Chapter 3: Producing descriptive statistics

<SAS Certification Prep Guide - Base Programming for SAS 9>

Objectives : learning about

◦ determine the n-count, mean, standard deviation, minimum, and maximum of numeric variables using the MEANS procedure

◦ control the number of decimal places used in PROC MEANS output

◦ specify the variables for which to produce statistics

◦ use the SUMMARY procedure to produce the same results as the MEANS procedure

◦ describe the difference between the SUMMARY and MEANS procedures

◦ create one-way frequency tables for categorical data using the FREQ procedure

◦ create two-way and n-way crossed frequency tables

◦ control the layout and complexity of crossed frequency tables.

Computing Statistics for Numeric Variables

■ Computing Statistics Using PROC MEANS

Descriptive statistics such as the mean, minimum, and maximum can provide useful information about numeric data. The MEANS procedure provides these and other data summarization tools, as well as helpful options for controlling your output.

The MEANS procedure can include many statements and options for specifying needed statistics. For simplicity, let's consider the procedure in its basic form.

```
PROC MEANS <DATA=SAS-data-set> <statistic-keywords> <options>;
run;
```

In its simplest form, PROC MEANS prints the n-count (number of nonmissing values), the mean, the standard deviation, and the minimum and maximum values of every numeric variable in a data set.

```
proc means data=clinic.survey;
run;
```

■ Specifying Statistics

The default statistics that the MEANS procedure produces (n-count, mean, standard deviation, minimum, and maximum) are not always the ones that you need. You might prefer to limit output to the mean of the values. Or you might need to compute a different statistic, such as the median or range of the values.

To specify statistics, include statistic keywords as options in the PROC MEANS statement. When you specify a statistic in the PROC MEANS statement, default

statistics are not produced. For example, to see the median and range of Perm.Survey numeric values, add the MEDIAN and RANGE keywords as options.

```
proc means data=clinic.survey median range;
run;
```

The following keywords can be used with PROC MEANS to compute statistics:

- Descriptive Statistics

Keyword	Description
CLM	Two-sided confidence limit for the mean
CSS	Corrected sum of squares
CV	Coefficient of variation
KURTOSIS / KURT	Kurtosis
LCLM	One-sided confidence limit below the mean
MAX	Maximum value
MEAN	Average
MIN	Minimum value
N	Number of observations with non-missing values
NMISS	Number of observations with missing values
RANGE	Range
SKEWNESS / SKEW	Skewness
STDDEV/STD	Standard deviation
STDERR/STDMEAN	Standard error of the mean
SUM	Sum
SUMWGT	Sum of the Weight variable values
UCLM	One-sided confidence limit above the mean
USS	Uncorrected sum of squares
VAR	Variance

- Quantile Statistics

Keyword	Description
MEDIAN / P50	Median or 50th percentile
P1	1st percentile
P5	5th percentile
P10	10th percentile
Q1 / P25	Lower quartile or 25th percentile
Q3/P75	Upper quartile or 75th percentile
P90	90th percentile
P95	95th percentile
P99	99th percentile
QRANGE	Difference between upper and lower quartiles: Q3-Q1

- Hypotheses testing

Keyword	Description
PROBT	Probability of a greater absolute value for the t value
T	Student's t for testing the hypothesis that the population mean is 0

■ Limiting Decimal Places

By default, PROC MEANS output uses the BEST. format to display values in the report. As the following example shows, this can result in unnecessary decimal places, making your output difficult to read.

```
proc means data=clinic.diabetes min max;
run;
```

To limit decimal places, use the MAXDEC= option in the PROC MEANS statement, and set it qual to the length that you prefer.

```
proc means data=clinic.diabetes min max maxdec=0;
run;
```

■ Specifying Variables in PROC MEANS

By default, the MEANS procedure generates statistics for every numeric variable in a data set, lthough you'll typically want to focus on only a few variables, particularly if the data set is large. It also makes sense to exclude certain types of variables. The values of employee identification numbers, for example, are unlikely to yield useful statistics.

