Power User's Guide to SAS Graph Templates

Visualising your data using SAS software

Philip R Holland

Power User's Guide to SAS® Graph Templates

Copyright © 2013, Philip R Holland, Royston, Herts, UK

All rights reserved.

First printing, February 2013.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Expert Reviews

A hugely informative book, written in Phil's usual, facts-first style. As a veteran of SAS/GRAPH, I've found Phil's book has opened my eyes to a whole world I did not know existed; these templates provide such control over SG graphics. I'm recommending Phil's book to all those who have an interest in SAS graphics.

Andrew Ratcliffe, Freelance SAS Consultant
web: RTSL.eu, blog: NoteColon.info

Phil Holland's Power User's Guide to SAS Graph Templates is a timely and helpful virtuoso stand-up tutorial presentation delivered as hard-copy. ODS Graphics and the SG (Statistical Graphics) Procedures are now packaged at no additional charge with Base SAS. This newer technology (than SAS/GRAPH) delivers a lot more types of graphs, but without all of the possibly desired customization options built-in. Graph Templates and their supporting extra language fill some of those gaps. The important added power from templates is the ability to build graph generation tools for reuse without recoding. This long-time graphics user shares his knowledge, makes the new SAS graphics look easy with an excellent walk through the ODS Graphics Designer, yet introduces the reader to the harder bits, and also considerately provides a clear explanation that spans both versions of SAS software that are widely in use at this time.

LeRoy Bessler PhD
Bessler Consulting and Research

Whether you've only just dipped your toe in the murky waters of SAS GTL, or you're an accomplished swimmer, there's something here for everyone. The book is easy to follow, and perfect to dip in and out of when you're stuck with a particular coding problem. With plenty of worked examples, and code for you to try yourself, you'll be creating all manner of graphs within a short space of time. A great desktop reference.

Neville Cope
Amgen

Holland has taken a topic that is complex and little understood, and has explained it clearly and concisely. This book is nominally about the Graph Template Language, but in the process of explaining what you should (and should not) do with GTL, Holland also explains how GTL interacts with the ODS Graphics Designer and SG procedures. This book is short. I read it in a couple hours. You won't find long tables of options here, just a concise overview of the various personas of ODS Graphics. The title is "Power User's Guide", not so much because it's ***for*** power users, as because it ***makes*** you a power user. With this book, Holland has done a great service to anyone producing graphics using SAS.

Susan J. Slaughter, author of *The Little SAS Book*
Avocet Solutions

Anyone who has seen one of Phil's presentations at a conference realizes what a wonderful teacher he is. His ability to organize material is first rate and coupled with his many examples and screens captures allows him to lead a reader through the material in a time efficient manner. I especially liked the fact that he starts with examples, so that people new to the topic build exposure, and then, when the reader is no longer a "newbie" presents technical details.

Russ Lavery
contractor

Table of Contents

Preface

When selecting a technical book for myself I tend to choose one where there are lots of examples and sample code snippets that I can use and adapt for my own development projects. I wanted to write a book that I could use for reference myself, so I have tried to make sure there are code snippets wherever possible.

This book was originally destined to be part of "PROC TEMPLATE Made Easy" with Kevin Smith, but it is now being published separately. The aim of both books is to give all SAS programmers the opportunity to make good use of a misunderstood and, to some, a frightening procedure. I hope that both books bring PROC TEMPLATE and the various templates it can create to more people.

I have spent the vast majority of my SAS programming career drawing graphs, first on pen plotters and, more recently, for web pages and books. SAS/GRAPH® is now a vast and complicated SAS component, requiring delicate configuration that changes from platform to platform, and even graph to graph. To find a way of drawing clear graphs with SAS software that is consistent, reusable and, mostly, simple to learn was a joy to me. I hope by the end of this book that you will agree with me.

Acknowledgements

- My wife, Angela, and our three daughters, Sarah, Rachel and Jessica, for their tolerance and encouragement.
- LeRoy Bessler, Andy Ratcliffe, Susan Slaughter and Russ Lavery, for their thorough proof-reading of everything SAS-related in this book.
- PhUSE Conference Committee members, for giving me the chance to present my ODS Graphics ideas at PhUSE Conferences.

Introduction to Graph Templates

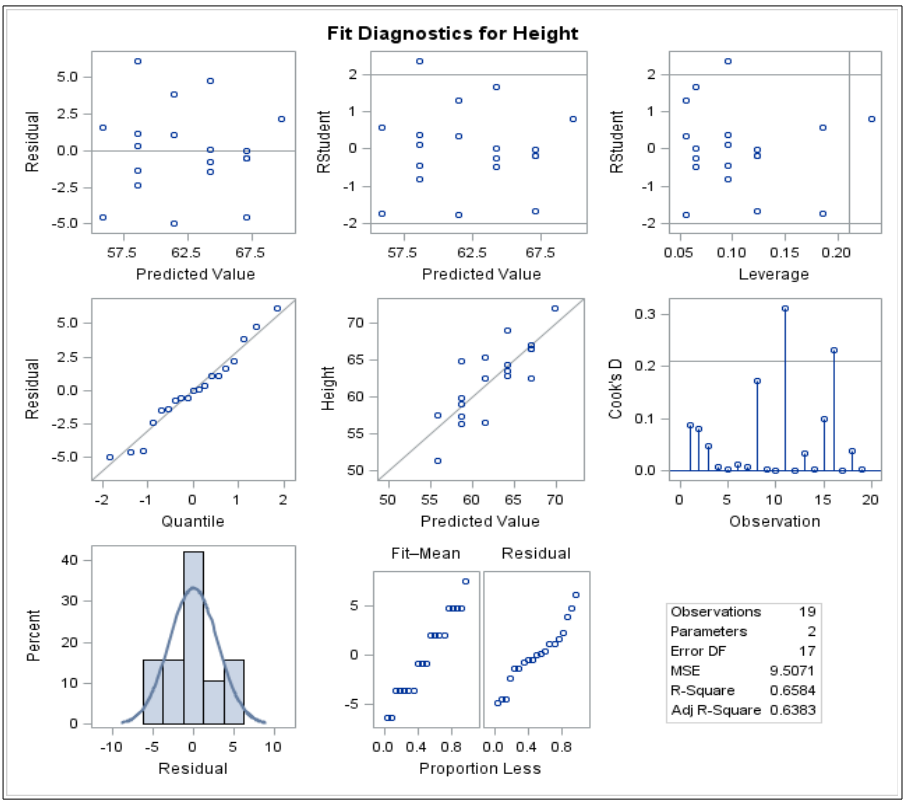
What are Graph Templates?

You will see a number of new terms mentioned throughout this book. STATGRAPH is the name of the template definition used for a Graph Template in PROC TEMPLATE, in the same way as STYLE and TABLE are the definitions used for Style and Table Templates. Graph Template Language (GTL) is the subset of the PROC TEMPLATE statements designed specifically with graphics in mind, and is similar to that used in other template subsets, but does have some unique features.

Graph Templates are very different to traditional SAS/GRAPH® programs, as they need to be displayed by rendering, i.e. creating an image from data, with another SAS program step, e.g. Data Step or PROC SGRENDER. At this point it is probably best to think of Graph Templates as similar to macros with parameters.

You may have used a Graph Template before now and not realised it, as many of the SAS/STAT® procedures can generate graphs by just adding ODS GRAPHICS statements around them. PROC REG can generate a Diagnostic Panel of graphs, amongst other output, to display the results of the regression model with very little additional SAS code:

```
ODS GRAPHICS ON;

PROC REG DATA = sashelp.class;
  MODEL height = age;
RUN;

ODS GRAPHICS OFF;
```



Note that templates written in SAS 9.1.3 are not compatible with SAS 9.2 and beyond.

Coming up...

The rest of this book will take you on a journey from easy Graph Template creation using the ODS Graphics Designer right through to some advanced techniques for updating generated templates to create flexible and reusable templates towards the end of the book.

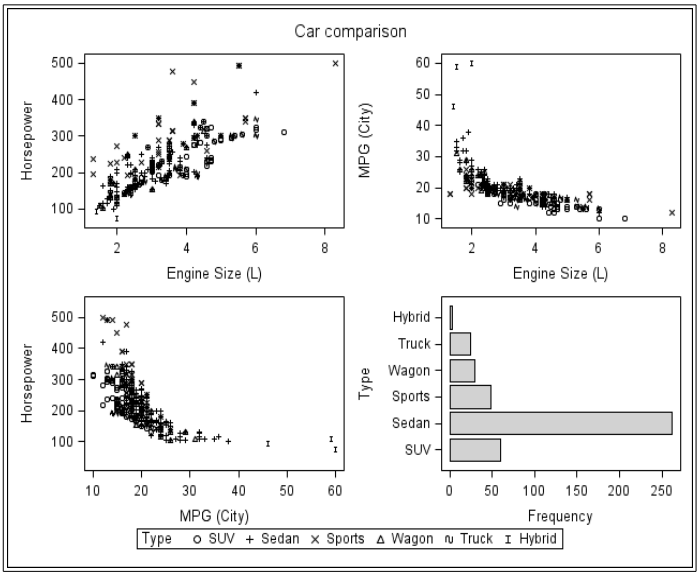
I have over the years standardised the way I write SAS code, so that I can tell at a glance whether the text shows standard SAS text, or I have supplied it. The majority of the code samples use the following conventions. Where the code appears not to follow these conventions, these programs were generated by SAS software, and not me.

