PSTAT 130

SAS BASE PROGRAMMING

- Lecture 10 -

Dr. Julie Swenson

Objectives

- PROC GPLOT
  - SYMBOL statement
  - PLOT statement
- Output Delivery System
  - HTML
  - CSV
- More SAS Functions
  - Parse text data
  - Truncate numeric data

PROC GPLOT

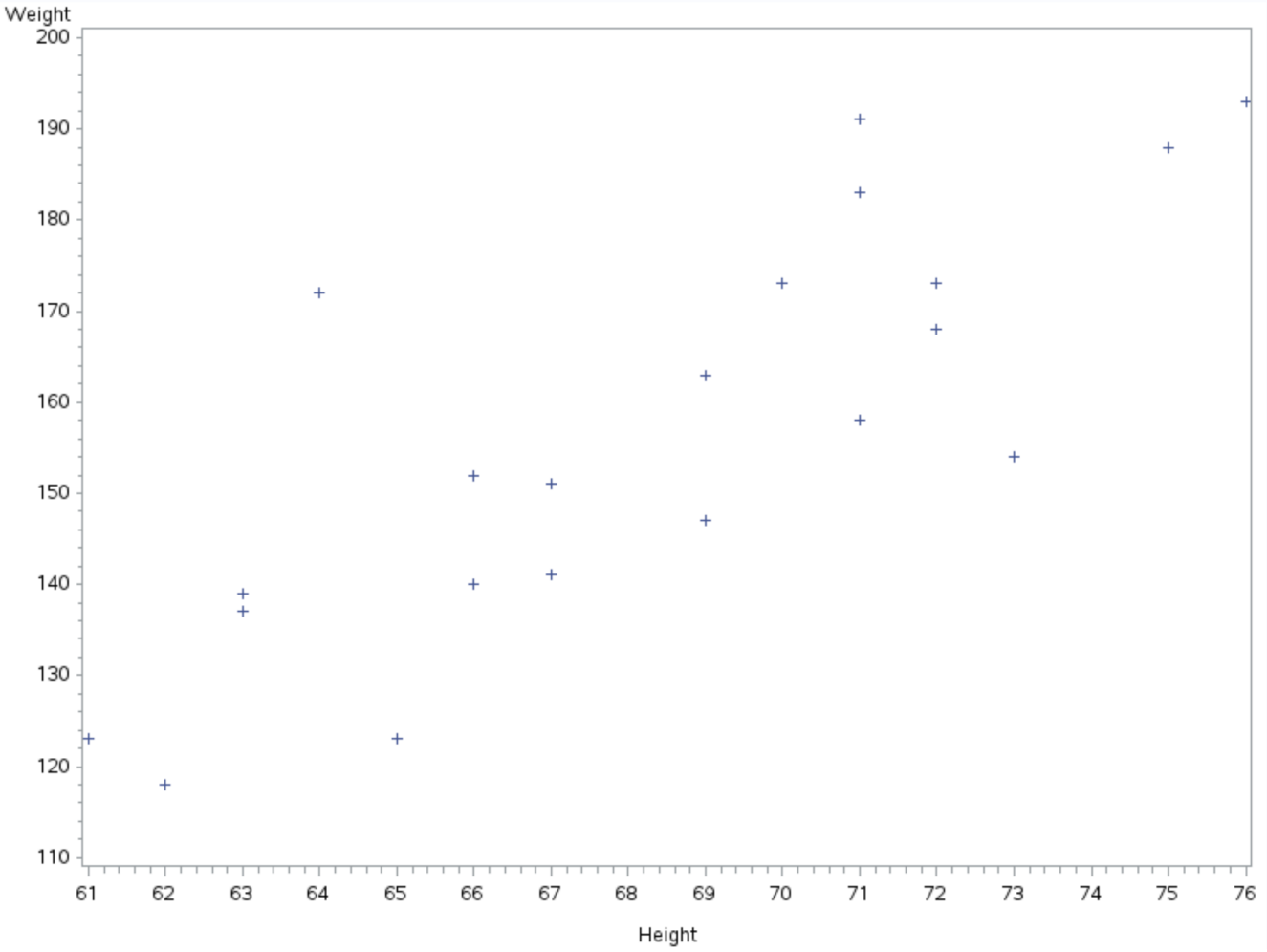
- Use the GPLOT procedure to produce scatterplots and line graphs
- General form

```
PROC GPLOT DATA=SAS-data-set;
   PLOT vertical-variable*horizontal-variable </options>;
RUN;
QUIT;
```

- Example

```
proc gplot data=data1.admit;
   plot weight * height;
run;
quit;
```

