SAS 9.2 Graphics Course October 2010

Kriss Harris

italjet125@yahoo.com

Part 2. SAS Version 9.2 Upgrades

SAS Version 9.2

- Proc SGPLOT
- Proc SGPANEL
- Proc SGPLOT and SGPANEL similarities
- Proc SGSCATTER
- Proc Template
  - Graphics Template Language
- Proc SGRENDER
- References
- Appendix

Proc SGPLOT

Proc SGPLOT
Concepts

- There are four basic types of plots that you can create with the SGPLOT procedure:
- Basic plots
  - scatter, series, step, band, and needle plots
- Fit and confidence plots
  - loess, regression, and penalized B-spline curves, and ellipses
- Distribution plots
  - box plots, histograms, and normal and kernel density estimates
- Categorization plots
  - dot plots, bar charts, and line charts

Proc SGPLOT
Plot Type Compatibility

	Basic	Fit and confidence	Distribution	Catergorization
Basic	x	x		
Fit and confidence	x	x		
Distribution			x	
Catergorization				x

Note: Box plots cannot be combined with any other plot types

Proc SGPLOT Syntax

```
PROC SGPLOT < option(s)>;
 BAND X= variable | Y= variable
 UPPER= numeric-value | numeric-variable LOWER= numeric-value | numeric-variable
 </option(s)>;
 DENSITY response-variable </option(s)>;
 DOT category-variable </option(s)>;
 ELLIPSE X= numeric-variable Y= numeric-variable </option(s)>;
 HBAR category-variable < /option(s) >
 HBOX response-variable </option(s)>;
 HISTOGRAM response-variable < /option(s)>
 HLINE category-variable < /option(s)>
 INSET "text-string-1" <... "text-string-n"> | (label-list);
 KEYLEGEND <"name-1" ... "name-n"> </option(s)>;
 LOESS X= numeric-variable Y= numeric-variable </option(s)>;
 NEEDLE X= variable Y= numeric-variable </option(s)>;
 PBSPLINE X= numeric-variable Y= numeric-variable </option(s)>;
 REFLINE value(s) </option(s)>;
 REG X= numeric-variable Y= numeric-variable </option(s)>;
 SCATTER X= variable Y= variable </option(s)>;
 SERIES X= variable Y= variable </option(s)>;
 STEP X= variable Y= variable </option(s)>;
 VBAR category-variable < /option(s)>
 VBOX response-variable </option(s)>;
 VLINE category-variable < /option(s)>
 XAXIS <option(s)>;
 X2AXIS <option(s)>;
 YAXIS <option(s)>;
 Y2AXIS <option(s)>;
```

Overview of plots covered

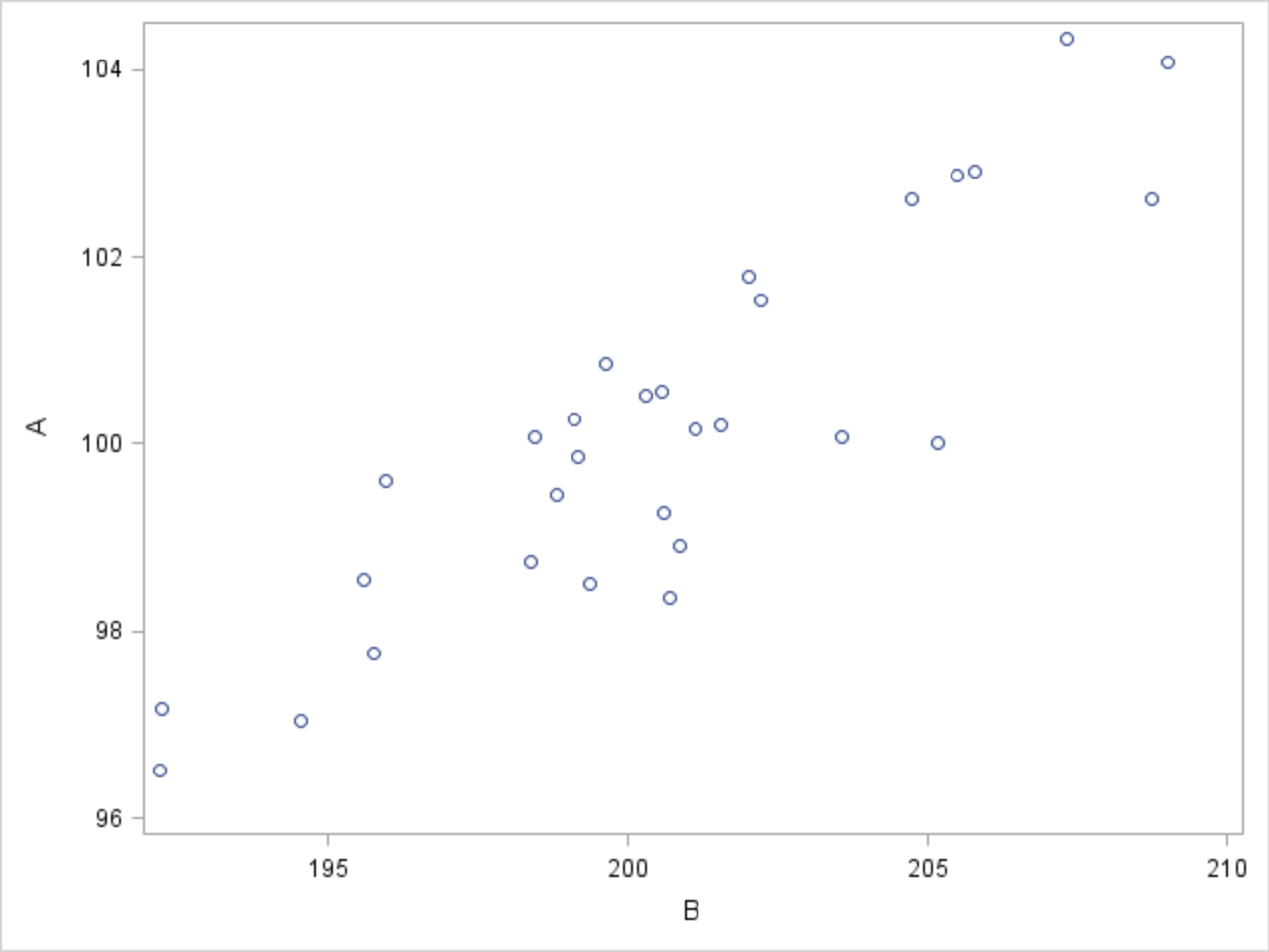
- Hbar and Vbar
  - Produces horizontal and vertical bar charts.
- Hbox and Vbox
  - Produces horizontal and vertical box plots.
- Hline and Vline
  - Produces a plot with error bars and statistics (e.g. mean) connected by line.
- Scatter
  - Produces a scatter plot.
- Reg
  - Produces a scatter plot with a linear regression fit.
- Series
  - Produces a plot where the datapoints are connected by a line.

Overview of other plot options covered

- Inset
  - Inserts text into the plot.
- Keylegend
  - Formats the display of the legend.
- Refline
  - Adds a reference line to the plot.
- Xaxis and Yaxis
  - Controls axis scales, titles, ticks, etc.

Basic Scatterplot

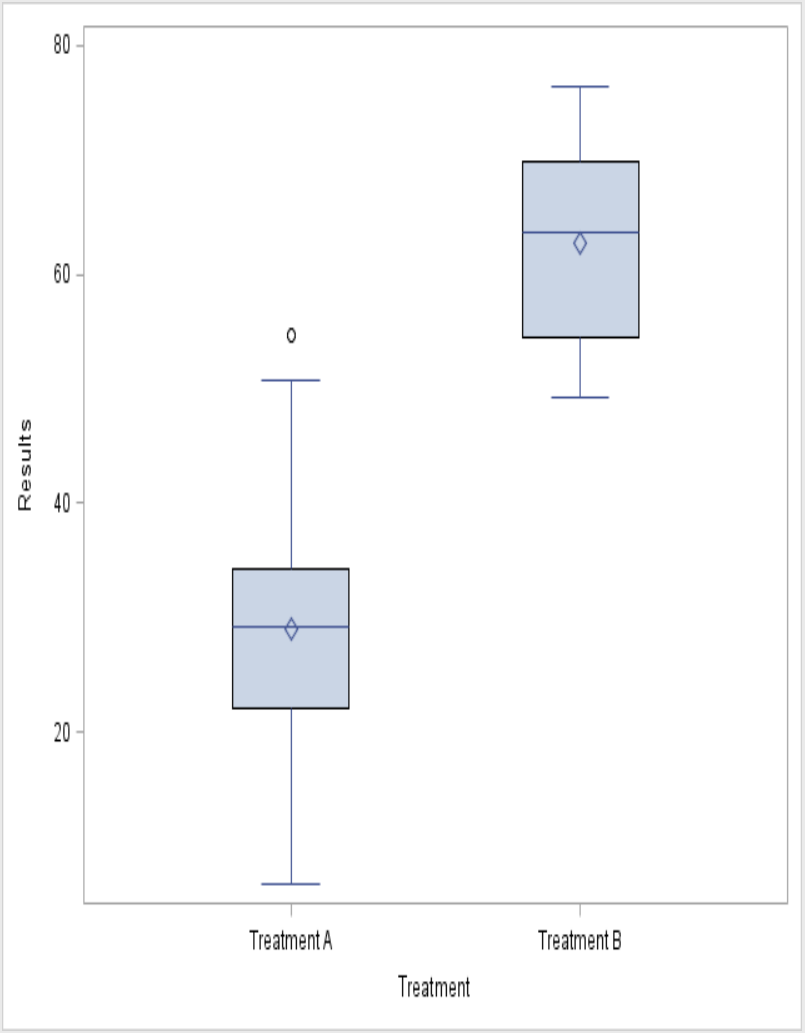
```
ods html style = statistical;
proc sgplot data = scatter;
scatter x = B  y = A;
run;
```

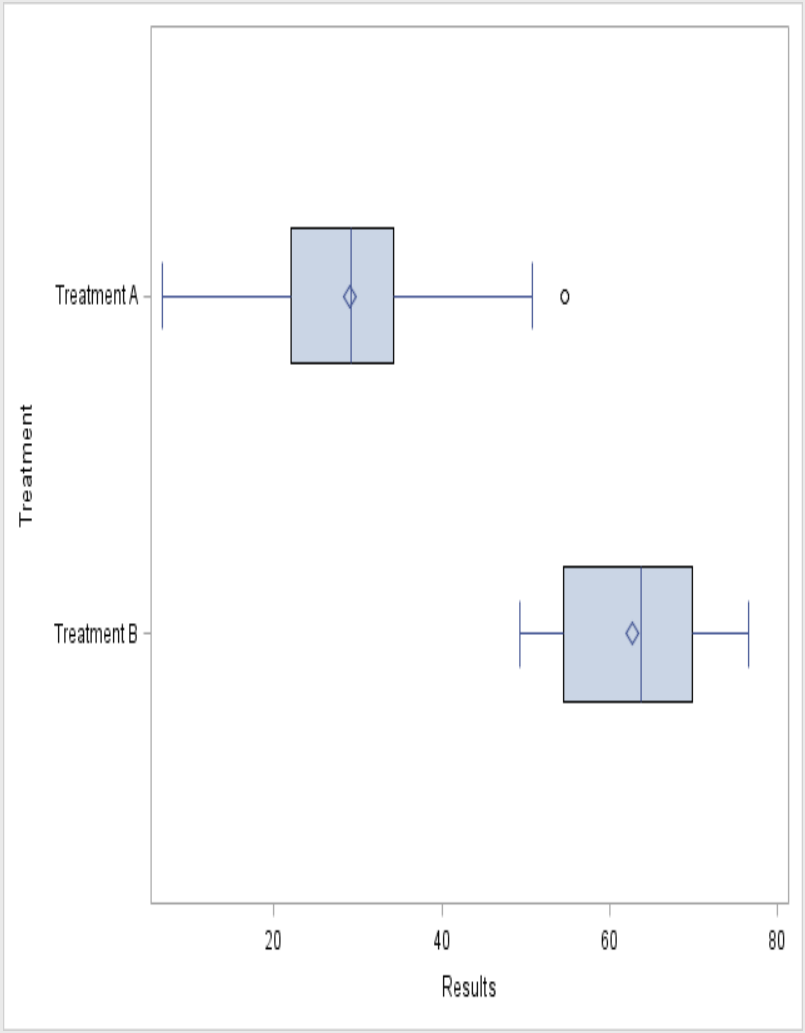

104
102
100
98
96
A
195
200
205
210
B

Basic Boxplot

```
proc sgplot data = boxplots;
vbox results / category = Treatment;
run;
```

Changing vbox to hbox produces horizontal box plots.


80
60
40
20
Results
Treatment A
Treatment B
Treatment

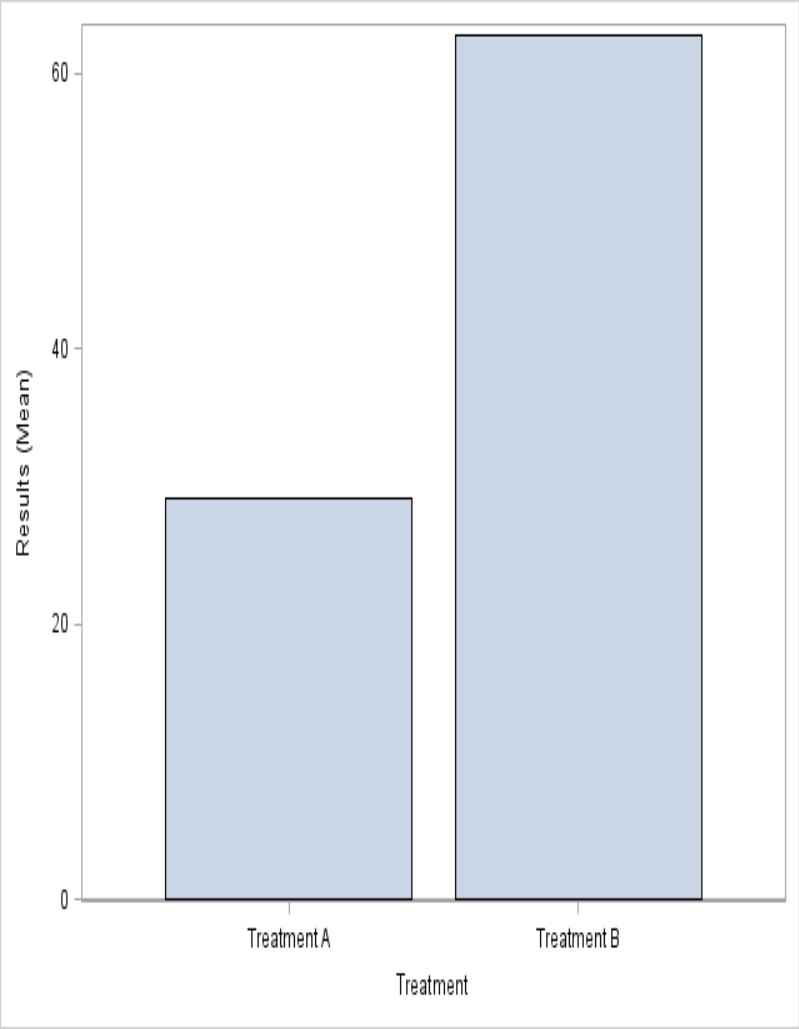

Treatment A
Treatment B
Treatment
20
40
60
80
Results

Basic Bar chart

```
proc sgplot data = boxplots;
vbar Treatment /  response = results STAT = MEAN
   ;
run;
```

Changing vbar to hbar produces horizontal box plots.

The other STAT options that are FREQ and SUM, SUM is the default.

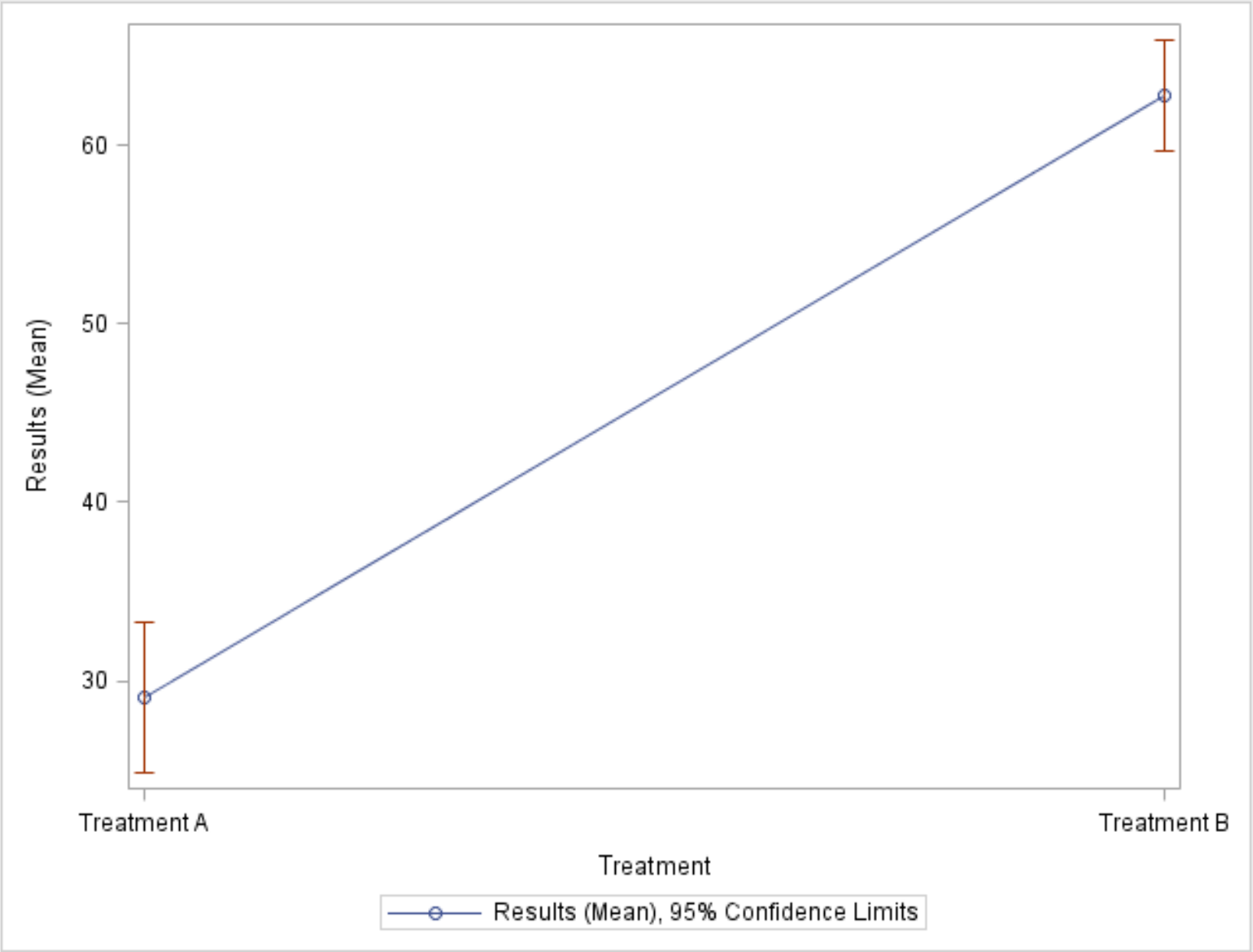

60
40
20
0
Results (Mean)
Treatment A
Treatment B
Treatment

Means and Confidence Limits (for ONE WAY ANOVA Models)

```
proc sgplot data = boxplots;
vline Treatment /  response = results STAT = MEAN
   LIMITSTAT = CLM markers;
run;
```

Uses the confidence limits as the statistic for the limit lines

Plots the mean of the data.