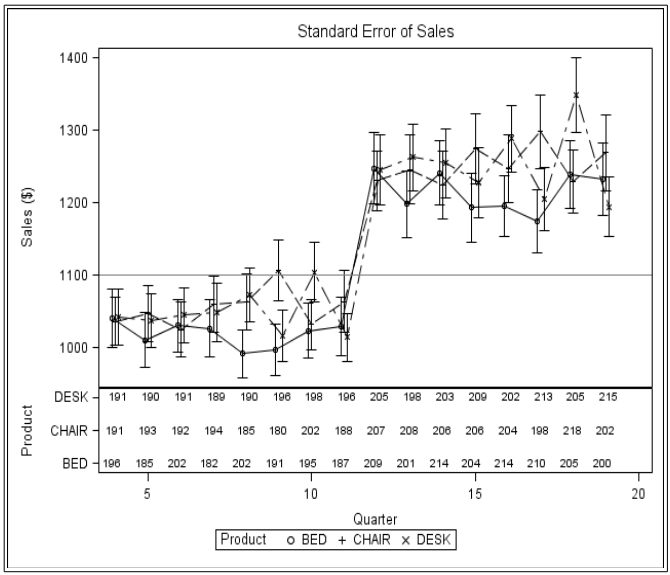
- Upper-case text is standard SAS text, e.g. **ODS GRAPHICS ON;**
- Mixed-case is user-supplied text, e.g. **x = y + z;**
- User-supplied parameters in DYNAMIC statements will begin with an underscore and will be in mixed-case, e.g. **_title1**

On this journey you will be shown some data preparation considerations, how to render data sets using Graph Templates, and how the Statistical Graphics (SG) procedures can be used to generate templates.

Because actually writing Graph Templates manually is not a task that is recommended without a lot of experience in generating them, you will not see any generated template programs until the SG procedures are introduced, and the actual template structure and syntax will not be shown until the final advanced user section.

You will see how to develop the templates that can generate the following graphs:





Generating Graph Templates

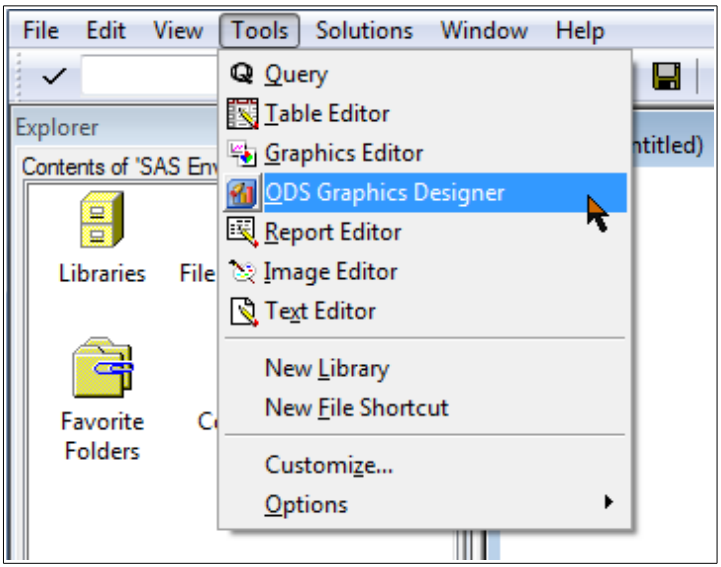
ODS Graphics Designer

How to Start the ODS Graphics Designer

The ODS Graphics Designer can be started from within SAS 9.2 and 9.3 with a macro call:

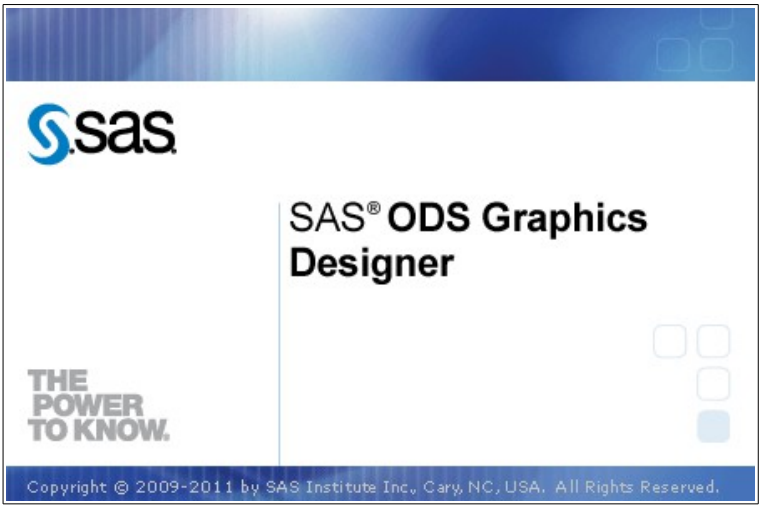
```
%SGDESIGN
```

It can also be started from a drop down menu option in SAS 9.3, which generates this macro call:

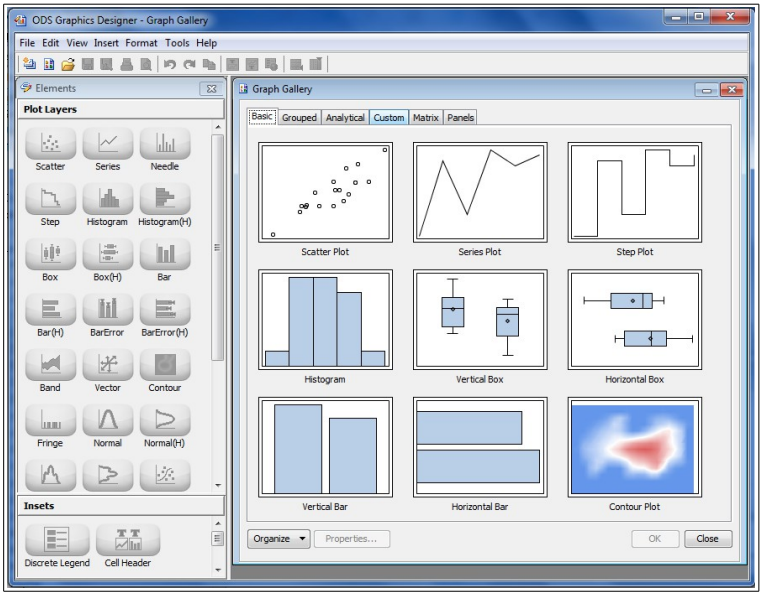


This starts a Java application which can be used to build Graph Templates, and then send them back to SAS for execution. The screen-shots in this section were taken of the SAS 9.3 version of the ODS Graphics Designer, but differences between SAS 9.2 and 9.3 will be noted.

Note that the ODS Graphics Designer will run on all SAS versions from SAS 9.2 M3 on Windows, UNIX and Linux platforms, but also requires access to a correctly configured Java installation and an interactive SAS session on the same platform. An Enterprise Guide Add-in for ODS Graphics Designer is available, but this also has these pre-requisites on the same Windows PC as Enterprise Guide.



The initial screen layout includes Elements and Graph Gallery.



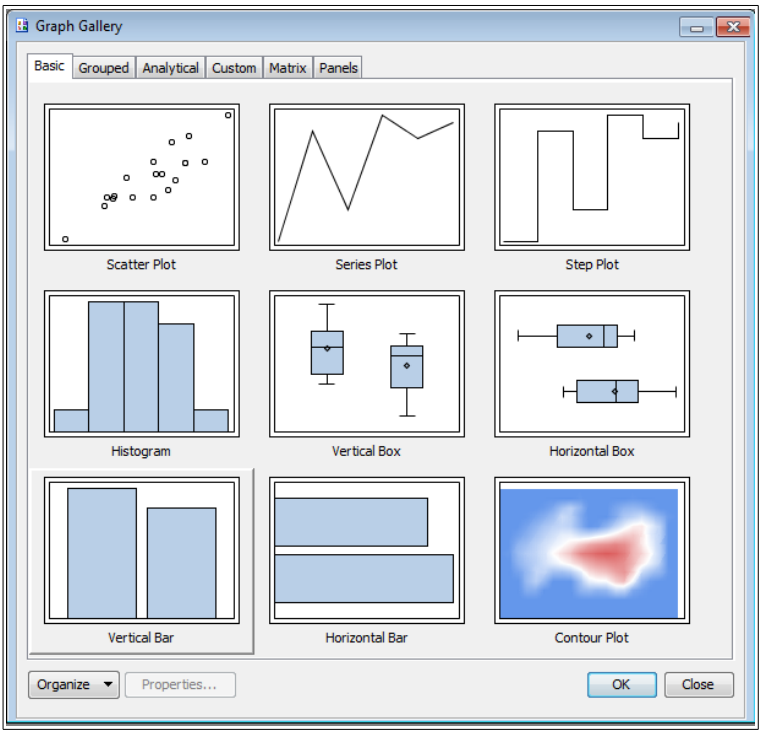
As the ODS Graphics Designer is not part of the SAS System, but an external program using the SAS software environment, the SAS data sets used to create templates must be stored as permanent SAS data sets in accessible libraries. The SAS System starts with the following permanent libraries allocated by default: SASHELP, SASUSER and MAPS.