GPLOT Example Output

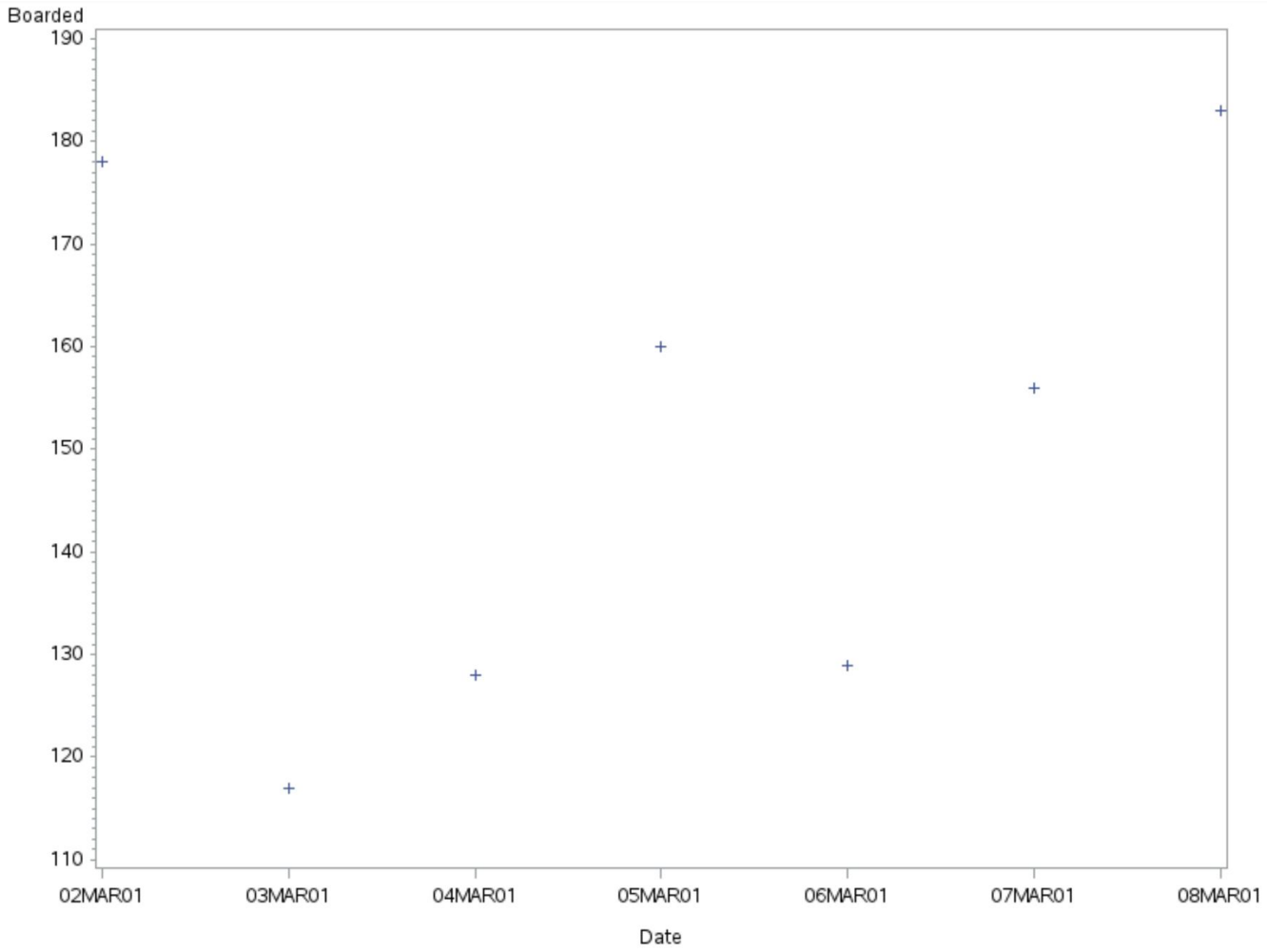


GPLOT Example

- Produce a plot of the number of passengers by date for flight number 114 over a one-week period.

```
proc gplot data=data1.flight114;
   *this selects one week of flights;
   where date between '02mar2001'd and '08mar2001'd;
   plot Boarded*Date;
run;
quit;
```

GPLOT Example Output



SYMBOL Statement

- You can use the SYMBOL statement to do the following:
  - Define plotting symbols
  - Draw lines through the data points
  - Specify the width and color of the plotting symbols and lines

- General Form

```
SYMBOLn options;
```

  - n = 1 - 255

SYMBOL Statement Options

- Options for the shape of the symbol

VALUE=*symbol* | *V=symbol*

- Selected *symbol* values include the following

PLUS	DIAMOND
STAR	TRIANGLE
SQUARE	NONE (no plotting symbol)

SYMBOL Statement Options

- Interpolation

  I=*interpolation*

- Selected *interpolation* values

JOIN	joins the points with straight lines.
SPLINE	joins the points with a smooth line.
NEEDLE	draws vertical lines from the points to the horizontal axes.

- Note: Combining `symbol value=none` with `interpolation=join` produces a line-only plot

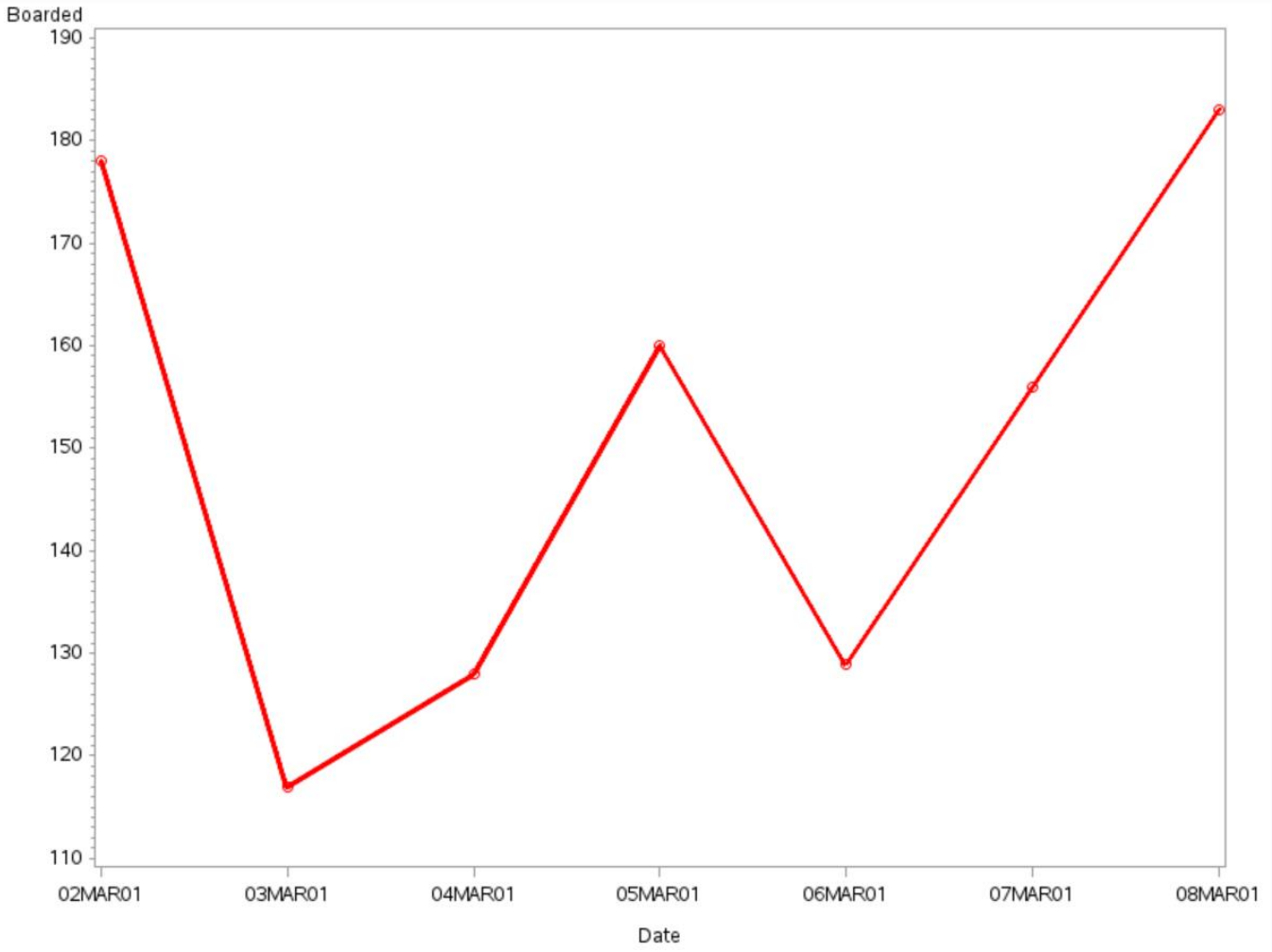
SYMBOL Statement Options

WIDTH=*width* **W=***width*	specifies the thickness of the line.
COLOR=*color* **C=***color*	specifies the color of the line.

