SAS 4. Sorting, printing and summarizing Your Data

1. Using SAS procedures

Using a procedure, or PROC, is like filling out a form. All procedures have required statements, and most have optional statements.

PROC: All procedures start with the keyword PROC followed by the name of the procedure, such as PRINT.

BY: The BY statement is required for only one procedure, PROC SORT, in which it tells SAS how to arrange the observations.

TITLE and **FOOTNOTE**: These global statements are not technically part of any step. You can put them anywhere in your program, but since they apply to the procedure output it generally makes sense to put them with the procedure. For example,

```
TITLE "Here's another title";
FOOTNOTE 'This is the third footnote';
```

2. Subsetting in procedures with the WHERE statements

One optional statement for any PROC that reads a SAS data set is the WHERE statement. The WHERE statement tells a procedure to use a subset of the data. The basic form of a WHERE statement is

```
WHERE condition;
```

Only observations satisfying the condition will be used by the PROC.

Example You have a database containing information about well-known painters. A subset of the data appears below. For each artist, the data include the painter's name, primary style, and nation of origin:

```
Mary Cassatt          Impressionism      U
Paul Cezanne          Post-impressionism F
Edgar Degas           Impressionism      F
Paul Gauguin          Post-impressionism F
Claude Monet          Impressionism      F
Pierre Auguste Renoir Impressionism      F
Vincent van Gogh      Post-impressionism N
```

To make this example more realistic, it has two steps: one to create a permanent SAS data set, the other to subset the data.

```
DATA style;
INPUT Name $ 1-21 Genre $ 23-40 Origin $ 42;
DATALINES;
Mary Cassatt          Impressionism      U
Paul Cezanne           Post-impressionism F
Edgar Degas            Impressionism      F
Paul Gauguin           Post-impressionism F
Claude Monet           Impressionism      F
Pierre Auguste Renoir Impressionism       F
Vincent van Gogh      Post-impressionism N
;
PROC PRINT DATA = style;
      WHERE Genre = 'Impressionism';
      TITLE 'Major Impressionist Painters';
      FOOTNOTE 'F = France N = Netherlands U = US';
RUN;

*WHERE Genre = 'Impressionism' and g ge 4;
```

Here are the most frequently used operators:

Symbolic	Mnemonic	Example
=	EQ	`WHERE Region = 'Spain';`
¬=, ~=, ^=	NE	`WHERE Region ~= 'Spain';`
>	GT	`WHERE Rainfall > 20;`
<	LT	`WHERE Rainfall < AvgRain;`
>=	GE	`WHERE Rainfall >= AvgRain + 5;`
<=	LE	`WHERE Rainfall <= AvgRain / 1.25;`
&	AND	`WHERE Rainfall > 20 AND Temp < 90;`
|,¦,!	OR	`WHERE Rainfall > 20 OR Temp < 90;`
	IS NOT MISSING	`WHERE Region IS NOT MISSING;`
	BETWEEN AND	`WHERE Region BETWEEN 'Plain' AND 'Spain';`
	CONTAINS	`WHERE Region CONTAINS 'ain';`
	IN (*list*)	`WHERE Region IN ('Rain', 'Spain', 'Plain');`

3. Sorting your data with PROC SORT

PROC SORT is a very simple form of sorting your data. The basic form of this procedure is

```
      PROC SORT;
             BY variable-1 ... variable-n;
```

The DATA= and OUT= options specify the input and output data sets. If you don't specify the DATA= option, then SAS will use the most recently created data set. If you don't specify the OUT= option, then SAS will replace the original data set with the newly sorted version. The NODUPKEY option tells SAS to eliminate any duplicate observations that have the same values for the BY variables. To use this option, just add NODUPKEY to the PROC SORT statement. For example,

```
      PROC SORT DATA = messy OUT = neat NODUPKEY;
```

By default SAS sorts data in ascending order, from lowest to highest or from A to Z. To have your data sorted from highest to lowest, add the keyword DESCENDING to the BY statement before each variable that should be sorted from highest to lowest. For instance,

```
       BY State DESCENDING City;
```

