        11         1        72
           2     Spice           2        11         2        83
           3     Spice           3        11         1        91
           4     Vanilla         .        11         1        84
           5     Vanilla         1        11         3        94
           6     Vanilla         2        11         2        87
           7     Chocolate       .        11         1        84
           8     Chocolate       1        11         5        85
           9     Chocolate       2        11         3        92
```

Example 11: Identifying an Extreme Value with the Output Statistics

Procedure features:

CLASS statement

OUTPUT statement options:

statistic keyword
MAXID

Other features:

PRINT procedure

Data set: CHARITY on page 671

This example

- □ identifies the observations with maximum values for two variables
- □ creates new variables for the maximum values
- □ displays the output data set.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

The statistic keywords specify the statistics and their order in the output.

```
proc means data=Charity n mean range;
```

The CLASS statement separates the analysis by School and Year.

```
   class School Year;
```

The VAR statement specifies the analysis variables and their order in the output.

```
   var MoneyRaised HoursVolunteered;
```

The OUTPUT statement outputs the new variables, MostCash and MostTime, which contain the names of the students who collected the most money and volunteered the most time, respectively, to the PRIZE data set.

```
   output out=Prize maxid(MoneyRaised(name)
          HoursVolunteered(name))= MostCash MostTime
          max= ;
   title 'Summary of Volunteer Work by School and Year';
run;
```

PROC PRINT displays the PRIZE data set.

```
proc print data=Prize;
   title 'Best Results: Most Money Raised and Most Hours Worked';
run;
```

Output

The first page of output shows the output from PROC MEANS with the statistics for six class levels: one for Monroe High for the years 1992, 1993, and 1994; and one for Kennedy High for each of the three years.

```
                 Summary of Volunteer Work by School and Year                1

                             The MEANS Procedure

                       N
School           Year  Obs  Variable            N          Mean          Range
------------------------------------------------------------------------------
Kennedy          1992   15  MoneyRaised        15    29.0800000     39.7500000
                            HoursVolunteered   15    22.1333333     30.0000000

                 1993   20  MoneyRaised        20    28.5660000     23.5600000
                            HoursVolunteered   20    19.2000000     20.0000000

                 1994   18  MoneyRaised        18    31.5794444     65.4400000
                            HoursVolunteered   18    24.2777778     15.0000000

Monroe           1992   16  MoneyRaised        16    28.5450000     48.2700000
                            HoursVolunteered   16    18.8125000     38.0000000

                 1993   12  MoneyRaised        12    28.0500000     52.4600000
                            HoursVolunteered   12    15.8333333     21.0000000

                 1994   28  MoneyRaised        28    29.4100000     73.5300000
                            HoursVolunteered   28    19.1428571     26.0000000
------------------------------------------------------------------------------
```

The output from PROC PRINT shows the maximum MoneyRaised and HoursVolunteered values and the names of the students who are responsible for them. The first observation contains the overall results, the next three contain the results by year, the next two contain the results by school, and the final six contain the results by School and Year.

```
            Best Results: Most Money Raised and Most Hours Worked           2

                                          Most     Most       Money     Hours
  Obs  School   Year  _TYPE_  _FREQ_  Cash     Time      Raised  Volunteered

    1              .     0      109   Willard  Tonya      78.65       40
    2           1992     1       31   Tonya    Tonya      55.16       40
    3           1993     1       32   Cameron  Amy        65.44       31
    4           1994     1       46   Willard  L.T.       78.65       33
    5  Kennedy     .     2       53   Luther   Jay        72.22       35
    6  Monroe      .     2       56   Willard  Tonya      78.65       40
    7  Kennedy  1992     3       15   Thelma   Jay        52.63       35
    8  Kennedy  1993     3       20   Bill     Amy        42.23       31
    9  Kennedy  1994     3       18   Luther   Che-Min    72.22       33
   10  Monroe   1992     3       16   Tonya    Tonya      55.16       40
   11  Monroe   1993     3       12   Cameron  Myrtle     65.44       26
   12  Monroe   1994     3       28   Willard  L.T.       78.65       33
```

Example 12: Identifying the Top Three Extreme Values with the Output Statistics

Procedure features:

PROC MEANS statement option:

NOPRINT

CLASS statement

OUTPUT statement options:

statistic keywords
AUTOLABEL
AUTONAME
IDGROUP

TYPES statement

Other features:

FORMAT procedure

FORMAT statement

PRINT procedure

RENAME = data set option

Data set: CHARITY on page 671

This example

- □ suppresses the display of default statistics
- □ analyzes the data for the one-way combination of the class variables and across all observations
- □ outputs the total and average amount of money raised to new variables
- □ outputs to new variables the top three amounts of money raised, the names of the three students who raised the money, the years when it occurred, and the schools the students attended
- □ automatically resolves conflicts in the variable names when names are assigned to the new variables in the output data set
- □ appends the statistic name to the label of the variables in the output data set that contain statistics that were computed for the analysis variable.
- □ assigns a format to the analysis variable so that the statistics that are computed from this variable inherit the attribute in the output data set
- □ renames the _FREQ_ variable in the output data set
- □ displays the output data set and its contents.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

PROC FORMAT creates user-defined formats that assign the value of **`All`** to the missing levels of the class variables.

