Summary Statistics in SAS

Math 3210
Dr. Zeng
Department of Mathematics
California State University, Bakersfield

Outline

- PROC SORT
- PROC PRINT
- PROC MEANS
- PROC FREQ
- PROC SGPLOT

Subsetting in Procedures with the WHERE statement

- The WHERE statement in PROC step tells SAS to read a subset of the data.
- While subsetting Ifs work only in DATA steps, the WHERE statement works in PROC steps.
- Unlike subsetting in a DATA step, using a WHERE statement in a procedure does not create a new data set.
- The basic form of a WHERE statement is

```
WHERE condition;
```

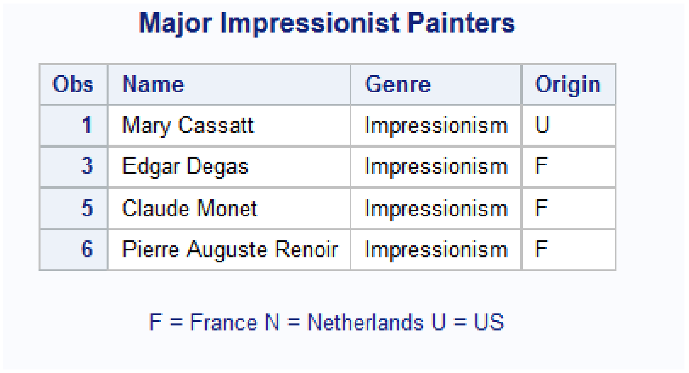
- It is similar to a subsetting IF.

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. Suppose a day later you wanted to print a list of just the impressionist painters.

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

```
DATA Artists;
  INFILE '/home/bzeng/my_content/Artists.dat';
  INPUT Name $ 1-21 Genre $ 23-40 Origin $ 42;
RUN;

PROC PRINT DATA = Artists;
  WHERE Genre = 'Impressionism';
  TITLE 'Major Impressionist Painters';
  FOOTNOTE 'F = France N = Netherlands U = US';
RUN;
```



Major Impressionist Painters

Obs	Name	Genre	Origin
1	Mary Cassatt	Impressionism	U
3	Edgar Degas	Impressionism	F
5	Claude Monet	Impressionism	F
6	Pierre Auguste Renoir	Impressionism	F

F = France N = Netherlands U = US

Sorting Your Data with PROC SORT

- Sorting your data is an important task when organizing data for a report.
- The basic form is

```
PROC SORT;
    BY variable-list;
```

- You can specify as many BY variables as you wish.
- With one BY variable, SAS sorts the data based on the values of that variable.
- With more than one variable, SAS sorts observations by the first variable, then by the second variable within categories of the first, and so on.

Controlling the output data set

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

- The DATA= and OUT= options specify the input and output data sets.
- Without the DATA= option, SAS will use the most recently created data set. If you don’t specify the OUT= option, 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.
- When use the DUPOUT=option, SAS will put the deleted observations in that data set.

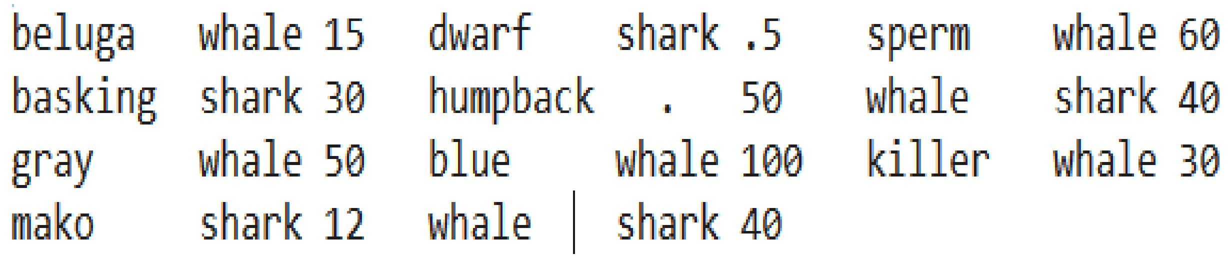
Ascending versus descending sorts

- By default, SAS sorts data in ascending order, from lowest to highest.
- If you prefer the opposite order, add the keyword DESCENDING to the BY statement before each variable that should be sorted in reverse order.
- For example, the following statement tells SAS to sort first by State (from A-Z) and then City (from Z to A) within State:

```
BY State DESCENDING city;
```

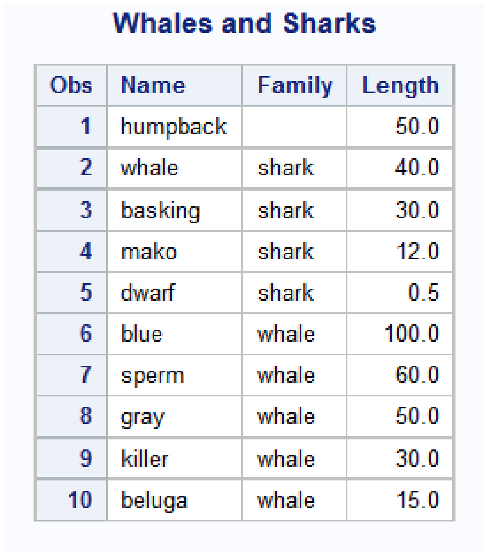
Example

The following data show the typical length in feet of selected whales and sharks. Notice that each line includes data for more than one species. Create a data set named seasort which rearranges the observations by Family in ascending order, and by Length in descending order.



```
beluga   whale 15   dwarf    shark .5    sperm    whale 60
basking  shark 30   humpback   .   50    whale    shark 40
gray     whale 50   blue     whale 100   killer   whale 30
mako     shark 12   whale    shark 40
```

```
DATA marine;
   INFILE '/home/bzeng/my_content/Lengths.dat';
   INPUT Name $ Family $ Length @@;
RUN;
* Sort the data;
PROC SORT DATA = marine OUT = seasort
NODUPKEY;
   BY Family DESCENDING Length;
PROC PRINT DATA = seasort;
   TITLE 'Whales and Sharks';
RUN;
```