Example The following data show the average length in feet of selected whales and sharks.

```
DATA marine;
INPUT Name $ Family $ Length;
DATALINES;
beluga whale 15
whale shark 40
basking shark 30
gray whale 50
mako shark 12
sperm whale 60
dwarf shark .5
whale shark 40
humpback . 50
blue whale 100
killer whale 30
;
* Sort the data;
PROC SORT DATA = marine OUT = seasort NODUPKEY;
       BY Family DESCENDING Length;
PROC PRINT DATA = seasort;
TITLE 'Whales and Sharks';
RUN;
```

4. Printing your data with PROC PRINT

The PRINT procedure can be specified as follows:

```
   PROC PRINT DATA = data-set;
```

The following are optional statements that sometimes come in handy:

BY variable-list: It starts a new section in the output for each new value of the BY variables and prints the values of the BY variables at the top of each section. The data must be presorted by the BY variables.

ID variable-list: When you use the ID statement, the observation numbers are not printed. Instead, the variables in the ID variable list appear on the left-hand side of the page.

SUM variable-list: It prints sums for the variables in the list.

VAR variable-list: It specifies which variables to print and the order.

Example Students from two fourth-grade classes are selling candy to earn money for a special field trip. The class earning more money gets a free box of candy. The following are the data for the results of the candy sale. The students' names are followed by their classroom number, the date they turned in their money, the type of candy: mint patties or chocolate dinosaurs, and the number of boxes sold. The class earns $1.25 for each box of candy sold. The teachers want a report giving the money earned for each classroom, the money earned by each student, the type of candy sold, and the date the students returned their money. The following program reads the data, computes money earned (Profit), and sorts the data by classroom using PROC SORT. Then, the PROC PRINT step uses a BY statement to print the data by Class and a SUM statement to give the totals for Profit. The VAR statement lists the variables to be printed:

```
DATA sales;
INPUT Name $ 1-11 Class @15 DateReturned MMDDYY10. CandyType $
Quantity;
Profit = Quantity * 1.25;
DATALINES;
Adriana    21 3/21/2000 MP  7
Nathan     14 3/21/2000 CD 19
Matthew    14 3/21/2000 CD 14
Claire     14 3/22/2000 CD 11
Caitlin    21 3/24/2000 CD  9
Ian        21 3/24/2000 MP 18
Chris      14 3/25/2000 CD  6
Anthony    21 3/25/2000 MP 13
Stephen    14 3/25/2000 CD 10
Erika      21 3/25/2000 MP 17
;
PROC SORT DATA = sales;
      BY Class;
PROC PRINT DATA = sales;
      BY Class;
      SUM Profit;
      VAR Name DateReturned CandyType Profit;
      TITLE 'Candy Sales for Field Trip by Class';
RUN;
```

5. Changing the format of the output

You can change the appearance of printed values using SAS formats. SAS has many formats for character, numeric, and date values. The general forms of a SAS format are

Character	**Numeric**	**Date**
$formatw.	formatw.d	formatw.

where the $ indicates character formats, format is the name of the format, w is the total width including any decimal point, and d is the number of decimal places. The period in the format is very important because it distinguishes a format from a variable name, which cannot, by default, contain any special characters except the underscore.

FORMAT FORMAT statements can go in either DATA steps or PROC steps. The FORMAT statement starts with the keyword FORMAT, followed by the variable names and the format. For example,

```
   FORMAT Profit Loss DOLLAR8.2 SaleDate MMDDYY8.;
```

Example We use the FORMAT statement to print the date in the last example.

```
DATA sales;
INPUT Name $ 1-11 Class @15 DateReturned MMDDYY10. CandyType $
Quantity;
Profit = Quantity * 1.25;
DATALINES;
Adriana    21 3/21/2000 MP  7
Nathan     14 3/21/2000 CD 19
Matthew    14 3/21/2000 CD 14
Claire     14 3/22/2000 CD 11
Caitlin    21 3/24/2000 CD  9
Ian        21 3/24/2000 MP 18
Chris      14 3/25/2000 CD  6
Anthony    21 3/25/2000 MP 13
Stephen    14 3/25/2000 CD 10
Erika      21 3/25/2000 MP 17
;
PROC PRINT DATA = sales;
      VAR Name DateReturned CandyType Profit;
      FORMAT DateReturned DATE9. Profit DOLLAR6.2;
      TITLE 'Candy Sale Data Using Formats';
RUN;
```