60
50
40
30
Results (Mean)
Treatment A
Treatment B
Treatment
Results (Mean), 95% Confidence Limits

Means and Confidence Limits (for more complicated models)

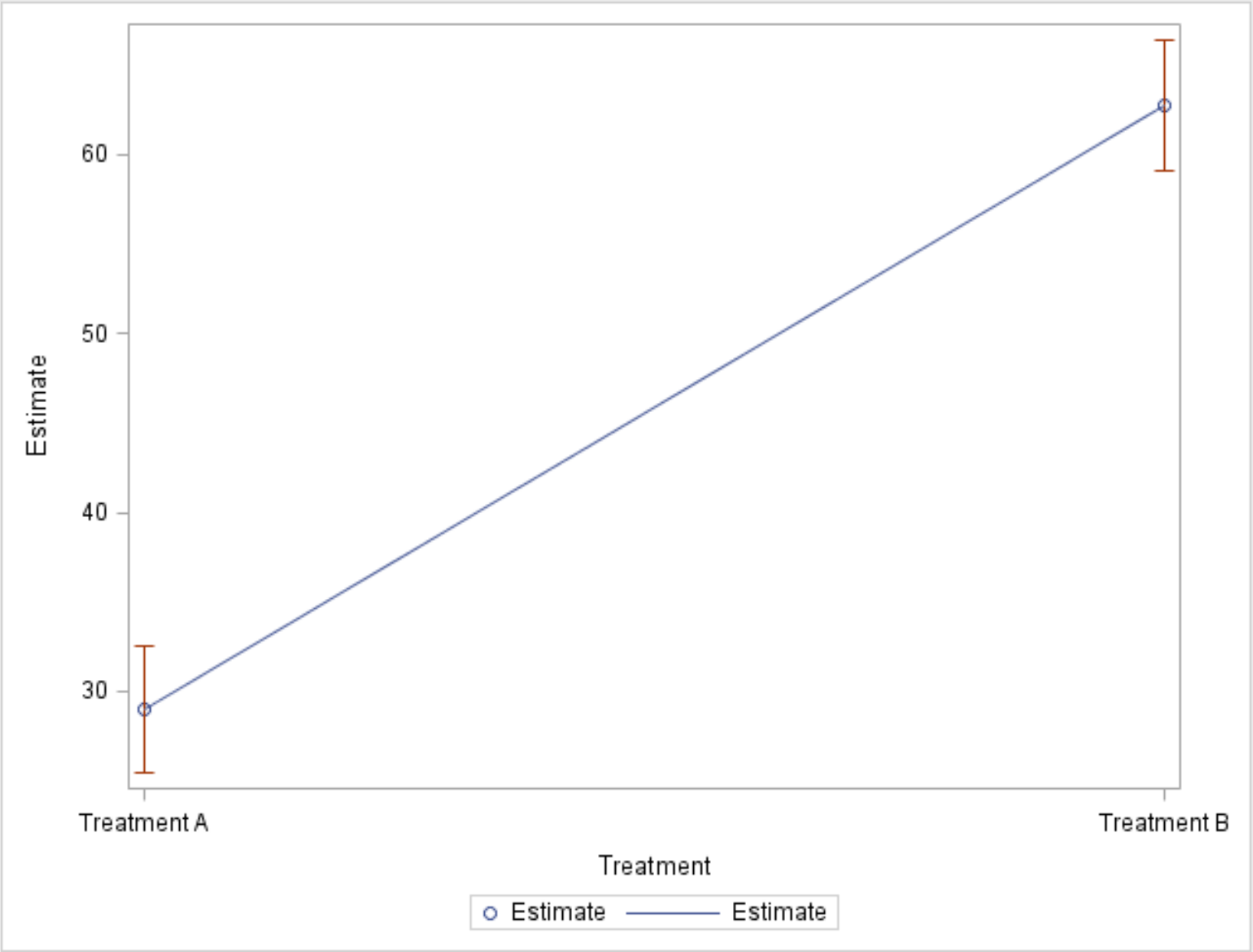
- Fit the model first

```
ods output lsmeans = lsmeans;
proc mixed data = boxplots;
class treatment day;
model results = treatment day;
lsmeans treatment / cl;
run;
```

- Then plot the graph

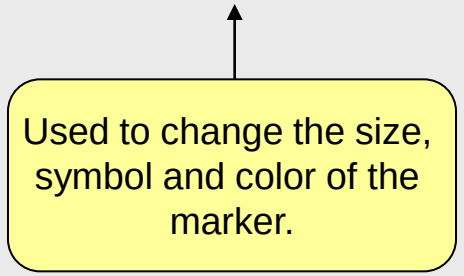
```
proc sgplot data = lsmeans;
scatter x = treatment y = estimate / YERRORLOWER =
   lower YERRORUPPER = upper;
series x = treatment y = estimate;
run;
```

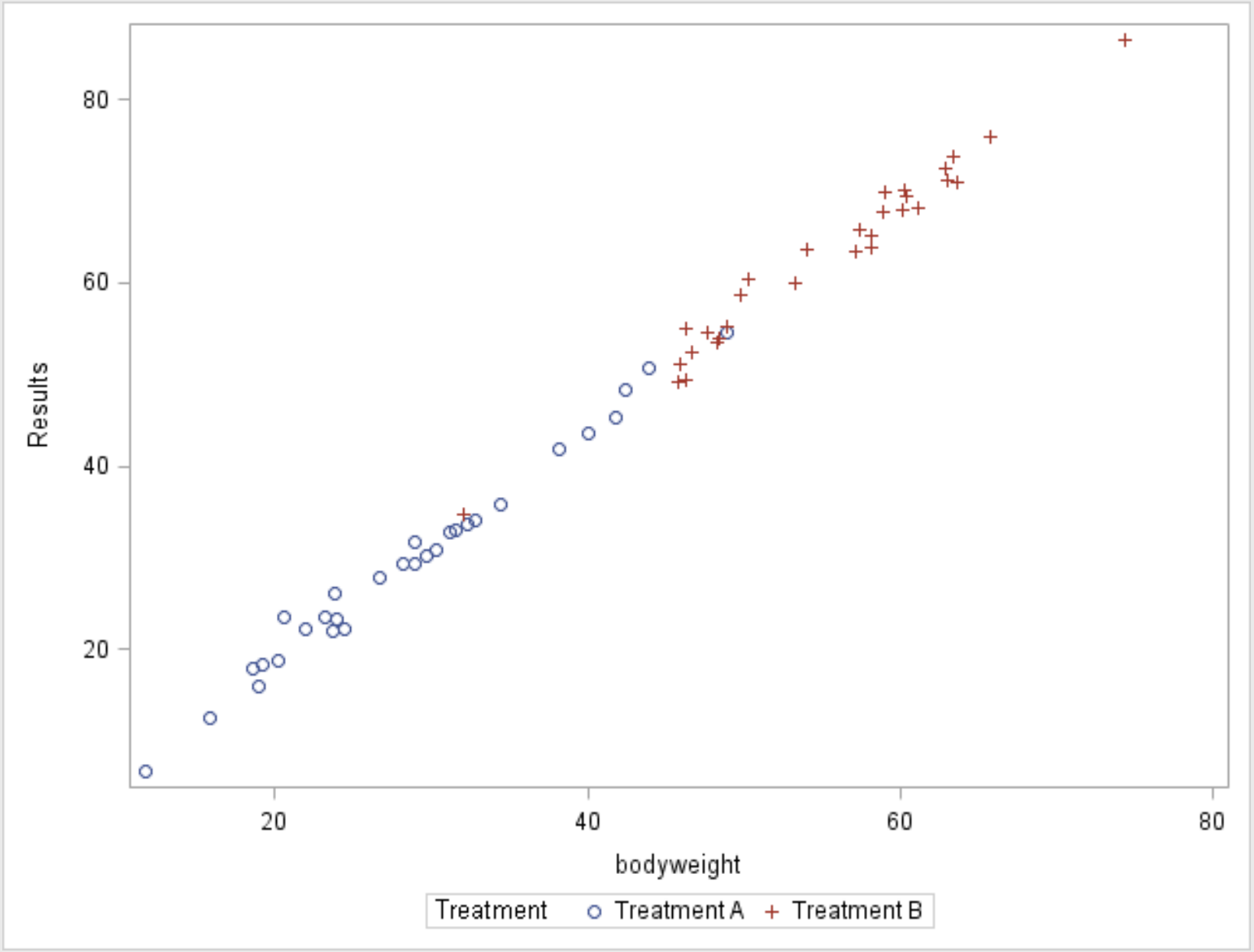
Series connects the datapoints with a line.


60
50
40
30
Estimate
Treatment A
Treatment B
Treatment
Estimate
Estimate

Scatter Plots with different markers for each treatment group

```
proc sgplot data = merged;
scatter x = bodyweight  y = Results / group =
   treatment;
run;
```



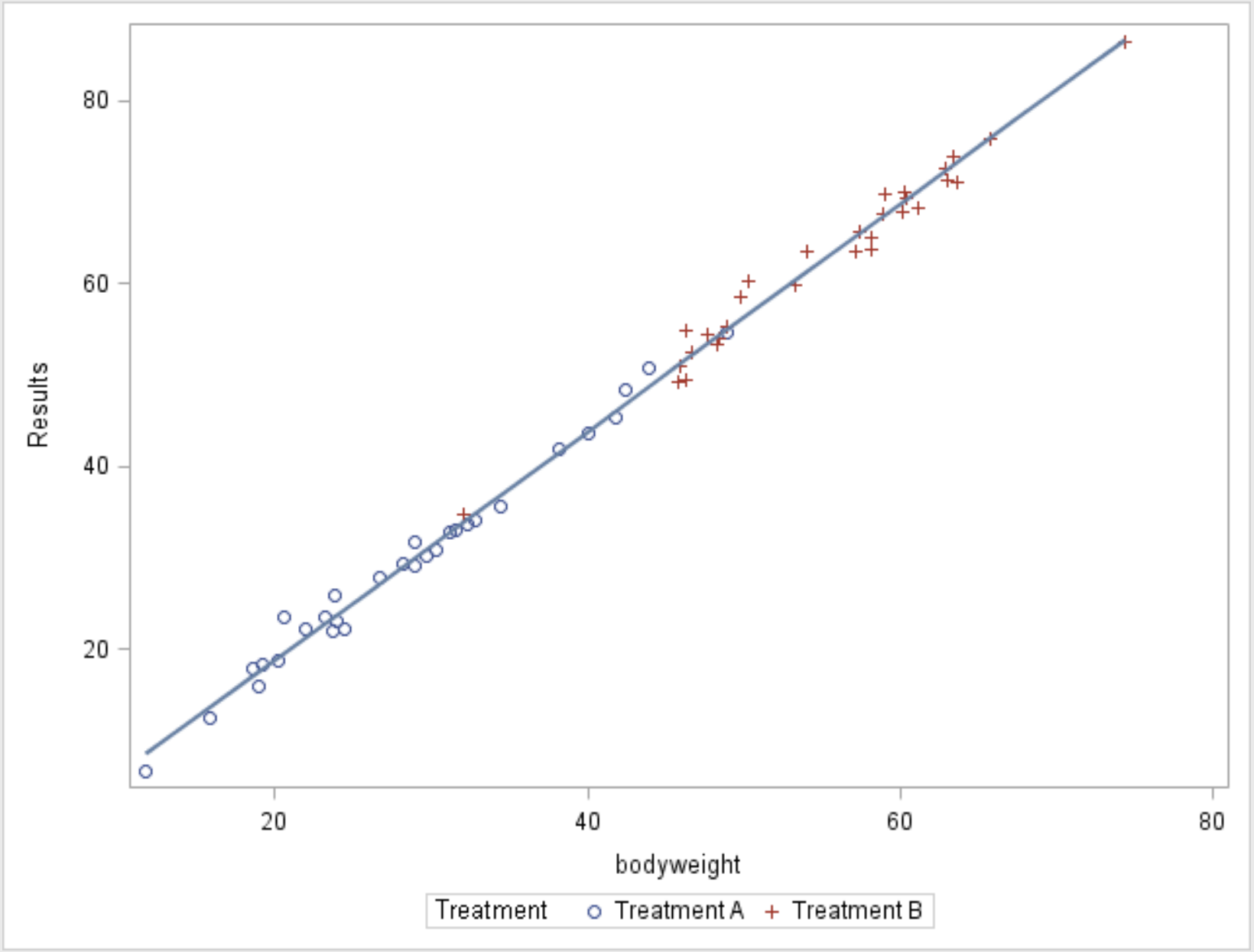

Results
80
60
40
20
20
40
60
80
bodyweight
Treatment
Treatment A
Treatment B

Scatter Plots with single regression slope

```
proc sgplot data = merged;
scatter x = bodyweight  y = Results / group =
   treatment;
reg x = bodyweight  y = Results /  markerattrs =
   (size = 0);
run;
```

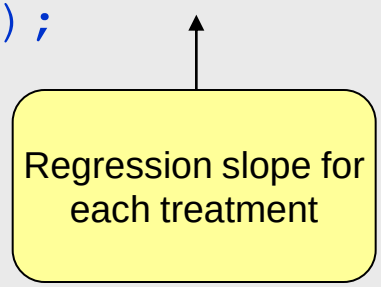
Size = 0 prevents more markers from being produced

Used to change the size, symbol and color of the marker.

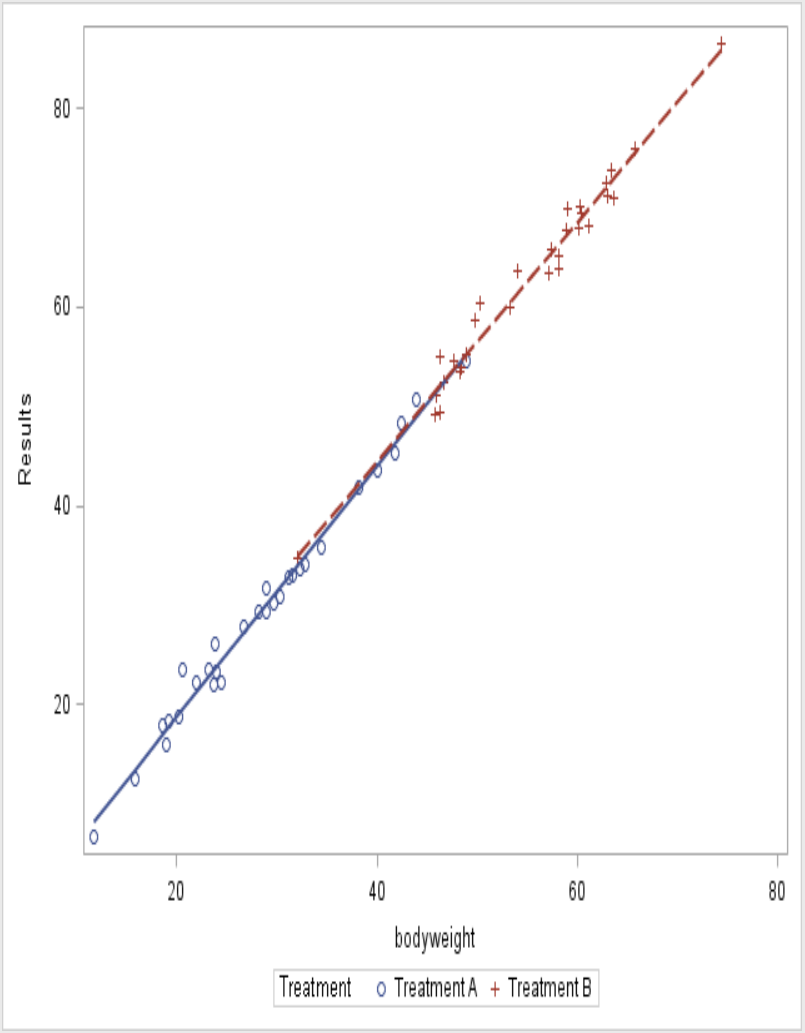

Results
80
60
40
20
20
40
60
80
bodyweight
Treatment
Treatment A
Treatment B

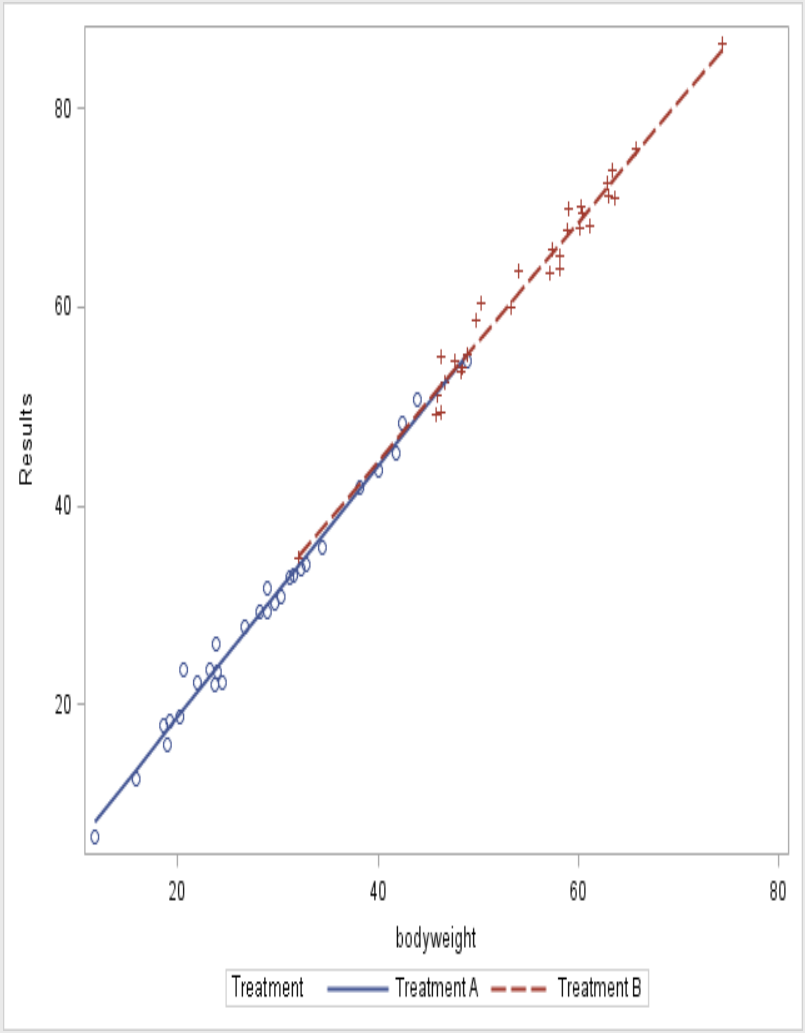
Scatter Plots with regression slopes for each treatment group

```
proc sgplot data = merged;
scatter x = bodyweight  y = Results / group =
  treatment;
reg x = bodyweight  y = Results / group =
  treatment markerattrs = (size = 0);
run;

proc sgplot data = merged;
reg x = bodyweight  y = Results / group =
  treatment;
run;
```



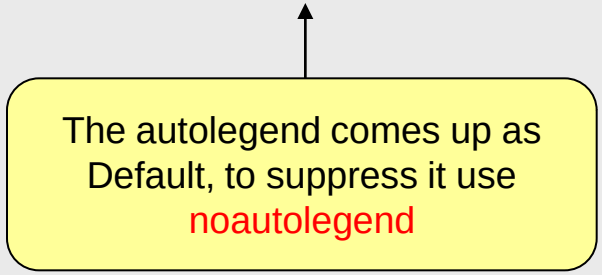
The legends below
are the only thing
that are different

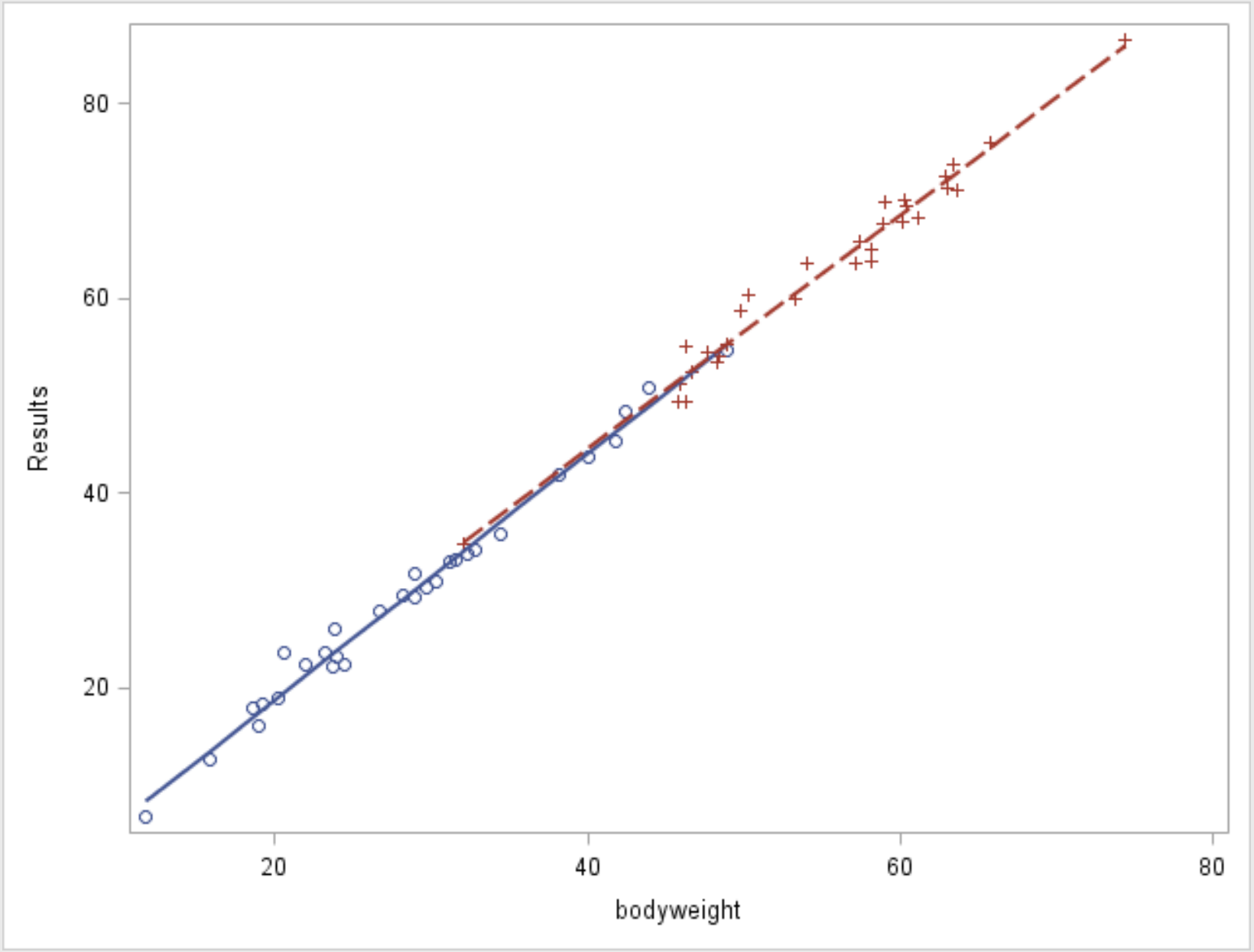




noautogend

```
proc sgplot data = merged noautolegend;
reg x = bodyweight  y = Results / group =
   treatment;
run;
```



