PhUSE 2008

Paper TU06

An Introduction to ODS Statistical Graphics

Ian Sedgwick, SAS, Marlow, Buckinghamshire UK

ABSTRACT

SAS has introduced a new and exciting way of generating plots and charts, referred to as ODS Statistical Graphics (or ODS Graphics for short), which complements the text output from statistical procedures. With the production release of SAS 9.2, over sixty statistical procedures have been modified to take advantage of this new functionality. In addition new SAS/GRAPH procedures allow you to create ODS graphics directly from SAS datasets. More complex modification can be made through the use of the underlying graphic template language (GTL). This tutorial shows you how to get started creating and customizing ODS Graphics within the SAS system.

INTRODUCTION

Different users will want to take advantage of the different functionality available. For statisticians it may be enough that they get the appropriate graphical output simply by running the appropriate SAS statistical procedure and making very minor changes to the code. In more complex situations, it might be that the data needed to produce the appropriate statistical plot has come from the results of a number of statistical procedures and data step code, in which case the use of the new family of statistical graphics procedures would be more appropriate. These new procedures start with the letters 'SG' to distinguish them from the 'traditional' SAS/GRAPH® procedures. In the cases where the requirements of the graphics output go beyond the capability of the 'SG' procedures there is the ability to create your own custom output using the graph template language (GTL).

STATISTICAL GRAPHICS FOR THE STATISTICIAN

Historically, in SAS, if you wanted to generate graphics output from a statistical procedure it would typically have involved two steps:

1. Run the statistical procedure and output results to a SAS dataset
2. Run a traditional SAS/GRAPH procedure to display graphics from the data

In the following statistical procedures this can now be reduced to one step.

Base	**SAS/STAT**			**SAS/ETS**
CORR	ANOVA	LIFEREG	PRINQUAL	ARIMA
FREQ	BOXPLOT	LIFETEST	PROBIT	AUTOREG
UNIVARIATE	CALIS	LOESS	QUANTREG	ENTROPY
	CLUSTER	LOGISTIC	REG	EXPAND
	CORRESP	MCMC	ROBUSTREG	MODEL
SAS/QC	FACTOR	MDS	RSREG	PANEL
ANOM	GAM	MI	SEQDESIGN	RISK
CAPABILITY	GENMOD	MIXED	SEQTEST	SIMILARITY
CUSUM	GLIMMIX	MULTTEST	SIM2D	SYSLIN
MACONTROL	GLM	NPAR1WAY	TCALIS	TIMESERIES
PARETO	GLMSELECT	PHREG	TRANSREG	UCM
RELIABILITY	KDE	PLS	TTEST	VARMAX
SHEWHART	KRIGE2D	PRINCOMP	VARIOGRAM	X12

All we need to do is add an ODS GRAPHICS statement to our code.

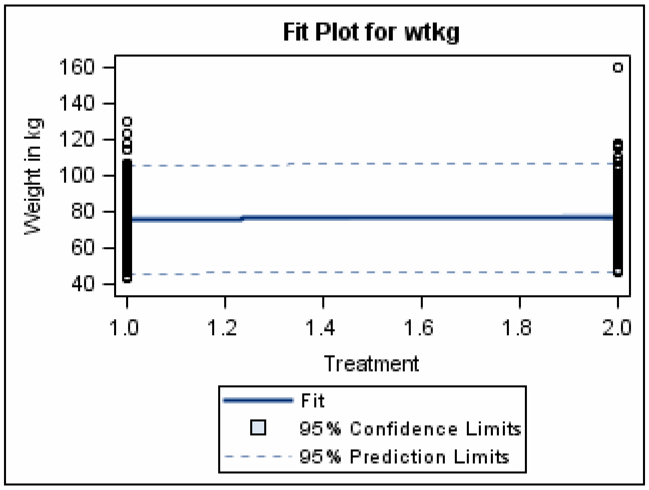
GETTING DEFAULT OUTPUT

We can use the ODS GRAPHICS statement to control if additional graphics are requested or not.
The example below illustrates this. Using ODS GRAPHICS ON requests that the GLM Procedure produce associated graphics and ODS GRAPHICS OFF turns this facility off. ODS graphics is turned off by default.

Simple ODS example

```
ods graphics on;
proc glm data=pharm.demog;
model wtkg=tmt;
run;
quit;
ods graphics off;
```