- Example

```
proc gplot data=data1.flight114;
   where date between '02mar2001'd and '08mar2001'd;
   plot Boarded*Date;
   symbol value=circle i=join color=red width=2;
run;
quit;
```

SYMBOL Example Output

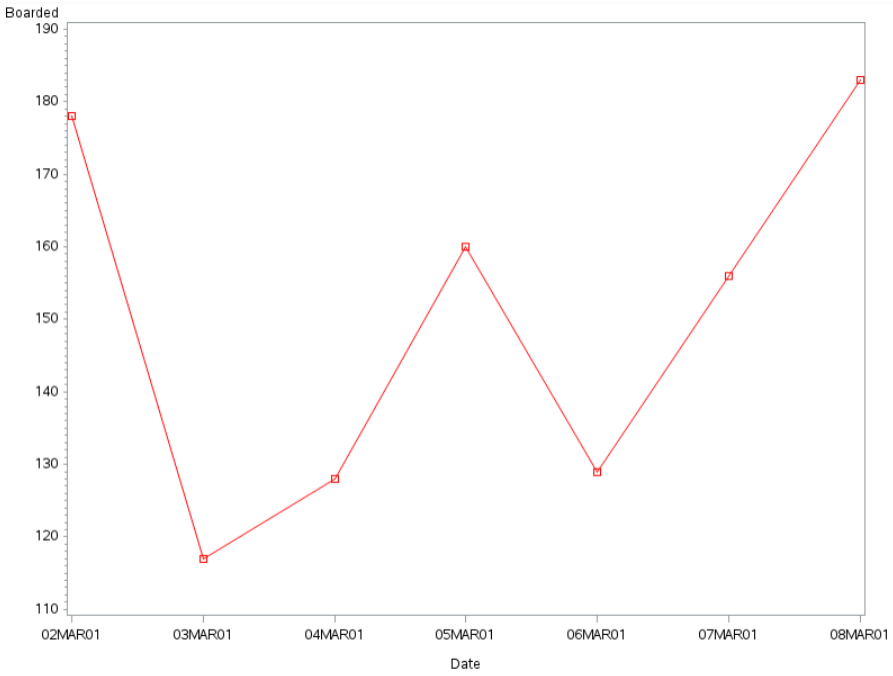


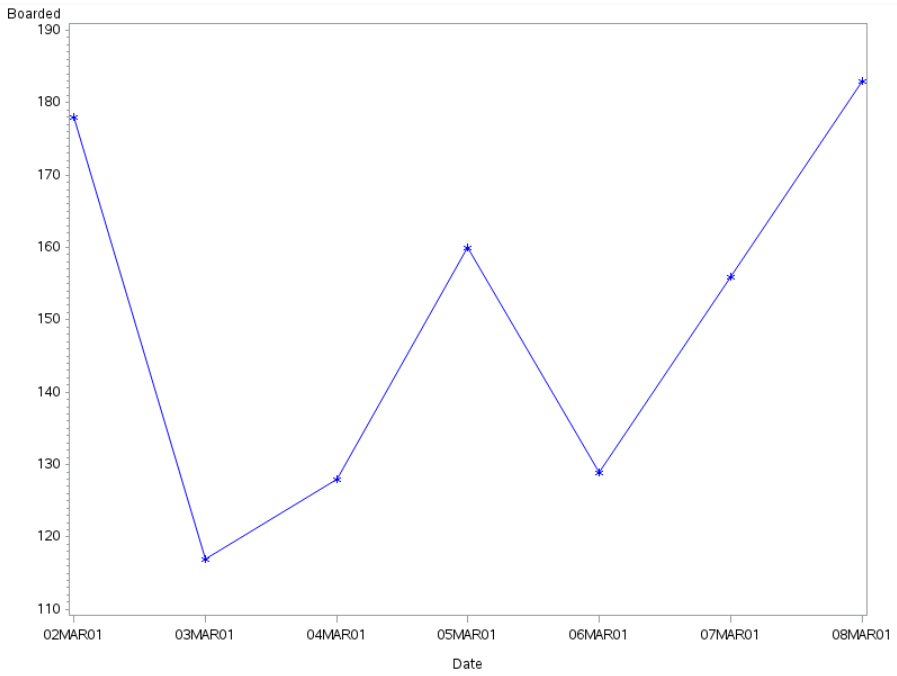
SYMBOL Statement Example

- Create one plot by modifying the previous code to use a red square as the plotting symbol, set the line width to 1, and join the symbols with straight lines.
- Create a second plot by modifying this code to use a blue star as the plotting symbol.

```
proc gplot data=data1.flight114;
   *this selects one week of flights;
   where date between '02mar2001'd and '08mar2001'd;
   plot Boarded*Date;
   symbol c=red v=square w=1 i=join;
run;
   where date between '02mar2001'd and '08mar2001'd;
   plot Boarded*Date;
   symbol c=blue v=star w=1 i=join;
run;
quit;
```

SYMBOL Example Output





SYMBOL Statement

- SYMBOL statements have the following characteristics:

global	After they are defined, they remain in effect until changed or until the end of the SAS session.
additive	Specifying the value of one option does not affect the values of other options.

Modify SYMBOL Options

- Set the attributes for SYMBOL1:

```
symbol1 c=blue v=diamond;
```

- Modify only the color of SYMBOL1, and not the value:

```
symbol1 c=green;
```

- To cancel SYMBOL statements

```
symbol1;
-OR-
goptions reset=symbol;
```

Control the Appearance of the Axis

- You can modify the appearance of the axes that PROC GPLOT produces with the following
  - PLOT statement options

HAXIS=*values*	scales the horizontal axis.
VAXIS=*values*	scales the vertical axis.
CAXIS=*color*	specifies the color of both axes.
CTEXT=*color*	specifies the color of the text on both axes.

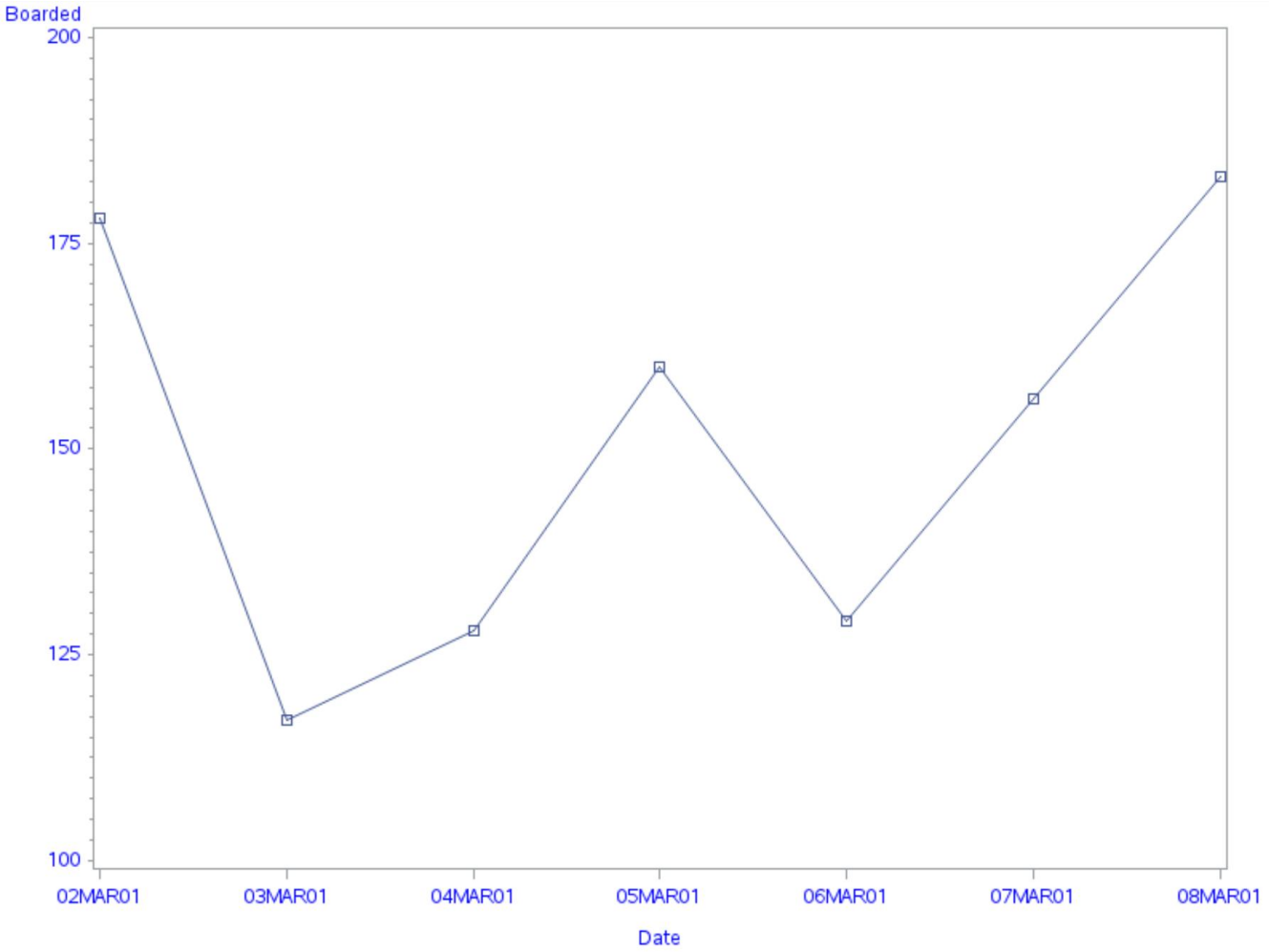
  - The LABEL statement
  - The FORMAT statement