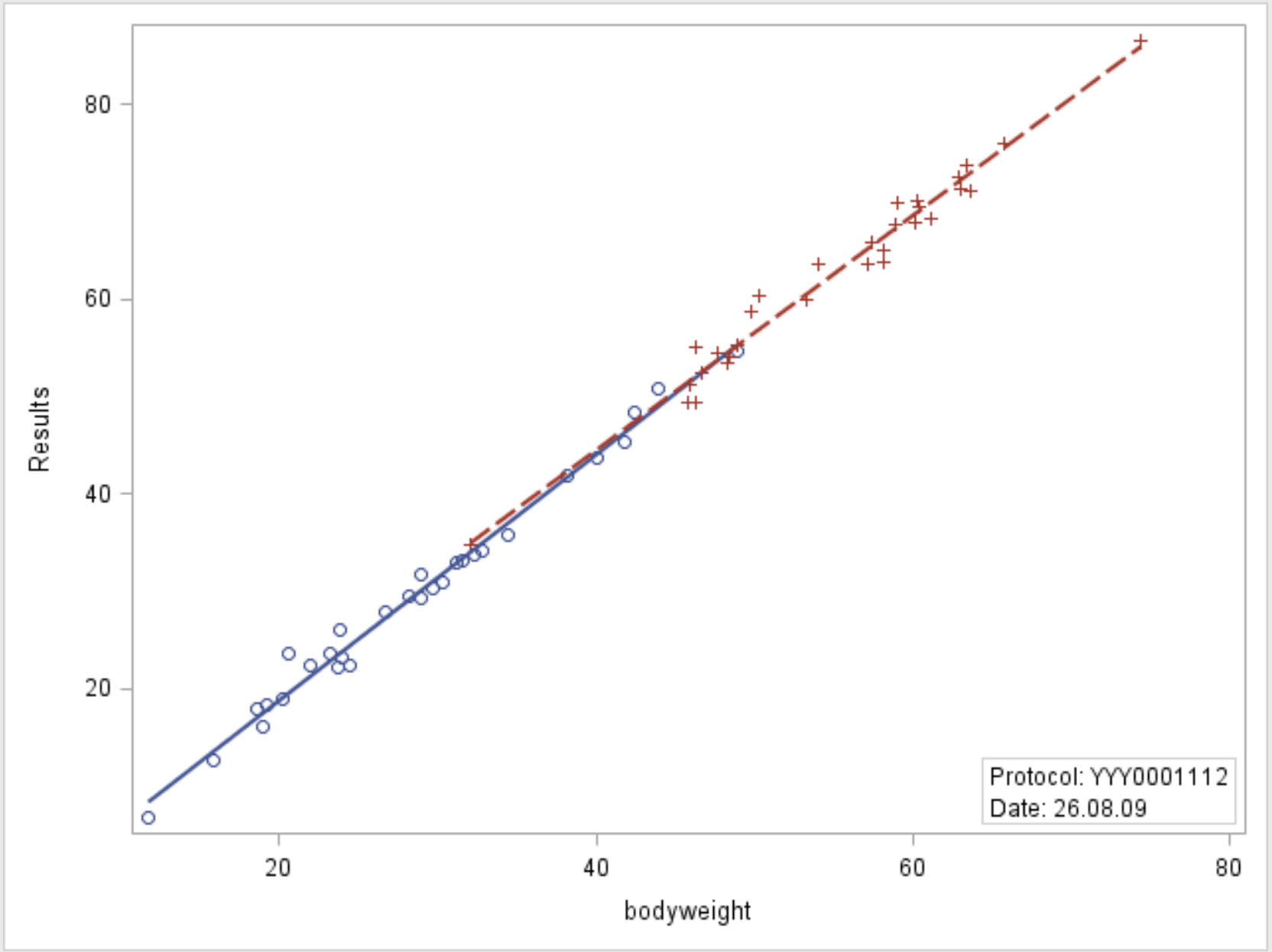

Results
80
60
40
20
20
40
60
80
bodyweight

Using INSET to insert text into the graph

```
proc sgplot data = merged noautolegend;
reg x = bodyweight  y = Results / group =
  treatment;
inset "Protocol: YYY0001112" "Date: 26.08.09" /
  position = bottomright BORDER;
run;
```

Other position options are Bottom, Bottomleft, Left, Right, Top, Topleft and Topright.

Using separate strings places the New string on a separate line.


Results
80
60
40
20
20
40
60
80
bodyweight
Protocol: YYY0001112
Date: 26.08.09

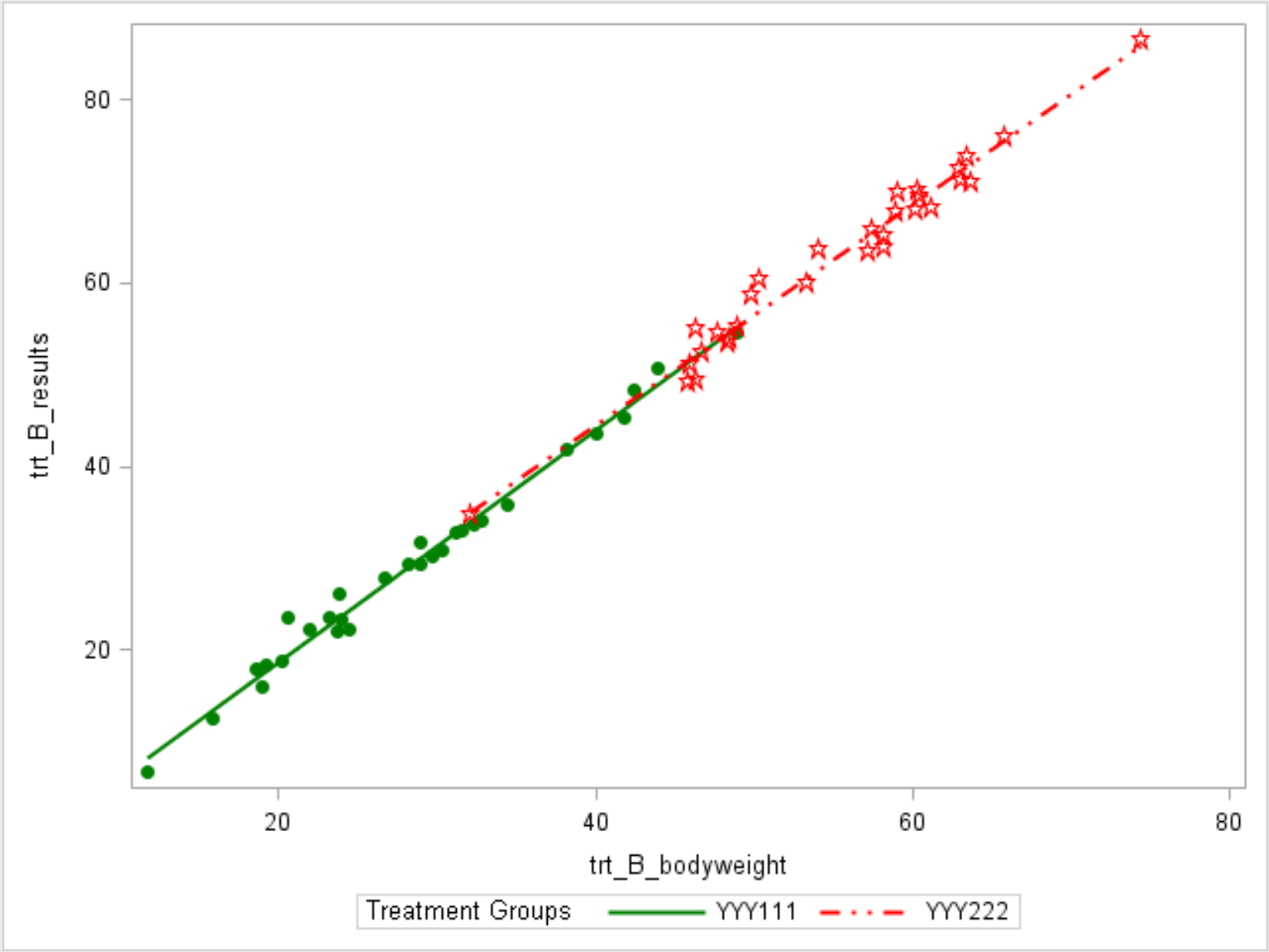
Correctly controlling symbols and legends for each treatment group

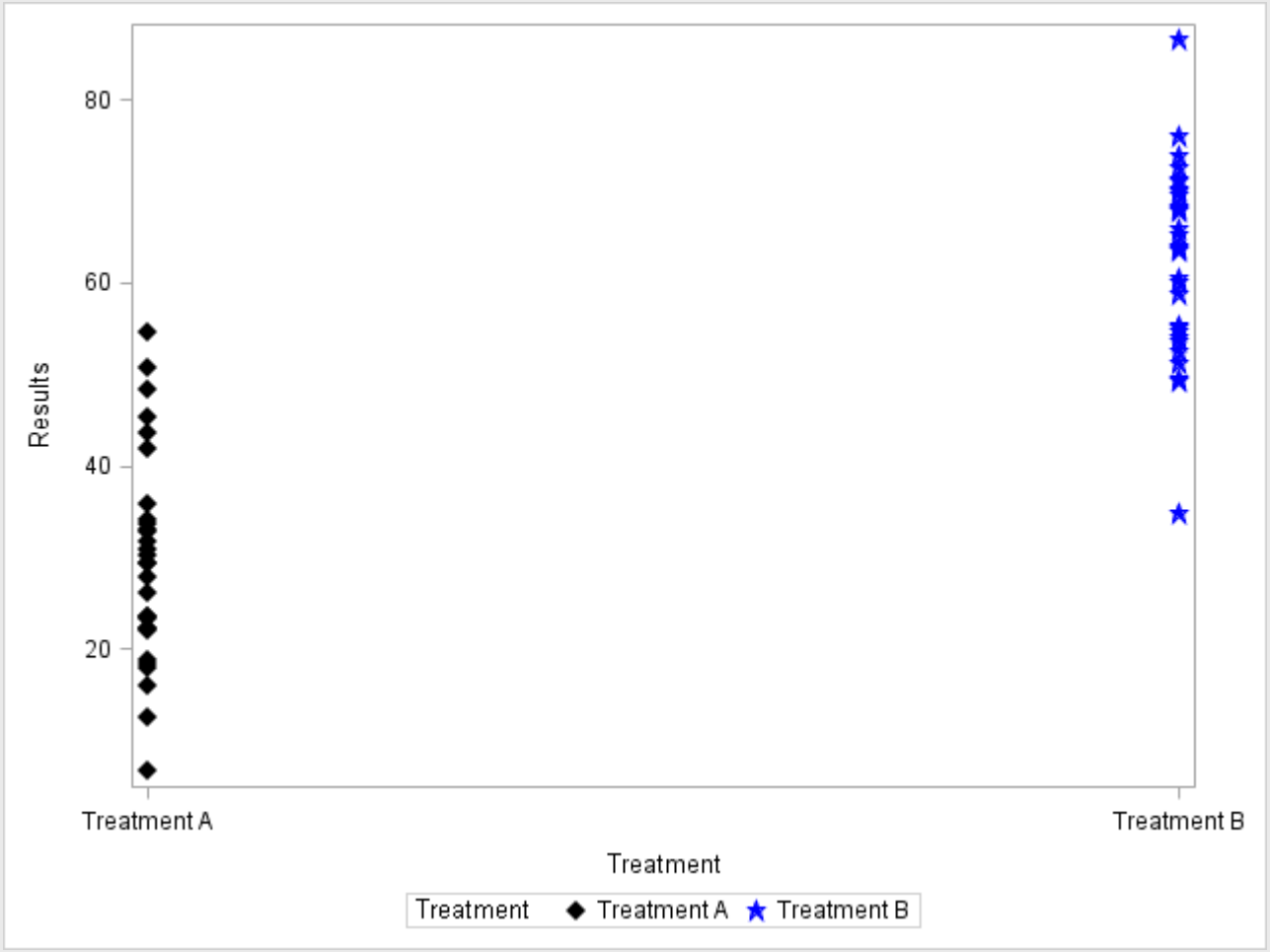
- Unstack the treatment groups first, eg;

```
data treatment_A treatment_B;
set merged;
if treatment = "Treatment A" then output
   treatment_A;
else output treatment_B;
drop i variable1;
run;

data unstacked;
merge treatment_A (rename = (results =
   trt_A_results bodyweight = trt_A_bodyweight))
   treatment_B (rename = (results = trt_B_results
   bodyweight = trt_B_bodyweight));
run;
```

Correctly controlling symbols and legends for each treatment group

```
proc sgplot data = unstacked;
reg x = trt_A_bodyweight y = trt_A_results /
   LEGENDLABEL= “YYY111" name = "lineA"
   MARKERATTRS = (symbol = circlefilled color =
   green)  LINEATTRS = (color = green) ;
reg x = trt_B_bodyweight y = trt_B_results /
   LEGENDLABEL= “YYY222" name = "lineB"
   MARKERATTRS = (symbol = star color = red)
   LINEATTRS = (color = red);
keylegend "lineA" "lineB" / title = "Treatment
   Groups";
run;
```


80
60
40
20
trt_B_results
20
40
60
80
trt_B_bodyweight
Treatment Groups
YYY111
YYY222


80
60
40
20
Results
Treatment A
Treatment B
Treatment
Treatment
Treatment A
Treatment B

Modstyle function

- %modstyle is a SAS defined function to set up cosmetic definitions. This takes the place of much of the symbol statement functionality from previous releases of SAS

```
%modstyle(parent=statistical, name=regressionst,type=CLM,
    colors=black blue, fillcolors=colors,
    markers=diamondfilled starfilled circle,
    linestyles=mediumdashdotdot solid shortdashdot);
ods html style = regressionst;
```

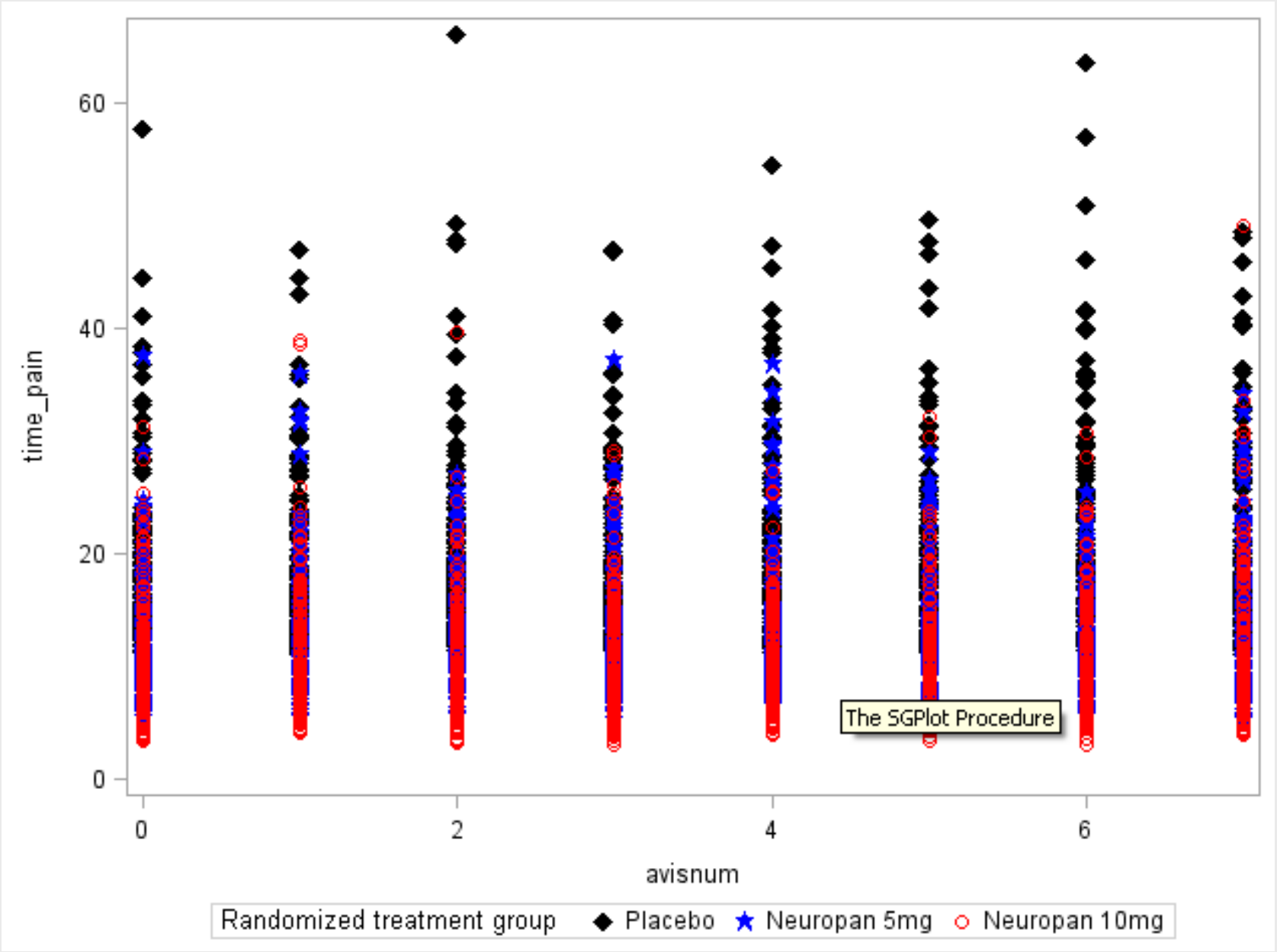
- The default style for a SAS session is Statistical.
- This replaces the original settings with new settings under the name regressionst.

Modstyle function

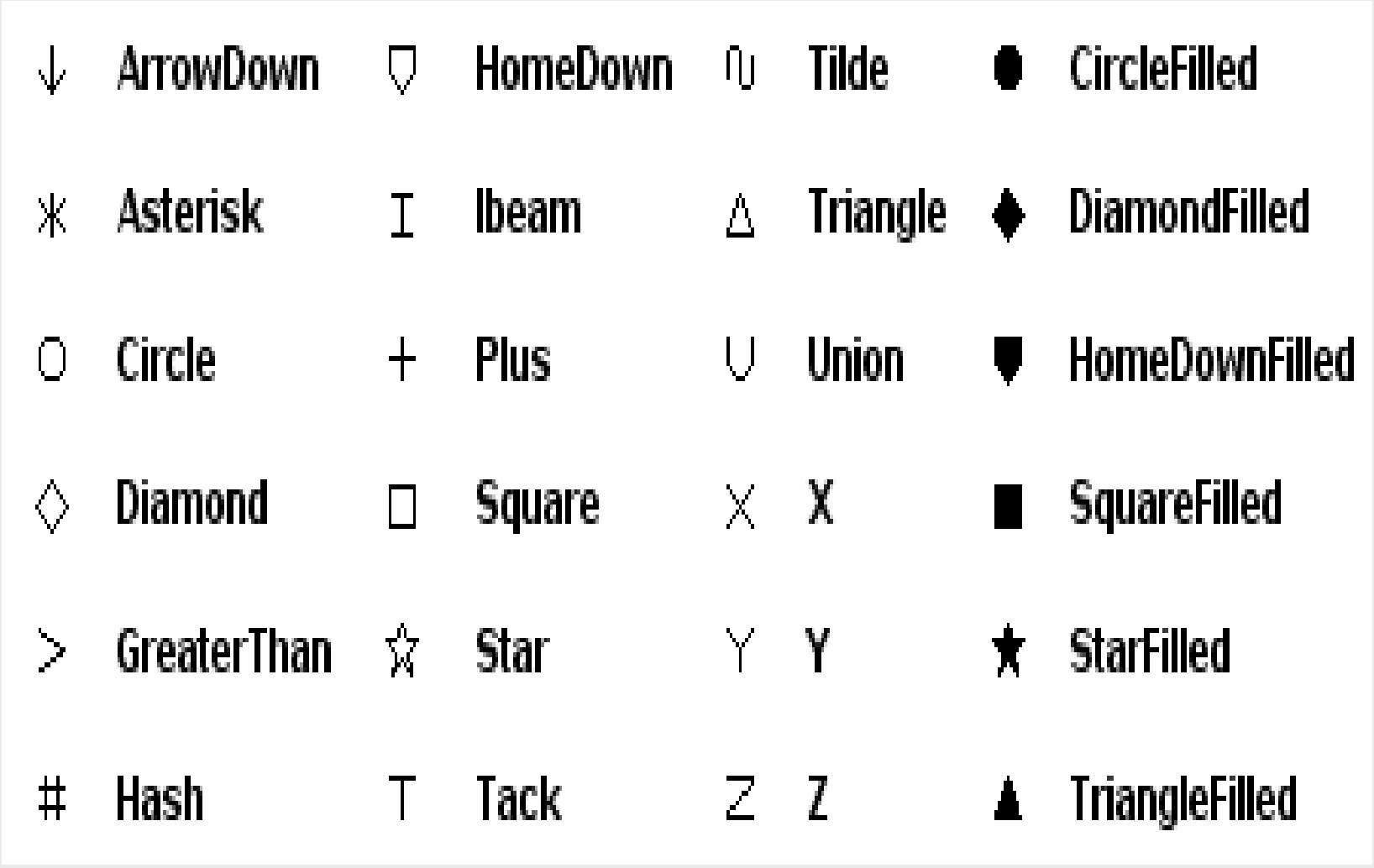
- When calling MARKERATTRS, or LINEATTRS in options of SAS statements (i.e. REG, SCATTER, VLINE, etc.), only one color, linestyle, or symbol can be specified. MODSTYLE can be used to specify specific orders of colors, etc when there is more than one grouping of data.
- Note when using modstyle, if you specify less colors, markers, or styles than what is in the data, SAS will create the entire graph in its defaults rather than in the new user-defined style.