Only graphics output shown



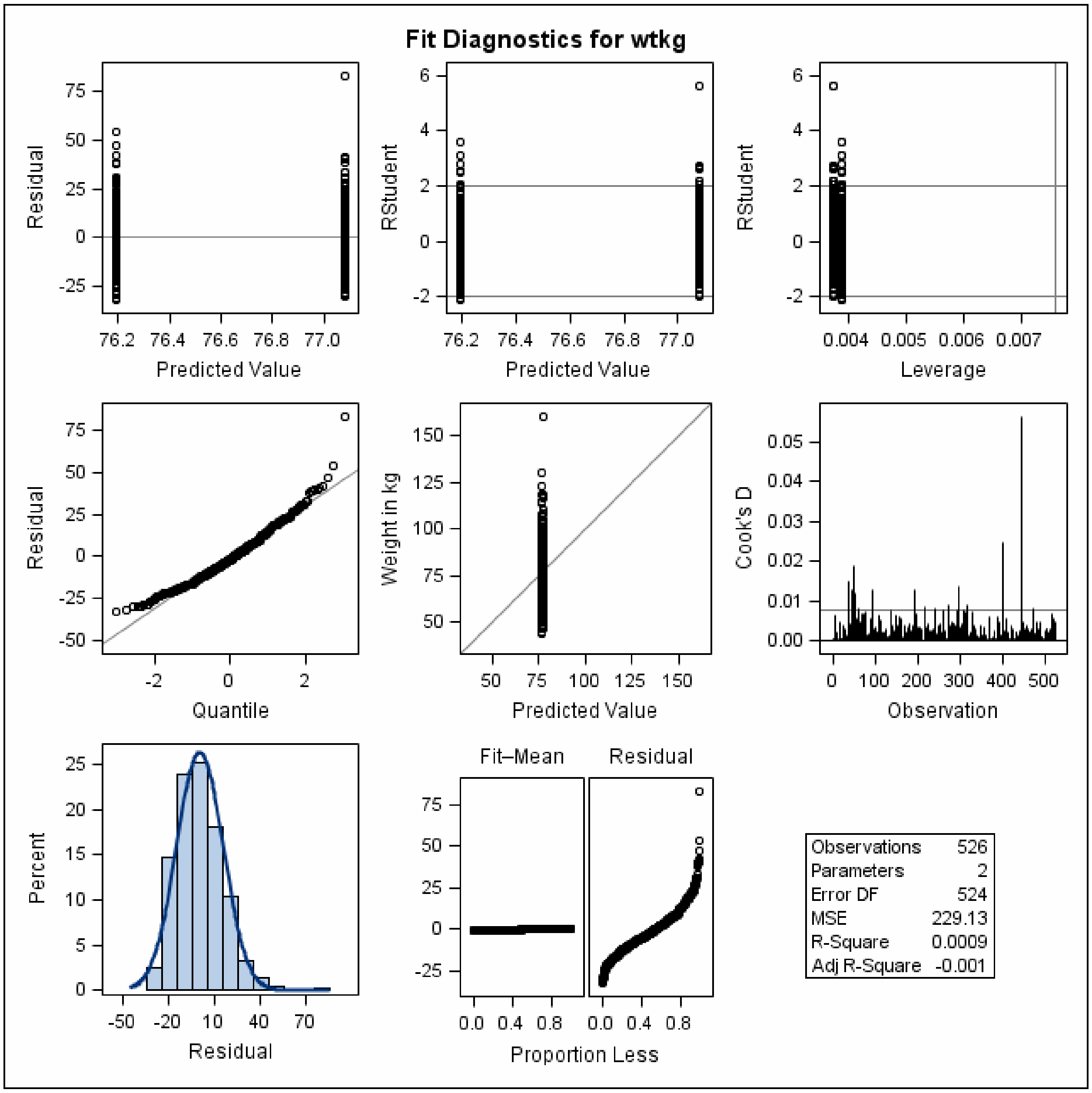
REQUESTING ADDITIONAL PLOTS
Sometimes the procedure is capable of producing additional plots or selecting specific ones from the default list. This functionality is provided in the statistical procedure by the use of a PLOTS option. The PLOTS option has the general form of PLOTS (global-plot-options) =plot-request (options) and the specific values will vary from procedure to procedure. Some examples for PROC GLM could be:

- PLOTS=NONE
  - No graphics are displayed
- PLOTS=(DIAGNOSTICS RESIDUALS)
  - Requests a panel of summary diagnostics for the fit, along with a scatter plot panel of the residuals for each continuous covariate.
- PLOTS(UNPACK)=(DIAGNOSTICS RESIDUALS)
  - Requests all the plots to be individual plots
- PLOTS=(DIAGNOSTICS RESIDUALS(UNPACK))
  - Requests only the residual plots to come out as individual plots
- PLOTS(ONLY)=MEANPLOT(CLBAND)
  - Requests only plot specified , which in this case produces a shaded confidence limits for a LSMEANS statement

Specific details of these options can be found under the appropriate procedure section of the SAS/STAT® 9.2 User's Guide.

Adding additional plots option

```
ods graphics on;
proc glm data=pharm.demog plots=(diagnostics);
model wtkg=tmt;
run;
quit;
ods graphics off;
```



Observations	526
Parameters	2
Error DF	524
MSE	229.13
R-Square	0.0009
Adj R-Square	-0.001

STATISTICAL GRAPHICS FOR THE SAS PROGRAMMER

The following three ‘SG’ procedures work directly against SAS datasets to produce a range of different statistical plots. They can be classified in the following ways:

- SGPLOT
  - This procedure is used only to produce a single-cell output. Within the output cell can be multiple overlaid plots which all share common axes.
- SGPANEL
  - This procedure typically generates multi-cell output where the contents of the individual cells represent graphs based on the value of a selected classification variable. One or more classification variables can be used.
- SGSCATTER
  - This procedure again produces multi-cell output where the graphs in each cell can be different or share common axes.

The ODS GRAPHICS statement does not need to be issued to run these procedures but you may use it to provide additional information such as size of output.

THE SGPLOT PROCEDURE

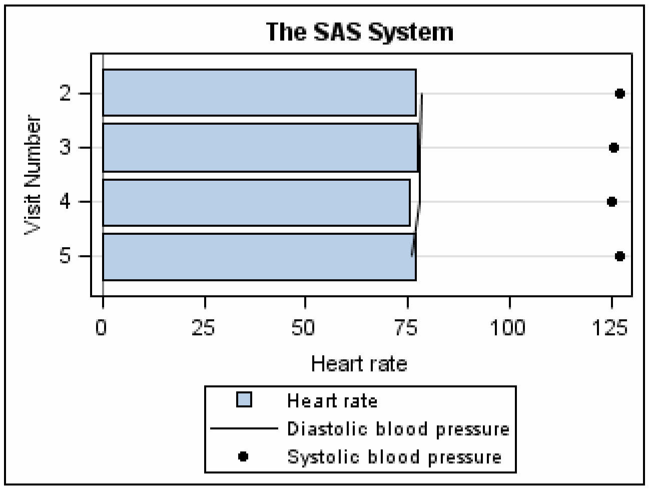
There are four types of plots that you can create:

- Categorization plots
  - bar charts, line charts, dot plots
- Distribution plots
  - histograms, normal & kernel density plots, box plots
- Basic plots
  - scatter, series, step, needle, band
- Fit and confidence plots
  - Regression, loess, penalized B-spline curves, ellipses

Each plot is generated by a separate statement. Overlaid plots can be produced by using multiple statements in one procedure, however the statements have to relate to the same type of plot. The order of the plot statements determines the order in which the individual plots are drawn.