Modify Axis Scale

- Example: Define the scale on the vertical axis and display the axis text in blue.

```
proc gplot data=data1.flight114;
   where date between '02mar2001'd and '08mar2001'd;
   plot Boarded*Date / vaxis=100 to 200 by 25
                       ctext=blue;
   symbol value=square i=join;
run;
quit;
```

Modify Axis Scale Output

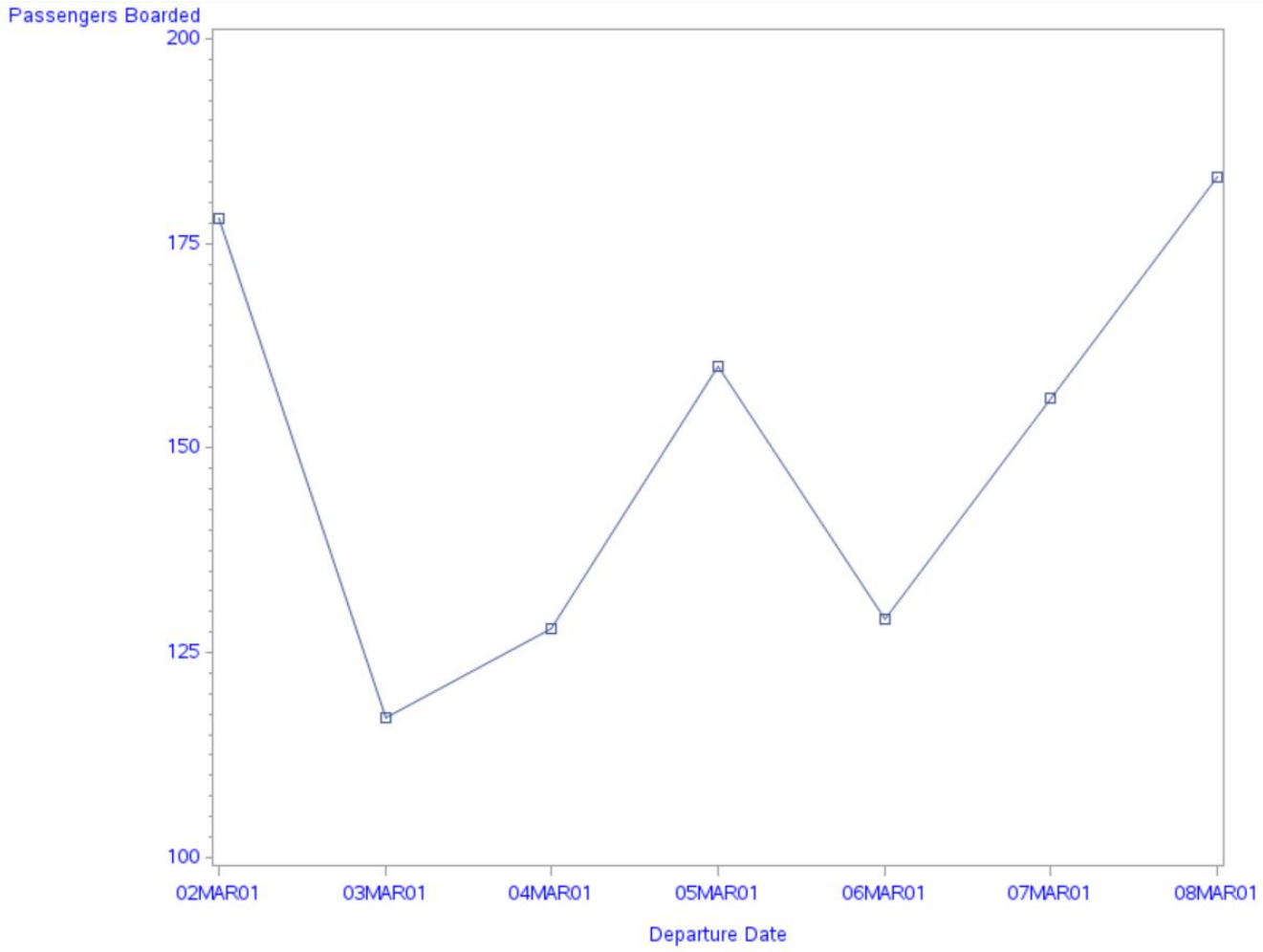


Modify Axis Labels

- Example: Display 'Passengers Boarded' for the variable Boarded, and 'Departure Date' for the variable Date.

```
proc gplot data=data1.flight114;
   where date between '02mar2001'd and '08mar2001'd;
   plot Boarded*Date / vaxis=100 to 200 by 25
                       ctext=blue;
   symbol value=square i=join;
   label Boarded='Passengers Boarded'
         Date='Departure Date';
run;
quit;
```