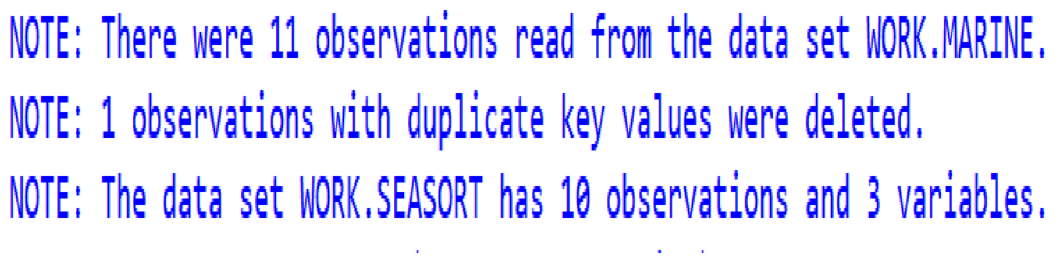


Whales and Sharks

Obs	Name	Family	Length
1	humpback		50.0
2	whale	shark	40.0
3	basking	shark	30.0
4	mako	shark	12.0
5	dwarf	shark	0.5
6	blue	whale	100.0
7	sperm	whale	60.0
8	gray	whale	50.0
9	killer	whale	30.0
10	beluga	whale	15.0

Note

- The missing values are always low for both numeric and character variables.
- The NODUPKEY option eliminated a duplicate observation for the whale shark.
- Messages from the log:



```
NOTE: There were 11 observations read from the data set WORK.MARINE.
NOTE: 1 observations with duplicate key values were deleted.
NOTE: The data set WORK.SEASORT has 10 observations and 3 variables.
```

Changing the Sort Order for Character Data

Here is the basic sort orders for character data from lowest to highest:

- ASCII (default)

Blank→numerals→uppercase letters → lowercase letters

- EBCDIC

Blank→lowercase letters →uppercase letters →numerals

To change the sort order, we can use the options SORTSEQ=EBCDIC and SORTSEQ=ASCII. For example,

```
PROC SORT SORTSEQ=EBCDIC;
```

- Linguistic sorting: by default, upper and lowercase letters will be sorted separately. To ignore case, you can use the SORTSEQ=LINGUISTIC option with the STRENGTH=PRIMARY suboption.
- For example,

```
PROC SORT SORTSEQ=LINGUISTIC (STRENGTH=PRIMARY);
```

unsorted order	**ASCII**	**Linguistic Sort (strength=primaray)**
ella	ANNA	amanda
amanda	Zoe	ANNA
Zoe	amanda	ella
ANNA	ella	Zoe

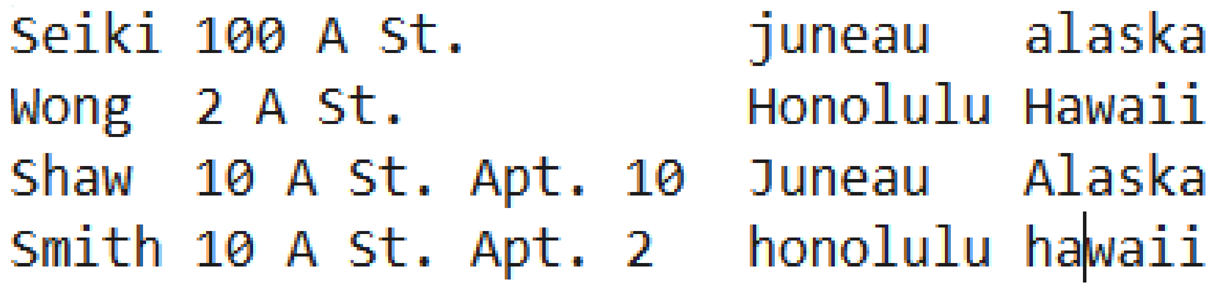
- When numerals are sorted as character data, the value “10” comes before “2”. To treat numerals as their numeric equivalent, use the NUMERIC_COLLATION=ON suboption.
- For example,

PROC SORT SORTSEQ=LINGUISTIC (NUMERIC_COLLATION=ON)

unsorted order	ASCII	Linguistic Sort (NUMERIC_COLLATION=ON)
1500m freestyle	100m backstroke	50m freestyle
200m breaststroke	1500m freestyle	100m backstroke
100m backstroke	200m breaststroke	200m breaststroke
50mm freestyle	50m freestyle	1500m freestyle

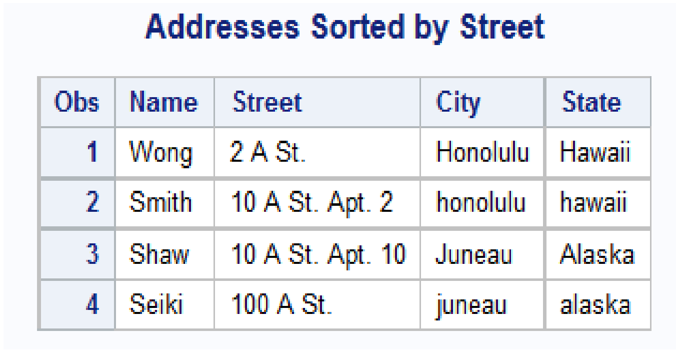
Example

The following data contain names and addresses. First sort the data by street using numeric collation, and then by state ignoring case.



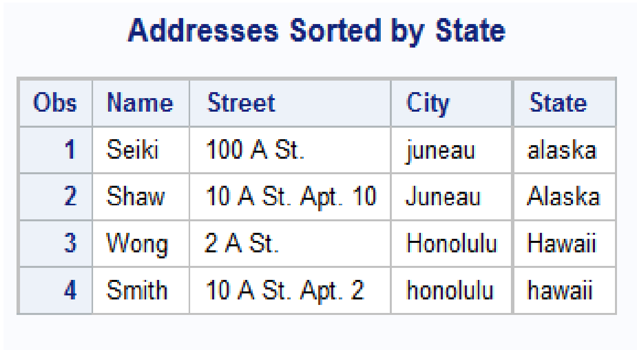
```
Seiki 100 A St.          juneau   alaska
Wong  2 A St.            Honolulu Hawaii
Shaw  10 A St. Apt. 10   Juneau   Alaska
Smith 10 A St. Apt. 2    honolulu hawaii
```

```
DATA addresses;
  INFILE '/home/bzeng/my_content/Mail.dat';
  INPUT Name $6. Street $18. City $9. State $6.;
RUN;

PROC SORT DATA = addresses OUT = sortone
    SORTSEQ = LINGUISTIC (NUMERIC_COLLATION = ON);
  BY Street;
PROC PRINT DATA = sortone;
  TITLE 'Addresses Sorted by Street';
RUN;

PROC SORT DATA = addresses OUT = sorttwo
    SORTSEQ = LINGUISTIC (STRENGTH = PRIMARY);
  BY State;
PROC PRINT DATA = sorttwo;
  TITLE 'Addresses Sorted by State';
RUN;
```