<u>Example - Categorization plots</u>

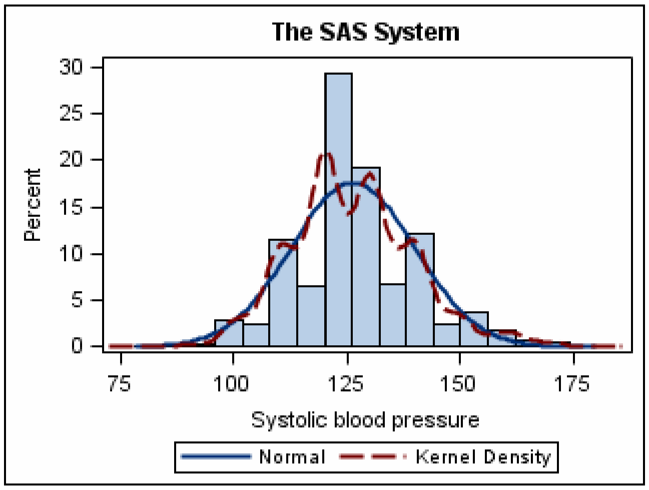
```
proc sgplot data=pharm.visits ;
hbar vis/response=hrtrte stat=mean;
hline vis/response=bpd stat=mean;
dot vis /response=bps stat=mean;
run;
```



In addition to hbar and hline there are some corresponding vbar and vline statements which are used to produce vertical bars and associated overlayed lines. If plots are overlaid and share axes they must be of the same orientation i.e. Hbar and vbar cannot be used together.

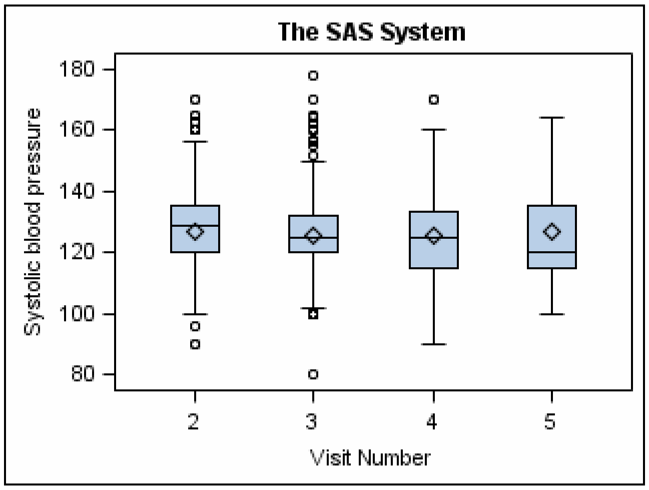
<u>Example – Distribution plots</u>

```
proc sgplot data=pharm.visits;
histogram bps;
density bps;
density bps / type=kernel;
run;
```



Example – Distribution plots

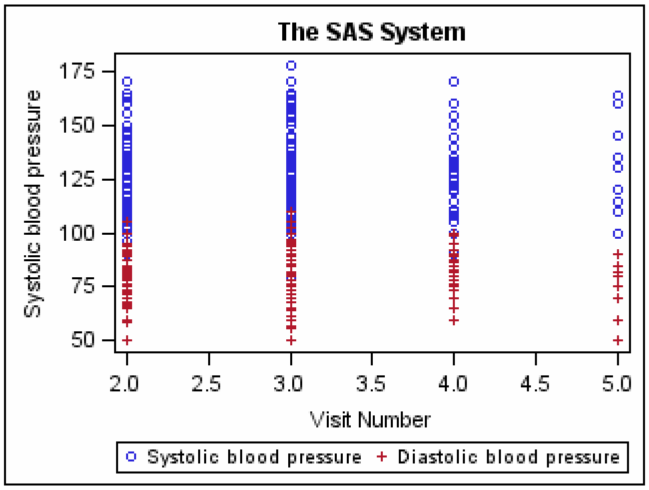
```
   proc sgplot data=pharm.visits;
   vbox bps /category=vis;
   run;
```



The hbox and vbox plots cannot be overlaid with other types of plot.

Example – Basic plots

```
   proc sgplot data=pharm.visits;
   scatter y=bps x=vis;
   scatter y=bpd x=vis;
   run;
```



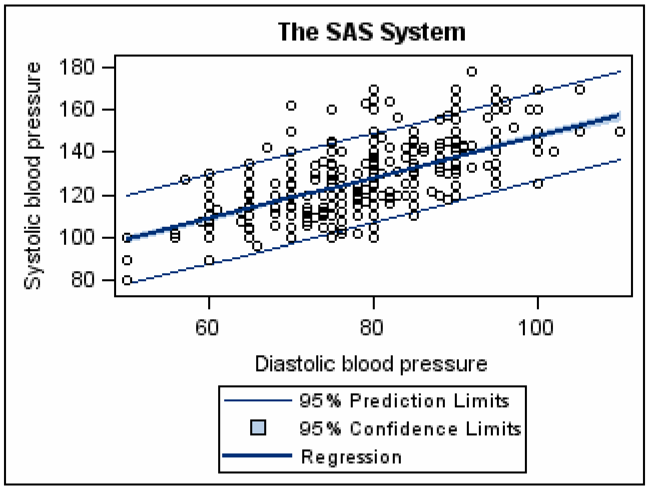
Basic plots are used to plot two variables against each other. In addition to the scatter plot which plots individual points, there are other statements which are used for basic interpolation, i.e.:

- SERIES – joins points directly with straight lines
- STEP – joins points with a stepped lines
- NEEDLE – joins the point to a given axis

In addition the BAND statement can be used to create a shaded region. This is useful for showing custom defined confidence intervals.