Modstyle function

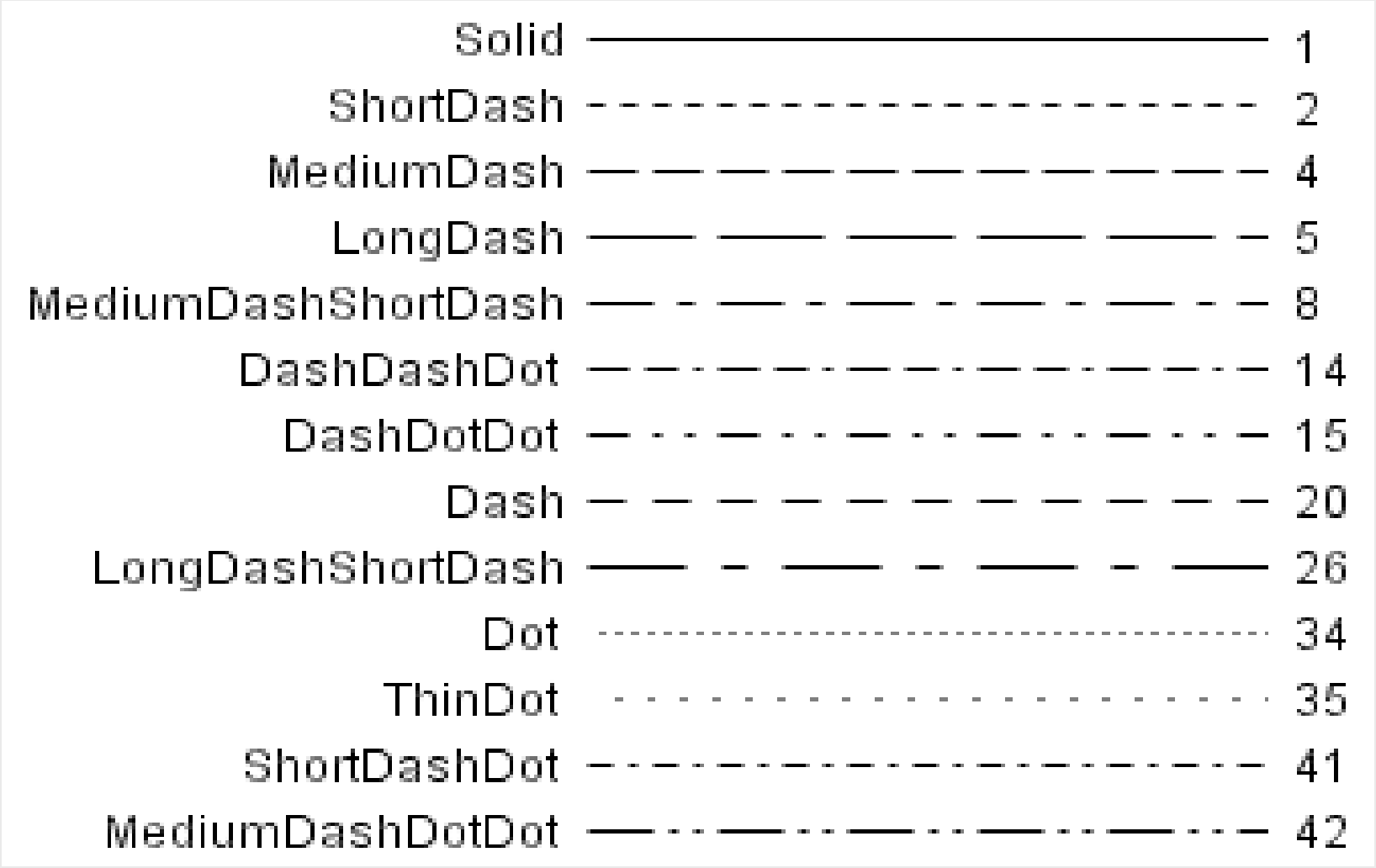
```
%modstyle(parent=statistical,
        name=regressionst,
        type=CLM,
        colors=black blue red,
        fillcolors=colors,
        markers=diamondfilled starfilled circle,
        linestyles=mediumdashdotdot solid shortdashdot);
ods html style = regressionst;

proc sgplot data=data.a_time;
  scatter x=avisnum y=time_pain/group=trtgrp;
run;
```


time_pain
60
40
20
0
0
2
4
6
avisnum
The SGPlot Procedure
Randomized treatment group
Placebo
Neuropan 5mg
Neuropan 10mg

Marker Symbols
MARKERATTRS= option



LINE PATTERNS
LINEATTRS= option



Changing Axis Labels

```
proc sgplot data = scatter;
yaxis label = "Treatment A Response";
xaxis label = "Treatment B Response";
scatter x = B  y = A;
run;
```

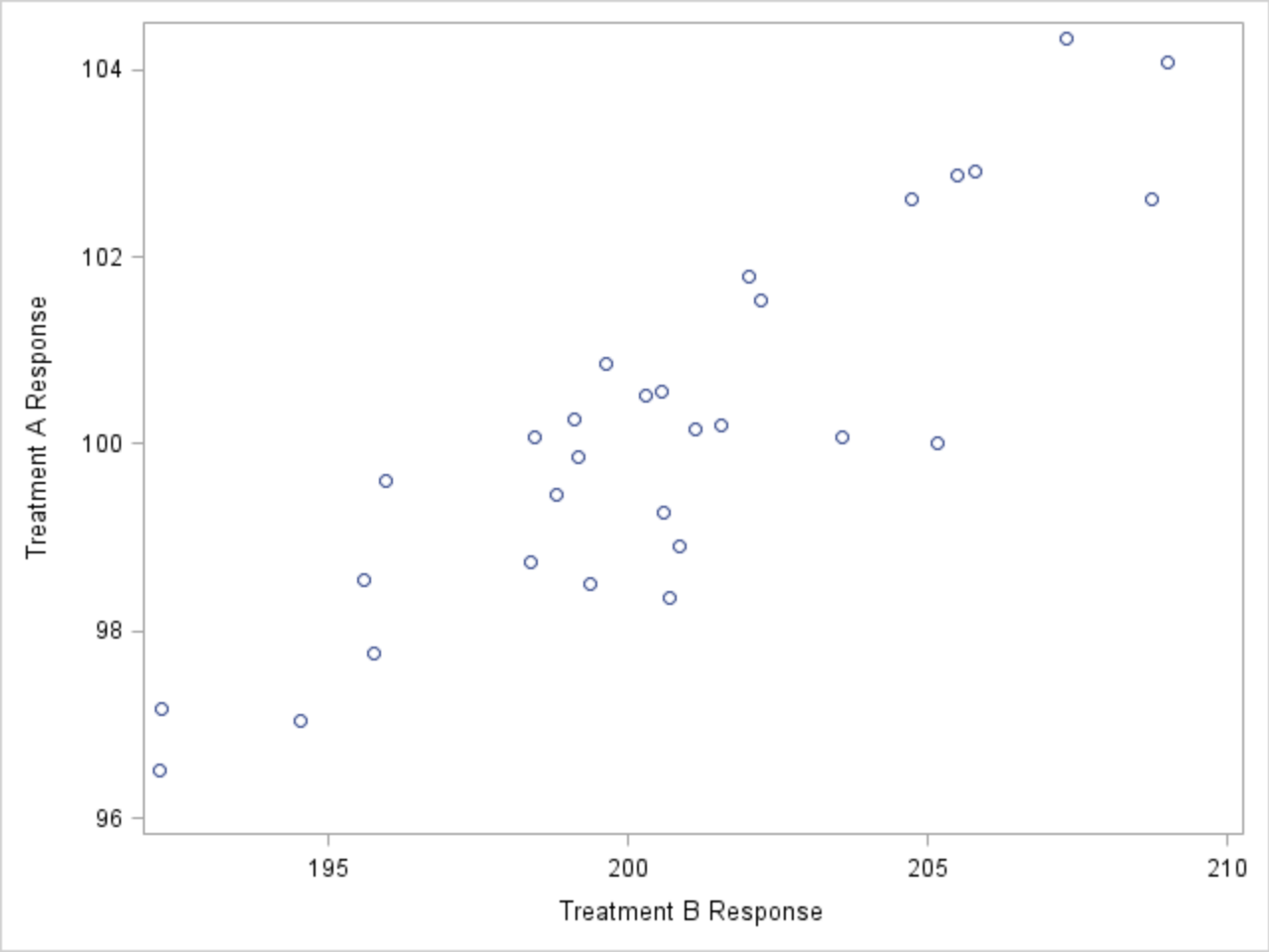
Other options for xaxis and yaxis statements:

values – specify values or intervals:

i.e. values=(0 to 20 by 2)
values=(1 3 10 to 50 by 5 100)
values=(2 3 4 5 6)

min – specify minimum value of axis – i.e. min=0

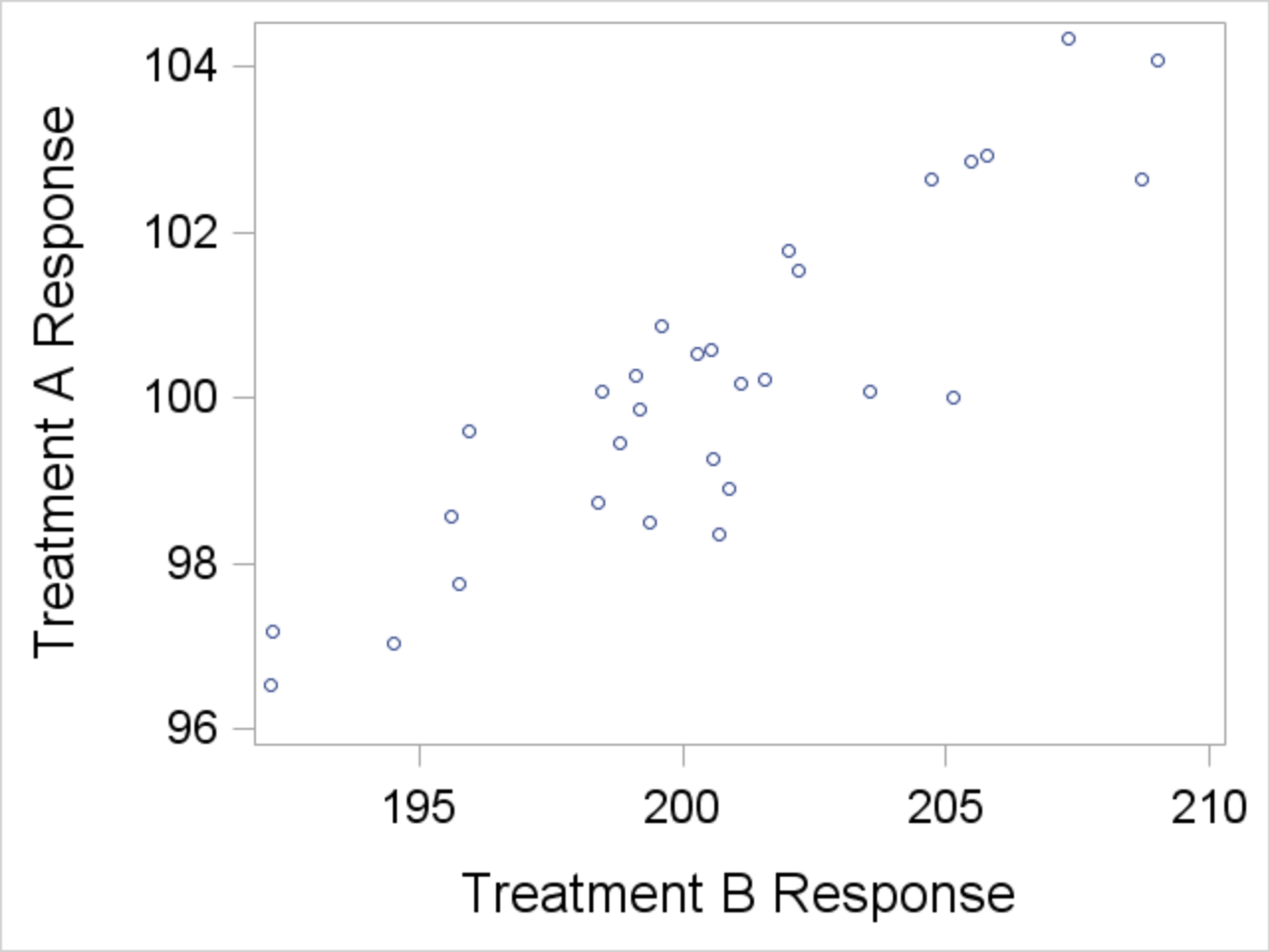
max – specify maximum value of axis – i.e. max=100


104
102
100
98
96
Treatment A Response
195
200
205
210
Treatment B Response

Changing Font Size

```
proc template;
define style MyStyleDefault;
parent=Styles.statistical;
style GraphLabelText from GraphLabelText / fontsize =
   14px;
style GraphValueText from GraphValueText / fontsize =
   12px;
end;
run;

ods html style = MyStyleDefault;
proc sgplot data = scatter;
yaxis label = "Treatment A Response";
xaxis label = "Treatment B Response";
scatter x = B  y = A;
run;
```

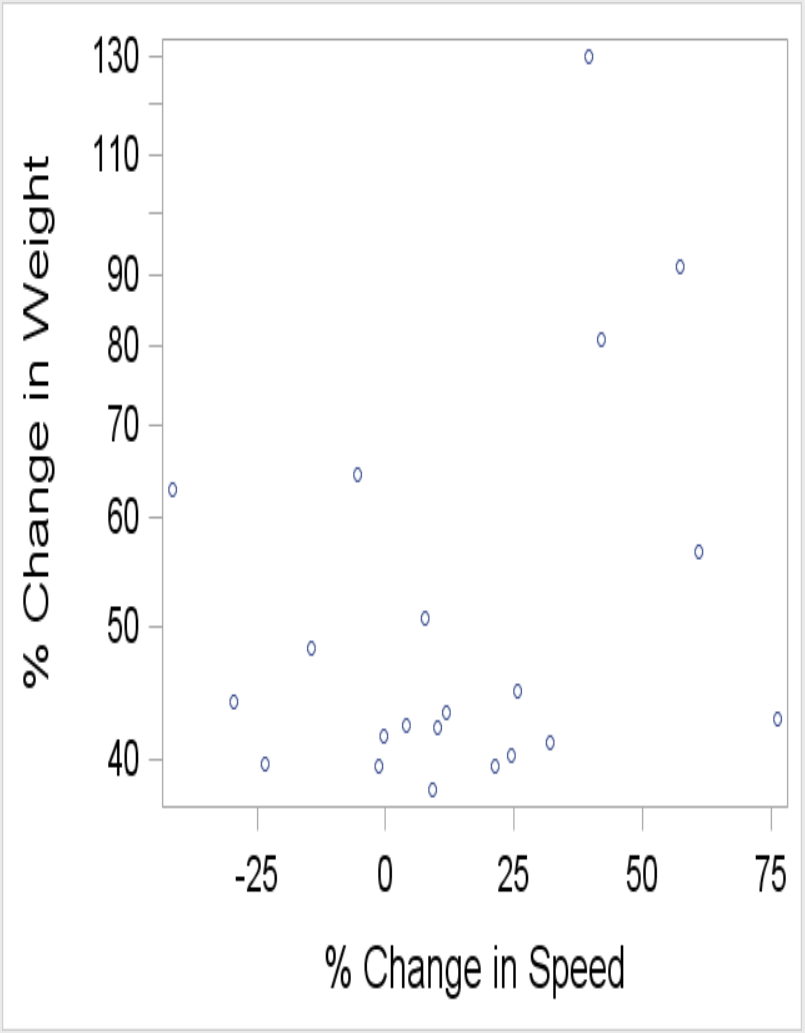
Sneak Preview of Proc Template. Proc Template will be covered again later on in the slides.

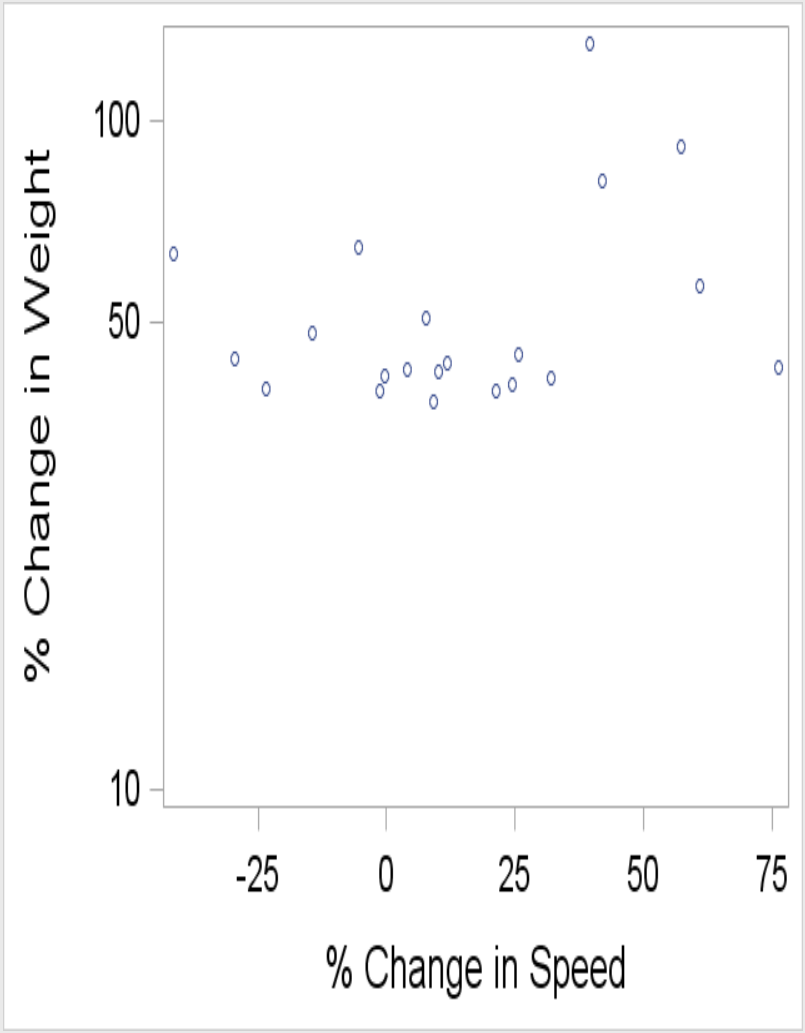

Treatment A Response
104
102
100
98
96
195
200
205
210
Treatment B Response

Plotting the data on the Log Transformed scale

```
proc sgplot data =
   scatter;
yaxis label = "Treatment A
   Response" type = log;
xaxis label = "Treatment B
   Response";
scatter x = B  y = A;
run;
```

```
proc sgplot data =
   scatter;
yaxis label = "Treatment A
   Response" type = log
   logstyle = logexpand;
xaxis label = "Treatment B
   Response ";
scatter x = B  y = A;
run;
```

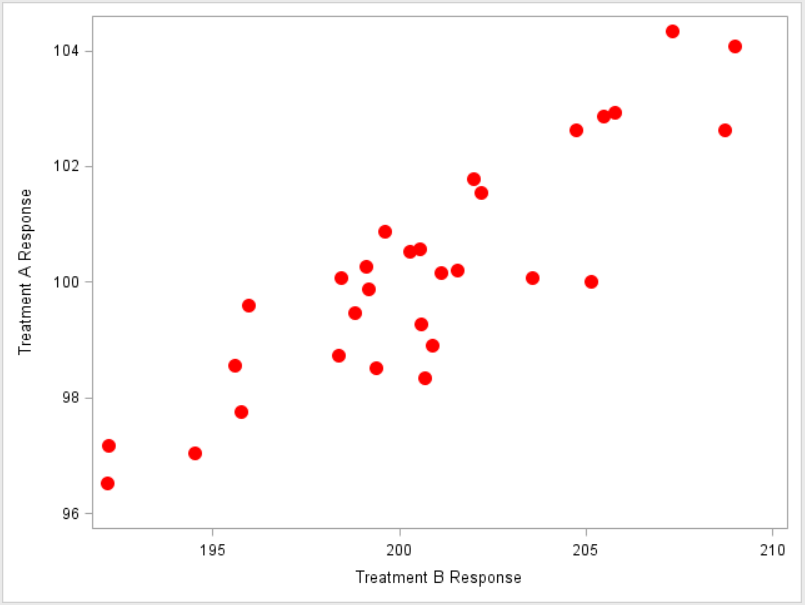

% Change in Weight
130
110
90
80
70
60
50
40
-25
0
25
50
75
% Change in Speed

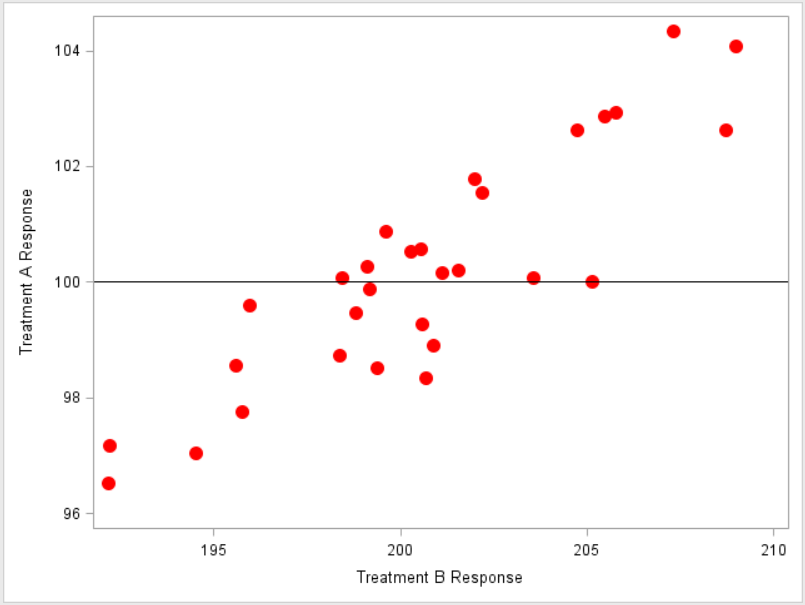

% Change in Weight
100
50
10
-25
0
25
50
75
% Change in Speed

Modifying the shape of the markers and adding a reference line

```
proc sgplot data = data;
yaxis label = "% Change in
   Weight" type = log
   logstyle = logexpand;
xaxis label = "% Change in
   Speed";
scatter x = B  y = A /
   MARKERATTRS = (color =
   red size = 12 symbol =
   circlefilled);
run;
```

```
proc sgplot data = data;
yaxis label = "% Change in
   Weight" type = log
   logstyle = logexpand;
xaxis label = "% Change in
   Speed";
scatter x = B  y = A /
   MARKERATTRS = (color =
   red size = 12 symbol =
   circlefilled);
refline 50 / axis = y
   LINEATTRS = (color =
   black) ;
run;
```