```
 proc format;
   value yrFmt . = " All";
   value $schFmt ' ' = "All    ";
run;
```

NOPRINT suppresses the display of default statistics.

```
proc means data=Charity noprint;
```

The CLASS statement separates the analysis by values of School and Year.

```
   class School Year;
```

The TYPES statement requests the analysis across all the observations and for each one-way combination of School and Year.

```
   types () school year;
```

The VAR statement specifies the analysis variable.

```
   var moneyraised;
```

The OUTPUT statement creates the TOP3LIST data set. RENAME= renames the _FREQ_ variable that contains frequency count for each class level. SUM= and MEAN= specify that the sum of money raised and the mean of money raised are output to automatically name the new variables. IDGROUP outputs 12 variables that contain the top three amounts of money raised and the three corresponding students, schools, and years. AUTOLABEL appends the analysis variable name to the label for the output variables that contain the sum and mean. AUTONAME resolves naming conflicts for these variables.

```
   output out=top3list(rename=(_freq_=NumberStudents))sum= mean=
          idgroup( max(moneyraised) out[3] (moneyraised name
            school year)=)/autolabel autoname;
```

The FORMAT statement assigns user-defined formats to Year and School and a SAS dollar format to MoneyRaised. The LABEL statement assigns a label to the analysis variable MoneyRaised.

```
   label MoneyRaised='Amount Raised';
   format year yrfmt. school $schfmt.
          moneyraised dollar8.2;
  run;
```

PROC PRINT displays the TOP3LIST data set.

```
proc print data=top3list;
   title1 'School Funding Raising Report';
   title2 'Top Three Students';
run;
```

PROC DATASETS displays the contents of the TOP3LIST data set. NOLIST suppresses the directory listing for the WORK data library.

```
proc datasets library=work nolist;
   contents data=top3list;
   title1 'Contents of the PROC MEANS Output Data Set';
  run;
```

Output

The output from PROC PRINT shows the top three values of MoneyRaised, the names of the students who raised these amounts, the schools the students attended, and the years when the money was raised. The first observation contains the overall results, the next three contain the results by year, and the final two contain the results by school. The missing class levels for School and Year are replaced with the value of **ALL**.

```
                         School Funding Raising Report                        1
                               Top Three Students

                                     Money    Money
                          Number    Raised_  Raised_  Money    Money    Money
Obs School  Year _TYPE_ Students       Sum     Mean  Raised_1 Raised_2 Raised_3

 1  All      All    0       109   $3192.75   $29.29   $78.65   $72.22   $65.44
 2  All     1992    1        31    $892.92   $28.80   $55.16   $53.76   $52.63
 3  All     1993    1        32    $907.92   $28.37   $65.44   $47.33   $42.23
 4  All     1994    1        46   $1391.91   $30.26   $78.65   $72.22   $56.87
 5  Kennedy  All    2        53   $1575.95   $29.73   $72.22   $52.63   $43.89
 6  Monroe   All    2        56   $1616.80   $28.87   $78.65   $65.44   $56.87



Obs Name_1  Name_2  Name_3  School_1 School_2 School_3 Year_1 Year_2 Year_3

 1  Willard Luther  Cameron Monroe   Kennedy  Monroe    1994   1994   1993
 2  Tonya   Edward  Thelma  Monroe   Monroe   Kennedy   1992   1992   1992
 3  Cameron Myrtle  Bill    Monroe   Monroe   Kennedy   1993   1993   1993
 4  Willard Luther  L.T.    Monroe   Kennedy  Monroe    1994   1994   1994
 5  Luther  Thelma  Jenny   Kennedy  Kennedy  Kennedy   1994   1992   1992
 6  Willard Cameron L.T.    Monroe   Monroe   Monroe    1994   1993   1994
```

The labels for the variables that contain statistics that were computed from MoneyRaised include the statistic name at the end of the label.