To specify the variables that PROC MEANS analyzes, add a VAR statement and list the variable names.

```
proc means data=clinic.diabetes min max maxdec=0;
        var age height weight;
run;
proc means data=perm.survey mean stderr maxdec=2;
        var item1-item5;
run;
```

■ Group Processing Using the CLASS Statement

You will often want statistics for grouped observations, instead of for observations as a whole. For example, census numbers are more useful when grouped by region than when viewed as a national total. To produce separate analyses of grouped observations, add a CLASS statement to the MEANS procedure.

PROC MEANS does not generate statistics for CLASS variables, because their values are used only to categorize data. CLASS variables can be either character or numeric, but they should contain a limited number of discrete values that represent meaningful groupings.

The output of the program shown below is categorized by values of the variables Survive and Sex. The order of the variables in the CLASS statement determines their order in the output table.

```
proc means data=clinic.heart maxdec=1;
        var arterial heart cardiac urinary;
        class survive sex;
run;
```

■ Group Processing Using the BY Statement

Like the CLASS statement, the BY statement specifies variables to use for categorizing observations.

But BY and CLASS processing differ in two key ways:

1. Unlike CLASS processing, BY processing requires that your data already be sorted or indexed in the order of the BY variables. Unless data set observations are already sorted, you will need to run the SORT procedure before using PROC MEANS with any BY group. Be careful when sorting data sets to enable group processing. If you don't specify an output data set by using the OUT= option, then PROC SORT will overwrite your initial data set with the newly sorted
observations.
2. BY group results have a layout that is different from the layout of CLASS

group results. Note that the BY statement in the program below creates four small tables; a CLASS statement would produce a single large tables.

```
proc sort data=clinic.heart out=work.heartsort;
        by survive sex;
run;
proc means data=work.heartsort maxdec=1;
        var arterial heart cardiac urinary;
        by survive sex;
run;
```

■ Group Processing Using the BY Statement

Creating a Summarized Data Set Using PROC MEANS

```
OUTPUT OUT=SAS-data-set <statistic-keyword=variable-name(s)>;
```

When you use the OUTPUT statement without specifying the statistic-keyword= option, the summary statistics N, MEAN, STD, MIN, and MAX are produced for all of the numeric variables or for all of the variables that are listed in a VAR statement.

To specify which statistics to produce in the output data set, you must specify the keyword for the statistic and then list all of the variables. The variables must be listed in the same order as in the VAR statement. You can specify more than one statistic in the OUTPUT statement.

The following program creates a typical PROC MEANS report and also creates a summarized output data set that includes only the MEAN and MIN statistics:

```
proc means data=clinic.diabetes;
        var age height weight;
        class sex;
        output out=work.sum_gender
        mean=AvgAge AvgHeight AvgWeight
        min=MinAge MinHeight MinWeight;
run;
proc print data=work.sum_gender;
run;
```

cf) You can use the NOPRINT option in the PROC MEANS statement to prevent the default report from being created.

■ Creating a Summarized Data Set Using PROC SUMMARY

You can also create a summarized output data set by using the SUMMARY procedure. When you use PROC SUMMARY, you use the same code to produce the output data set that you would use with PROC MEANS.

The difference between the two procedures is that PROC MEANS produces a report by default (remember that you can use the NOPRINT option to suppress the default report). By contrast, to produce a report in PROC SUMMARY, you must include a PRINT option in the PROC SUMMARY statement.

```
proc summary data=clinic.diabetes;
        var age height weight;
        class sex;
        output out=work.sum_gender
        mean=AvgAge AvgHeight AvgWeight;
run;
```

cf) If you placed a PRINT option in the PROC SUMMARY statement above, this program would produce the same report as if you replaced the word SUMMARY with MEANS.