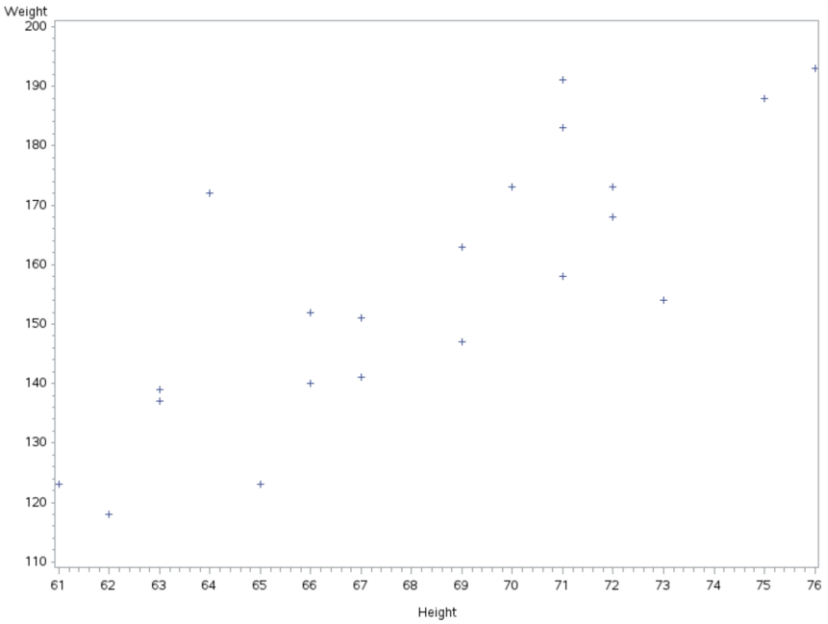
Modify Axis Labels Output



Produce a Scatterplot

- A scatterplot typically plots two continuous variables
- Example

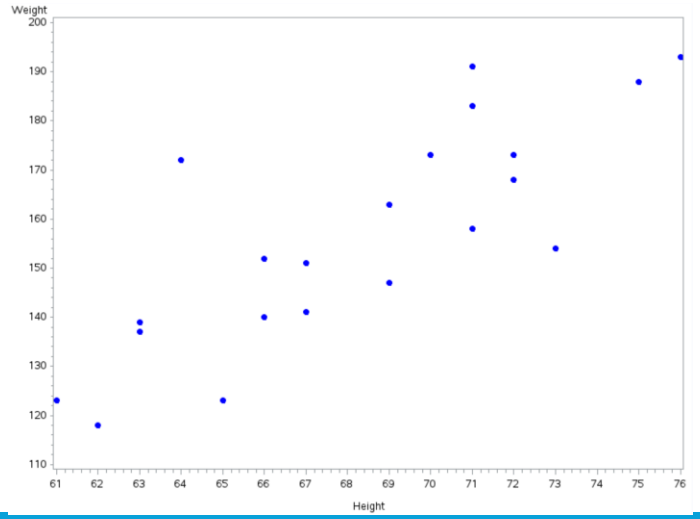
```
proc gplot data=data1.admit;
    plot weight*height;
run;
quit;
```



Add Options

- You can modify the symbol, axis labels and axis tick marks
- You usually do not connect the dots in a scatterplot
- Example

```
proc gplot data=data1.admit;
   plot weight*height;
   symbol v=dot color=blue;
run;
quit;
```



Scatterplot with Regression

- You can also add a
  - Regression equation
  - Regression line
  - Regression line and prediction confidence interval

Regression Options

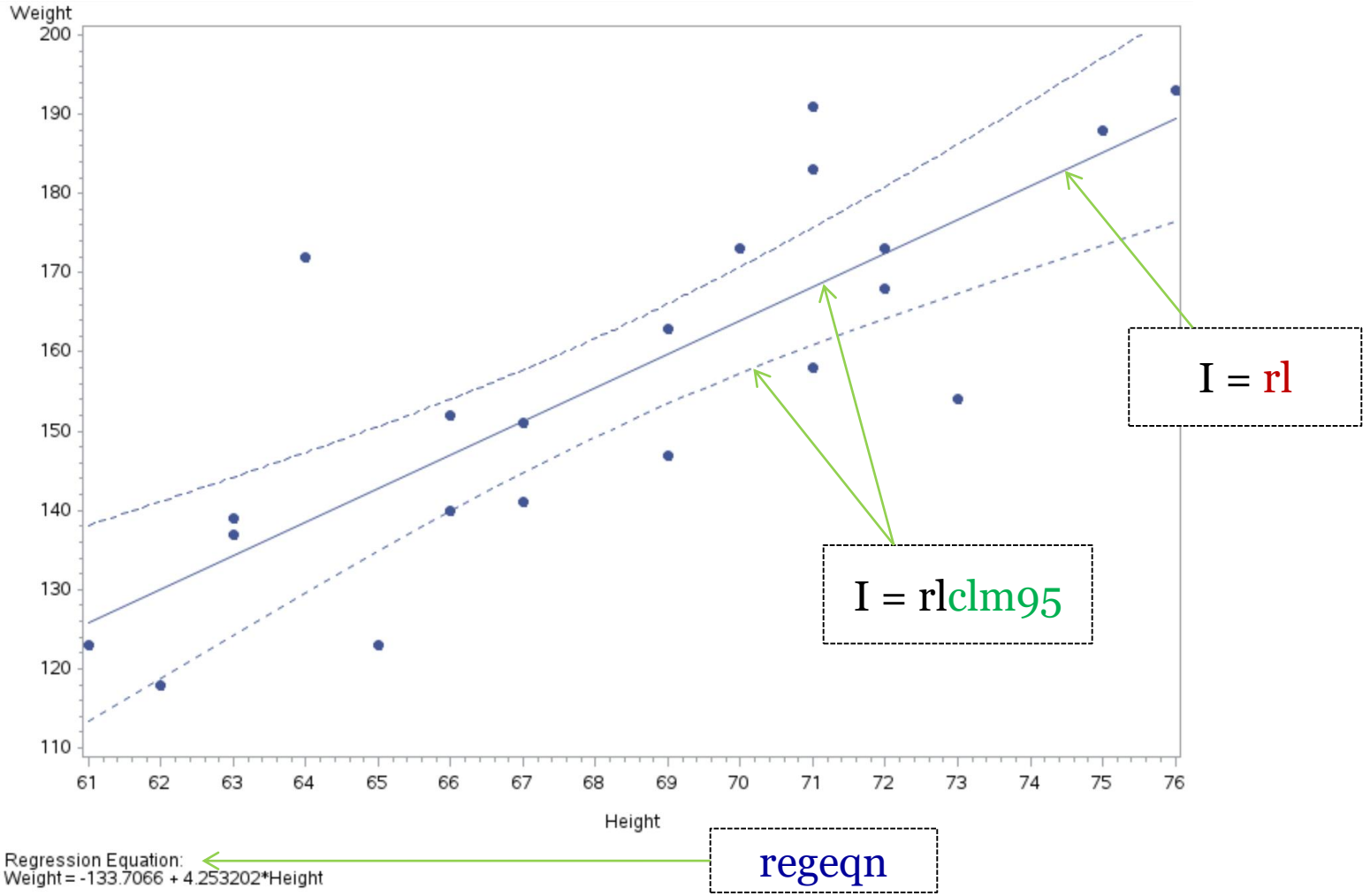
- Regression equation
  - Use `regeqn` as an option to the plot statement
- Regression line (linear)
  - Use an interpolation method of `i=rl`
- Regression line (linear) + Confidence limits
  - Use an interpolation method of `i=rlclm__`
    - Set the confidence level by writing it at the end of the interpolation
    - i.e. 90% CL: `i=rlclm90`