Treatment A Response
104
102
100
98
96
195
200
205
210
Treatment B Response


Treatment A Response
104
102
100
98
96
195
200
205
210
Treatment B Response

Adding Jitter

```
data boxplots_jittered;
set boxplots;
if treatment = "Treatment A" then trtcode = 10;
else if treatment = "Treatment B" then trtcode = 20;
treatment_jittered = trtcode + ranuni(10) - 0.5;
label treatment_jittered = "Treatment";
run;

proc format;
value trtfmt    9.5 - 11.5 = "Treatment A"
                19.5 - 20.5 = "Treatment B"
                other = " ";
run;
```

Making a new numerical variable for treatment

Adding a value between 0 and 1 to each of the treatment codes, and then subtracting 0.5 so the tick mark is in the middle

Using the Format Statement to recode the Treatments so that the character value is displayed on the axis

Adding Jitter

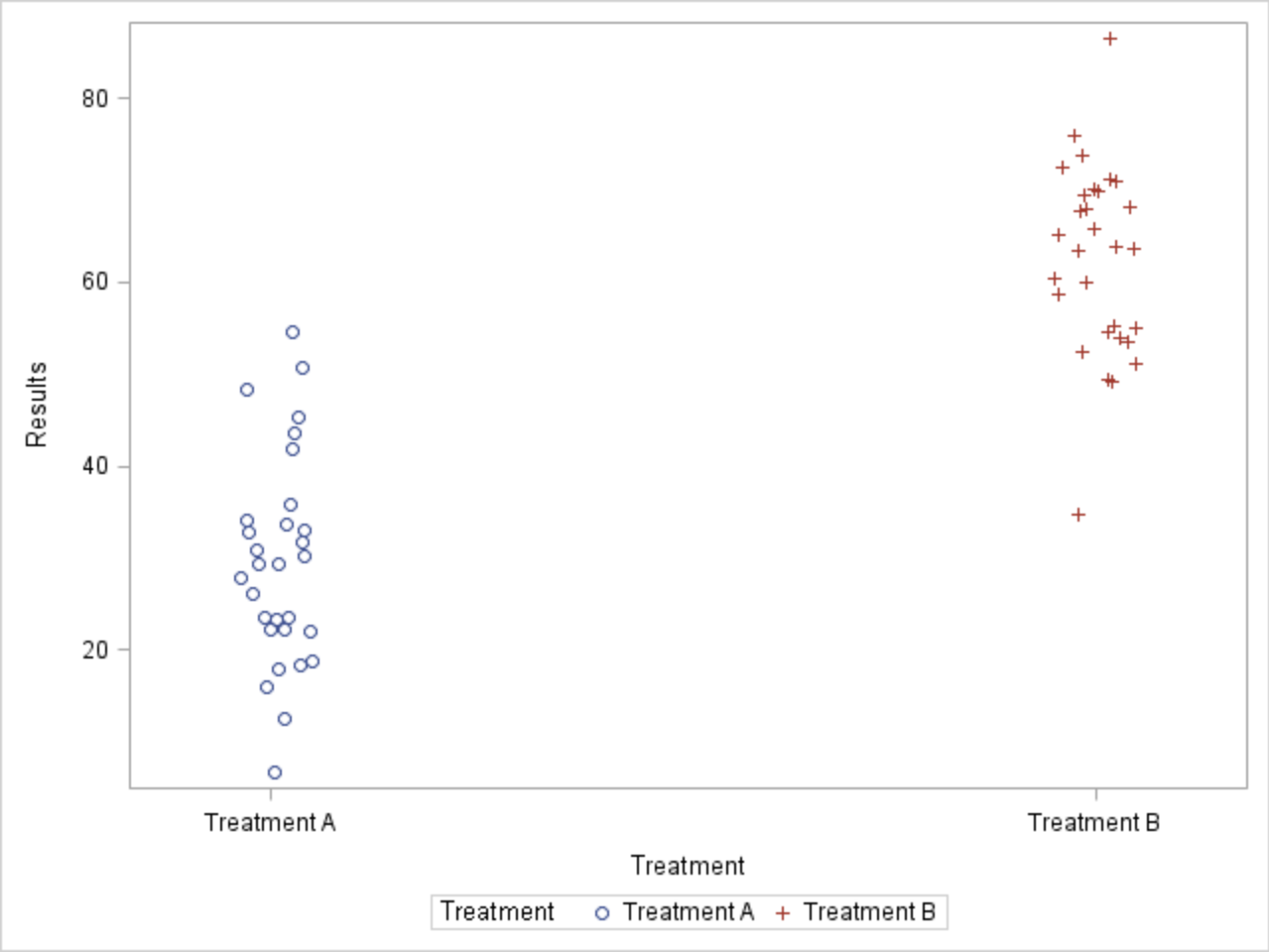
```
proc sgplot data = boxplots_jittered;
scatter y = results x = treatment_jittered / group =
   treatment_jittered ;
format treatment_jittered trtfmt. ;
xaxis label = "Treatment" values = (10,20)
valueshint offsetmax = 0.1 offsetmin = 0.1;
run;
```

Displaying only the tickmarks of interest

Leaving some room between the
left hand and right hand side of the graph

Minimum and Maximum axis values are determined independently of the values you specify in the VALUES= option.

Useful as some of the jittered values will be less than 10 and more than 20.



Jittering will be performed more easily in SAS 9.3.

Removing the border

```
ods graphics on / border = off;
```

SGPLOT Questions

- Questions?

Proc SGPANEL

Proc SGPANEL
Concepts

- The SGPANEL procedure has a required PANELBY statement that is used to define the classifier variables for the panel. This statement must be specified before any plot, axis, or legend statement or else an error occurs.

- The SGPANEL procedure creates the same plots as the SGPLOT procedure, the only difference is that you can panel them in SGPANEL.

- The SGPANEL and SGPLOT procedures contain the same statements except that there is a PANELBY statement in Proc SGPANEL and there are COLAXIS and ROWAXIS statements instead of XAXIS and YAXIS.

Proc SGPANEL Syntax

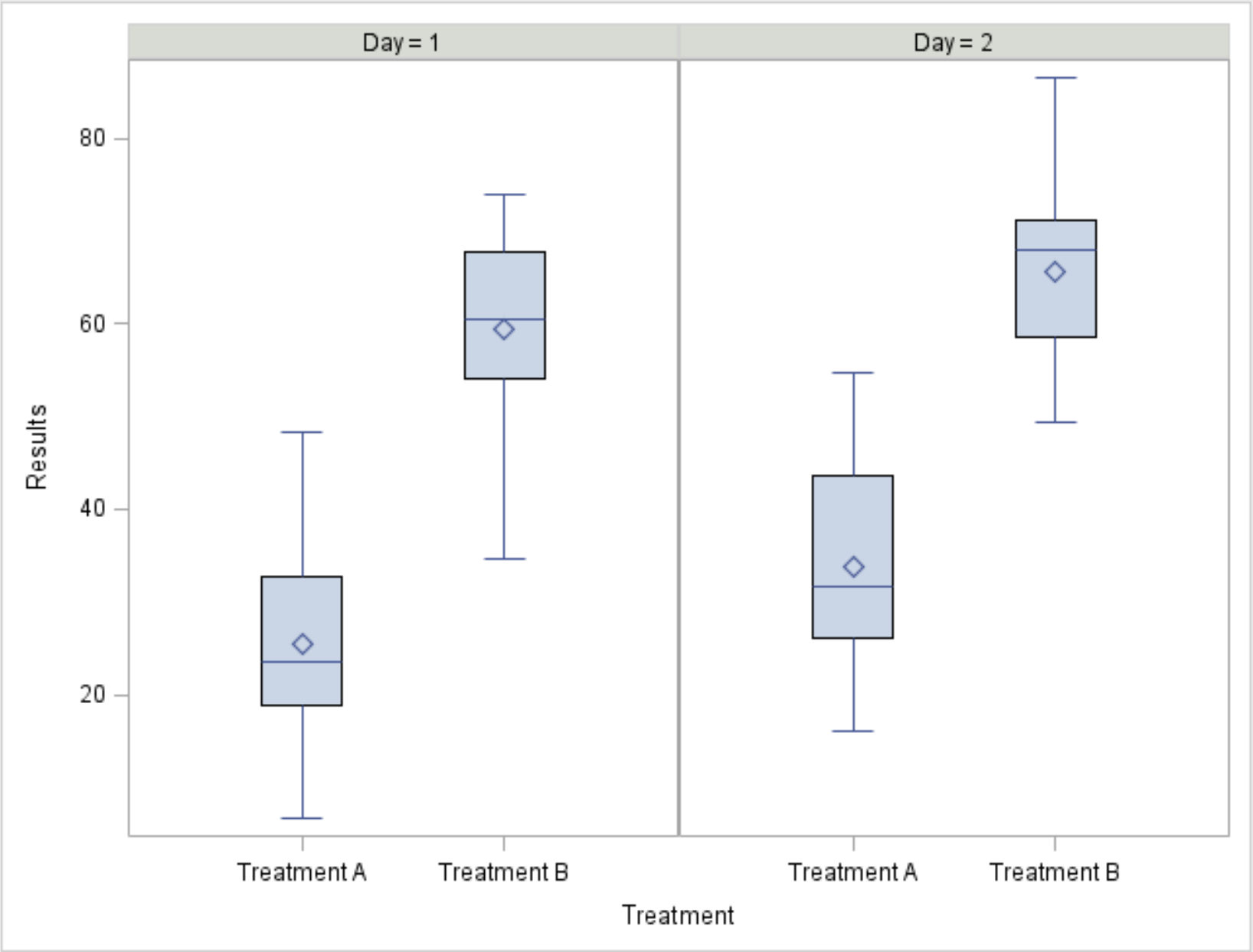
```
PROC SGPANEL < option(s)>;
PANELBY variable(s)</option(s)>;
 BAND X= variable | Y= variable
 UPPER= numeric-value | numeric-variable LOWER= numeric-value | numeric-variable
 </option(s)>;
 COLAXIS <option(s)>;
 DENSITY response-variable </option(s)>;
 DOT category-variable </option(s)>;
 HBAR category-variable </option(s)>
 HBOX response-variable </option(s)>;
 HISTOGRAM response-variable </option(s)>
 HLINE category-variable </option(s)>
 KEYLEGEND <"name(s)"> </option(s)>;
 HLINE variable </option(s)>;
 LOESS X= numeric-variable Y= numeric-variable </option(s)>;
 NEEDLE X= variable Y= numeric-variable </option(s)>;
 PBSPLINE X= numeric-variable Y= numeric-variable </option(s)>;
 REFLINE value(s) </option(s)>;
 REG X= numeric-variable Y= numeric-variable </option(s)>;
 ROWAXIS <option(s)>;
 SCATTER X= variable Y= variable </option(s)>;
 SERIES X= variable Y= variable </option(s)>;
 STEP X= variable Y= variable </option(s)>;
 VBAR category-variable </option(s)>
 VBOX response-variable </option(s)>;
 VLINE category-variable </option(s)>
```

PANELBY Statement

- **Syntax**
- PANELBY variable(s) </ option(s)> option(s) can be one or more of the following:
- COLUMNS= n
- GRIDLAYOUT= LATTICE | PANEL
- MISSING
- NOVARNAME
- ROWS= n
- SPACING= n
- SPARSE
- UNISCALE= ROW | COLUMN | ALL

Boxplot panelled by day

```
proc sgpanel data = boxplots;
panelby day;
vbox results / category = Treatment;
run;
```

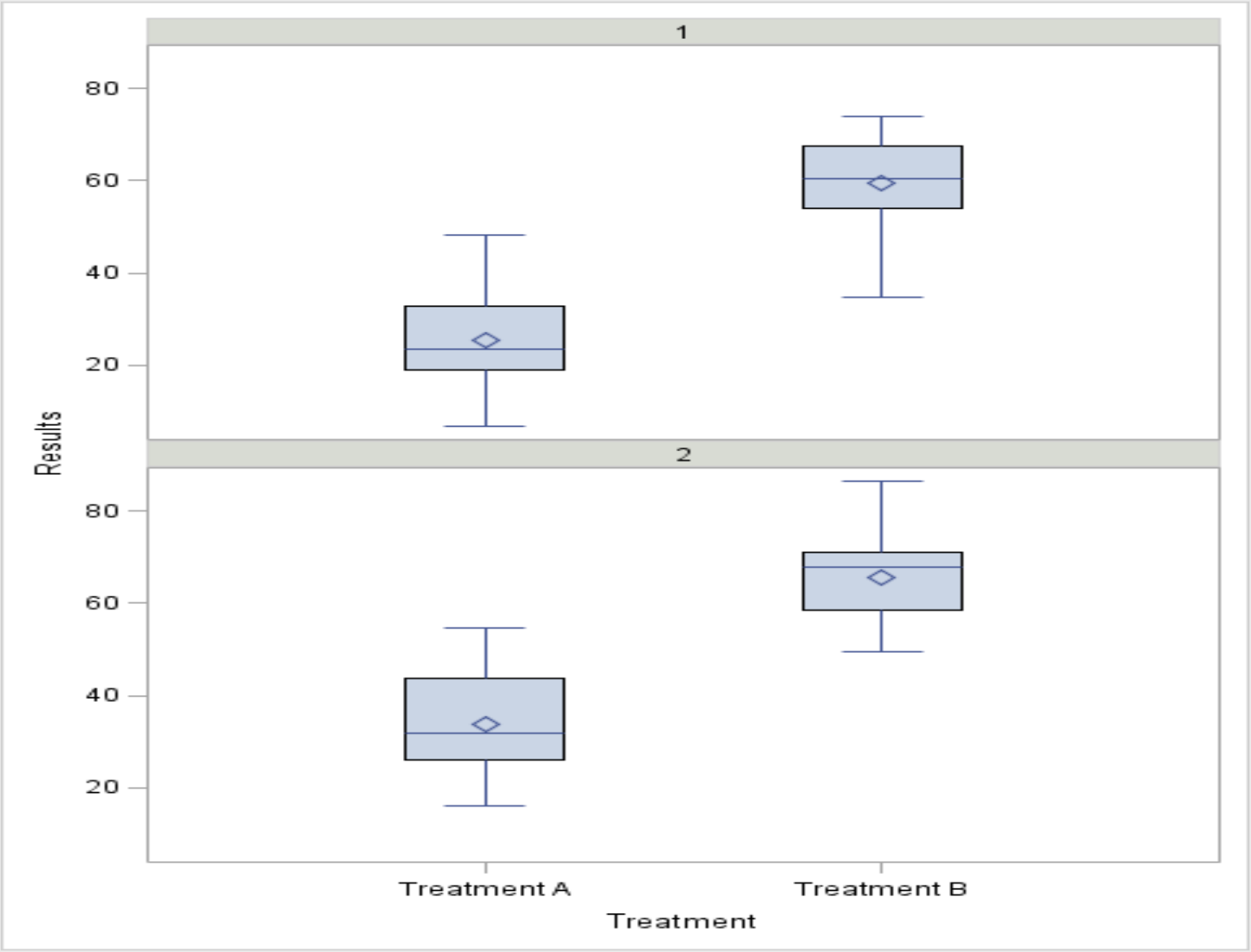

Day = 1
Day = 2
80
60
40
20
Results
Treatment A
Treatment B
Treatment A
Treatment B
Treatment

Boxplot panelled by day

```
proc sgpanel data = boxplots;
panelby day / columns = 1 novarname;
vbox results / category = Treatment;
run;
```

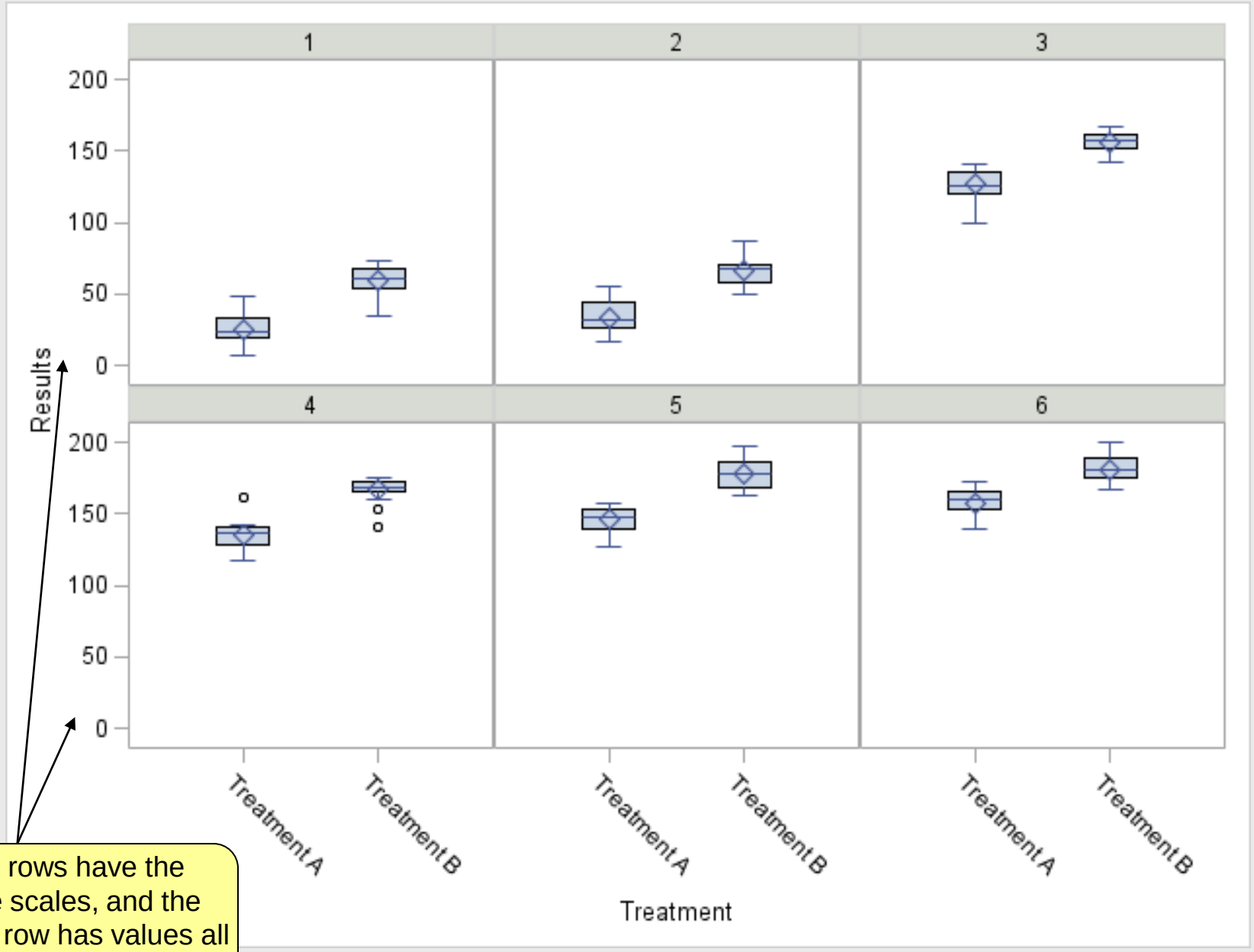
Controlling the amount of **columns**. Rows can be controlled with rows = n.

Novarname omits the variable name from the panel headers. i.e. instead of **Day = 1** being produced, just **1** will be produced in the header.