Addresses Sorted by Street

Obs	Name	Street	City	State
1	Wong	2 A St.	Honolulu	Hawaii
2	Smith	10 A St. Apt. 2	honolulu	hawaii
3	Shaw	10 A St. Apt. 10	Juneau	Alaska
4	Seiki	100 A St.	juneau	alaska



Addresses Sorted by State

Obs	Name	Street	City	State
1	Seiki	100 A St.	juneau	alaska
2	Shaw	10 A St. Apt. 10	Juneau	Alaska
3	Wong	2 A St.	Honolulu	Hawaii
4	Smith	10 A St. Apt. 2	honolulu	hawaii

<u>Practice</u>: what will the result be without using these options?

Printing Your Data with PROC PRINT

By default, SAS prints the observation numbers along with the variables' values. If you don't want observation numbers, use the NOOBS option in the PROC PRINT statement.

For example,

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

The following are optional statements that sometimes come in handy:

- **BY variable-list;**

The BY statement 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 **sorted** first by the BY variables in the PROC SORT statement.

- **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;**

The sum statement prints sums for the variables in the list.

- **VAR variable-list;**

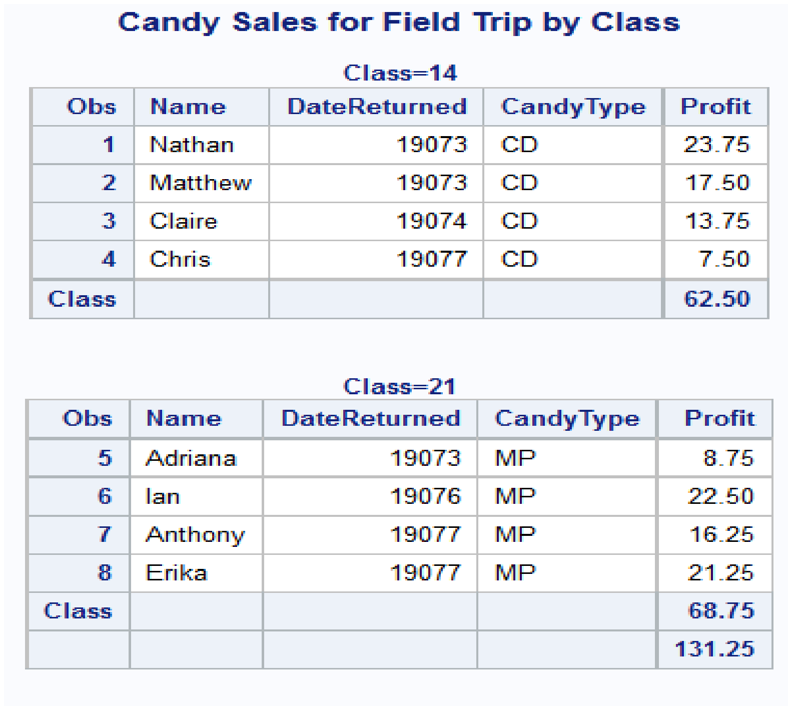
The VAR statement specifies which variables to print and the order. Without a VAR statement, all variables in the SAS data set are printed in the order that they occur in the data set.

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 data they turned in their money, the type of candy: mint patties or chocolate dinosaurs, and the number of boxes 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.

```
Adriana    21  3/21/2012 MP  7
Nathan     14  3/21/2012 CD 19
Matthew    14  3/21/2012 CD 14
Claire     14  3/22/2012 CD 11
Ian        21  3/24/2012 MP 18
Chris      14  3/25/2012 CD  6
Anthony    21  3/25/2012 MP 13
Erika      21  3/25/2012 MP 17
```

```
DATA sales;
   INFILE '/home/bzeng/my_content/CandySales.dat';
   INPUT Name $ 1-11 Class @15 DateReturned MMDDYY10.
CandyType $ Quantity;
   Profit = Quantity * 1.25;

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;
```



Candy Sales for Field Trip by Class

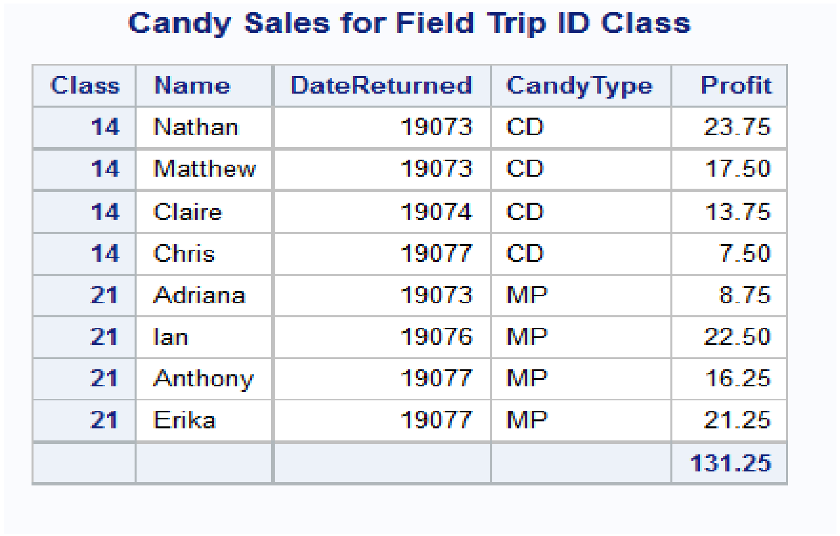
Class=14

Obs	Name	DateReturned	CandyType	Profit
1	Nathan	19073	CD	23.75
2	Matthew	19073	CD	17.50
3	Claire	19074	CD	13.75
4	Chris	19077	CD	7.50
Class				62.50