Scatterplot with Regression

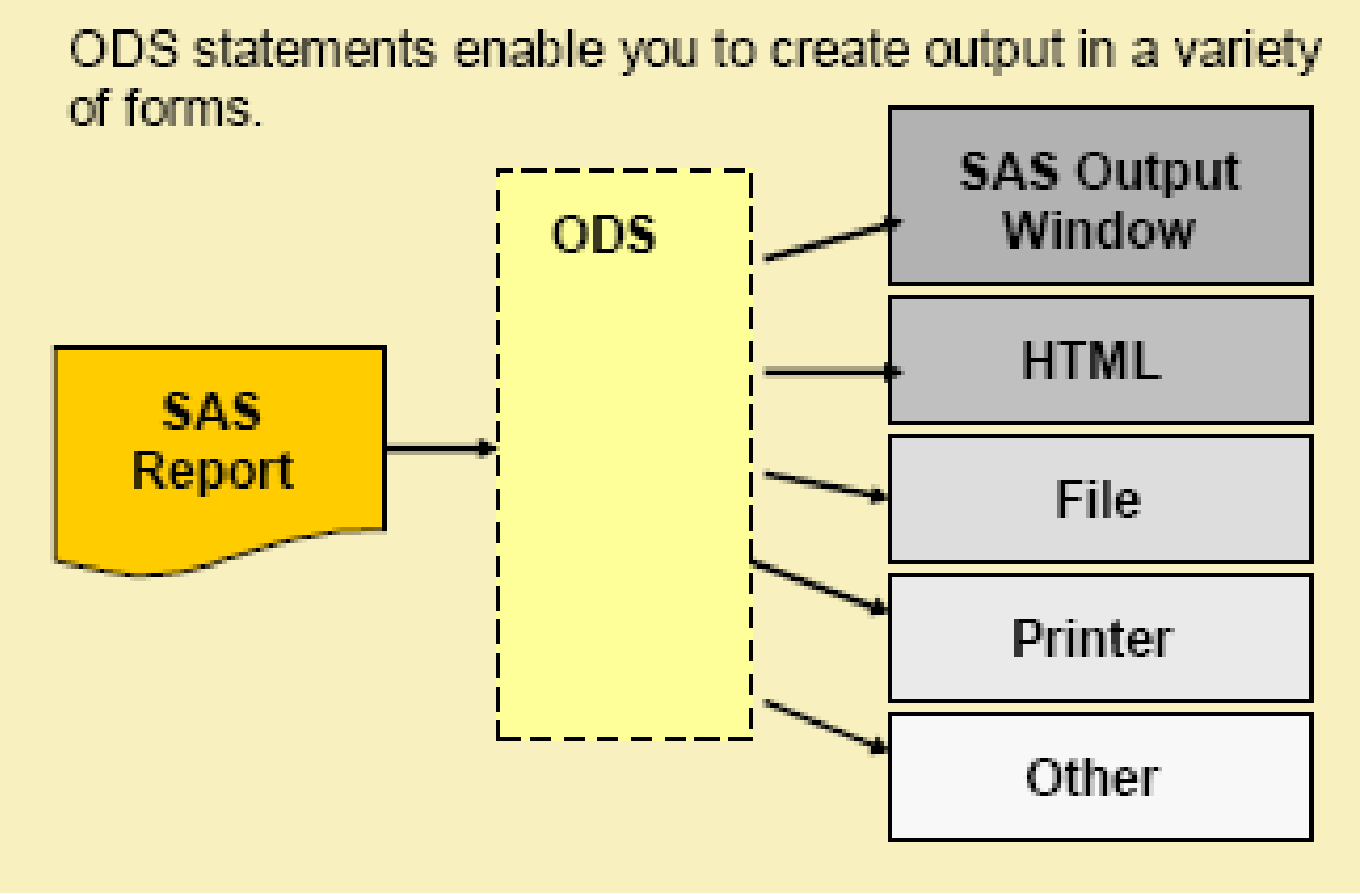
- Example

```
proc gplot data=data1.admit;
   plot weight*height / regeqn;
   symbol v=dot i=rlCLM95;
run;
quit;
```

Scatterplot with Regression Output



The Output Delivery System



Generate a LST File

- The ODS LISTING statement opens, closes, and manages the LST file destination.

- General form of the ODS LISTING statement:

```
ODS LISTING FILE='LST-file-specification' <options>;
    SAS code that generates output
ODS LISTING CLOSE;
```

Generate a HTML File

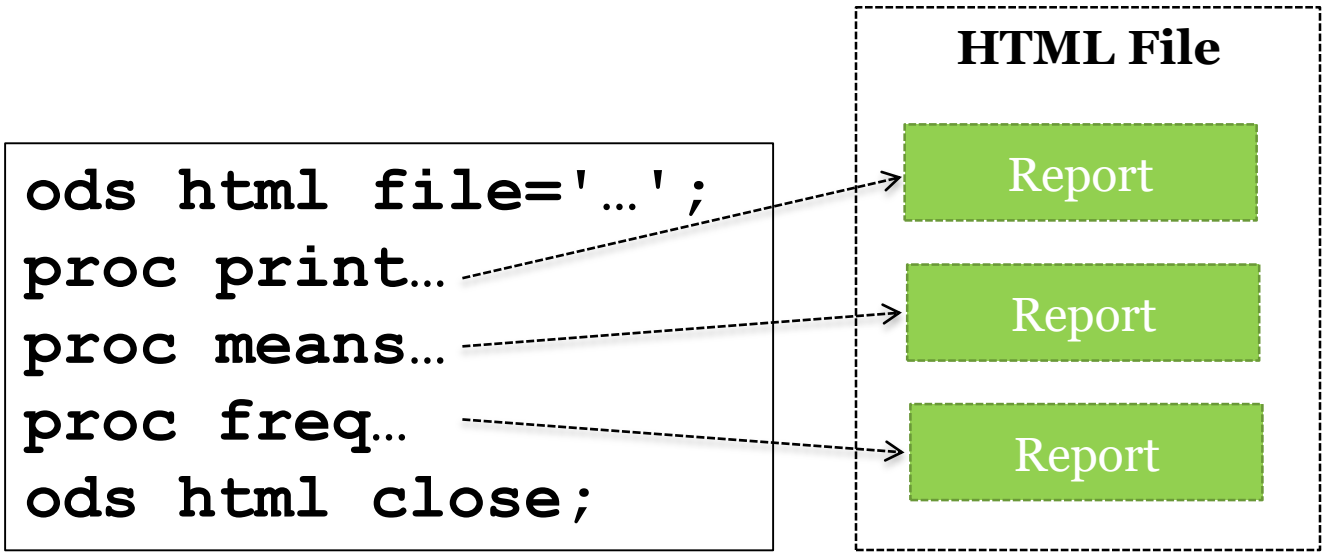
- The ODS HTML statement opens, closes, and manages the HTML destination.