Example – Fit & confidence plots

```
   proc sgplot data=pharm.visits;
   reg x=bpd y=bps /clm cli;
   run;
```



The reg statement is used to produce regression lines. Other statements apply different fits to the data:

- LOESS – Loess fit
- PBSPLINE – Penalized B-spline curves
- ELLIPSE – Ellipse curves

THE SGPANEL PROCEDURE

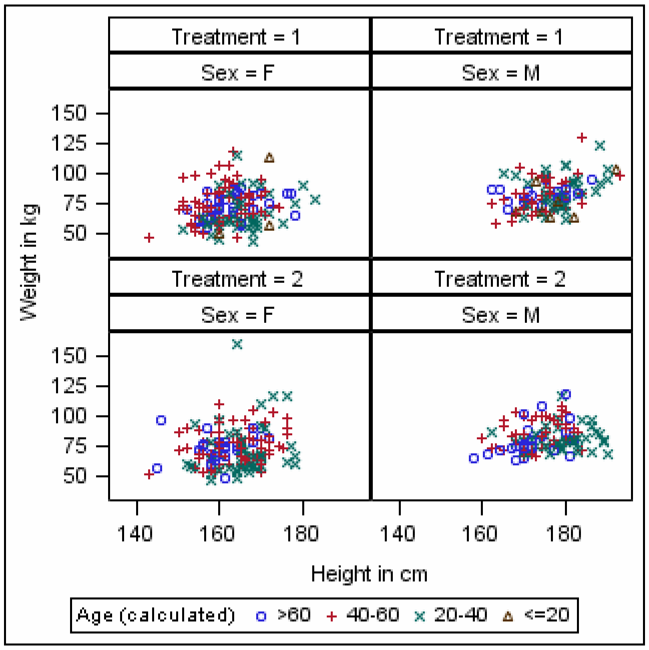
The category variable whose value is used in each plot is defined on the PANELBY statement. There are two styles of output that can be created:

- PANEL – this has no limit to the number of categorical variables used and displays the values of each variable in each cell-plot
- LATTICE – this restricts the use of categorical variables to 2 and uses one in the row dimension and the other in the column dimension

When this procedure is used a legend is automatically produced for each category variable. The COLAXIS and ROWAXIS statements can be used to control the common row and column axes attributes. Any plot statement (except ELLIPSE) used in SGPLOT can be used with SGPANEL.

Example – Panel Output

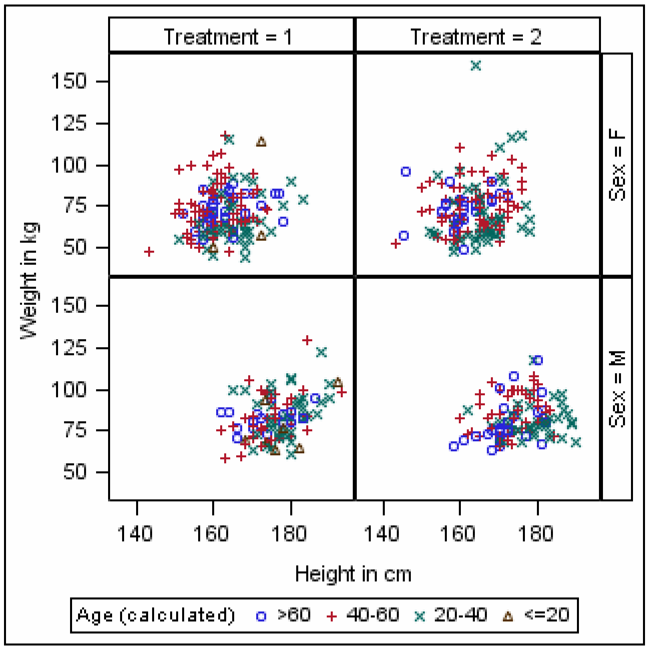
```
proc format;
value agefmt
    low-20='<=20'
    20-40='20-40'
    40-60='40-60'
    60-high='>60';
run;
proc sgpanel data=pharm.demog;
panelby tmt sex ;
scatter x=htcm y=wtkg / group=age;
format age agefmt.;
run;
```



Specifying tmt and sex on the PANELBY statement has generated a separate plot for each value combination of those variables existing in the data. Individual points have different attributes based on the group variable age. Ages are grouped based on the format statement and the custom format agefmt.

<u>Example – Lattice Output</u>

```
proc sgpanel data=pharm.demog;
panelby tmt sex / layout=lattice ;
scatter x=htcm y=wtkg / group=age;
format age agefmt.;
run;
```



Adding the layout=lattice to the panalby statement here rearranges the panel labels and saves some space.

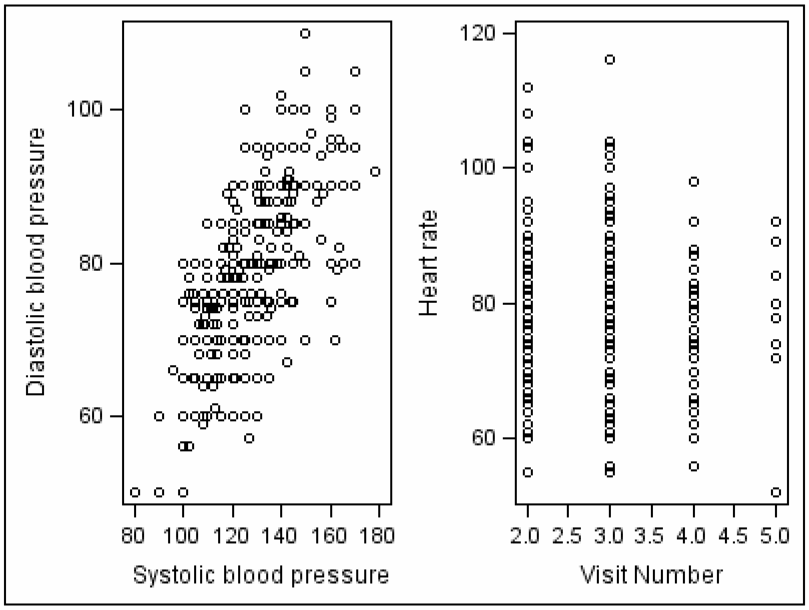
THE SGSCATTER PROCEDURE

This procedure has three statements, each of which is designed to create a specific paneled graph:

- PLOT
  - Each graph cell has an independent axis
- COMPARE
  - Axes are shared where appropriate
- MATRIX
  - Each variable is graphed against each other