Class=21

Obs	Name	DateReturned	CandyType	Profit
5	Adriana	19073	MP	8.75
6	Ian	19076	MP	22.50
7	Anthony	19077	MP	16.25
8	Erika	19077	MP	21.25
Class				68.75
				131.25

```
DATA sales;
  INFILE '/home/bzeng/my_content/CandySales.dat';
  INPUT Name $ 1-11 Class @15 DateReturned MMDDYY10.
CandyType $ Quantity;
  Profit = Quantity * 1.25;

PROC SORT DATA = sales;
  BY Class;

PROC PRINT DATA = sales;
  ID Class;
  SUM Profit;
  VAR Name DateReturned CandyType Profit;
  TITLE 'Candy Sales for Field Trip ID Class';
RUN;
```

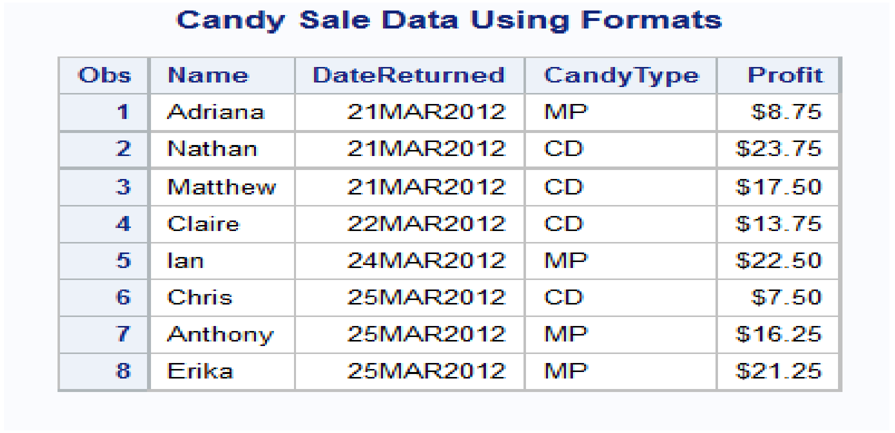

Candy Sales for Field Trip ID Class

Class	Name	DateReturned	CandyType	Profit
14	Nathan	19073	CD	23.75
14	Matthew	19073	CD	17.50
14	Claire	19074	CD	13.75
14	Chris	19077	CD	7.50
21	Adriana	19073	MP	8.75
21	Ian	19076	MP	22.50
21	Anthony	19077	MP	16.25
21	Erika	19077	MP	21.25
				131.25

Example

From the previous example, use the FORMAT statement in the PRINT procedure to print the dates in a readable form. Meanwhile, print the variable Profit using the DOLLAR6.2 format so dollar signs appear before the numbers.

```
DATA sales;
   INFILE '/home/bzeng/my_content/CandySales.dat';
   INPUT Name $ 1-11 Class @15 DateReturned MMDDYY10.
CandyType $ Quantity;
   Profit = Quantity * 1.25;

PROC PRINT DATA = sales;
   VAR Name DateReturned CandyType Profit;
   FORMAT DateReturned DATE9. Profit DOLLAR6.2;
   TITLE 'Candy Sale Data Using Formats';
RUN;
```



Candy Sale Data Using Formats

Obs	Name	DateReturned	CandyType	Profit
1	Adriana	21MAR2012	MP	$8.75
2	Nathan	21MAR2012	CD	$23.75
3	Matthew	21MAR2012	CD	$17.50
4	Claire	22MAR2012	CD	$13.75
5	Ian	24MAR2012	MP	$22.50
6	Chris	25MAR2012	CD	$7.50
7	Anthony	25MAR2012	MP	$16.25
8	Erika	25MAR2012	MP	$21.25

Writing Simple Custom Reports

You can write data in a DATA step the same way you read data by using a FILE statement and PUT statements (instead of INFILE and INPUT). Here is the general form of a FILE statement for creating a report:

```
FILE ‘file-specification’ PRINT;
```

Where ‘file-specification’ specifies the location and name of the generated report

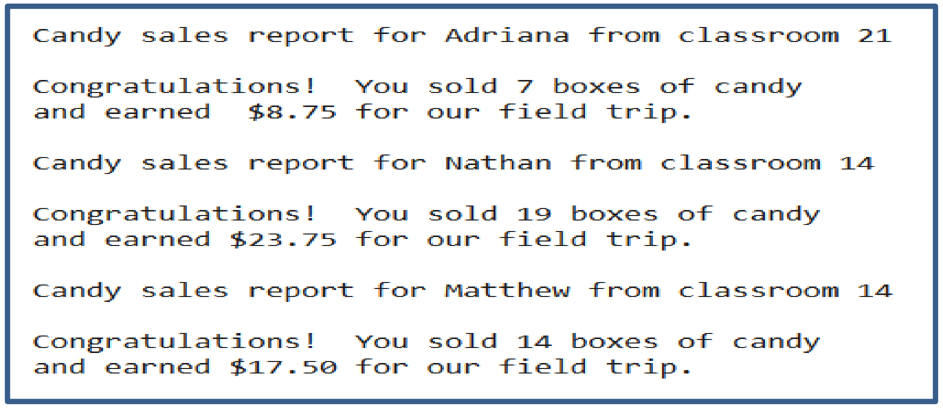
PUT statements

- It allows list, column or formatted style.
- You don’t have to put a $ after character variables.
- For list inputs, SAS will automatically put a space between each variable.
- For column and formatted inputs, SAS will put the variables wherever you specify.
- Use the same pointer controls that INPUT statements use to control spacing such as @n, +n, #n, and /.
- To insert a text string, you need to enclosing it in quotation marks.

Example

Based on the candy sales example, the teachers want a report for each student showing how much money that student earned. They want each student’s report on a separate page so it is easy to hand out. Lastly, they want it to be easy for fourth grades to understand, with complete sentences.

```
DATA _NULL_;
  INFILE '/home/bzeng/my_content/CandySales.dat';
  INPUT Name $ 1-11 Class @15 DateReturned MMDDYY10.
CandyType $ Quantity;
  Profit = Quantity * 1.25;
  FILE '/home/bzeng/my_content/Student.txt' PRINT;
  TITLE;
  PUT @5 'Candy sales report for ' Name 'from classroom ' Class
   // @5 'Congratulations!  You sold ' Quantity 'boxes of candy'
   / @5 'and earned ' Profit DOLLAR6.2 ' for our field trip.';
  PUT _PAGE_;
RUN;
```



```
Candy sales report for Adriana from classroom 21

Congratulations!  You sold 7 boxes of candy
and earned  $8.75 for our field trip.

Candy sales report for Nathan from classroom 14

Congratulations!  You sold 19 boxes of candy
and earned $23.75 for our field trip.

Candy sales report for Matthew from classroom 14

Congratulations!  You sold 14 boxes of candy
and earned $17.50 for our field trip.
```

Remark:

- The keyword _NULL_ appears in the DATA statement tells SAS not to bother writing a SAS data set since the goal is to create a report, which makes the program run faster.
- The PRINT option tells SAS to include carriage returns and page breaks.
- Two slash lines indicate SAS to skip two lines. You could also use multiple PUT statements instead of slashes to skip lines because SAS goes to a new line every time there is a new PUT statement.
- The statement PUT_PAGE_ inserts a page break after each student's report.