The following code shows the allocating of a permanent SAS library, called **test**, and a new SAS data set, called **test.cars**, for use with %SGDESIGN, prior to calling the macro:

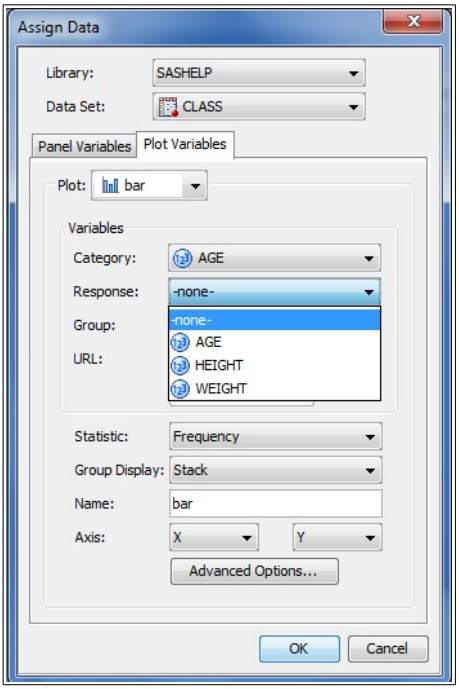
```
LIBNAME test "C:\saslibrary\";

DATA test.cars;
  SET sashelp.cars;
  percent_saving = 100 * (msrp - invoice) / msrp;
  highway_increase = mpg_highway - mpg_city;
  cylinder_size = engine_size / cylinders;
RUN;

%SGDESIGN
```

Using the Gallery to create Simple Templates

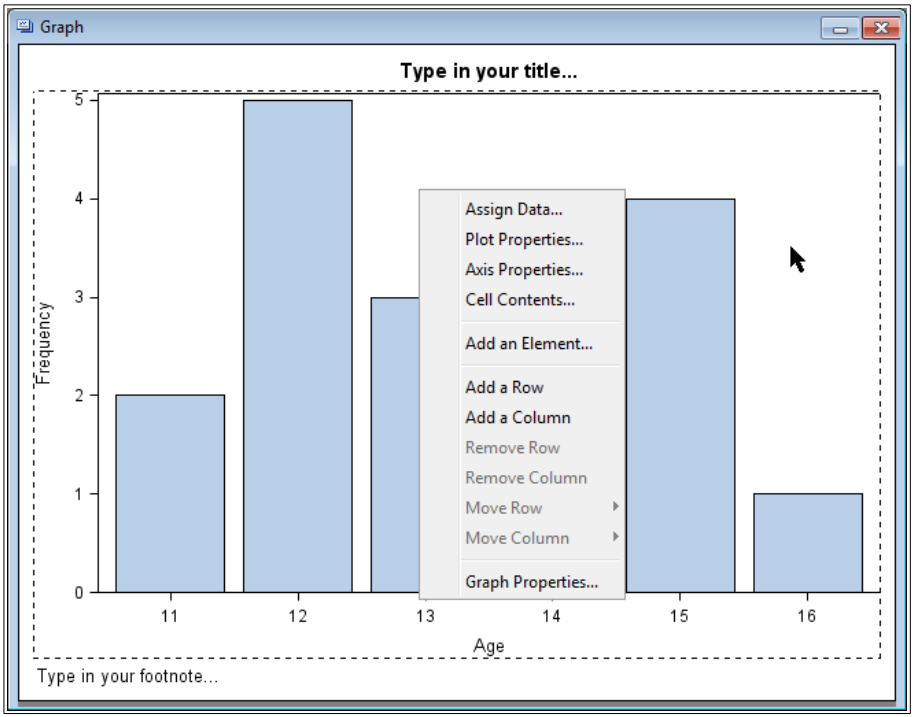
The Gallery provides a collection of typical graphical reports that can be used as starting points for more complex reports, but can be used with minimal customization to generate simple templates.



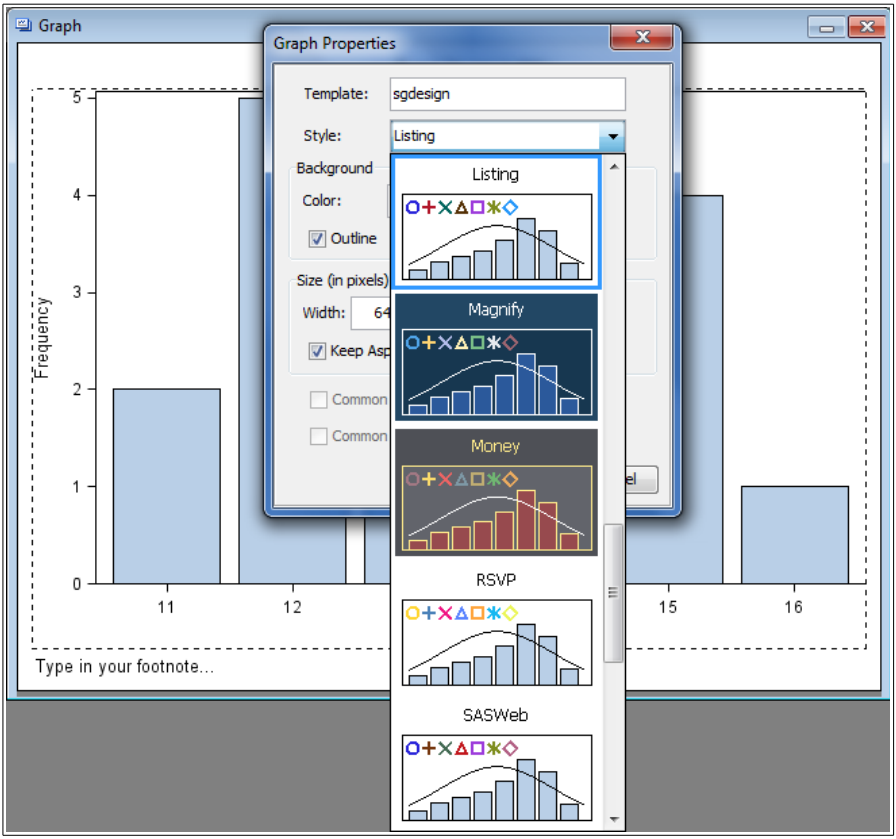
Once a Gallery entry has been selected you will be asked for details of the data to be plotted. Note that in SAS 9.2 there is no Group Display option.



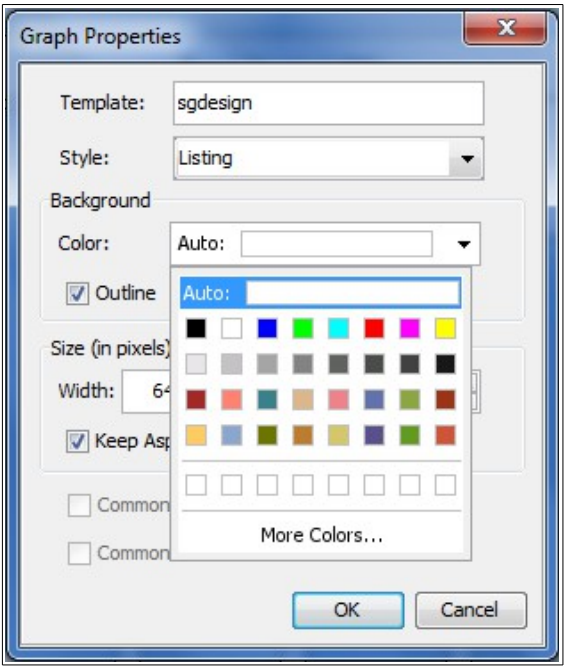
The style and graph layout can also be selected from the drop-down menu, which can be shown with a right-click on the graph.



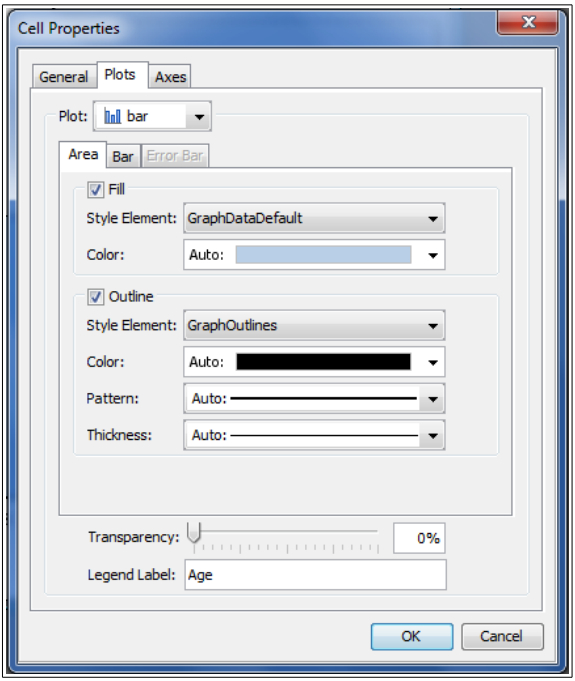
The Graph Properties... option is used to change the style.