Producing Frequency Tables

■ Group Processing Using the BY Statement

The FREQ procedure is a descriptive procedure as well as a statistical procedure. It produces oneway and n-way frequency tables, and it concisely describes your data by reporting the distribution of variable values. You can use the FREQ procedure to create crosstabulation tables that summarize data for two or more categorical variables by showing the number of observations for each combination of variable values.

The FREQ procedure can include many statements and options for controlling frequency output.

By default, PROC FREQ creates a one-way table with the frequency, percent, cumulative frequency, and cumulative percent of every value of all variables in a data set. But, if you want to specify the variables to be processed by the FREQ procedure, include a TABLES statement.

```
proc freq data=clinic.admit;
        tables sex actlevel;
run;
```

■ Creating Two-Way Tables

So far, you have used the FREQ procedure to create one-way frequency tables. The table results show total frequency counts for the values within the data set. However, it is often helpful to crosstabulate frequencies with the values of other variables. For example, census data is typically crosstabulated with a variable that represents geographical regions.

The simplest crosstabulation is a two-way table. To create a two-way table, join two variables with an asterisk (*) in the TABLES statement of a PROC FREQ step.

When crosstabulations are specified, PROC FREQ produces tables with cells that contain

- cell frequency
- cell percentage of total frequency
- cell percentage of row frequency
- cell percentage of column frequency.

```
proc freq data=clinic.admit;
        tables sex*actlevel;
run;
```

■ Creating N-Way Tables

For a frequency analysis of more than two variables, use PROC FREQ to create n-way crosstabulation tables. A series of two-way tables is produced, with a table for each level of the other variables.

```
proc freq data=clinic.admit;
        tables sex*actlevel*fee;
run;
```

■ Creating Tables in List Format

When three or more variables are specified, the multiple levels of n-way tables can produce considerable output. Such bulky, often complex crosstabulations are often easier to read as a continuous list. Although this eliminates row and column frequencies and percents, the results are compact and clear.

To generate list output for crosstabulations, add a slash (/) and the LIST option to the TABLES statement in your PROC FREQ step.

```
proc freq data=clinic.admit;
        tables sex*actlevel*fee / list;
run;
```

■ Suppressing Table Information

You can use options to suppress any of these statistics. To control the depth of crosstabulation results, add a slash (/) and any combination of the following options to the TABLES statement:

- NOFREQ suppresses cell frequencies. (for multi-way tables)
- NOPERCENT suppresses cell percentages
- NOROW supresses row percentages. (for multi-way tables)
- NOCOL suppresses column percentages. (for multi-way tables)
- NOCUM suppresses cumulative frequencies and percentages. (for 1-way tables)

sas-code

```
proc means data=clinic.survey;
run;
proc means data=clinic.survey median range;
run;
proc means data=clinic.diabetes min max;
run;
proc means data=clinic.diabetes min max maxdec=0;
run;
proc means data=clinic.diabetes min max maxdec=0;
        var age height weight;
run;
proc means data=perm.survey mean stderr maxdec=2;
        var item1-item5;
run;
proc means data=clinic.heart maxdec=1;
        var arterial heart cardiac urinary;
        class survive sex;
run;
proc sort data=clinic.heart out=work.heartsort;
        by survive sex;
run;
proc means data=work.heartsort maxdec=1;
        var arterial heart cardiac urinary;
        by survive sex;
run;
proc means data=clinic.diabetes;
        var age height weight;
        class sex;
        output out=work.sum_gender
        mean=AvgAge AvgHeight AvgWeight
        min=MinAge MinHeight MinWeight;
run;
proc print data=work.sum_gender;
run;
proc summary data=clinic.diabetes;
        var age height weight;
        class sex;
        output out=work.sum_gender
        mean=AvgAge AvgHeight AvgWeight;
run;



proc freq data=clinic.admit;
        tables sex actlevel;
run;
proc freq data=clinic.admit;
        tables sex*actlevel;
run;
proc freq data=clinic.admit;
        tables sex*actlevel*fee;
run;
proc freq data=clinic.admit;
        tables sex*actlevel*fee / list;
run;
```