Summarizing Your Data Using PROC MEANS

The MEANS procedure provides simple statistics for numeric variables. The general form of the procedure:

$$\boxed{\text{PROC MEANS options;}}$$

Here are some of the options:

MAXDEC=n	specifies the number of decimal places to be displayed
MISSING	treats missing values as valid summary groups
MAX (default)	maximum value
MIN (default)	minimum value
MEAN (default)	mean
MEDIAN	median
MODE	mode
N	unmber of non-missing values
NMISS (default)	number of missing values
RANGE	range
STDDEV (default)	standard deviation
SUM	sum

By default, PROC MEANS statement will generate statistics for all numeric variables in your data set. To control which variables are used, here are some of the optional statements:

- **BY variable-list;**

The BY statement performs separate analyses for each level of the variables in the list. The data must be **sorted** first by the BY variables in the PROC SORT statement.

- **CLASS variable-list;**

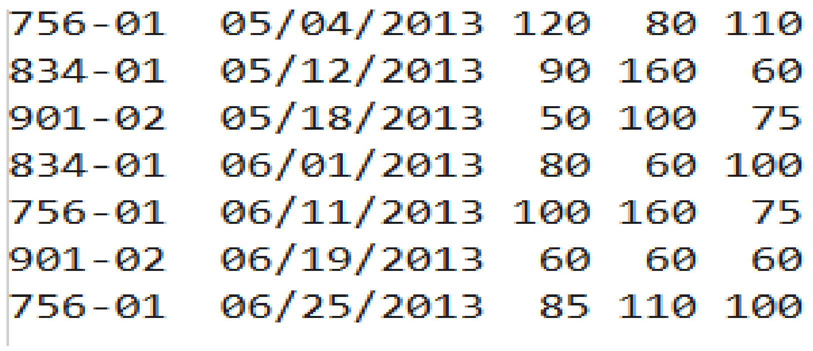
The CLASS statement also performs separate analyses for each level of the variables in the list, but its output is more compact than with the BY statement, and the data do not have to be sorted first.

- **VAR variable-list;**

The VAR statement specifies which numeric variables to use in the analysis. If it is absent, then SAS uses all numeric variables.

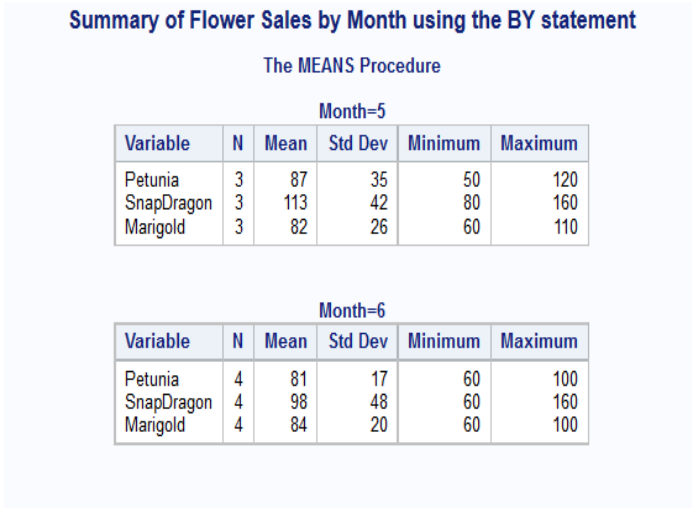
Example

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, and number of petunias, snapdragons, and marigolds sold:



```
756-01  05/04/2013 120  80 110
834-01  05/12/2013  90 160  60
901-02  05/18/2013  50 100  75
834-01  06/01/2013  80  60 100
756-01  06/11/2013 100 160  75
901-02  06/19/2013  60  60  60
756-01  06/25/2013  85 110 100
```

```
/* Program 1*/
DATA sales;
   INFILE '/home/bzeng/my_content/Flowers.dat';
   INPUT CustID $ @9 SaleDate MMDDYY10. Petunia SnapDragon Marigold;
   Month = MONTH(SaleDate);
PROC SORT DATA = sales;
   BY Month;
* Calculate means by Month for flower sales;
PROC MEANS DATA = sales MAXDEC = 0;
   BY Month;
   VAR Petunia SnapDragon Marigold;
   TITLE 'Summary of Flower Sales by Month using the BY statement';
RUN;
```



Summary of Flower Sales by Month using the BY statement

The MEANS Procedure

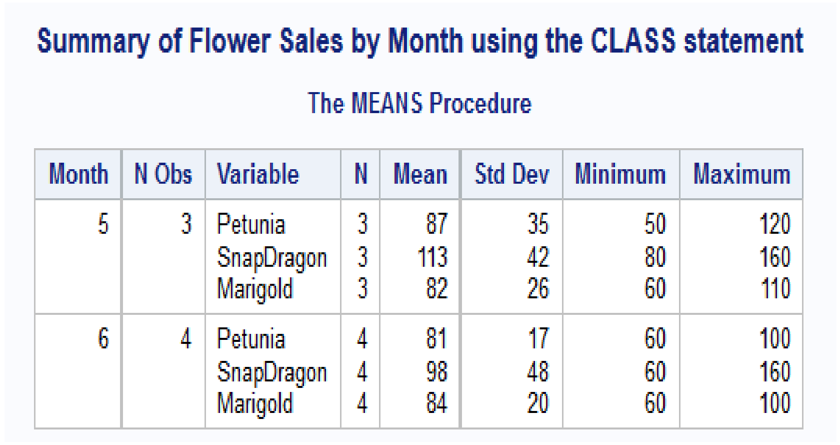
Month=5

Variable	N	Mean	Std Dev	Minimum	Maximum
Petunia	3	87	35	50	120
SnapDragon	3	113	42	80	160
Marigold	3	82	26	60	110