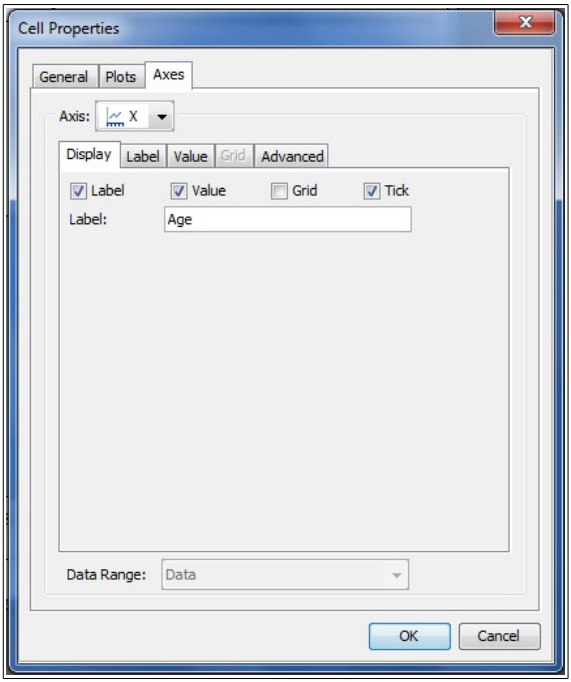
It is also possible to change the background and image size. Note that in SAS 9.2 it is not possible for the user to specify the Template name, and it is fixed as “sgdesign”.



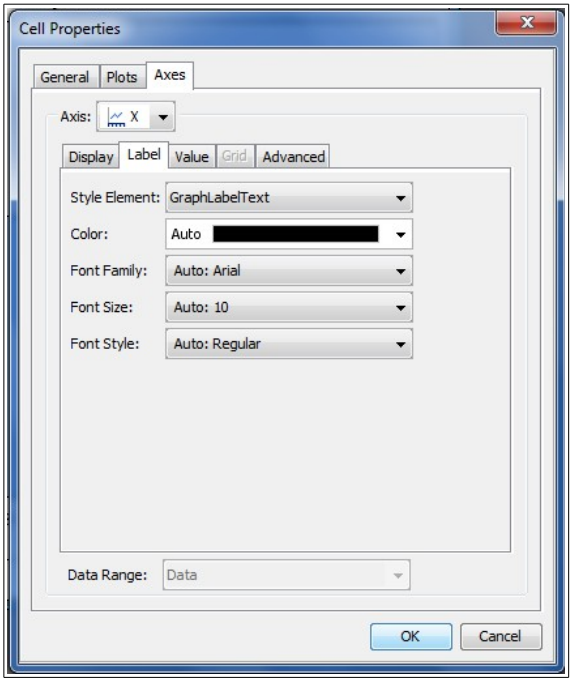
The Plot Properties... option provides a place to change the styles used for text and other graph features.



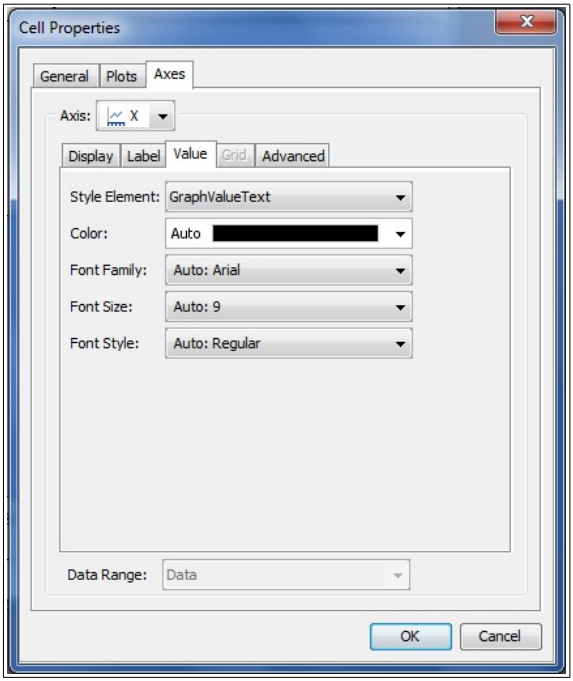
The Axis Properties... option opens the Axes tab on the window opened by the Plot Properties... option. The Display tab can be used to update the way each of the axes is displayed.



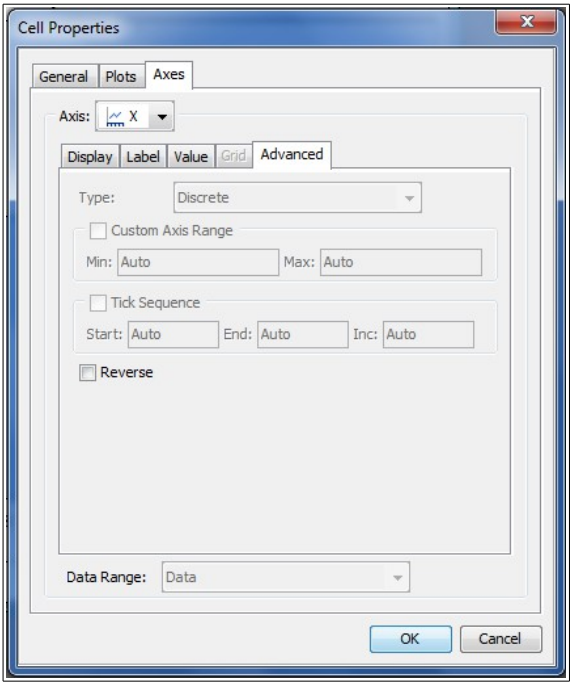
The Label tab can be used to update the label style for each axis.



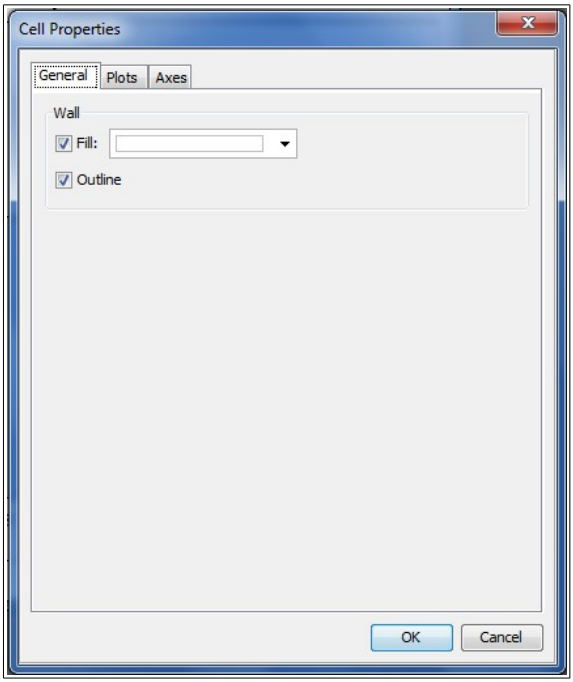
The Value tab can be used to update the value style for each axis.



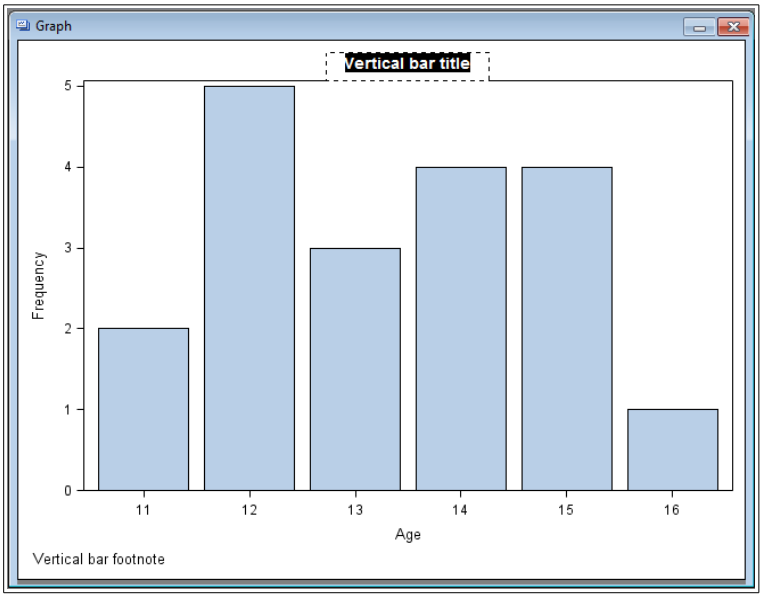
The Advanced tab gives options to update the way the values on the axis are arranged. These may be greyed out, depending on the type of graph displayed. In this case the only option available is to reverse the order of the tick marks on the x-axis, as the axis range itself has been automated.



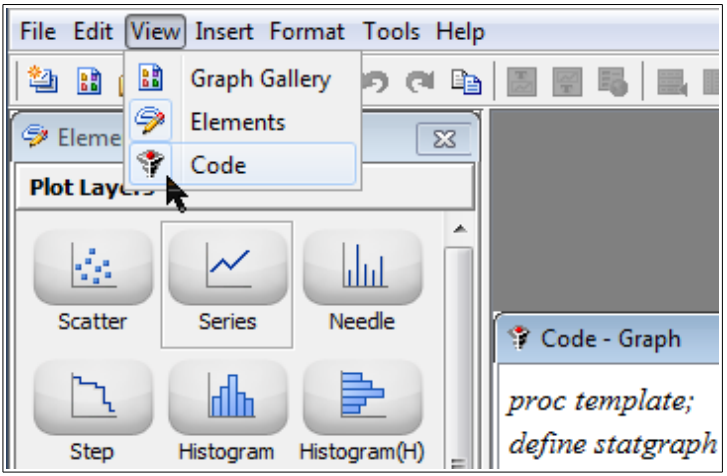
The Cell Contents... option can be used to update the general appearance of the graph.