<u>Example – Plot example</u>

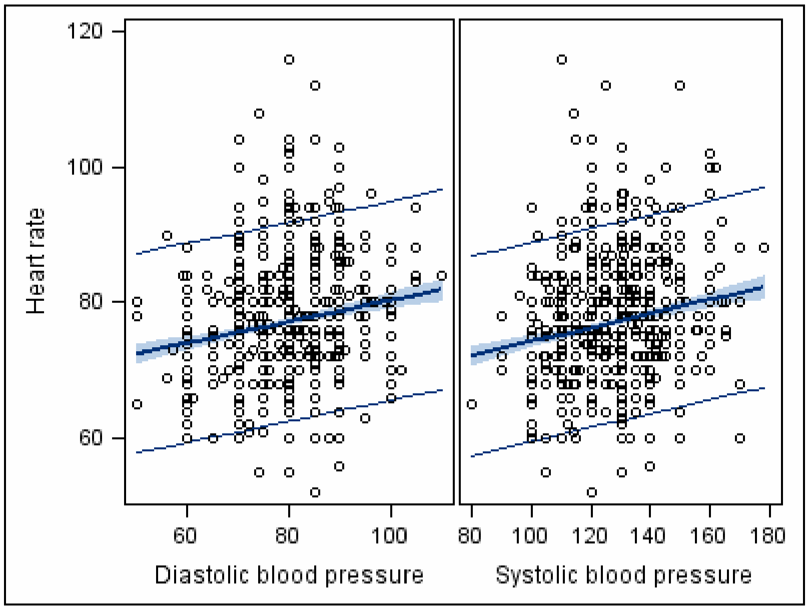
```
proc sgscatter data=pharm.visits;
plot bpd*bps hrtrte*vis ;
run;
```



With the plot statement each axis is created independently.

Example – Compare example

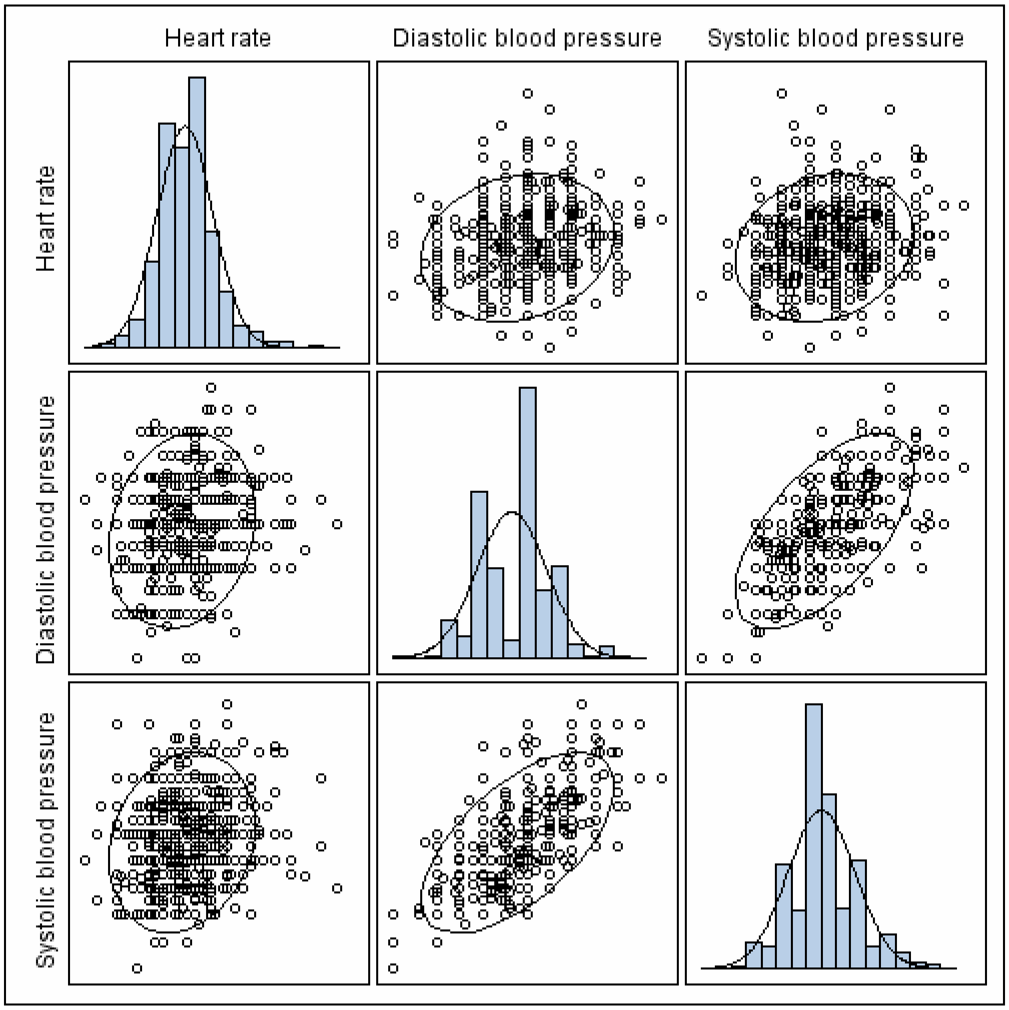
```
proc sgscatter data=pharm.visits;
compare y=hrtrte x=(bpd bps)/ reg=(cli clm) ;
run;
```



In this case the compare statement allows the yaxis to be shared and interpolation is provided through the options.

Example – Matrix example

```
proc sgscatter data=pharm.visits;
matrix hrtrte bpd bps / diagonal=(histogram normal) ellipse=(type=predicted);
run;
```



SIMPLE CUSTOMISATION

CUSTOMISATION OF OUTPUT

Many aspects of statistical graphics can be changed eg. Colour, size, format , name of output etc.
Control exists through ODS statements, ODS GRAPHICS statement options and ODS Graphics procedure options.

ODS STATEMENTS

These statements, as with earlier versions of SAS, control the destination (how the output is marked up) and style (the colour scheme of the output). It is also possible to control the image resolution with the IMAGE_DPI=. A default output device is associated with each destination and this can be changed on the ODS GRAPHICS statement.