1
2
80
60
40
20
Results
Treatment A
Treatment B
Treatment

Boxplot panelled by day (for 6 days)

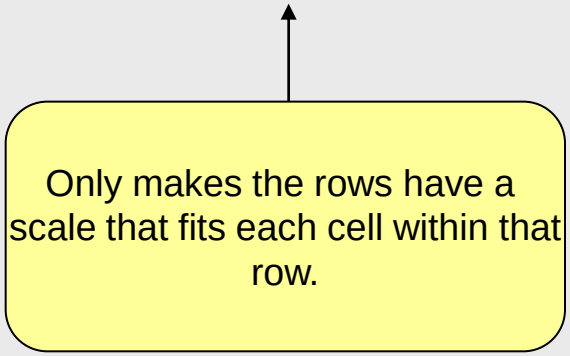
```
proc sgpanel data = boxplots_all;
panelby day / novarname;
vbox results / category = Treatment;
run;
```

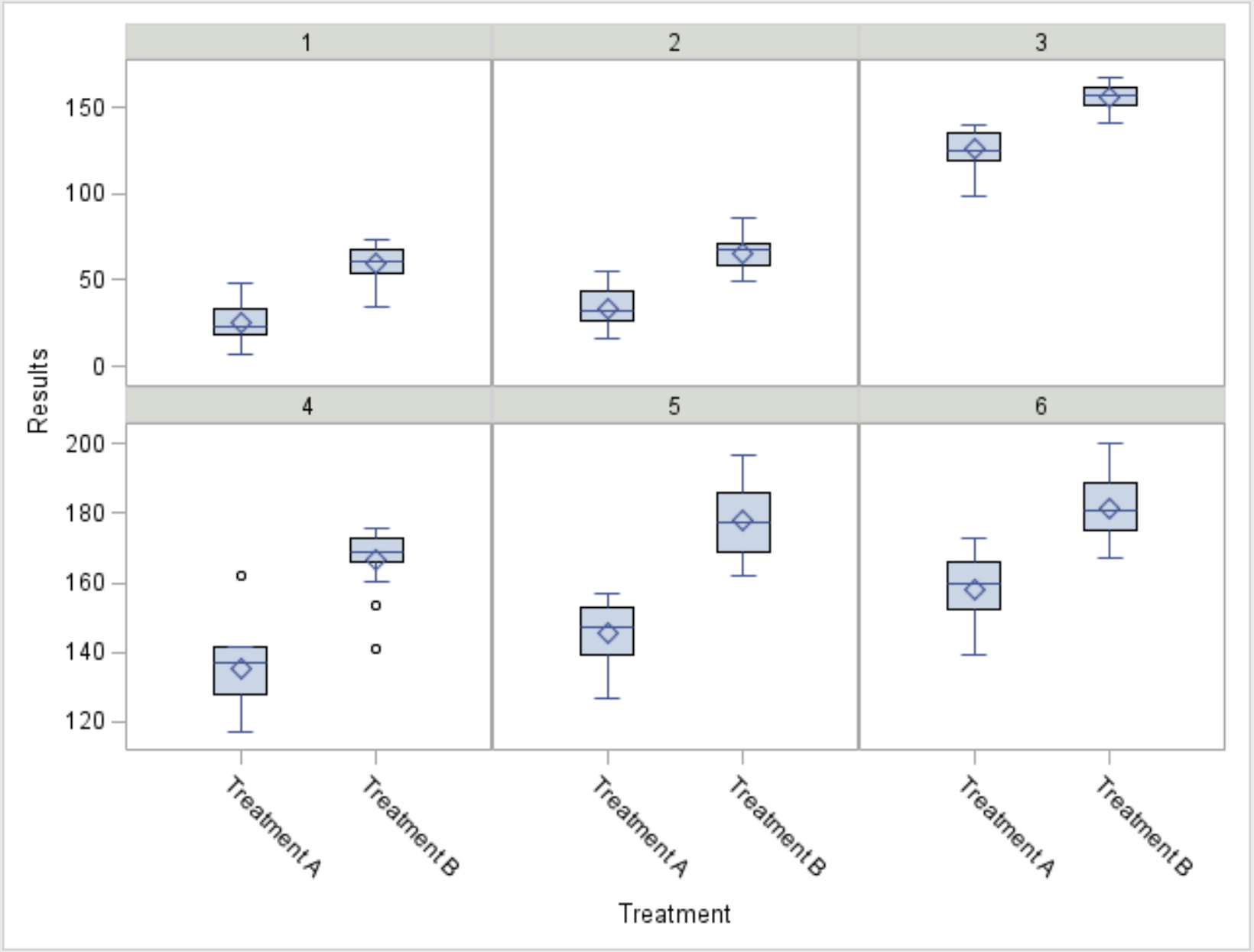


Both rows have the same scales, and the second row has values all above 100.

Boxplot panelled by day
Separate scales for each row

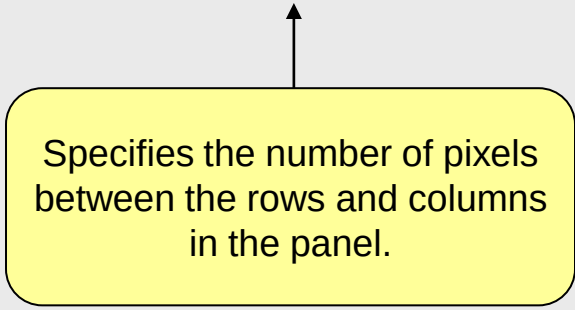
```
proc sgpanel data = boxplots_all;
panelby day / novarname uniscale = column;
vbox results / category = Treatment;
run;
```

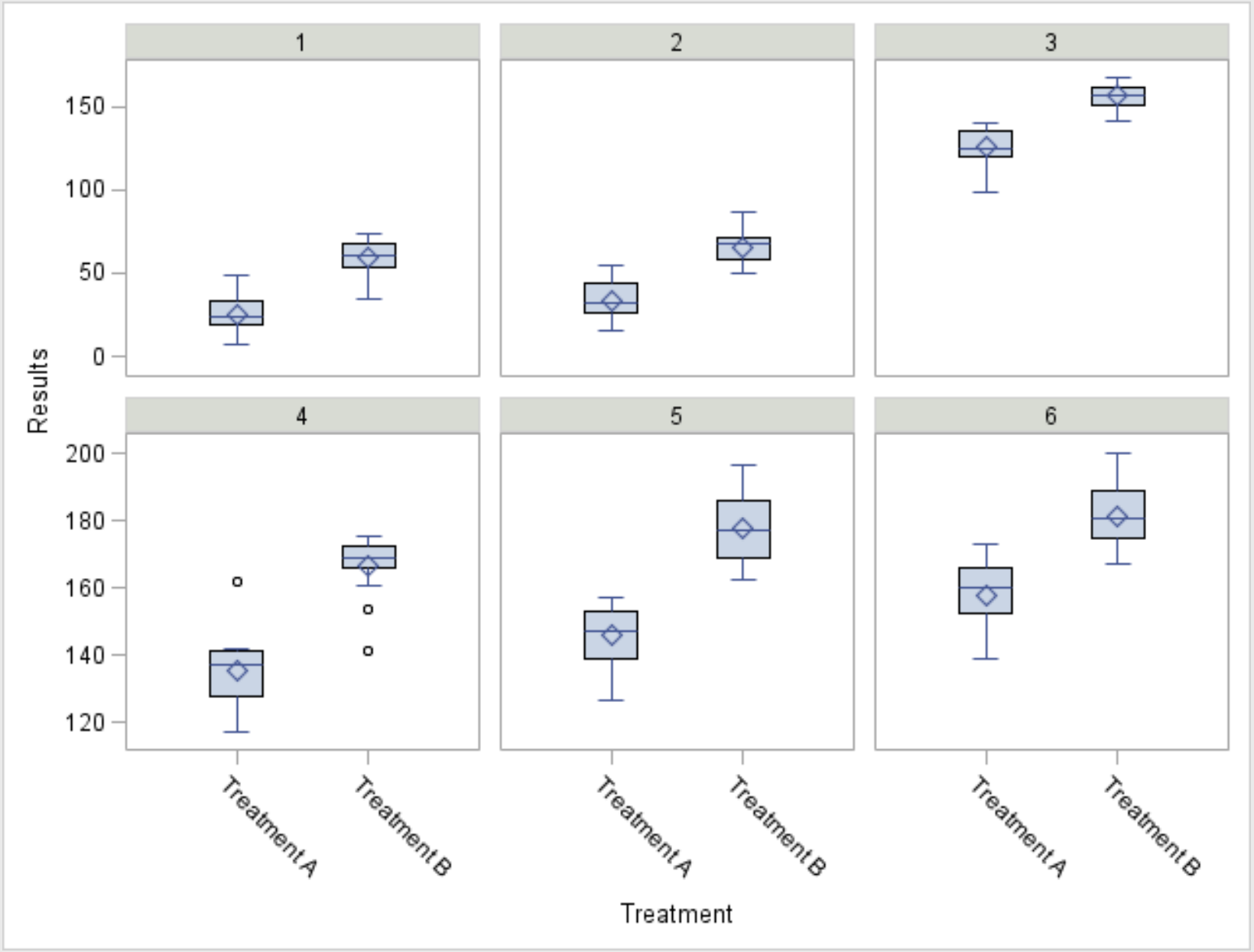



1
2
3
4
5
6
Results
150
100
50
0
200
180
160
140
120
Treatment A
Treatment B
Treatment A
Treatment B
Treatment A
Treatment B
Treatment

Boxplot panelled by day with spacing between the panels

```
proc sgpanel data = boxplots_all;
panelby day / novarname uniscale = column spacing
   = 10;
vbox results / category = Treatment;
run;
```




1
2
3
4
5
6
Results
150
100
50
0
200
180
160
140
120
Treatment A
Treatment B
Treatment

SGPANEL Questions?

- Questions?

Proc SGPLOT and SGPANEL similarities

Basic Plots	Categorical Plots	Fit Plots	Distribution Plots	Other
BAND	DOT	LOESS	DENSITY	KEYLEGEND
NEEDLE	HBAR	PBSPLINE	HISTOGRAM	REFLINE
SCATTER	HBOX	REG		
SERIES	HLINE			
STEP	VBAR			
	VBOX			
	VECTOR			
	VLINE			

VECTOR is not currently In this version.

Proc SGSCATTER

Proc SGSCATTER Syntax

```
PROC SGSCATTER < options>;
  COMPARE X= variable | (variable-1 ... variable-
  n) Y= variable | (variable-1 ... variable-
  n)</options>;
  MATRIX variable-1 < ... variable-n> </options>;
  PLOT plot-request(s) </options>;
```

PLOT Statement

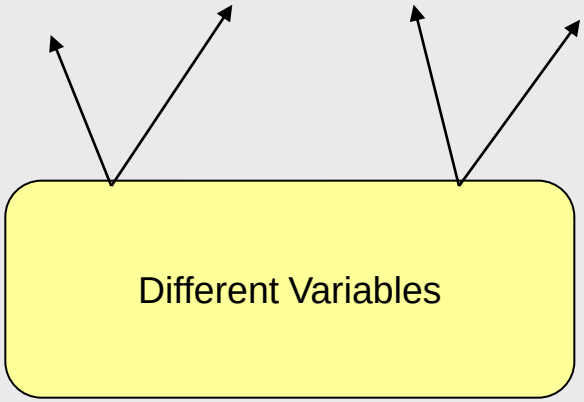
- The PLOT statement is used to create a paneled graph of scatter plots where each graph cell has its own independent set of axes. Each variable pair that you specify in the PLOT statement creates an independent graph cell. You can also overlay fit plots and ellipses on each cell by using options.

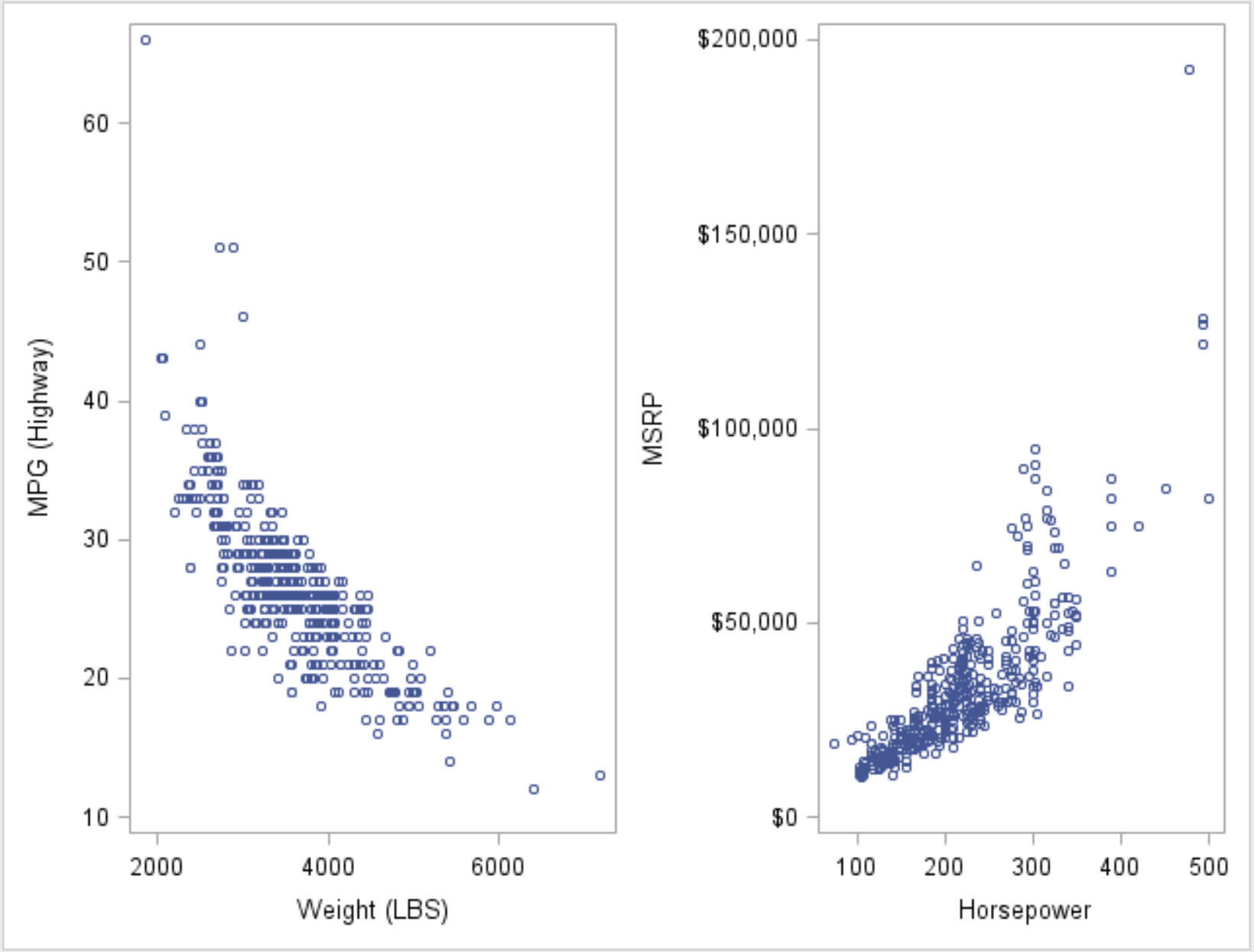
PLOT statement syntax

- PLOT plot-request(s) </ options>;
- COLUMNS= n
- DATALABEL <= variable>
- ELLIPSE <= (options)>
- GRID
- GROUP= variable
- LEGEND = (options)
- LOESS <= (options)>
- MARKERATTRS= style-element <(options)> | (options)
- NOLEGEND
- PBSPLINE <= (options)>
- REFTICKS
- REG <= (options)>
- ROWS= n
- SPACING= n
- UNISCALE= X | Y | ALL

Using PLOT statement

```
proc sgscatter data=sashelp.cars;
  plot mpg_highway*weight msrp*horsepower;
run;
```




MPG (Highway)
60
50
40
30
20
10
2000
4000
6000
Weight (LBS)
MSRP
$200,000
$150,000
$100,000
$50,000
$0
100
200
300
400
500
Horsepower

COMPARE statement

- The COMPARE statement is used to create a shared axis panel, also called an MxN matrix

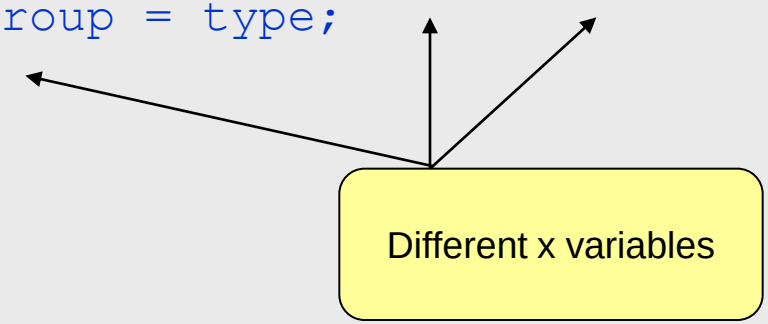
- Basically it is used to compare different x-variables with the same y variable.

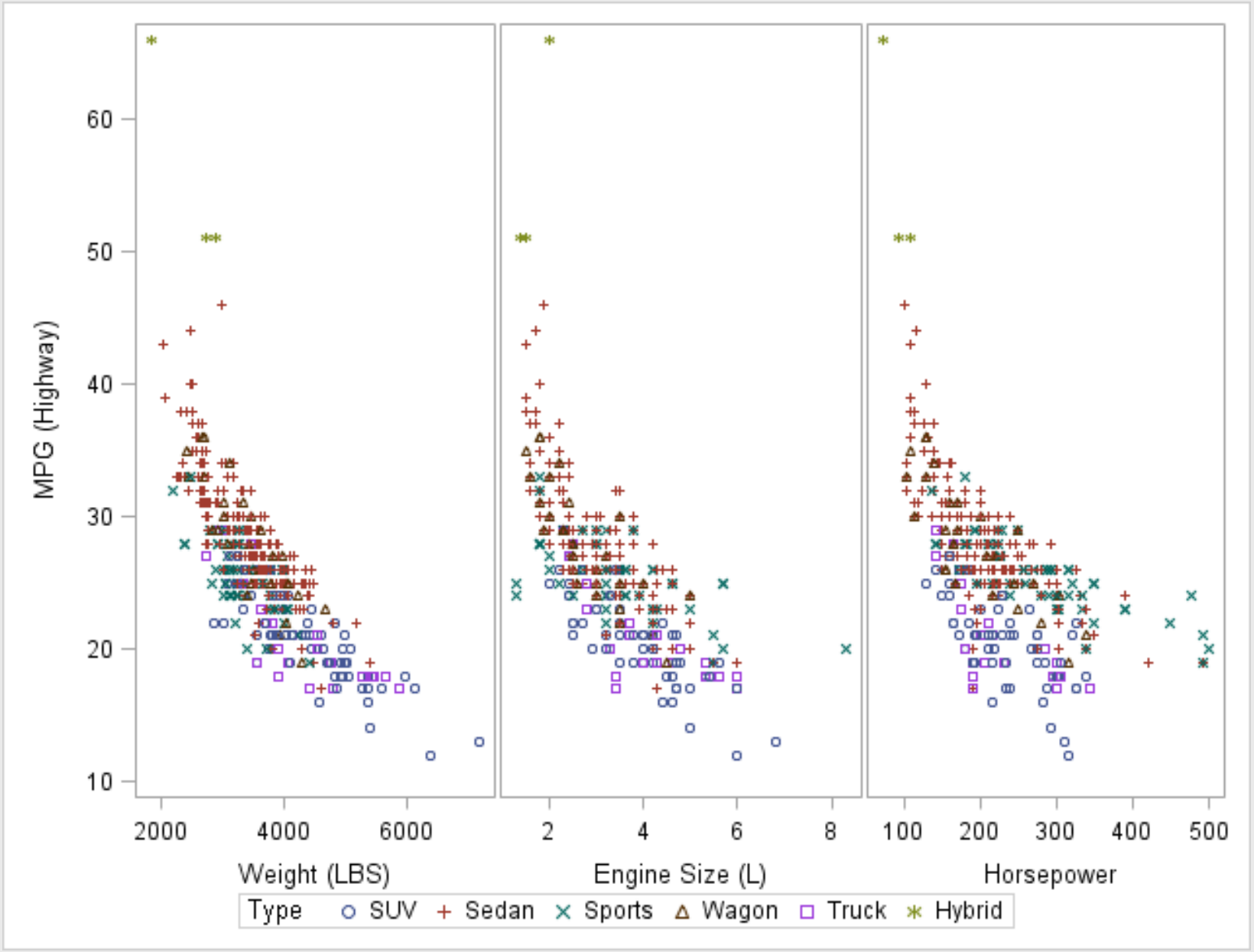
COMPARE statement syntax

- COMPARE X= variable | (variable-1 ... variable-n) Y= variable | (variable-1 ... variable-n) </options>;
- DATALABEL <= variable>
- ELLIPSE <= (options)>
- GRID
- GROUP= variable
- LEGEND = (options)
- LOESS <= (options)>
- MARKERATTRS= style-element <(options)> | (options)
- NOLEGEND
- PBSPLINE <= (options)>
- REFTICKS
- REG <= (options)>
- SPACING= n

Using COMPARE statement

```
proc sgscatter data=sashelp.cars;
  compare y=mpg_highway x=(weight enginesize
   horsepower) / group = type;
run;
```




MPG (Highway)
60
50
40
30
20
10
2000
4000
6000
Weight (LBS)
2
4
6
8
Engine Size (L)
100
200
300
400
500
Horsepower
Type
SUV
Sedan
Sports
Wagon
Truck
Hybrid

MATRIX statement

- The MATRIX statement is used to create a scatter plot matrix

MATRIX statement syntax

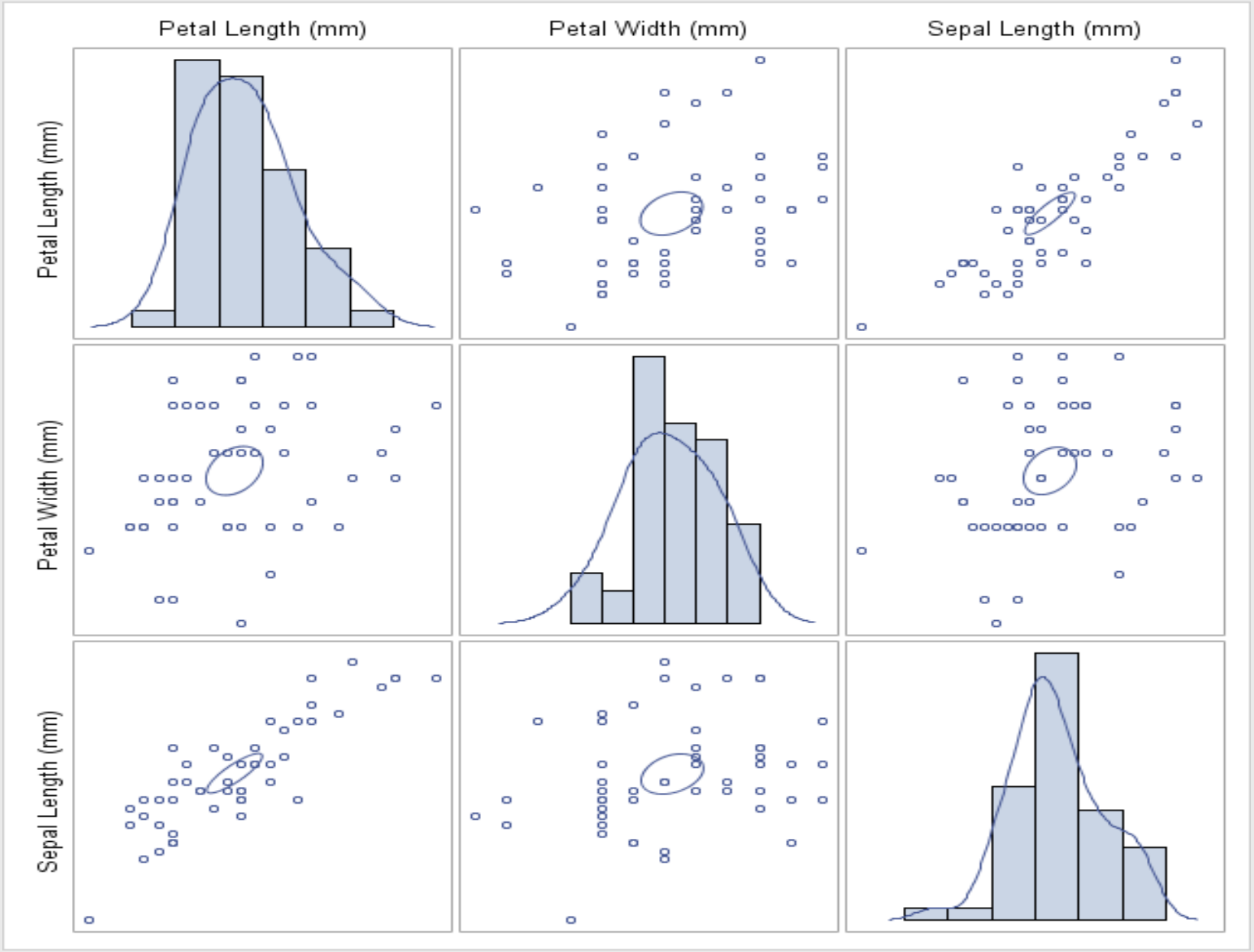
- MATRIX variable-1 < ... variable-n> </ options>;
- DATALABEL= variable
- DIAGONAL= (graph-list)
- ELLIPSE <= (options)>
- GROUP= variable
- LEGEND = (options)
- MARKERATTRS= style-element <(options)> | (options)
- NOLEGEND
- START= BOTTOMLEFT | TOPLEFT

Using MATRIX statement

```
proc sgscatter data=sashelp.iris
                (where=(species eq 'Virginica'));
matrix petallength petalwidth sepallength
      / ellipse=(type=mean)
        diagonal=(histogram kernel);
run;
```

Subsetting the dataset

Produces a 3 * 3 matrix plot


Petal Length (mm)
Petal Width (mm)
Sepal Length (mm)
Petal Length (mm)
Petal Width (mm)
Sepal Length (mm)

SGSCATTER Questions?