The titles and footnotes can be updated by double-clicking the text, and overwriting whatever text is already present.

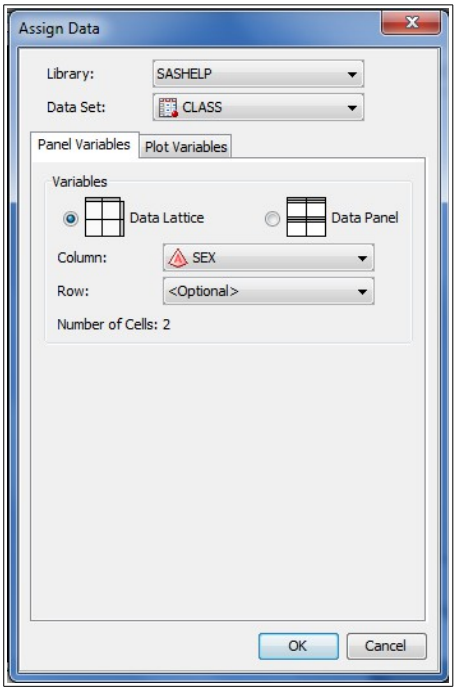


The graph and the code can be viewed together, if the Code window is not already visible, by clicking on the View menu option at the top of the ODS Graphics Designer window and selecting Code.

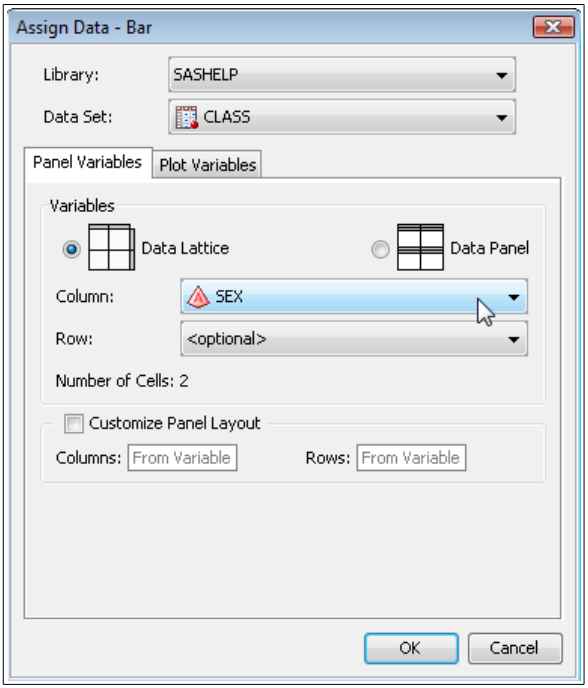


The template code for this graph can be seen as Template A in “Graph Template Contents” on page 65.

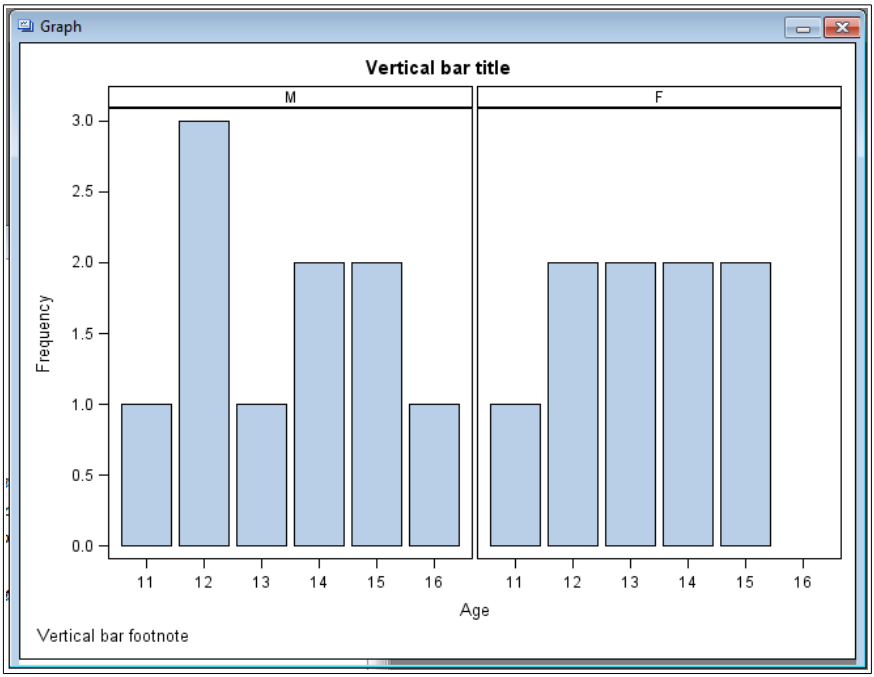
There is also an opportunity now to split the graph into a panelled layout using the Assign Data option and specifying the Panel Variables.



The following screen-shot is what would be seen, instead of the previous screen-shot, in the SAS 9.2 version, including an option to Customize Panel Layout which is no longer available in SAS 9.3.

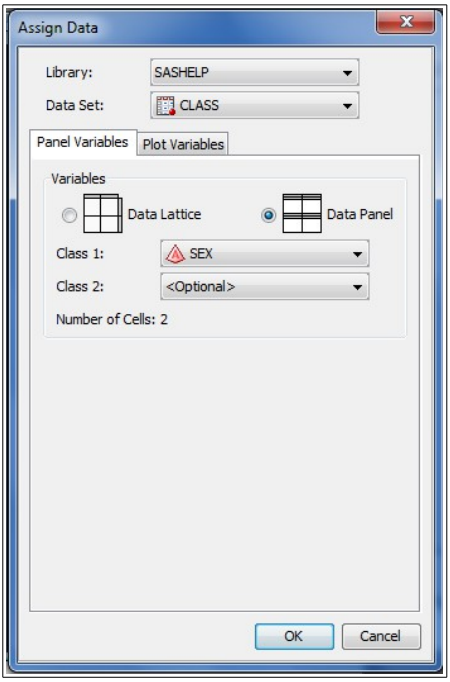


The resulting graph shows the data split between Sex=M and F, with a gap where there is no corresponding data for Age=16 when Sex=F.



The template code for this graph can be seen as Template B in “Graph Template Contents” on page 66.

Instead of arranging the data into a lattice, selecting the Data Panel option can create a different layout.

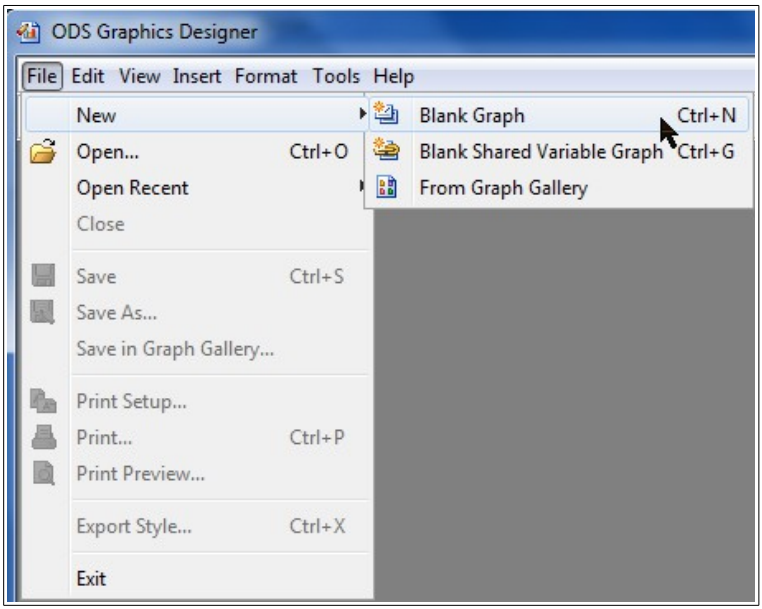


Panels are drawn with all of the corresponding categorical variable values in boxes above each individual cell. Lattices are drawn with first categorical variable value above each corresponding column of cells, and the second, where used, next to each corresponding row of cells. Therefore, because there are only two cells in the panel, the resulting graph is indistinguishable from that generated using the Data Lattice.