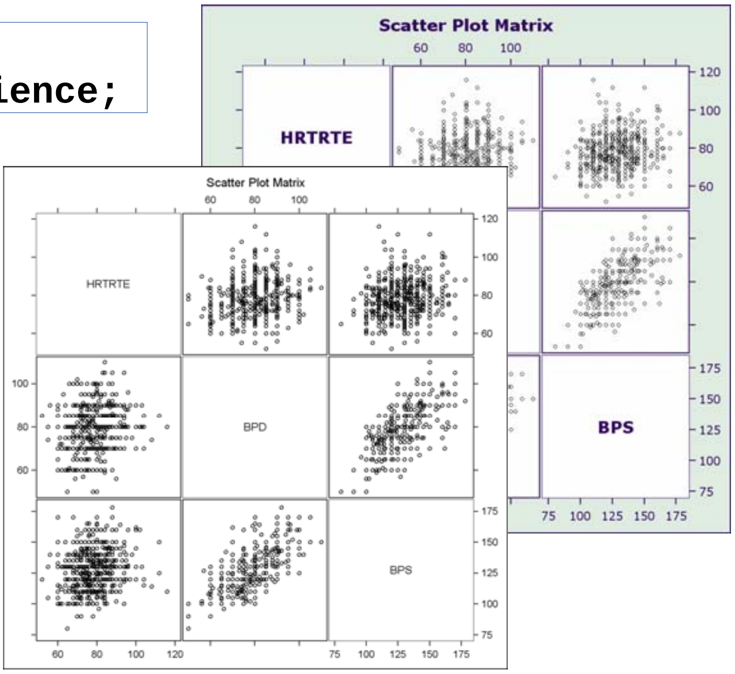
DESTINATION	DEFAULT FILE TYPE	ALTERNATIVE
HTML	PNG	GIF, JPEG
LATEX	POSTSCRIPT	EPSI, GIF, JPEG, PDF, PNG
LISTING	PNG	GIF, BMP, DIB, EMF, EPSI, JFIF, JPEG, PBM, PS, TIFF, WMF
PDF,PS,PCL	EMBEDDED	
RTF	EMBEDDED	
DOCUMENT	N/A	

Each destination is associated with a particular SAS style and can be overriden using style= on the ODS statement. Styles are made up of style elements (these are associated with areas of the output eg. Fill color, marker symbols etc. Each style element is made up of attributes (eg. Font= , Markersize= etc.)
The default styles are stored in the SAS file SASHELP.TEMPLAT and you have the ability to create your own styles.

<u>Example – using destinations & styles</u>

```
ods pdf
 style=science;
```

```
ods rtf
 style=journal;
```



ODS GRAPHICS STATEMENT OPTIONS

These options are placed on the ODS GRAPHICS statement and typically control the following:

- File type & Name
    - IMAGEFMT= , IMAGENAME=, INDEX=
- Size
    - HEIGHT=, WIDTH=
- Display characteristics
    - ANTIALIAS=, ANTIALIASMAX=, BORDER=, LABELMAX=, MAXLEGENDAREA=, SCALE=
- Tooltips
    - IMAGEMAP=, TIPMAX=

Options can be reset back at any time usin the RESET= option.

<u>Example – ODS graphics options</u>

```
ods graphics on /
  imagename='myplot' imagefmt=jpeg height=240px width=320px;
```

This would create a file called MYPLOT.JPEG which had a size of 240x320.

ODS GRAPHICS PROCEDURE OPTIONS

Using these options you can affect how an individual plot looks. There are many options you can change and some of the common ones that affect lines, markers and fills are :

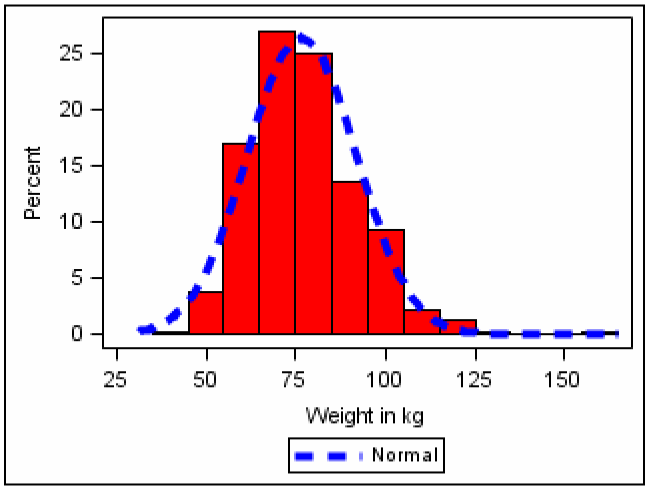
- LINEATTRS=(color= pattern= thickness=)
- MARKERATTRS=(color= size= symbol=)

- FILLATTRS=(color=)

Not all plots have every attribute.

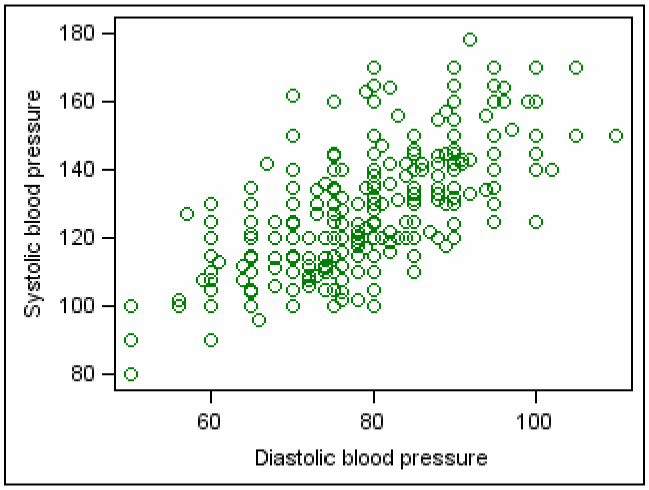
Example - attributes

```
proc sgplot data=pharm.demog;
histogram wtkg /
fillattrs=(color=red);
density wtkg /
lineattrs=(color=blue pattern=dash
thickness=5);
run;
```



Example - attributes

```
proc sgplot data=pharm.visits;
scatter y=bps x=bpd /
markerattrs=(size=10 color='green'
symbol='circle') ;
run;
```



Another common attribute useful for overlayed graphs is CYCLEATTRS which gives a different set of attributes to each plot.

ADVANCED USAGE

There are times when you need a specific type of output to be produced automatically by the statistician or used by SAS programmers in a number of ways. To do this you need to understand the underlying language that is used by the SG procedures and statistical graphics. This is the graph template language (GTL).