- Questions?

Proc TEMPLATE
Graph Template Language (GTL)

Graph Template Language

- The SAS/GRAPH ***Graph Template Language*** (GTL) is an extension to the Output Delivery System (ODS) that enables you to create sophisticated analytical graphics that are not available from traditional SAS/GRAPH procedure statements.
- The GTL templates are defined with PROC TEMPLATE.
- The GTL templates are rendered using the SGRENDER procedure, which specifies a data source that contains appropriate data values and the template to use for rendering the graph.
- GTL Reference link: http://support.sas.com/documentation/cdl/en/grstatgraph/63878/HTML/default/viewer.htm#/documentation/cdl/en/grstatgraph/63878/HTML/default/p0891gx3y0z8xqn1k9ijhv5xughi.htm

BEGINGRAPH statement syntax

- BEGINGRAPH </ option(s)> ;
- <GTL-global-statements >
- GTL-layout-block
- <GTL-global-statements >
- ENDGRAPH ;

LAYOUT statements

- LAYOUT DATALATTICE Statement
- LAYOUT DATAPANEL Statement
- LAYOUT GRIDDED Statement
- LAYOUT LATTICE Statement
- LAYOUT OVERLAY Statement
- LAYOUT OVERLAYEQUATED Statement
- LAYOUT OVERLAY3D Statement
- LAYOUT PROTOTYPE Statement

Layout GRIDDED

- A gridded layout is the simplest organization of cells. By default, the output of each contained statement is placed in its own cell. The cells are arranged into one column and each is center aligned.

Layout LATTICE

- The lattice layout is a multicell layout that combines features of gridded and overlay layouts and offers reserved areas for additional formatting:
  - 4 sidebars (top, bottom, left, and right) that span all rows and columns
  - individual row and column headings
  - individual cell headings
  - axis scaling on a per cell, per row, per column, all rows, or all columns basis

Layout OVERLAY

- The overlay layout creates a single-cell plot. It is used primarily to combine (superimpose) plots.

- Plots are "stacked"in the order you declare them, with the first on the bottom.

- All plots in the overlay "lose" their individual axes. Instead consolidated sets of axes "owned" by the layout are created.

PLOT statements

- BANDPLOT Statement
- BARCHART Statement
- BARCHARTPARM Statement
- BIHISTOGRAM3DPARM Statement
- BLOCKPLOT Statement
- BOXPLOT Statement
- BOXPLOTPARM Statement
- CONTOURPLOTPARM Statement
- DENSITYPLOT Statement
- DROPLINE Statement
- ELLIPSE Statement
- ELLIPSEPARM Statement
- FRINGEPLOT Statement
- HISTOGRAM Statement
- HISTOGRAMPARM Statement
- LINEPARM Statement
- LOESSPLOT Statement
- MODELBAND Statement
- NEEDLEPLOT Statement
- PBSPLINEPLOT Statement
- REFERENCELINE Statement
- REGRESSIONPLOT Statement
- SCATTERPLOT Statement
- SCATTERPLOTMATRIX Statement
- SERIESPLOT Statement
- STEPPLOT Statement
- SURFACEPLOTPARM Statement
- VECTORPLOT Statement

Obtaining GTL for a boxplot and for a scatterplot

```
proc sgplot data = boxplots tmplout =
   "C:\Documents and Settings\KYH44612\Desktop\SAS
   Graphics\Analysis\Template_for_boxplots.sas" ;
vbox results / category = Treatment;
run;

proc sgplot data = boxplots tmplout =
   "C:\Documents and Settings\KYH44612\Desktop\SAS
   Graphics\Analysis\Template_for_scatterplot.sas"
   ;
scatter y = results  x = Treatment;
run;
```

GTL for a boxplot

```
proc template;
define statgraph sgplot;
dynamic _ticklist_;
begingraph;
layout overlay / xaxisopts=(type=Discrete
  discreteOpts=(tickValueList=_ticklist_));
   BoxPlot X=Treatment Y=Results /
  SortOrder=Internal primary=true
  LegendLabel="Results" NAME="VBOX";
   ;
endlayout;
endgraph;
end;
run;
```

The Define statement specifies the type as **statgraph** and provides the name of the template.

GTL for a boxplot

```
proc template;
define statgraph sgplot;
dynamic _ticklist_;
begingraph;
layout overlay / xaxisopts=(type=Discrete
  discreteOpts=(tickValueList=_ticklist_));
   BoxPlot X=Treatment Y=Results /
  SortOrder=Internal primary=true
  LegendLabel="Results" NAME="VBOX";
   ;
endlayout;
endgraph;
end;
run;
```

The template definition is provided inside the **begingraph/endgraph** block.

GTL for a boxplot

```
proc template;
define statgraph sgplot;
dynamic _ticklist_;
begingraph;
layout overlay / xaxisopts=(type=Discrete
  discreteOpts=(tickValueList=_ticklist_));
   BoxPlot X=Treatment Y=Results /
  SortOrder=Internal primary=true
  LegendLabel="Results" NAME="VBOX";
   ;
endlayout;
endgraph;
end;
run;
```

The body of this template contains a **layout overlay** block that contains a **Boxplot**.

GTL for a scatterplot

```
proc template;
define statgraph sgplot;
begingraph;
layout overlay;
   ScatterPlot X=Treatment Y=Results /
  primary=true LegendLabel="Results"
  NAME="SCATTER";
   ;
endlayout;
endgraph;
end;
run;
```

Proc SGRENDER

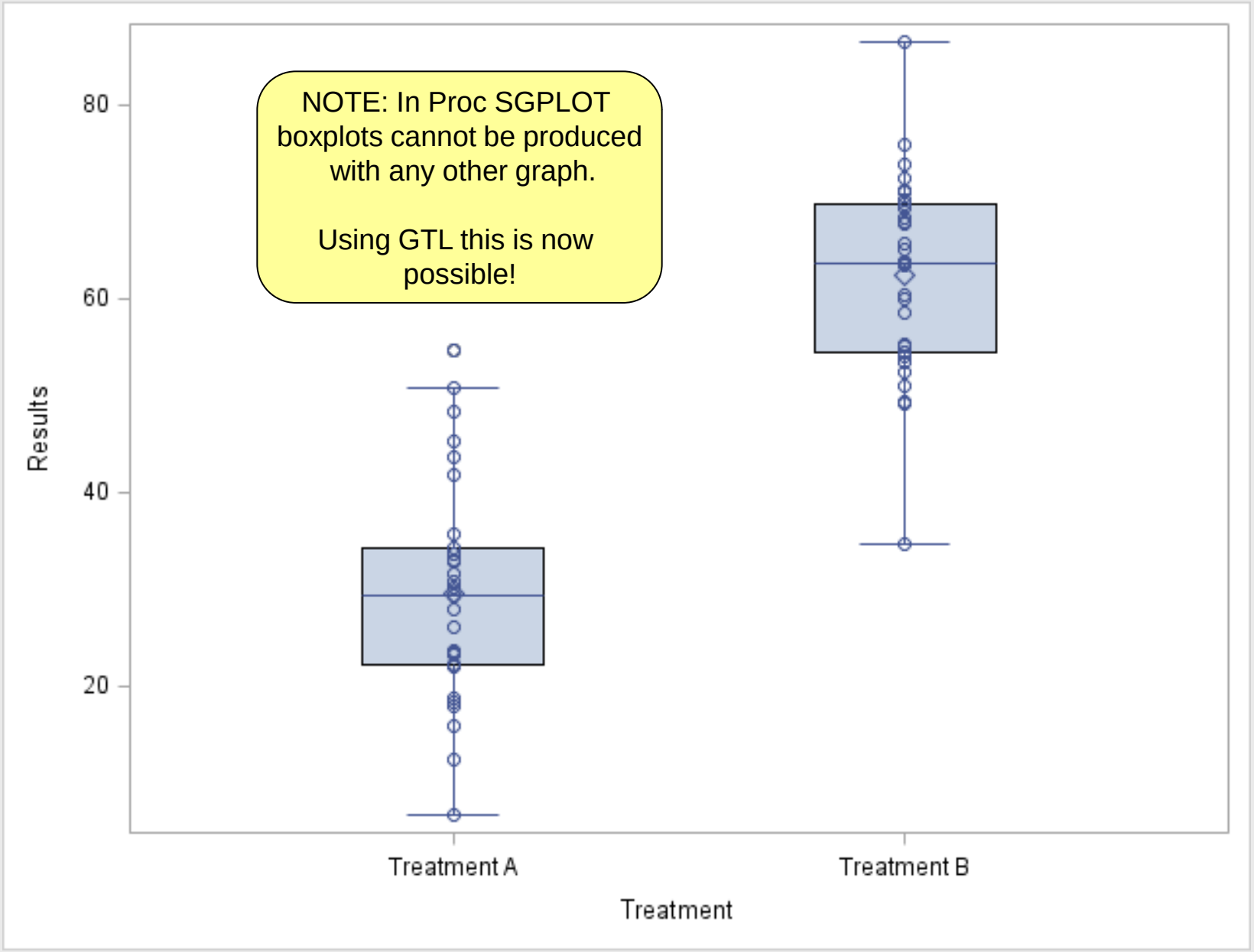
- The SGRENDER procedure produces graphical output from templates that are created with the Graph Template Language (GTL). The GTL is a comprehensive language for creating statistical graphics, which can be used to create customized layouts and graphs that are beyond the scope of the Statistical Graphics procedures.

Proc SGRENDER syntax

- PROC SGRENDER < option(s)>;
  DYNAMIC variable-assignment(s);

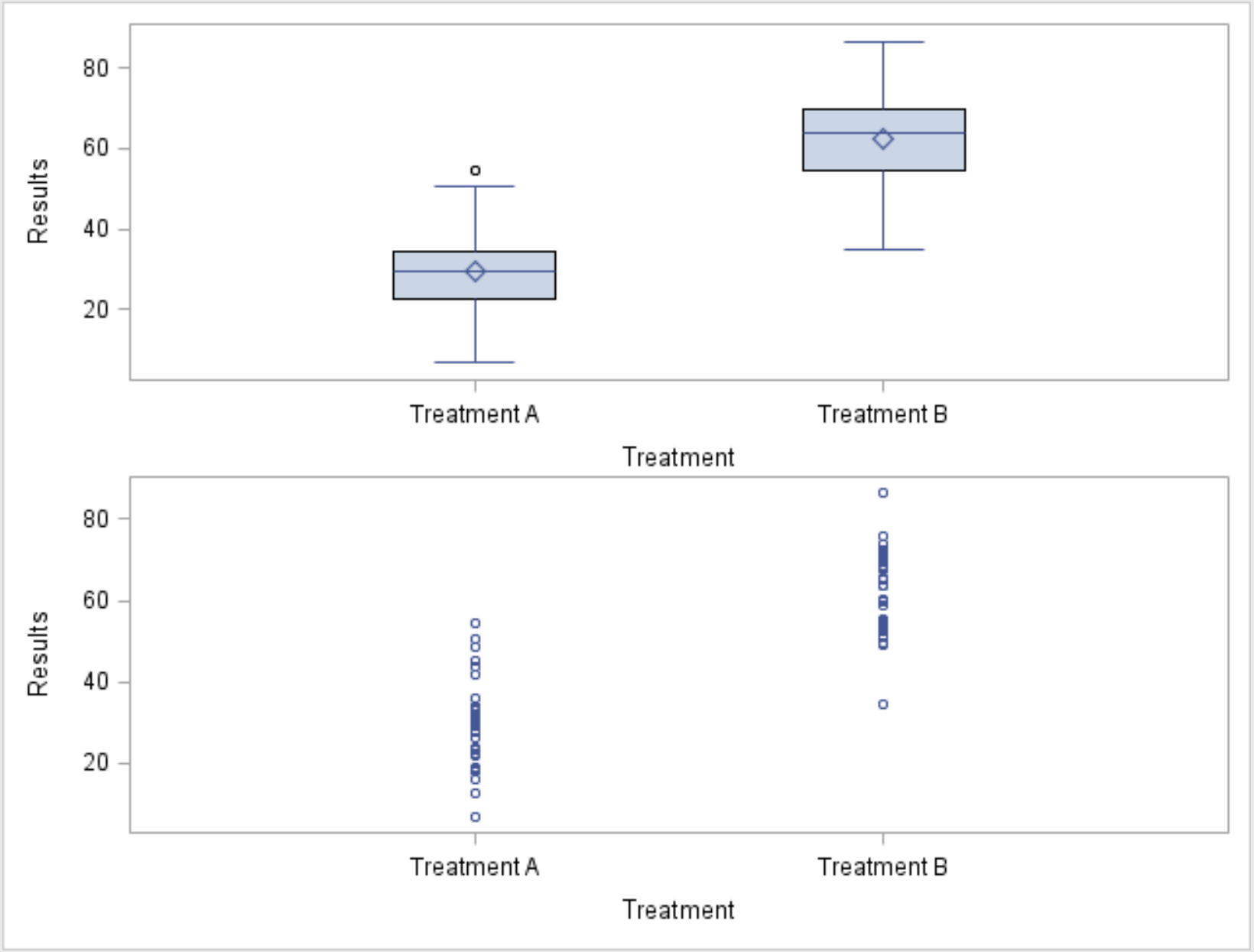
Using GTL to plot a scatterplot and boxplot on one graph

```
proc template;
define statgraph sgplot;
dynamic _ticklist_;
begingraph;
layout overlay / xaxisopts=(type=Discrete
   discreteOpts=(tickValueList=_ticklist_));
    BoxPlot X=Treatment Y=Results / SortOrder=Internal
   primary=true LegendLabel="Results" NAME="VBOX";
    ScatterPlot X=Treatment Y=Results / primary=true
   LegendLabel="Results" NAME="SCATTER";
    ;
endlayout;
endgraph;
end;
run;

proc sgrender data=boxplots template=sgplot;
run;
```


NOTE: In Proc SGPLOT boxplots cannot be produced with any other graph.
Using GTL this is now possible!
80
60
40
20
Results
Treatment A
Treatment B
Treatment

Using GTL to plot a scatterplot and boxplot on one graph in separate panels

```
proc template;
define statgraph sgmypanel;
dynamic _ticklist_;
begingraph;
layout lattice;
   layout overlay / xaxisopts=(type=Discrete
   discreteOpts=(tickValueList=_ticklist_) LABEL = "Treatment")
                          yaxisopts = (LABEL = "Results");
        BoxPlot X=Treatment Y=Results / SortOrder=Internal
   primary=true LegendLabel="Results" NAME="VBOX";
   ;
   endlayout;

   layout overlay;
        ScatterPlot X=Treatment Y=Results / primary=true
   LegendLabel="Results" NAME="SCATTER";
   ;
   endlayout;
endlayout;
endgraph;
end;
run;
```


Results
80
60
40
20
Treatment A
Treatment B
Treatment
Results
80
60
40
20
Treatment A
Treatment B
Treatment

Boxplot and Scatterplot with text in the Scatterplot

Produces two columns of plots

```
layout lattice / columns = 2;
   layout overlay / xaxisopts=(type=Discrete
   discreteOpts=(tickValueList=_ticklist_) LABEL = "Treatment")
                          yaxisopts = (LABEL = "Results");
   BoxPlot X=Treatment Y=Results / SortOrder=Internal primary=true
   LegendLabel="Results" NAME="VBOX";
   ;
   endlayout;

   layout overlay;
   ScatterPlot X=Treatment Y=Results / primary=true
   LegendLabel="Results" NAME="SCATTER";
   ;
        layout gridded / columns=3 border=TRUE autoalign=(BottomLeft
   TopLeft);
                          entry "";      entry "Treatment A";   entry
   "Treatment B";
                          entry "Mean:";     entry "Mean of A";   entry
   "Mean of B";
                          entry "Std Dev:";     entry "SD of A";
   entry "SD of B";
         endlayout;
   endlayout;
endlayout;
```

Boxplot and Scatterplot with text in the Scatterplot

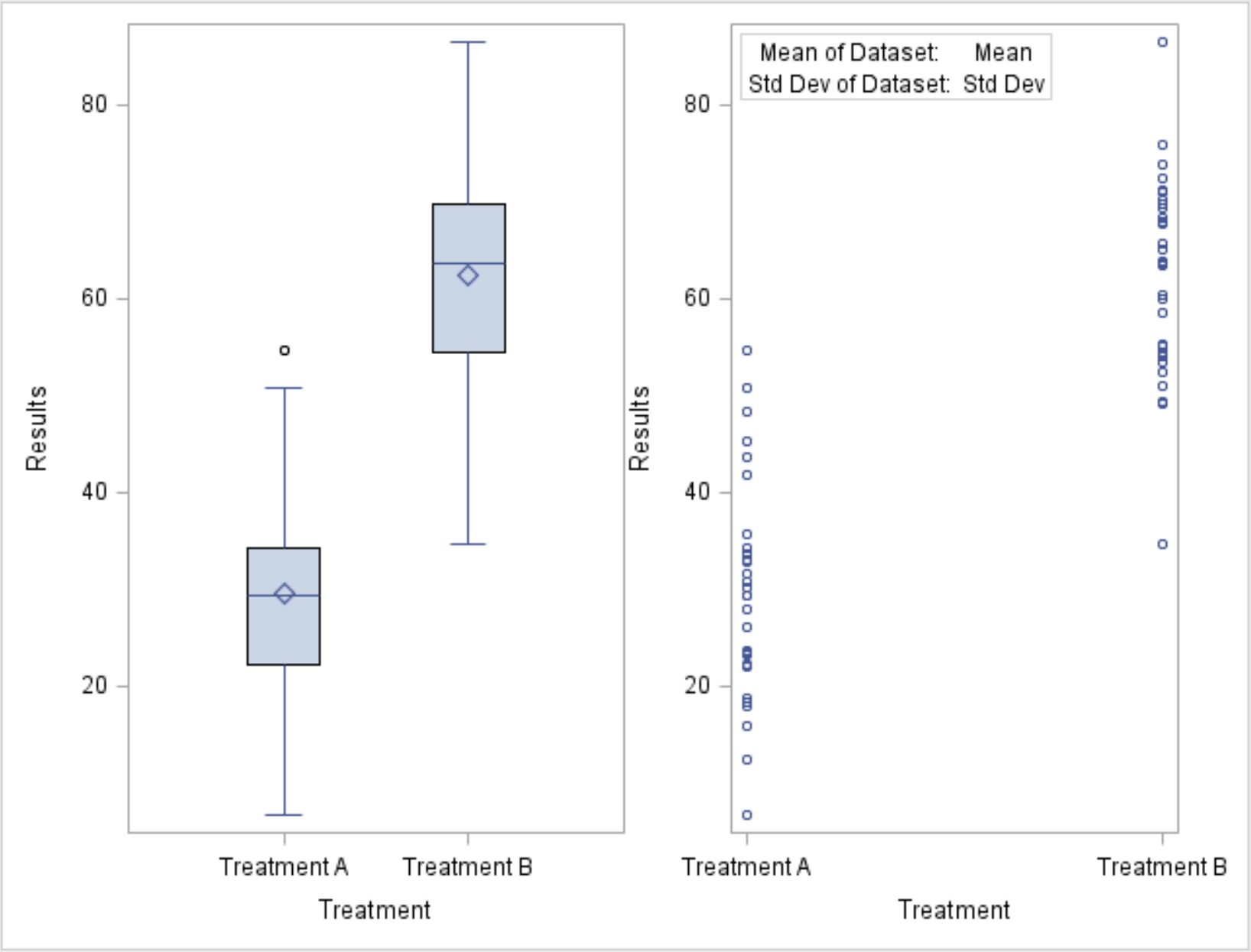
```
layout lattice / columns = 2;
   layout overlay / xaxisopts=(type=Discrete
   discreteOpts=(tickValueList=_ticklist_) LABEL = "Treatment")
                            yaxisopts = (LABEL = "Results");
   BoxPlot X=Treatment Y=Results / SortOrder=Internal primary=true
   LegendLabel="Results" NAME="VBOX";
   ;
   endlayout;

   layout overlay;
   ScatterPlot X=Treatment Y=Results / primary=true
   LegendLabel="Results" NAME="SCATTER";
   ;
        layout gridded / columns=3 border=TRUE autoalign=(BottomLeft
   TopLeft);
                           entry "";      entry "Treatment A";   entry
   "Treatment B";
                           entry "Mean:";     entry "Mean of A";   entry
   "Mean of B";
                           entry "Std Dev:";     entry "SD of A";
   entry "SD of B";
          endlayout;
   endlayout;
endlayout;
```