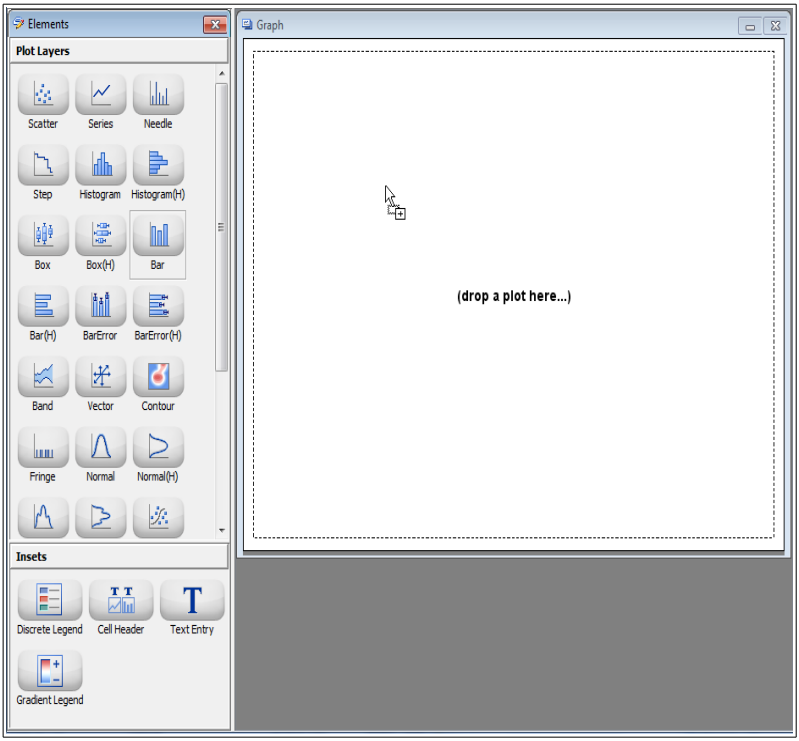
The template code for this graph can be seen as Template C in “Graph Template Contents” on page 67.

Building a Template from a Blank Graph

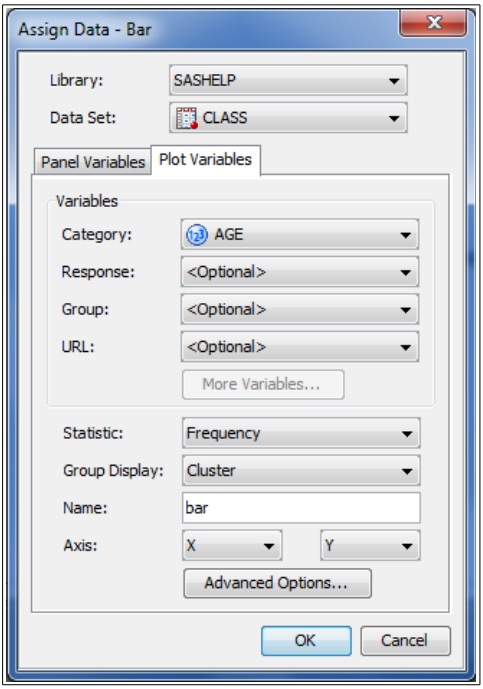
The difference between a Gallery entry and a blank graph is that no defaults have been set, so you have much more control over the content and layout of the final template.



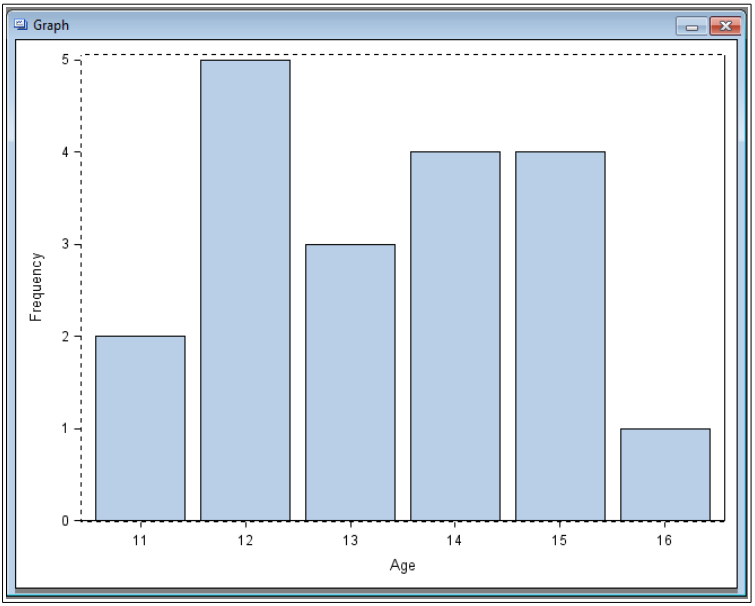
A blank graph is created ready for you to add what you need. It is now straightforward to add multiple graphical objects that use the same data to a single graphical report. You just drag the Elements from the appropriate Element group, i.e. Plot Layers, or Insets, onto the graph area.



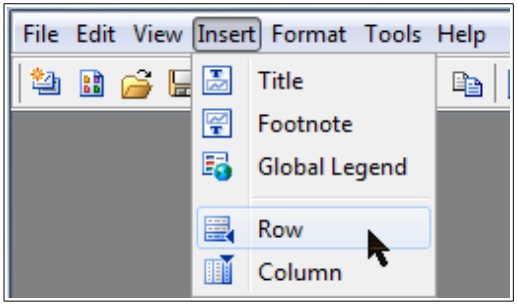
Once a plotting Element has been dragged, then the SAS data is requested. Note that in SAS 9.2 there is no Group Display option.



This generates a familiar graph, but without any titles or footnotes.

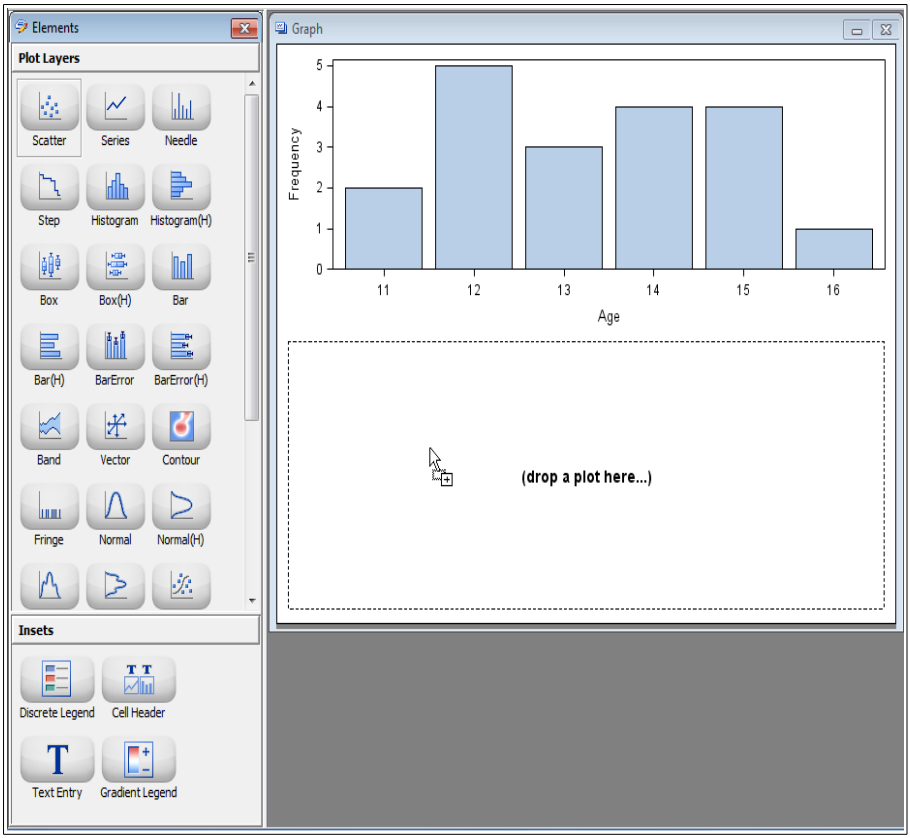


An additional row for another graph on the same page can then be created by adding a new row under the existing graph.



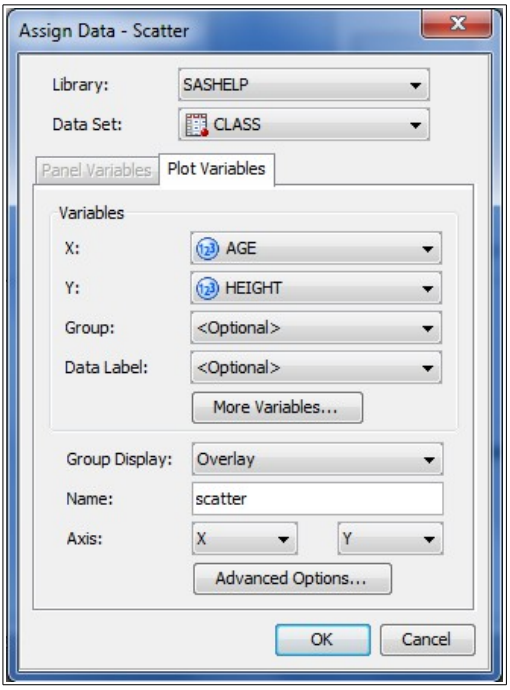
Note that there is also an option to insert a column, instead of a row. There is no limit to the number of rows and columns that can be added, other than the fact that each graph cell ought to be, at least, big enough to show something readable.

This generates a blank row in the image ready for another graph to be dragged into place. However, using the Column menu option would have added a new column instead.