Month=6

Variable	N	Mean	Std Dev	Minimum	Maximum
Petunia	4	81	17	60	100
SnapDragon	4	98	48	60	160
Marigold	4	84	20	60	100

```
/* Program 2*/
DATA sales;
   INFILE '/home/bzeng/my_content/Flowers.dat';
   INPUT CustID $ @9 SaleDate MMDDYY10. Petunia SnapDragon
Marigold;
   Month = MONTH(SaleDate);
PROC MEANS DATA = sales MAXDEC = 0;
   CLASS Month;
   VAR Petunia SnapDragon Marigold;
   TITLE 'Summary of Flower Sales by Month using the CLASS
statement';
RUN;
```



Summary of Flower Sales by Month using the CLASS statement

The MEANS Procedure

Month	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
5	3	Petunia	3	87	35	50	120
		SnapDragon	3	113	42	80	160
		Marigold	3	82	26	60	110
6	4	Petunia	4	81	17	60	100
		SnapDragon	4	98	48	60	160
		Marigold	4	84	20	60	100

Counting Your Data with PROC FREQ

- A frequency table is a simple list of counts answering the question "How many?" When you have counts for one variable, they are called one-way frequencies. When you combine two or more variables, the counts are called two-way frequencies, three-way frequencies, and so on. Tables combining two or more variables are also called cross-tabulations or contingency tables.
- The basic form of PROC FREQ is

```
PROC FREQ;
      TABLES   variable-combinations / options;
```

- For example, this statement produces a frequency table listing the number of observations for each value of YearsEducation:

```
TABLES  YearsEducation;
```

- To produce a cross-tabulation, list the variables separated by an asterisk.

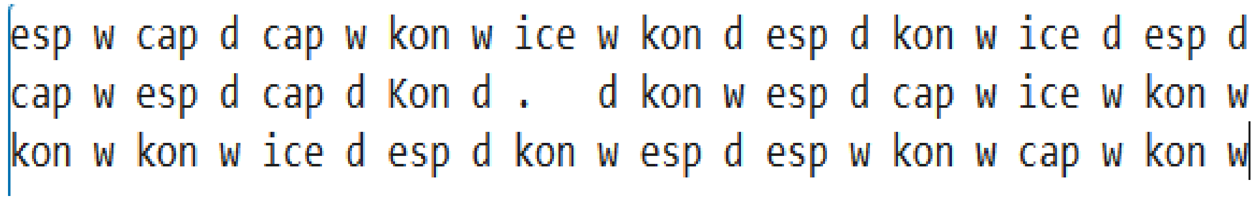
```
TABLES  gender*YearsEducation;
```

For PROC FREQ, options appear after a slash in the TABLES statement. Options for controlling the output of PROC FREQ include

LIST	prints cross-tabulations in list format rather than grid
MISSPRINT	includes missing values in frequencies but not in percentages
MISSING	includes missing values in frequencies and in percentages
NOCOL	suppresses printing of column percentages in cross-tabulations
NOPERCENT	suppresses printing of percentage
NOROW	suppresses printing of row percentages in cross-tabulations
OUT=data-set	writes a data set containing frequencies

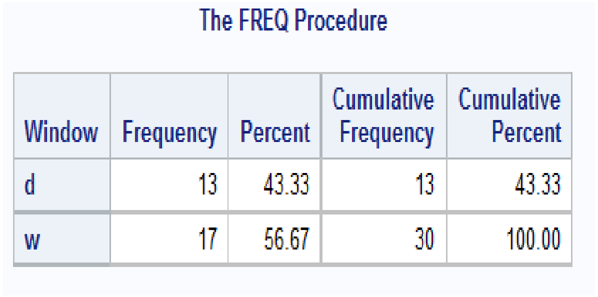
Example

The proprietor of a coffee shop keeps a record of sales. For each drink sold, she records the type of coffee (cappuccino, espresso, kona, or iced coffee), and whether the customer walked in or came to the drive-up window. Here are the data with ten observations per line.



```
esp w cap d cap w kon w ice w kon d esp d kon w ice d esp d
cap w esp d cap d Kon d .   d kon w esp d cap w ice w kon w
kon w kon w ice d esp d kon w esp d esp w kon w cap w kon w
```

```
DATA orders;
  INFILE '/home/bzeng/my_content/Coffee.dat';
  INPUT Coffee $ Window $ @@;
/*Print tables for Window and Window by Coffee*/
PROC FREQ DATA = orders;
  TABLES Window  Window * Coffee;
RUN;
```



The FREQ Procedure

Window	Frequency	Percent	Cumulative Frequency	Cumulative Percent
d	13	43.33	13	43.33
w	17	56.67	30	100.00

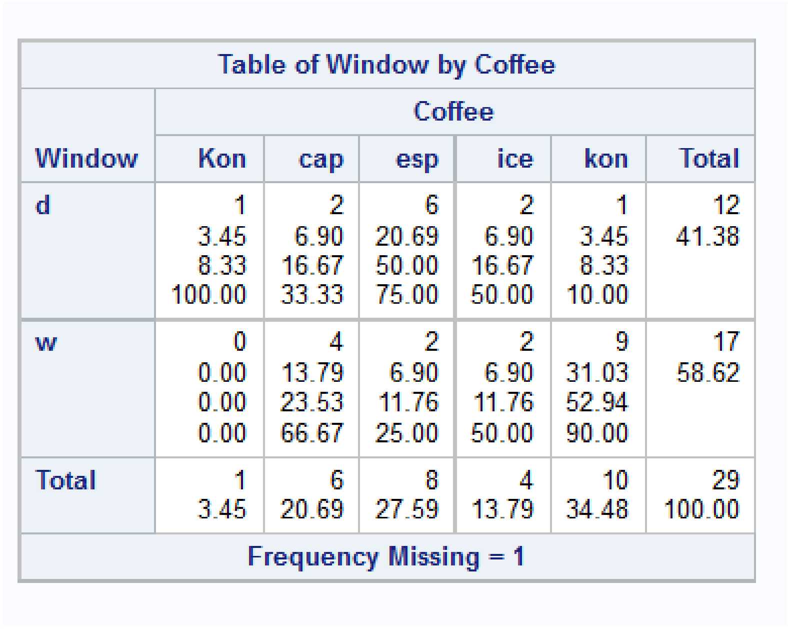


Table of Window by Coffee

Window	Coffee Kon	 cap	 esp	 ice	 kon	 Total
d	1 3.45 8.33 100.00	2 6.90 16.67 33.33	6 20.69 50.00 75.00	2 6.90 16.67 50.00	1 3.45 8.33 10.00	12 41.38
w	0 0.00 0.00 0.00	4 13.79 23.53 66.67	2 6.90 11.76 25.00	2 6.90 11.76 50.00	9 31.03 52.94 90.00	17 58.62
Total	1 3.45	6 20.69	8 27.59	4 13.79	10 34.48	29 100.00