- General form of the ODS HTML statement:

```
ODS HTML FILE='HTML-file-specification' <options>;
    SAS code that generates output
ODS HTML CLOSE;
```

Generate a LST or a HTML File

- Output is directed to the specified LST or HTML file until you
  - Close the LST or HTML destination
  - Specify another destination file



Apply ODS Styles

- ODS Styles are pre-defined formats for output.
- Example:

```
ods html file='output.html' style=analysis;
```

- Complete list of styles:

```
proc template;
   list styles;
run;
```

ODS Style Examples

Analysis Variable : WT_IN_Analysis_style

FEEDTYPE	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	7	7	51.46	4.75	44.80	56.00
2	6	6	54.97	4.79	51.30	64.30

Analysis Variable : WT_IN_Astronomy_style

FEEDTYPE	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	7	7	51.46	4.75	44.80	56.00
2	6	6	54.97	4.79	51.30	64.30

Analysis Variable : WT_IN_Banker_style

FEEDTYPE	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	7	7	51.46	4.75	44.80	56.00
2	6	6	54.97	4.79	51.30	64.30

Analysis Variable : WT_IN_BarrettsBlue_style

FEEDTYPE	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	7	7	51.46	4.75	44.80	56.00
2	6	6	54.97	4.79	51.30	64.30

Analysis Variable : WT_IN_Beige_style

FEEDTYPE	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	7	7	51.46	4.75	44.80	56.00
2	6	6	54.97	4.79	51.30	64.30

Analysis Variable : WT_IN_Brick_style

FEEDTYPE	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	7	7	51.46	4.75	44.80	56.00
2	6	6	54.97	4.79	51.30	64.30

Analysis Variable : WT_IN_Brown_style

FEEDTYPE	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	7	7	51.46	4.75	44.80	56.00
2	6	6	54.97	4.79	51.30	64.30

Analysis Variable : WT_IN_Curve_style

FEEDTYPE	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	7	7	51.46	4.75	44.80	56.00
2	6	6	54.97	4.79	51.30	64.30

Analysis Variable : WT_IN_D3d_style

FEEDTYPE	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	7	7	51.46	4.75	44.80	56.00
2	6	6	54.97	4.79	51.30	64.30

Analysis Variable : WT_IN_Default_style

FEEDTYPE	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	7	7	51.46	4.75	44.80	56.00
2	6	6	54.97	4.79	51.30	64.30

ODS File Formats

- With ODS you can create file formats:
  - HTML: HyperText Markup Language – for web pages
  - LST: Listing Reports
  - RTF: Rich Text Format – for Word
  - PDF: Portable Document Format – for Adobe
  - PS: Post-Script – for printers
  - CSV: Comma Separated Values
  - and many others

Write a Comma-Separated File

- Many programs can read in a comma-separated values (CSV) file, including Excel.
- Use the ODS CSVALL statement to convert a SAS data set to a CSV file

```
ods csvall file='/home/user/admit.csv';
title;
proc print data=data1.admit noobs;
run;
ods csvall close;
```

- You can use titles, footnotes, labels, and formats to change the appearance of the data.

CSVALL Output

```
"ID","Name","Sex","Age","Date","Height","Weight","ActLevel","Fee"
2458,"Murray, W","M",27,1,72,168,"HIGH",85.20
2462,"Almers, C","F",34,3,66,152,"HIGH",124.80
2501,"Bonaventure, T","F",31,17,61,123,"LOW",149.75
2523,"Johnson, R","F",43,31,63,137,"MOD",149.75
2539,"LaMance, K","M",51,4,71,158,"LOW",124.80
2544,"Jones, M","M",29,6,76,193,"HIGH",124.80
2552,"Reberson, P","F",32,9,67,151,"MOD",149.75
2555,"King, E","M",35,13,70,173,"MOD",149.75
2563,"Pitts, D","M",34,22,73,154,"LOW",124.80
2568,"Eberhardt, S","F",49,27,64,172,"LOW",124.80
2571,"Nunnelly, A","F",44,19,66,140,"HIGH",149.75
2572,"Oberon, M","F",28,17,62,118,"LOW",85.20
2574,"Peterson, V","M",30,6,69,147,"MOD",149.75
2575,"Quigley, M","F",40,8,69,163,"HIGH",124.80
2578,"Cameron, L","M",47,5,72,173,"MOD",124.80
2579,"Underwood, K","M",60,22,71,191,"LOW",149.75
2584,"Takahashi, Y","F",43,29,65,123,"MOD",124.80
2586,"Derber, B","M",25,23,75,188,"HIGH",85.20
2588,"Ivan, H","F",22,20,63,139,"LOW",85.20
2589,"Wilcox, E","F",41,16,67,141,"HIGH",149.75
2595,"Warren, C","M",54,7,71,183,"MOD",149.75
```

SAS Functions

A SAS function is often categorized by the type of data manipulation performed:

- truncation
- character
- date and time
- mathematical
- trigonometric
- sample statistics
- arithmetic
- financial
- random number
- state and ZIP code

Example: Mailing Labels

- The **`data2.freqflyers`** data set contains information about frequent flyers.

ID	Name	Address1	Address2
F31351	Farr,Sue	15 Harvey Rd.	Macon,Bibb,GA,31298
F161	Cox,Kay B.	163 McNeil Pl.	Kern,Pond,CA,93280
F212	Mason,Ron	442 Glen Ave.	Miami,Dade,FL,33054
F25122	Ruth,G. H.	2491 Brady St.	Munger,Bay,MI,48747

- How do we use this data set to create another data set suitable for mailing labels?

FullName	Address1	Address2
Ms. Sue Farr	15 Harvey Rd.	Macon, GA 31298
Ms. Kay B. Cox	163 McNeil Pl.	Kern, CA 93280
Mr. Ron Mason	442 Glen Ave.	Miami, FL 33054
Mr. G. H. Ruth	2491 Brady St.	Munger, MI 48747

The LENGTH Function

- The LENGTH function returns the number of characters in a string

```
NewVar = LENGTH(string);
```

- Example
  - `LENGTH('SMITH, JOHN')` = **11**

The INDEX Function

- Recall that the INDEX function returns the position of specific character (or characters) within a string

```
NewVar = INDEX(string,target);
```

- Example
  - `INDEX('SMITH-JOHN', '-') = 6`

- Returns ZERO (0) if the target isn't in the string
  - Recall that we previously used this function to mimic the CONTAINS special operator

The SUBSTR Function

- The SUBSTR function extracts a portion of a character variable:

```
NewVar=SUBSTR(string,start<,length>);
```

- Example:
  - `SUBSTR('PSTAT130 M20',6)` = '130 M20'
  - `SUBSTR('PSTAT130 M20',6,3)` = '130'

Parse a Text String

- How can we turn '`SMITH, JOHN`' into '`JOHN SMITH`'?
  - Find the location of the comma
  - Last Name = text before the comma
  - First Name = text after the comma

Put It All Together

```
DATA mail_labels;
input name $25.;
name_len = length(name);
comma_pos = index(name,',');
last_name = substr(name,1,comma_pos-1);
first_name = substr(name,comma_pos+2,name_len-comma_pos-1);

datalines;
Smith, John
Johnson, Davy
Quincy, Elizabeth
;
run;

proc print;
run;
```