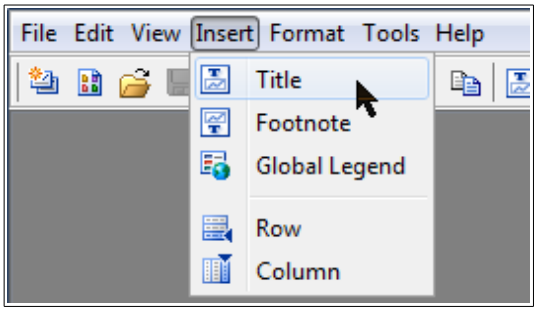


Note that the ODS Graphics Designer is only able to generate a lattice containing N1 x N2 cells, where every row contains the same number of columns, and every column contains the same number rows. A lattice where there were, for example, more cells in row 1 than in row 2 would require a manual amendment.

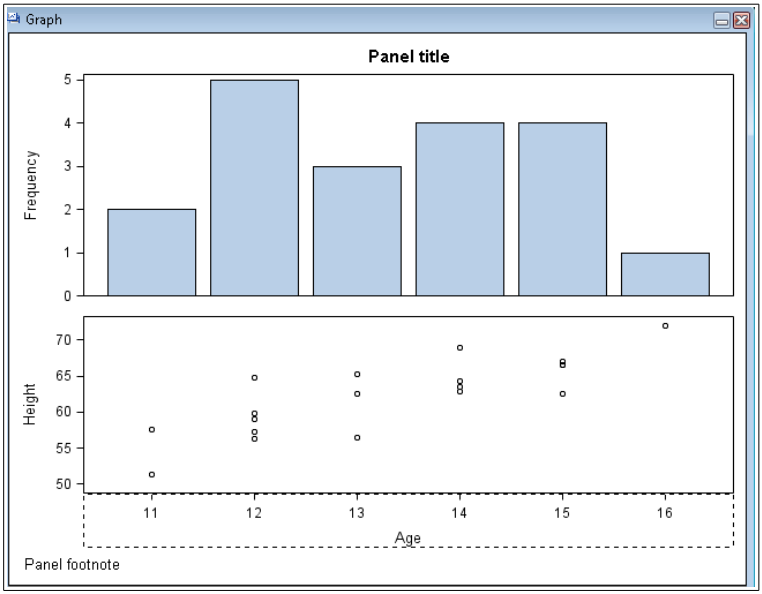
The data for the new graph is automatically requested. Note that the Group Display option is not seen in SAS 9.2.



Titles and footnotes can now be added, but, in ODS Graphics Designer, they can only be “global” titles and footnotes, not specific to each cell.



The graph is now ready for use.

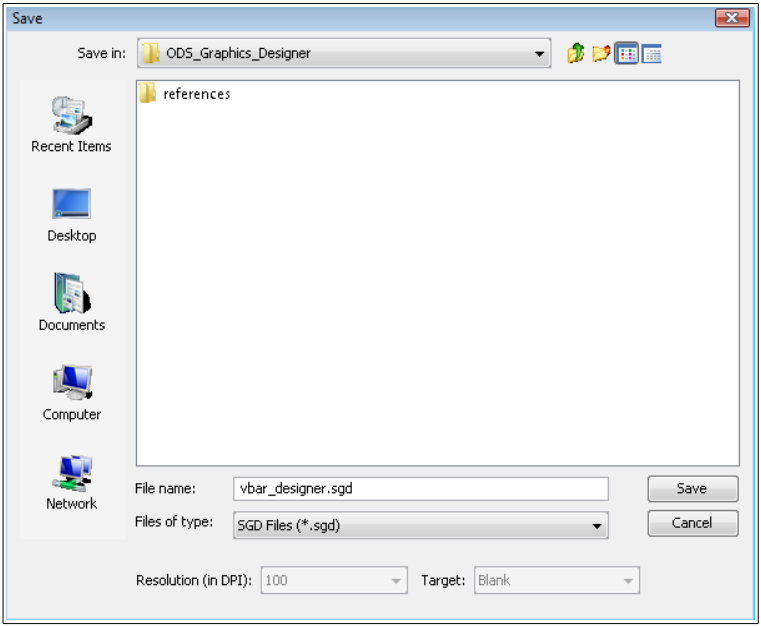


Note that common column axes are only possible in the ODS Graphics Designer when there is only a single column of graph cells. As soon as an additional column of cells is added, then the common column axes are changed back to individual axes again. However, common row axes are possible in ODS Graphics Designer, no matter how many rows of cells are present.

The template code for this graph can be seen as Template D in “Graph Template Contents” on page 68.

Saving the Template as a Designer File (*.sgd)

If you click on the Graph Window, then click the File menu option, the template can be saved as a Designer File with file type *.sgd by clicking Save As...

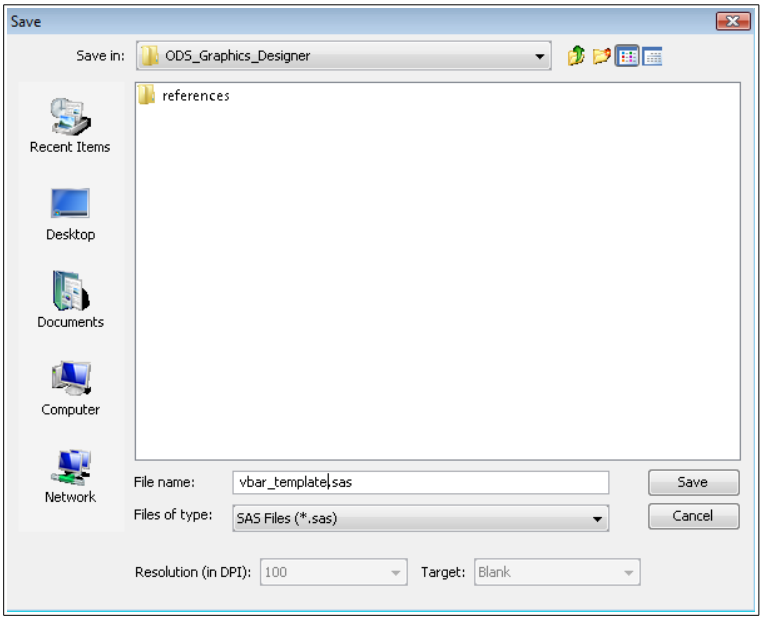


These files can be opened and edited with ODS Graphics Designer, so can be used to store the latest versions of the templates you are developing for later use. They could also be saved as backup copies, if you want to develop a range of templates from a single starting template. Note that these files can only be opened with ODS Graphics Designer, or PROC SGDESIGN, which is not described in this book.

The template can also be saved to the Gallery within ODS Graphics Designer by clicking Save to Gallery, which will make the template immediately available when ODS Graphics Designer is next used.

Saving the Template as a SAS Program (*.sas)

If you click on the Code Window, then click the File menu option, the template can be saved as a SAS Program by clicking Save As...



Graph Template Usage

Preparing Data for Graph Templates

The following code generates simulated clinical data with visit numbers, products, an absolute value with a standard error (for the simple line plot). The original data in **sashelp.prdsal2** is very uniform, so a filter is used to make the value counts less even.

```
PROC SQL;
  CREATE TABLE plotdata AS
    SELECT INTCK('QTR', '01jan1994'd, monyr) AS visitnum
          ,product
          ,MEAN(predict) AS value1
           /*used for the simple line plots*/
          ,STDERR(predict) AS value1_se
           /*used for the simple line plots*/
          ,COUNT(*) AS count
    FROM   sashelp.prdsal2
             (WHERE = (product IN ('BED' 'CHAIR' 'DESK')
                       AND predict > 400))
    GROUP BY
           visitnum
          ,product;
QUIT;
```

The extra calculations in **plotdata_ods** are applied to **visitnum** to offset the points to prevent them from overlapping and obscuring data, to **price_upper** and **price_lower** to add upper and lower standard error points for the error bars, and to **ccount** to convert the numeric counts to text for the final graph to be generated from a template by PROC SGRENDER.

```
DATA plotdata_ods;
  SET plotdata;
  LENGTH ccount $4;
  SELECT (product);
    WHEN ('BED') visitnum = visitnum - 0.1;
    WHEN ('DESK') visitnum = visitnum + 0.1;
    OTHERWISE;
  END;
  value1_upper = value1 + value1_se;
  value1_lower = value1 - value1_se;
  ccount = STRIP(PUT(count, 4.));
  LABEL value1 = 'Sales ($)'
        visitnum = 'Quarter';
RUN;
```

Displaying Graph Templates

*DATA _NULL_*

When Graph Templates were first introduced in SAS 9.1.3, and before the introduction of PROC SGRENDER, the DATA _NULL_ step with FILE PRINT ODS and PUT _ODS_ statements was the only way to display input data using Graph Templates. This method is still available in SAS 9.3, but PROC SGRENDER is now considered to be the preferred method. The **sgplot_count** template itself will be described in detail later.

```
ODS GRAPHICS ON;

DATA _NULL_;
   LENGTH ccount $4;
   SET plotdata;
   BY visitnum;
   value1_upper = value1 + price_se;
   value1_lower = value1 - price_se;
   ccount = STRIP(PUT(count, 4.));
   FILE PRINT ODS =
      (TEMPLATE = 'sgplot_count'
       DYNAMIC = (_title = "Figure 1. Standard Error of Sales"
                  _title2 = "Overall"
                  _footnote = "Program: &pgm..sas"
                  _xvar = "visitnum"
                  _xlabel = "Quarter"
                  _ylabel = "Bed and Chair sales ($)"
                  _yvar1 = "value1"
                  _yupper1 = "value1_upper"
                  _ylower1 = "value1_lower"
                  _nvar1 = "ccount"
                  _group = "product"
                 )
      );
   PUT _ODS_;
RUN;

ODS GRAPHICS OFF;
```