It can be used in two ways:

- Take an existing statistical graphics template and modify it for use with a statistical procedure
- Create a standalone template for use against data and render it using Proc SGRENDER

MODIFY AN EXISTING TEMPLATE

The process of modifying an existing template is the same process as modifying a table template in previous releases of SAS only we are dealing with a STATGRAPH template and not a TABLE template. The steps to go through re as follows:

1. Identify STATGRAPH template that is used (looking at the properties of the object or running ODS TRACE ON)
2. Extract existing template code
3. Make modifications to code
4. Re-run the new template code making your own version of the template
5. Ensure the correct ODS PATH is in place

6. Statistical procedure now runs with modified code

<u>Example – modifying an existing template</u>

If we go back to the first example

```
ods graphics on;
proc glm data=pharm.demog;
model wtkg=tmt;
run;
quit;
```

The template used here is Stat.GLM.Graphics.Fit, which has the following graphics template language (GTL) code:

```
proc template;
   link Stat.GLM.Graphics.Fit to Common.Zreg.Graphics.Fit;
   define statgraph Common.Zreg.Graphics.Fit;
      dynamic _PREDLABEL _CONFLABEL _DEPLABEL _DEPNAME _INDLABEL
         _SHORTINDLABEL _OBSNUM _Y _XVAR _UCL _LCL _UCLM _LCLM _TITLE _Y_OBS
         _XVAR_OBS _PREDICTED _UCL_OBS _LCL_OBS _UCLM_OBS _LCLM_OBS _FREQ
         _WEIGHT _ID1 _ID2 _ID3 _ID4 _ID5;
      BeginGraph;
         entrytitle _TITLE " for " _DEPNAME;
         layout overlay / xaxisopts=(label=_INDLABEL shortlabel=_SHORTINDLABEL
            ) yaxisopts=(label=_DEPLABEL shortlabel=_DEPNAME);
            bandplot limitupper=_UCLM limitlower=_LCLM x=_XVAR / connectorder=
               axis name="Confidence" LegendLabel=_CONFLABEL outlineattrs=
               GRAPHCONFIDENCE fillattrs=GRAPHCONFIDENCE datatransparency=0.5;
            if (EXISTS(_WEIGHT))
               scatterplot x=_XVAR_OBS y=_PREDICTED / markerattrs=(size=0)
               yerrorlower=_LCL_OBS yerrorupper=_UCL_OBS rolename=(_tip1=
               _OBSNUM _tip2=_LCL_OBS _tip3=_UCL_OBS _tip4=_LCLM_OBS _tip5=
               _UCLM_OBS _tip6=_WEIGHT _id1=_ID1 _id2=_ID2 _id3=_ID3 _id4=_ID4
               _id5=_ID5) tip=(y x _tip1 _tip2 _tip3 _tip4 _tip5 _tip6 _id1
               _id2 _id3 _id4 _id5) datatransparency=0.5 name="Prediction"
               LegendLabel=_PREDLABEL;
            else
               bandplot limitupper=_UCL limitlower=_LCL x=_XVAR / connectorder
               =axis name="Prediction" LegendLabel=_PREDLABEL outlineattrs=
               GRAPHPREDICTIONLIMITS fillattrs=GRAPHPREDICTIONLIMITS
               datatransparency=0.5 display=(outline);
            endif;
            seriesplot x=_XVAR y=_Y / connectorder=xaxis name="Fit"
               LegendLabel="Fit" lineattrs=GRAPHFIT;
            scatterplot x=_XVAR_OBS y=_Y_OBS / primary=true rolename=(_tip1=
               _OBSNUM _tip2=_LCL_OBS _tip3=_UCL_OBS _tip4=_LCLM_OBS _tip5=
               _UCLM_OBS _id1=_ID1 _id2=_ID2 _id3=_ID3 _id4=_ID4 _id5=_ID5)
               tip=(y x _tip1 _tip2 _tip3 _tip4 _tip5 _id1 _id2 _id3 _id4 _id5
               );
            if (
               EXISTS(_LCLM) OR (EXISTS(_WEIGHT) AND EXISTS(_LCL_OBS)) OR (NOT
               EXISTS(_WEIGHT) AND EXISTS(_LCL)))
               discretelegend "Fit" "Confidence" "Prediction";
            endif;
         endlayout;
      EndGraph;
   end;
run;
```

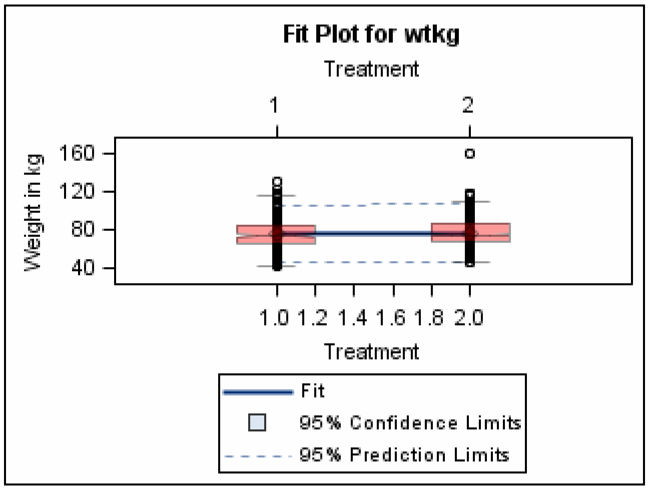
Without going into too many details here, the following code can be added to generate a boxplot to overlay the existing plots.

```
boxplot x=_XVAR_OBS y=_Y_OBS /
```

```
    xaxis=x2 datatransparency=0.6 fillattrs=(color=red) display=all;
```

When this code is run a version of Stat.GLM.Graphics.Fit will be written back to the first updateable path (which is typically SASUSER.TEMPLAT).

Now ruining the PROC GLM code, as before, will create the new graph automatically.



Users can revert back to the default graph by making SASHELP.TMPLMST the default search path.