```
                         School Funding Raising Report                        1
                               Top Three Students

                                     Money    Money
                          Number   Raised_  Raised_  Money    Money    Money
Obs School   Year _TYPE_ Students      Sum     Mean Raised_1 Raised_2 Raised_3

 1  All       All    0       109  $3192.75   $29.29   $78.65   $72.22   $65.44
 2  All      1992    1        31   $892.92   $28.80   $55.16   $53.76   $52.63
 3  All      1993    1        32   $907.92   $28.37   $65.44   $47.33   $42.23
 4  All      1994    1        46  $1391.91   $30.26   $78.65   $72.22   $56.87
 5  Kennedy   All    2        53  $1575.95   $29.73   $72.22   $52.63   $43.89
 6  Monroe    All    2        56  $1616.80   $28.87   $78.65   $65.44   $56.87



Obs Name_1  Name_2  Name_3  School_1 School_2 School_3 Year_1 Year_2 Year_3

 1  Willard Luther  Cameron Monroe   Kennedy  Monroe    1994   1994   1993
 2  Tonya   Edward  Thelma  Monroe   Monroe   Kennedy   1992   1992   1992
 3  Cameron Myrtle  Bill    Monroe   Monroe   Kennedy   1993   1993   1993
 4  Willard Luther  L.T.    Monroe   Kennedy  Monroe    1994   1994   1994
 5  Luther  Thelma  Jenny   Kennedy  Kennedy  Kennedy   1994   1992   1992
 6  Willard Cameron L.T.    Monroe   Monroe   Monroe    1994   1993   1994
```

```
                   Contents of the PROC MEANS Output Data Set                  2

                             The DATASETS Procedure

  Data Set Name: WORK.TOP3LIST                        Observations:         6
  Member Type:   DATA                                 Variables:            18
  Engine:        V8                                   Indexes:              0
  Created:       14:41 Tuesday, May 4, 1999           Observation Length:   144
  Last Modified: 14:41 Tuesday, May 4, 1999           Deleted Observations: 0
  Protection:                                         Compressed:           NO
  Data Set Type:                                      Sorted:               NO
  Label:


                    -----Engine/Host Dependent Information-----

Data Set Page Size:         16384
Number of Data Set Pages:   1
First Data Page:            1
Max Obs per Page:           113
Obs in First Data Page:     6
Number of Data Set Repairs: 0
File Name:                  UNIX-pathname
Release Created:            8.00.00B
Host Created:               HP-UX
Inode Number:               313604
Access Permission:          rw-r--r--
Owner Name:                 UNIX-userid
File Size (bytes):          24576


               -----Alphabetic List of Variables and Attributes-----

 #    Variable           Type    Len    Pos    Format       Label
-------------------------------------------------------------------------------
 7    MoneyRaised_1      Num       8     40    DOLLAR8.2    Amount Raised
 8    MoneyRaised_2      Num       8     48    DOLLAR8.2    Amount Raised
 9    MoneyRaised_3      Num       8     56    DOLLAR8.2    Amount Raised
 6    MoneyRaised_Mean   Num       8     32    DOLLAR8.2    Amount Raised_Mean
 5    MoneyRaised_Sum    Num       8     24    DOLLAR8.2    Amount Raised_Sum
10    Name_1             Char      7     95
11    Name_2             Char      7    102
12    Name_3             Char      7    109
 4    NumberStudents     Num       8     16
 1    School             Char      7     88    $SCHFMT.
13    School_1           Char      7    116    $SCHFMT.
14    School_2           Char      7    123    $SCHFMT.
15    School_3           Char      7    130    $SCHFMT.
 2    Year               Num       8      0    YRFMT.
16    Year_1             Num       8     64    YRFMT.
17    Year_2             Num       8     72    YRFMT.
18    Year_3             Num       8     80    YRFMT.
 3    _TYPE_             Num       8      8
```

See the TEMPLATE procedure in *The Complete Guide to the SAS Output Delivery System* for an example of how to create a custom table definition for this output data set.

References

Jain R. and Chlamtac I., (1985) “The P^2 Algorithm for Dynamic Calculation of Quantiles and Histograms Without Sorting Observations,” *Communications of the Association of Computing Machinery*, 28:10.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Procedures Guide, Version 8,* Cary, NC: SAS Institute Inc., 1999. 1729 pp.

SAS® Procedures Guide, Version 8

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1–58025–482–9

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999