6. Summarizing your data using PROC MEANS

The MEANS procedure provides simple statistics on numeric variables. The MEANS procedure starts with the keywords PROC MEANS, followed by options listing the statistics you want printed:

```
      PROC MEANS options;
```

If you do not specify any options, MEANS will print the number of non-missing values, the mean, the standard deviation, and the minimum and maximum values for each variable. There are over 30 different statistics you can request with the MEANS procedure, such as,

MAX	the maximum value
MIN	the minimum value
MEAN	the mean
MEDIAN	the median
N	number of non-missing values
NMISS	number of missing values
RANGE	the range
STDDEV	the standard deviation
SUM	the sum

There are also some optional statements you may want to use:

BY variable-list	It performs separate analyses for each level of the variables in the list.1 The data must first be sorted in the same order as the variable-list. (You can use PROC SORT to do this.)

CLASS variable-list;	It performs separate analyses for each level of the variables in the list, 1 but its output is more compact than with the BY statement, and the data do not have to be sorted first.
VAR variable-list	It specifies which numeric variables to use in the analysis. If it is absent then SAS uses all numeric variables.

Example A wholesale nursery is selling garden flowers, and they want to summarize their sales figures by month. The data file which follows contains the customer ID, date of sale, and number of petunias, snapdragons, and marigolds sold. The following program reads the data; computes a new variable, Month, which is the month of the sale; sorts the data by Month using PROC SORT; then summarizes the data by Month using PROC MEANS with a BY statement:

```
DATA sales;
INPUT CustomerID $ @9 SaleDate MMDDYY10. Petunia SnapDragon
Marigold;
Month = MONTH(SaleDate);
DATALINES;
756-01 05/04/2001 120 80 110
834-01 05/12/2001 90 160 60
901-02 05/18/2001 50 100 75
834-01 06/01/2001 80 60 100
756-01 06/11/2001 100 160 75
901-02 06/19/2001 60 60 60
756-01 06/25/2001 85 110 100
;
PROC SORT DATA =
sales; BY Month;
* Calculate means by Month for flower sales;
PROC MEANS DATA = sales;
      BY Month;
      VAR Petunia SnapDragon Marigold;
      TITLE 'Summary of Flower Sales by Month';
RUN;
```

7. Writing summary statistics to a SAS data set

There are two methods in PROC MEANS for saving summary statistics in a SAS data set. One way is to use the OUTPUT statement, which has the following form:

```
OUTPUT OUT = data-set output-statistic-list;
```

Here, `data-set` is the name of the SAS data set which will contain the results (this can be either temporary or permanent), and `output-statistic-list` defines which statistics you want and the associated variable names. You can have more than one OUTPUT statement and multiple output statistic lists. The following is one of the possible forms for `output-statistic-list`:

```
statistic(variable-list) = name-list
```

Here, `statistic` can be any of the statistics available in PROC MEANS (SUM, N, MEAN, for example), `variable-list` defines which of the variables in the VAR statement you want to output, and `name-list` defines the new variable names for the statistics. The new variable names must be in the same order as their corresponding variables in `variable-list`.

Example For the data used in the last example, we want to summarize the data so that you have only one observation per customer containing the sum and mean of the number of plant sets sold, and we also want to save the results in a SAS data set for further analysis.

```
DATA sales;
INPUT CustomerID $ @9 SaleDate MMDDYY10. Petunia SnapDragon
Marigold; DATALINES;
756-01 05/04/2001 120 80 110
834-01 05/12/2001 90 160 60
901-02 05/18/2001 50 100 75
834-01 06/01/2001 80 60 100
756-01 06/11/2001 100 160 75
901-02 06/19/2001 60 60 60
756-01 06/25/2001 85 110 100
;
PROC SORT DATA = sales;
BY CustomerID;
* Calculate means by CustomerID, output sum and mean to new data set;
PROC MEANS NOPRINT DATA =
       sales; BY CustomerID;
       VAR Petunia SnapDragon Marigold;
       OUTPUT OUT = totals MEAN(Petunia SnapDragon Marigold) =
              MeanPetunia MeanSnapDragon MeanMarigold
              SUM(Petunia SnapDragon Marigold) = Petunia SnapDragon Marigold;
PROC PRINT DATA = totals;
TITLE 'Sum of Flower Data over Customer ID';
FORMAT MeanPetunia MeanSnapDragon MeanMarigold 3.;
RUN;
```