Results

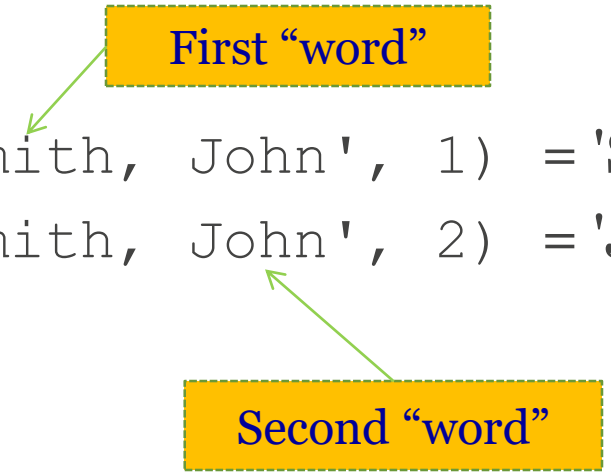
name	name_len	comma_pos	last_name	first_name
Smith, John	11	6	Smith	John
Johnson, Davy	13	8	Johnson	Davy
Quincy, Elizabeth	17	7	Quincy	Elizabeth

The SCAN Function

- The SCAN function “parses” a character string into a set of “words” using a delimiter.

```
NewVar=SCAN(string,n<,delimiters>);
```

- Example:
  - `SCAN('Smith, John', 1)` = 'Smith'
  - `SCAN('Smith, John', 2)` = 'John'



The SCAN Function

- When the SCAN function is used
  - The default delimiters include
    - blank . < (+ | & ! $ *) ; ¬ - / , % | ¢
  - Delimiters before the first "word" have no effect
  - Any character or set of characters can serve as delimiters
  - Two or more contiguous delimiters are treated as a single delimiter
  - A missing value is returned if there are fewer than *n* words in the *string*
  - If n is *negative*, SCAN returns the "word" in the *string* starting from the *end* (of the string)

Concatenation Operator

- Use the || operator to “concatenate” or join two strings together

- Examples
  - `'John' || 'Smith'` = 'JohnSmith'
  - `'John' || ' ' || 'Smith'` = 'John Smith'

A Better Mailing Label Program

```
DATA mail_labels2;
input name $25.;
last_name = scan(name,1);
first_name = scan(name,2);
datalines;
Smith, John
Johnson, Davy
Quincy, Elizabeth
;
run;
proc print;
run;
```

name	last_name	first_name
Smith, John	Smith	John
Johnson, Davy	Johnson	Davy
Quincy, Elizabeth	Quincy	Elizabeth

Numeric Truncation Functions

- Selected functions that truncate numeric values include
  - ROUND function
  - CEIL function
  - FLOOR function
  - INT function

The ROUND Function

- The ROUND function performs a traditional Round Up/Round Down operation:

```
NewVar = ROUND(argument<,round-off-unit>);
```

- Examples:

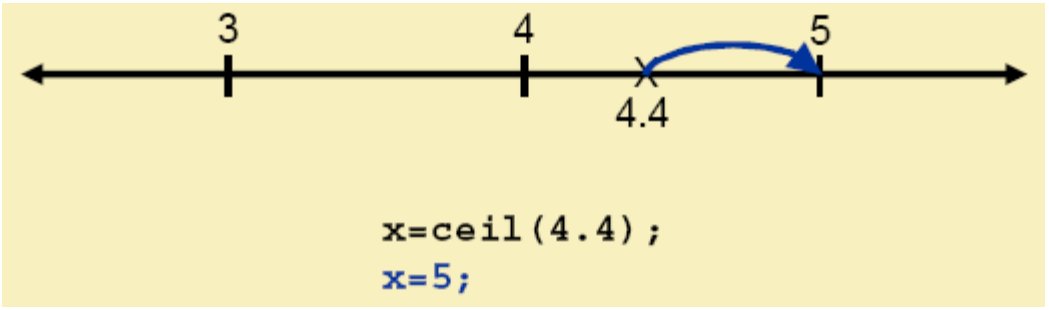
```
data truncate;
   NewVar1=round(12.12);
   NewVar2=round(42.65,.1);
   NewVar3=round(6.478,.01);
   NewVar4=round(96.47,10);
run;
```

NEWVAR1	NEWVAR2	NEWVAR3	NEWVAR4
12	42.7	6.48	100

The CEIL Function

- The CEIL function performs a *Round Up* operation only

```
NewVar = CEIL(argument);
```

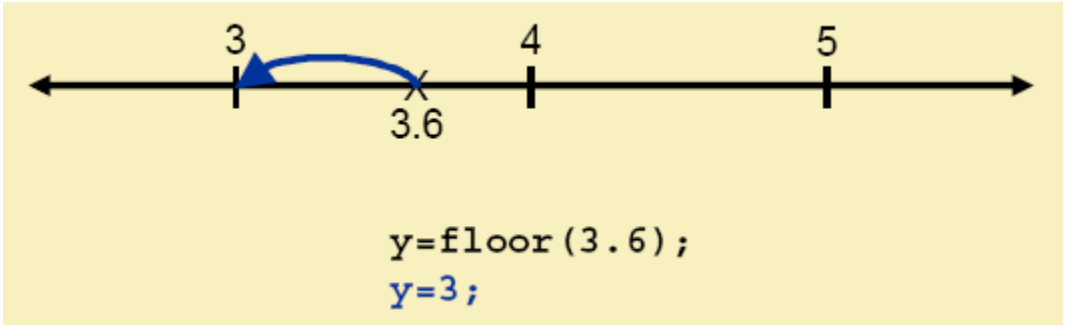


- Note: CEIL(4) = 4

The FLOOR Function

- The FLOOR function performs a *Round Down* operation only

```
NewVar = FLOOR(argument);
```



- Note: FLOOR(4) = 4

The INT Function

- The INT function removes any decimals from an number

```
NewVar = INT(argument);
```

- Examples:
  - INT(3.2) = 3
  - INT(-4.8) = -4

- For positive numbers, INT = FLOOR
- For negative numbers, INT = CEIL

```
data truncate;
   Var1=-6.478;
   NewVar1=ceil(Var1);
   NewVar2=floor(Var1);
   NewVar3=int(Var1);
run;
```

VAR1	NEWVAR1	NEWVAR2	NEWVAR3
-6.478	-6	-7	-6

Class Exercise 1

- Use the `pilots` data set in the data1 folder
  - Create a scatterplot of `salary` by `age` (assume the current date is 1/1/82)
    - Use a blue square symbols
    - Label the axes as 'Annual Salary' and 'Age'
    - Display a regression line and 95% confidence limits.
  - Create an HTML file (pilots.html) containing the following
    - The descriptor of the data set with an appropriate title
    - The data portion of the data set with an appropriate title.

Class Exercise 2

- The **`data2.ffhistory`** data set contains information about the history of each frequent flyer. This history information consists of
  - Each membership level the flyer has attained (bronze, silver, or gold)
  - The year the flyer attained each level.
- Create a report that shows all frequent flyers who have attained **silver** membership status and the **year** each became silver members.

Class Exercise 2 - continued

- Data:

ID	Status	Seat Pref
F31351	Silver 1998,Gold 2000	AISLE
F161	Bronze 1999	WINDOW
F212	Bronze 1992,silver 1995	WINDOW
F25122	Bronze 1994,Gold 1996,Silver 1998	AISLE

- Hint: Think about how you would
  - Parse the membership levels?
  - Parse the year each level was attained?
  - Select flyers that have achieved Silver status?