CREATE A STAND ALONE TEMPLATE

Templates are made from a number of standard components

- Graph Block
  - Begingraph and endgraph are the required outermost container statements for any graph template
- Layout Statements
  - These statements create the grid of cells into which graphs are placed
- Plot Statements
  - These determine the type of plot to be created. There are options here which are not available in the ‘SG’ plots such as a FRINGEPLOT and 3D plots.
- Axis options
  - Four axes can be controlled (top, bottom, left and right) for 2D plots and three axes for 3D plots
- Legend statements
  - Legends exist and can be customised for displaying both continuous and discrete data
- Text statements
  - Statements are used for adding text into the title ,footnote and body of a specific graph.

Templates are rendered with the data using PROC SGRENDER
Basic template code, with the required statements would look like the following:

```
PROC TEMPLATE;
  DEFINE STATGRAPH statgraph-template;
      BEGINGRAPH;
          LAYOUT OVERLAY;
             <at least one plot type >;
          ENDLAYOUT;
      ENDGRAPH;
  END;
RUN;
```

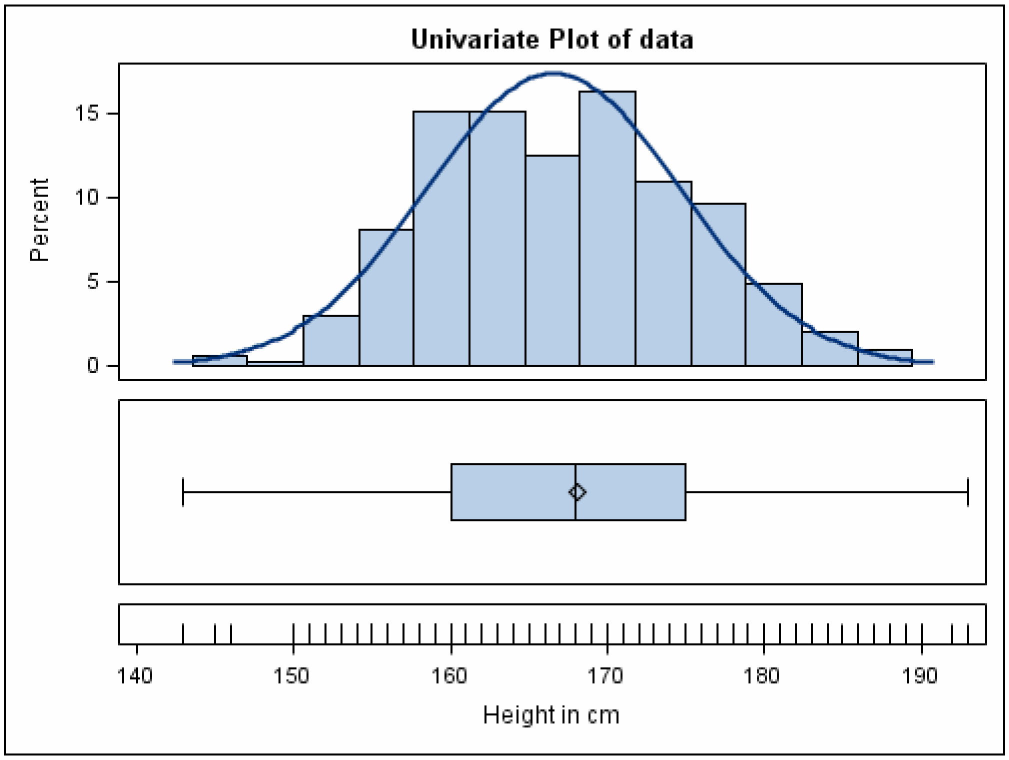
Once the template has been created the following code can be used to run the data against that template to create the graph:

```
PROC SGRENDER TEMPLATE=statgraph-template
  <DATA=input-data-set>
  <OBJECT=object-name>
  <OBJECTLABEL="text-string">;
  <DYNAMIC variable-assignment(s);>
RUN;
```

<u>Example – customised template</u>

```
/* Generate template */
proc template;
/* create a statgraph template called statgraph.custom */
  define statgraph statgraph.custom;
/* create a dynamic variable called myvar to take the value of an actual variable */
    dynamic myvar;
/* start the graph block section */
    begingraph;
/* create a title for the whole graph region */
      entrytitle "Univariate Plot of data";
/* create a region made of 3 cells in the row proportion 0.5 0.3 0.2 */
      layout lattice / rowgutter=5
                       rowweights=(0.5 0.3 0.2);
/* start of cell 1 */
        layout overlay / xaxisopts=(display=(line))
                         yaxisopts=(offsetmin=0.05);
/* content of cell 1 */
          histogram myvar;
          densityplot myvar /normal();
/* end of cell 1 */
        endlayout;
/* start of cell 2 */
        layout overlay / xaxisopts=(display=(line));
/* content of cell 2 */
          boxplot y=myvar / orient=horizontal;
/* end of cell 2 */
        endlayout;
/* start of cell 3 */
        layout overlay / xaxisopts=(display=all);
/* content of cell 3 */
          fringeplot myvar;
/* end of cell 3 */
        endlayout;
/* end of the lattice layout */
      endlayout;
/* end of the graph block section */
    endgraph;
/* end the define statement */
  end;
run;

/* Run SGRENDER */
proc sgrender data=pharm.demog
 template=statgraph.custom;
 dynamic myvar='htcm';
run;
```



CONCLUSION

ODS GRAPHICS provides relevant default graphics output for many statistical procedures. The 'SG' procedures allow access to many statistical graphics features with simple code and are not explicitly tied to a given statistical procedure. The graph template language (GTL) allows for the ability to modify existing templates or to create new ones.

RECOMMENDED READING

SAS® 9.2 Output Delivery System: User's Guide
SAS/STAT® 9.2 User's Guide (Chapter 21) Statistical Graphics using ODS
SAS/GRAPH® 9.2 Statistical Graphics Procedure Guide
SAS/GRAPH® 9.2 Graph Template Language Reference
SAS/GRAPH® 9.2 ODS Graphics Editor User's Guide

http://support.sas.com/resources/papers/ - for technical papers at SAS conferences

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ian Sedgwick
SAS Institute
Wittington House, Henley Road
Medmenham, Marlow
Bucks, SL7 2EB

Work Phone: +44(0)1628 404224
Email: Ian.sedgwick@suk.sas.com
Web: www.sas.com/uk