Frequency Missing = 1

- The first is a one-way frequency table for the variable window.
- The second table is a two-way cross-tabulation of window by coffee. Inside each cell, SAS prints the frequency, percentage, percentage for that row, and percentage for that column; while cumulative frequencies and percents appear along the right side and bottom
- Missing value is mentioned but not included in the statistics.
- http://support.sas.com/documentation/cdl/en/procstat/63104/HTML/default/viewer.htm#procstat_freq_sect026.htm
- **Practice:** use the MISSING OR MISSPRINT options if you want missing values to be included in the table. Try other options such as NOCOL, LIST, NOPERCENT, NOROW and OUT.

Creating Bar Charts

Bar charts show the distribution of a categorical variable. The length of each bar is proportional to the number of observations in that category. To create a vertical bar chart, use a VBAR statement with this general form:

```
PROC SGPLOT;
        VBAR variable-name/options;
```

For horizontal bars, specify the keyword HBAR instead of VBAR. Here are some options:

- BARWIDTH=n

sets the width of bars. Values range from 0.1 to 1 with a default 0.8.

- MISSING

includes a bar for missing values.

- GROUP=variable-name

specifies a variable used to group the data.

- GROUPDISPLAY=type

Specifies how to display grouped bars, either STACK (default) or CLUSTER

- RESPONSE=variable-name

specifies a numeric variable to be summarized.

- TRANSPARENCY=n

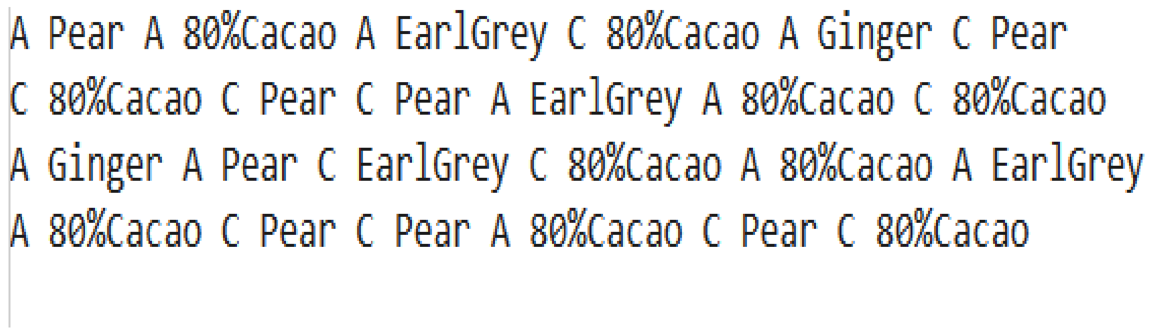
specifies the degree of transparency for the bars from 0 to 1 with a default of 0

- STAT=statistic

Specifies a statistic, either FREQ, MEAN, or SUM. FREQ is the default if there is no response variable. SUM is the default when you specify a response variable.

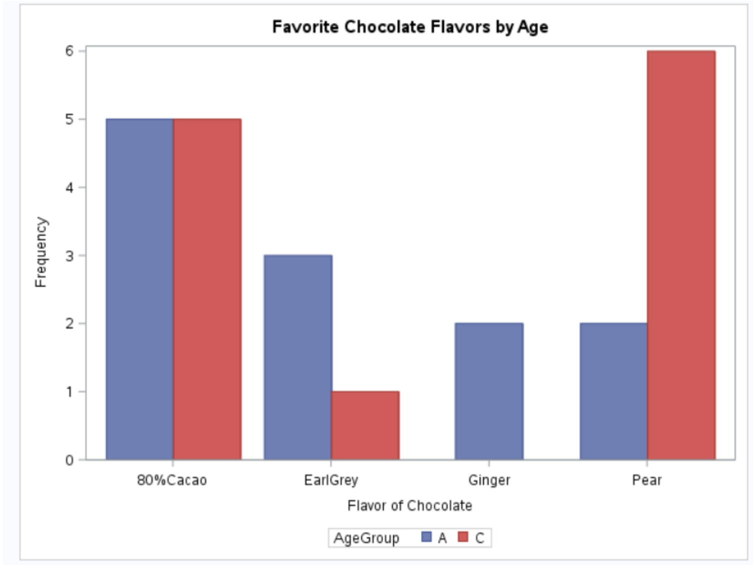
Example

A chocolate manufacturer is considering whether to add four new varieties of chocolate to its line of products. The company asked volunteers to taste the new flavors. The data contain each person’s age group (A for adult, C for Child) followed by their favorite flavor (80%Cacao, Earl Grey, Ginger, or Pear). Notice that each line of data contain six responses.

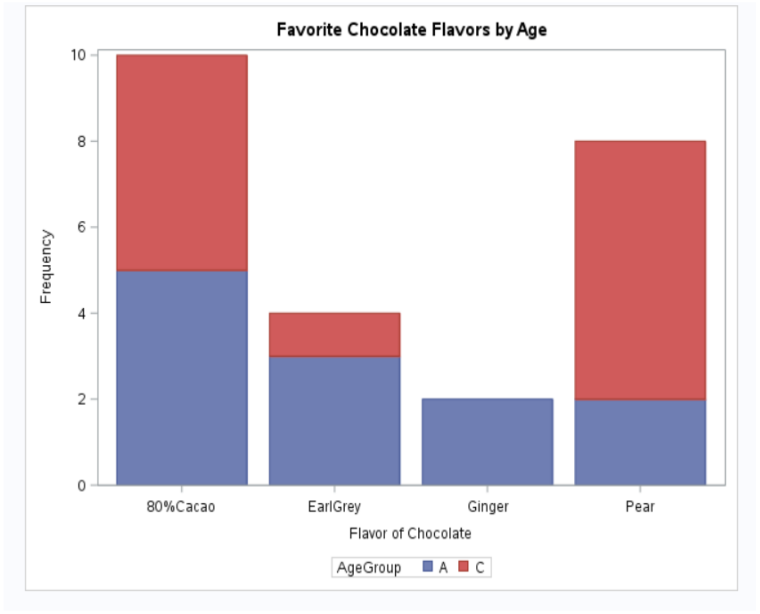


```
A Pear A 80%Cacao A EarlGrey C 80%Cacao A Ginger C Pear
C 80%Cacao C Pear C Pear A EarlGrey A 80%Cacao C 80%Cacao
A Ginger A Pear C EarlGrey C 80%Cacao A 80%Cacao A EarlGrey
A 80%Cacao C Pear C Pear A 80%Cacao C Pear C 80%Cacao
```

```
DATA chocolate;
   INFILE '/home/bzeng/my_content/Choc.dat';
   INPUT AgeGroup $ FavoriteFlavor $ @@;
RUN;


PROC SGPLOT DATA = chocolate;
   VBAR FavoriteFlavor / GROUP = AgeGroup GROUPDISPLAY =
CLUSTER;
   LABEL FavoriteFlavor = 'Flavor of Chocolate';
   TITLE 'Favorite Chocolate Flavors by Age';
RUN;
```