Gridded layout produces the text. The gridded layout is nested within the layout that produces the scatterplot

Boxplot and Scatterplot with text in the Scatterplot

```
layout lattice / columns = 2;
   layout overlay / xaxisopts=(type=Discrete
   discreteOpts=(tickValueList=_ticklist_) LABEL = "Treatment")
                           yaxisopts = (LABEL = "Results");
   BoxPlot X=Treatment Y=Results / SortOrder=Internal primary=true
   LegendLabel="Results" NAME="VBOX";
   ;
   endlayout;

   layout overlay;
   ScatterPlot X=Treatment Y=Results / primary=true
   LegendLabel="Results" NAME="SCATTER";
   ;

        layout gridded / columns=2 border=TRUE autoalign=(BottomLeft
   TopLeft TopRight);
               entry "Mean of Dataset:";     entry "Mean";
               entry "Std Dev of Dataset:";  entry "Std Dev";
        endlayout;

   endlayout;
endlayout;
```

Autoalign automatically aligns the box where there is no Points (if possible)


80
60
40
20
Results
Treatment A
Treatment B
Treatment
Mean of Dataset: Mean
Std Dev of Dataset: Std Dev

Using Dynamic Variables

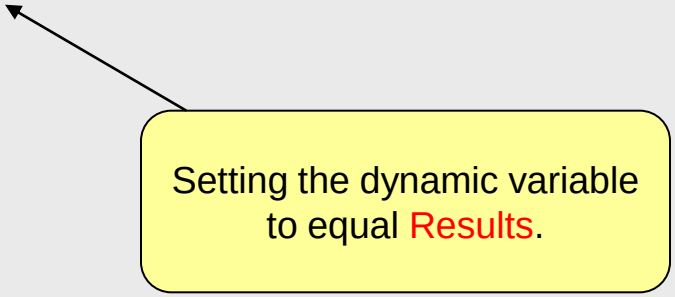
```
layout lattice / columns = 2;
   layout overlay / xaxisopts=(type=Discrete
   discreteOpts=(tickValueList=_ticklist_) LABEL = "Treatment")
                            yaxisopts = (LABEL = "Results");
   BoxPlot X=Treatment Y=Results / SortOrder=Internal primary=true
   LegendLabel="Results" NAME="VBOX";
   ;
   endlayout;

   layout overlay;
   ScatterPlot X=Treatment Y=Results / primary=true
   LegendLabel="Results" NAME="SCATTER";
   ;
        layout gridded / columns=2 border=TRUE autoalign=(BottomLeft
   TopLeft TopRight);
                  entry "Mean of Dataset:";     entry
   eval(strip(put(mean(VAR),8.2)));
                  entry "Std Dev of Dataset:";  entry
   eval(strip(put(stddev(VAR),8.2)));
        endlayout;
   endlayout;
endlayout;
```

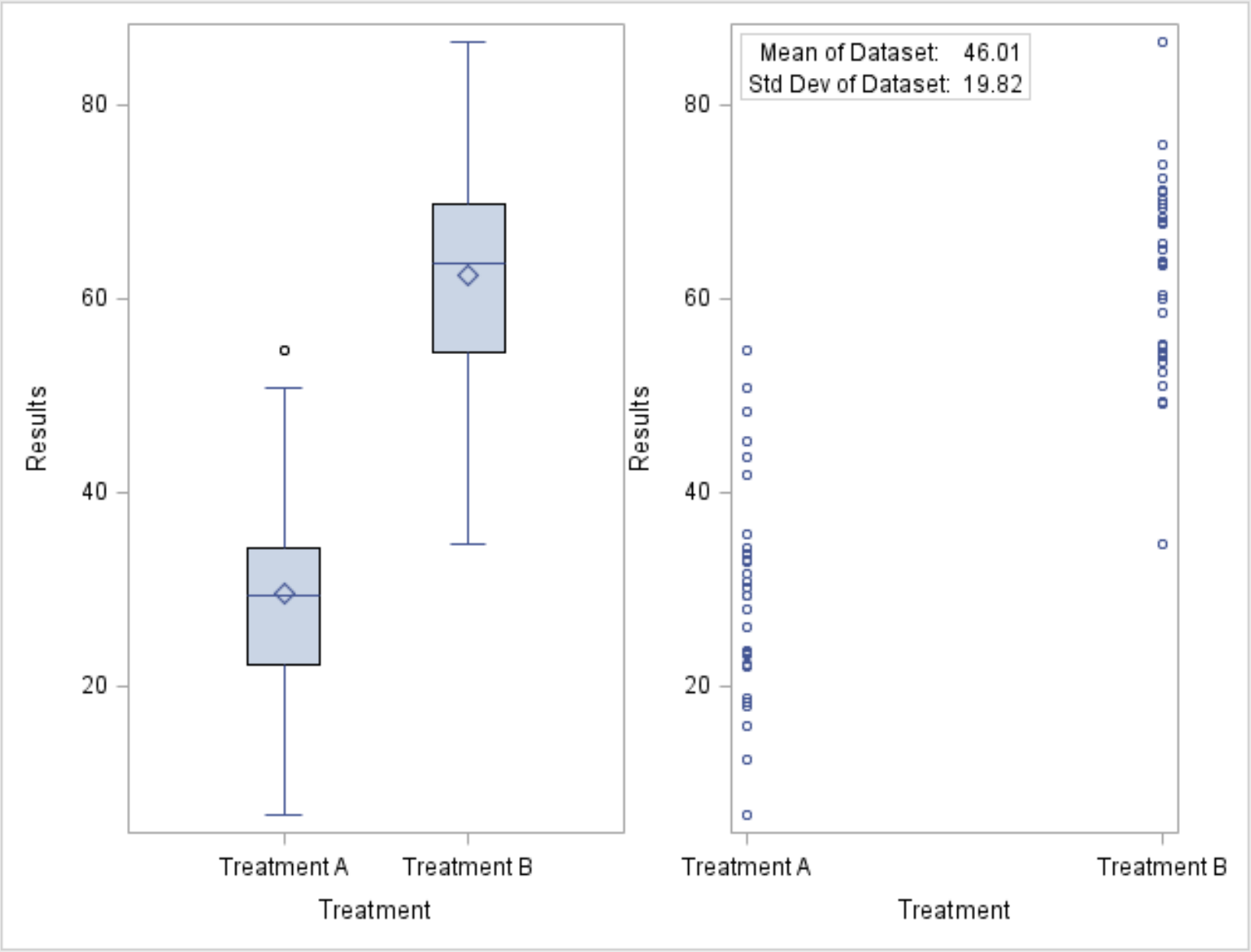
Using EVAL to calculate the mean.

Created a Dynamic Variable called VAR.
Using the statement: (after Proc Template)
Dynamic VAR;

Using Dynamic Variables
Rendering it

```
proc sgrender data=boxplots template=sgmypanel;
dynamic var = "Results";
run;
```




Mean of Dataset: 46.01
Std Dev of Dataset: 19.82
Results
80
60
40
20
Treatment A
Treatment B
Treatment
Results
80
60
40
20
Treatment A
Treatment B
Treatment

Using Blockquote to Display N, Mean and StdDev

- First need to calculate the statistics for each treatment using:

```
proc summary data=boxplots nway;
class treatment;
var results;
output out=class(drop=_type_ _freq_)
n = n mean=mean stddev = stddev;
run;
```

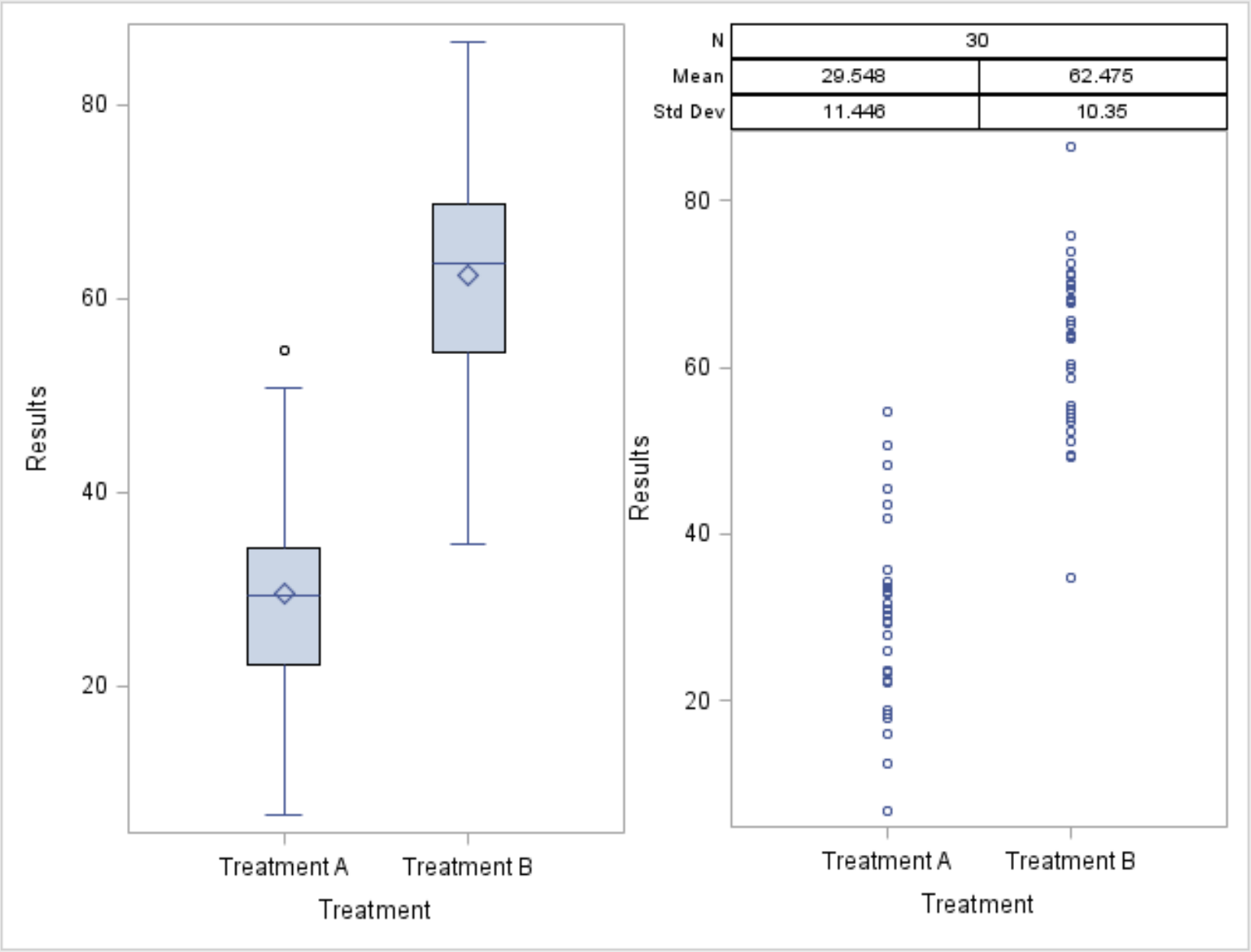
- Merging the statistics with the original dataset:

```
proc sql;
create table boxplots_merged as
select a.*, b.*
from boxplots as a inner join class as b
on a.treatment = b.treatment;
quit;
```

Using Blockquote to Display N, Mean and StdDev

```
layout lattice/ rowweights=(.04 .04 .04 .88); ;
;
            blockplot x=treatment block=n / label = "N"
valuehalign=center
        labelattrs=graphdatatext valueattrs=graphdatatext
display=(values label outline);
     blockplot x=treatment block=mean / label = "Mean"
valuehalign=center
        labelattrs=graphdatatext valueattrs=graphdatatext
display=(values label outline);
     blockplot x=treatment block=stddev / label = "Std
Dev"valuehalign=center
        labelattrs=graphdatatext valueattrs=graphdatatext
display=(values label outline);

     ScatterPlot X=Treatment Y=Results / primary=true
LegendLabel="Results" NAME="SCATTER";
;

endlayout;
```

Controlling the row heights.


80
60
40
20
Results
Treatment A
Treatment B
Treatment
N
30
Mean
29.548
62.475
Std Dev
11.446
10.35
80
60
40
20
Results
Treatment A
Treatment B
Treatment

GTL and SGRENDER Questions?

- Questions?

SAS 9.2 Shortcomings

1. Box plots can only have one category variable. In other words, if you want to produce a box plot by both time and treatment, you must concatenate the variables before the proc sgplot.
2. Also, the box plot axis must be discrete. If there is a numeric categorical variable, it will be treated as discrete by SAS.
3. Graph Template code is different syntax than SG plot code
4. Differences from older SAS Graph code may not be intuitive

References for Version 9.2

- **New Features and Enhancements in SAS 9.2 SAS/GRAPH® Software**

- **Creating Statistical Graphics in SAS 9.2: What Every Statistical User Should Know**
  Robert N. Rodriguez and Tonya E. Balan. SAS Institute Inc. Cary, North Carolina, USA.

- **A Programmer's Introduction to the Graphics Template Language**
  Jeff Cartier, SAS Institute Inc., Cary, NC.

References for Version 9.2

- **Using PROC SGPLOT for Quick High-Quality Graphs**
  Lora D. Delwiche, University of California, Davis, CA
  Susan J. Slaughter, Avocet Solutions, Davis, CA

- **Effective Graphics Made Simple Using SAS/GRAPH® SG Procedures**
  Dan Heath, SAS Institute Inc., Cary, NC

- **SAS/GRAPH® Procedures for Creating Statistical Graphics in Data Analysis**
  Dan Heath, SAS Institute Inc., Cary, NC

- **Secrets of the SG Procedures**
  Dan Heath, SAS Institute Inc., Cary, NC

References for Version 9.2

- **Introduction to the Graph Template Language**
  Sanjay Matange, SAS Institute, Cary, NC

- **Enhancements to SAS/GRAPH® Software in SAS 9.2**
  Himesh Patel, SAS Institute Inc., Cary, NC

Proc SGPLOT Syntax

```
PROC SGPLOT < option(s)>;
 BAND X= variable | Y= variable
 UPPER= numeric-value | numeric-variable LOWER= numeric-value | numeric-variable
 </option(s)>;
 DENSITY response-variable </option(s)>;
 DOT category-variable </option(s)>;
 ELLIPSE X= numeric-variable Y= numeric-variable </option(s)>;
 HBAR category-variable < /option(s) >
 HBOX response-variable </option(s)>;
 HISTOGRAM response-variable < /option(s)>
 HLINE category-variable < /option(s)>
 INSET "text-string-1" <... "text-string-n"> | (label-list);
 KEYLEGEND <"name-1" ... "name-n"> </option(s)>;
 LOESS X= numeric-variable Y= numeric-variable </option(s)>;
 NEEDLE X= variable Y= numeric-variable </option(s)>;
 PBSPLINE X= numeric-variable Y= numeric-variable </option(s)>;
 REFLINE value(s) </option(s)>;
 REG X= numeric-variable Y= numeric-variable </option(s)>;
 SCATTER X= variable Y= variable </option(s)>;
 SERIES X= variable Y= variable </option(s)>;
 STEP X= variable Y= variable </option(s)>;
 VBAR category-variable < /option(s)>
 VBOX response-variable </option(s)>;
 VLINE category-variable < /option(s)>
 XAXIS <option(s)>;
 X2AXIS <option(s)>;
 YAXIS <option(s)>;
 Y2AXIS <option(s)>;
```

HBAR, VBAR statement syntax

- HBAR *(or VBAR)* category-variable </ option(s)>;
- Bar options:
- ALPHA= numeric-value
- BARWIDTH= numeric-value
- FILL | NOFILL
- FILLATTRS= style-element < (fill-options) >| (fill-options)
- FREQ= numeric-variable
- LIMITS= BOTH | LOWER | UPPER
- LIMITSTAT= CLM | STDDEV | STDERR
- MISSING
- NUMSTD= n
- OUTLINE | NOOUTLINE
- RESPONSE= response-variable
- STAT= FREQ | MEAN | SUM
- URL= variable
- WEIGHT= numeric-variable
-
- Plot options:
- GROUP= variable
- LEGENDLABEL= "text-string"
- NAME= "text-string"
- TRANSPARENCY= numeric-value

HBOX, VBOX statement syntax

- HBOX *(or VBOX)* response-variable </ option(s)>;
- Box options:
- BOXWIDTH= numeric-value
- CATEGORY= category-variable
- DATALABEL <= variable>
- EXTREME
- FREQ= numeric-variable
- LABELFAR
- MISSING
- PERCENTILE= numeric-value
- SPREAD
-
- Plot options:
- LEGENDLABEL= "text-string"
- NAME= "text-string"
- TRANSPARENCY= numeric-value
- X2AXIS
- Y2AXIS

KEYLEGEND statement syntax

- KEYLEGEND <"name-1" ... "name-n"> </ option(s)>;
- ACROSS= n
- BORDER | NOBORDER
- DOWN= n
- LOCATION= OUTSIDE | INSIDE
- POSITION= position-value
- TITLE= "text-string"

INSET statement syntax

- INSET "text-string-1" <... "text-string-n"></ option(s)>;
- INSET (label-list) </ option(s)>;
- BORDER | NOBORDER
- LABELALIGN= LEFT | CENTER | RIGHT
- POSITION= position-value
- TEXTATTRS= style-element
- TITLE= "text-string"
- TITLEATTRS= style-element
- VALUEALIGN= LEFT | CENTER | RIGHT

REG statement syntax

- REG X= numeric-variable Y= numeric-variable </ option(s)>;
- REG options:
- ALPHA= numeric-value
- CLI <= "text-string">
- CLIATTRS= style-element
- CLM <= "text-string">
- CLMATTRS= style-element
- CLMTRANSPARENCY= value
- CURVELABEL <= "text-string">
- CURVELABELLOC= OUTSIDE | INSIDE
- CURVELABELPOS= MIN | MAX | START | END
- DATALABEL <= variable>
- DEGREE= n
- FREQ= numeric-variable
- LINEATTRS= style-element <(options)> | (options)
- MARKERATTRS= style-element <(options)> | (options)
- MAXPOINTS= n
- NOLEGCLI
- NOLEGCLM
- NOLEGFIT
- NOMARKERS
- WEIGHT= numeric-variable
-
- Plot options:
- GROUP= variable
- LEGENDLABEL= "text-string"
- NAME= "text-string"
- X2AXIS
- Y2AXIS

REFLINE statement syntax

- REFLINE value-1 <... value-n> </ option(s)>;
- REFLINE options:
- AXIS= X | X2 | Y | Y2
- LABEL <= ("text-string-1" ... "text-string-n")>
- LABELLOC= INSIDE | OUTSIDE
- LABELPOS= MIN | MAX
- LINEATTRS= style-element <(options)> | (options)

- Plot options:
- LEGENDLABEL= "text-string"
- NAME= "text-string"
- TRANSPARENCY= numeric-value

SCATTER statement syntax

- SCATTER X= numeric-variable Y= numeric-variable < / option(s)>;
- SCATTER options:
- DATALABEL <= variable>
- FREQ= numeric-variable
- MARKERATTRS= style-element <(options)> | (options)
- MARKERCHAR= variable
- MARKERCHARATTRS= style-element <(options)> | (options)
- URL= character-variable
- XERRORLOWER= numeric-variable
- XERRORUPPER= numeric-variable
- YERRORLOWER= numeric-variable
- YERRORUPPER= numeric-variable
-
- Plot options:
- GROUP= variable
- LEGENDLABEL= "text-string"
- NAME= "text-string"
- TRANSPARENCY= numeric-value
- X2AXIS
- Y2AXIS

SERIES statement syntax

- SERIES X= variable Y= variable < / option(s)>;
- SERIES options:
- BREAK
- CURVELABEL <= text-string>
- CURVELABELLOC= INSIDE | OUTSIDE
- CURVELABELPOS= MIN | MAX | START | END
- DATALABEL <= variable>
- LINEATTRS= style-element <(options)> | (options)
- MARKERATTRS= style-element <(options)> | (options)
- MARKERS
- URL= character-variable

- Plot options:
- GROUP= variable
- LEGENDLABEL= "text-string"
- NAME= "text-string"
- TRANSPARENCY= numeric-value
- X2AXIS
- Y2AXIS

XAXIS, YAXIS, XAXIS2, YAXIS2 statement syntax

- XAXIS *(or YAXIS etc)* option(s);
- options:
- DISCRETEORDER= DATA | FORMATTED | UNFORMATTED
- DISPLAY= ALL | NONE | (options)
- FITPOLICY= policy-value
- GRID
- INTEGER
- LABEL= "text-string "
- LOGBASE= 2 | 10 | e
- LOGSTYLE= LINEAR | LOGEXPAND | LOGEXPONENT
- MIN= numeric-value
- MINOR
- MAX= numeric-value
- NOTIMESPLIT
- REFTICKS
- TYPE= DISCRETE | LINEAR | LOG | TIME
- VALUES= (value-1 < ... value-n >)
- VALUESHINT