- The LABEL statement replaced the name of the variable FavoriteFlavor with the words ‘Flavor of Chocolate’ in the X axis label.
- FREQ is the default if there is no response variable.
- Without the GROUPDISPLAY=CLUSTER option, the default is STACK. See the bar chart below.



Creating Histograms and Density Curves

Histograms show the distribution of continuous data. In a histogram, the data are divided into discrete intervals called bins. The basic form of a HISTOGRAM statement is:

```
    PROC SGPLOT;
        HISTOGRAM variable-name/options;
```

Possible options include:

- BINSTART=n

Specifies the midpoint for the first bin.

- BINWIDTH=n

Specifies the bin width. This option is ignored if you specify the NBINS=option.

- NBINS=n

Specifies the number of bins.

- SCALE=scaling-type

Specifies the scale for the vertical axis, either PERCENT (the default), COUNT, or PROPORTION.

- SHOWBINS

Place tick marks at the midpoints of the bins. By default, tick marks are placed at regular intervals based on minimum and maximum values.

- TRANSPARENCY=n

specifies the degree of transparency for the bars from 0 to 1 with a default of 0

Density Curves:

You can also plot density curves for your data. The basic form of a DENSITY statement is

```
PROC SGPLOT;
        DENSITY variable-name/options;
```

Common options are:

- TYPE=distribution-type

Specifies the type of distribution curve, either NORMAL (the default) or KERNEL.

- TRANSPARENCY=n

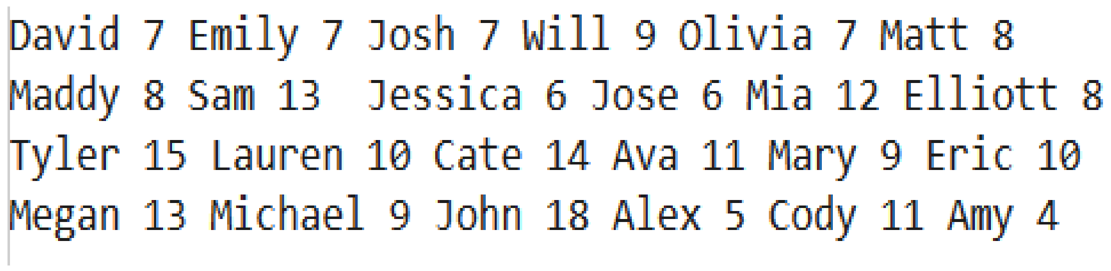
specifies the degree of transparency for the bars from 0 to 1 with a default of 0

Notes:

- The HISTOGRAM and DENSITY statements can be used together, but not with other types of graphs.
- When you overlay graphs, the order of statements is important because the second graph will be drawn on top of the first and could hide it.

Example

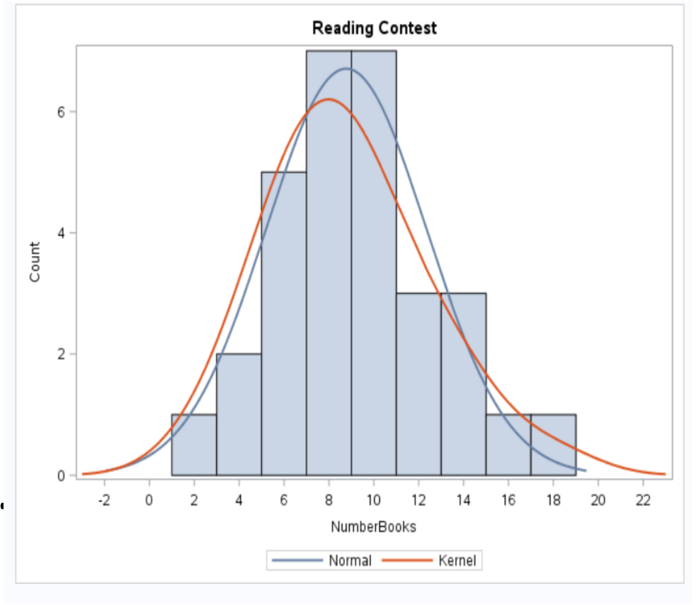
A fourth grade class has a competition to see who can read the most books in one month. For each student, the teacher records the students' name and the number of books read. Notice that each line of data includes six students.



```
David 7 Emily 7 Josh 7 Will 9 Olivia 7 Matt 8
Maddy 8 Sam 13  Jessica 6 Jose 6 Mia 12 Elliott 8
Tyler 15 Lauren 10 Cate 14 Ava 11 Mary 9 Eric 10
Megan 13 Michael 9 John 18 Alex 5 Cody 11 Amy 4
```

```
DATA contest;
  INFILE '/home/bzeng/my_content/Reading.dat';
  INPUT Name $ NumberBooks @@;
RUN;

PROC SGPLOT DATA = contest;
  HISTOGRAM NumberBooks / BINWIDTH = 2 SHOWBINS SCALE =
COUNT;
  DENSITY NumberBooks;
  DENSITY NumberBooks / TYPE = KERNEL;
   TITLE 'Reading Contest';
RUN;
```

<u>Practice: </u> Try other options for this program.


Reading Contest
Count
NumberBooks
Normal
Kernel