PROC SGRENDER

PROC SGRENDER was introduced in SAS 9.2, but is probably more closely related to PROC GANNO than PROC GPLOT, as it is used to render input data using pre-defined Graph Templates. The **sgplot_count** template itself will be described in detail later.

```
PROC SGRENDER DATA = plotdata_ods
                        (WHERE = (product IN ('BED' 'CHAIR')))
              TEMPLATE = 'sgplot_count';
  DYNAMIC _title = "Sales"
          _title2 = "Bed and Chair"
          _footnote = "Program: &pgm..sas"
          _xvar = "visitnum"
          _xlabel = "Quarter"
          _ylabel = "Sales ($)"
          _yvar1 = "value1"
          _yupper1 = "value1_upper"
          _ylower1 = "value1_lower"
          _nvar1 = "ccount"
          _group = "product";
RUN;
```

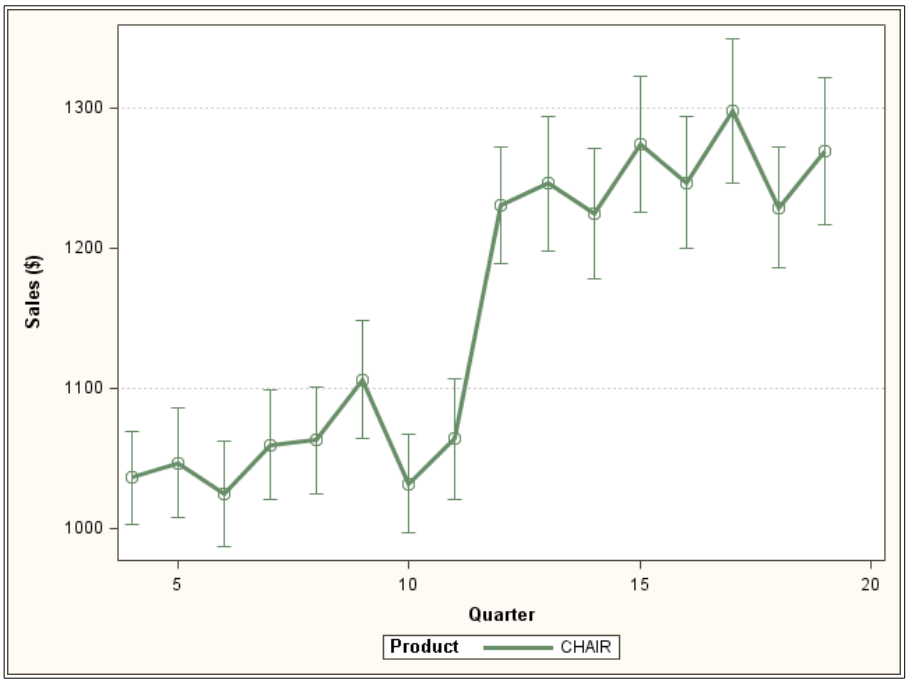
Note that the highlighted ODS GRAPHICS statements used with the Data Step is no longer necessary, but only optional with any of the Statistical Graphics (SG) procedures.

Statistical Graphics (SG) Procedures

PROC SGPLOT

PROC SGPLOT was introduced in SAS 9.2 and effectively replaces PROC GPLOT and PROC GCHART for most of the standard graphs they are able to produce. PROC SGPLOT also provides facilities to combine different plots by overlaying them on the same axes.

The example given here only displays a subset of the data with error bars and vertical reference lines:



The following code was used to generate the graph. Note that a SAS program containing PROC TEMPLATE code to recreate the graph is saved to **sgplot_template.sas** using the TMPLOUT= option:

```
PROC SGPLOT DATA = plotdata_ods
                    (WHERE = (product = 'CHAIR'))
            TMPLOUT = "sgplot_template.sas";
  SERIES X = visitnum Y = value1 /
         MARKERATTRS = (SIZE = 10PX)
         LINEATTRS = (THICKNESS = 3PX)
         GROUP = product;
  SCATTER X = visitnum Y = value1 /
          YERRORUPPER = value1_upper
          YERRORLOWER = value1_lower
          MARKERATTRS = (SIZE = 10PX)
          GROUP = product;
  REFLINE 1100 / AXIS = Y LINEATTRS = (PATTERN = DOT);
  REFLINE 1300 / AXIS = Y LINEATTRS = (PATTERN = DOT);
RUN;
```

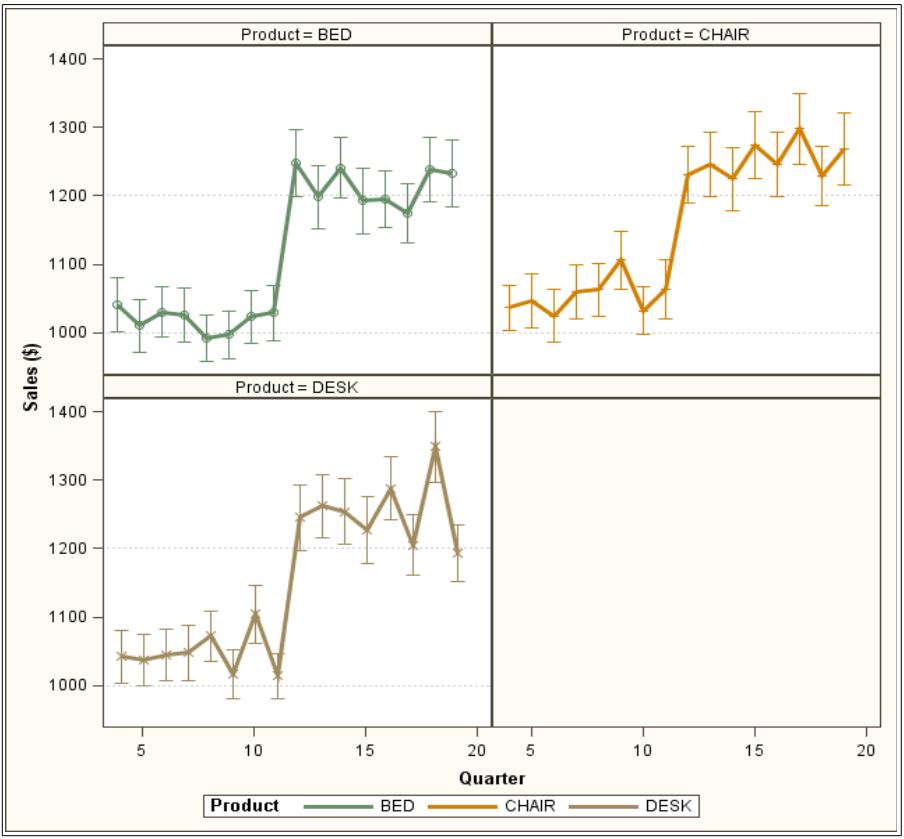
By comparing the graphs created by PROC GPLOT and PROC SGPLOT we can see there are a number of obvious differences in their default behaviour. In particular the y-axis labels have been rotated and the tick marks on both axes are sensibly spaced in PROC SGPLOT, as they are with the other SG procedures. Both features are available in PROC GPLOT, but require additional parameters to achieve.

The generated Graph Template created by this PROC SGPLOT example is given below:

```
proc template;
define statgraph sgplot;
begingraph;
layout overlay;
   SeriesPlot X='visitnum'n Y='value1'n / Group='PRODUCT'n
              Markerattrs=( Size=10px)
              Lineattrs=( Thickness=3px)
              LegendLabel="Sales ($)"
              NAME="series";
   ScatterPlot X='visitnum'n Y='value1'n /
               primary=true Group='PRODUCT'n
               Markerattrs=( Size=10px)
               YErrorUpper='value1_upper'n
               YErrorLower='value1_lower'n
               LegendLabel="Sales ($)"
               NAME="SCATTER";
   DiscreteLegend "series"/ title="Product";
endlayout;
endgraph;
end;
run;
```

PROC SGPANEL

PROC SGPANEL was introduced in SAS 9.2, and makes the production of multiple graphs in a grid very straightforward. It includes the majority of the features in PROC SGPLOT, but also includes the PANELBY statement that specifies how the data for each panel is selected. Single or multiple panels can be generated per page, and multiple graph pages will be created if the number of panels exceeds the number available on each page. Single panels, i.e. 1 x 1 grids, are functionally similar to using a BY statement with PROC SGPLOT, except that the panel variable values are presented in a box above each cell.



The example given here only displays the three sub-graphs in the available spaces in a 2 x 2 grid, but it can also be used to create a grid of graphs where the rows are based on one category value, and the columns are based on another category value, allowing direct comparison of 4 different category value combinations in a single image.

The following code was used to generate the graph above. Note that no SAS program containing PROC TEMPLATE code to recreate the graph can be generated from PROC SGPANEL in SAS 9.3, as the TMPLOUT= option is no longer available:

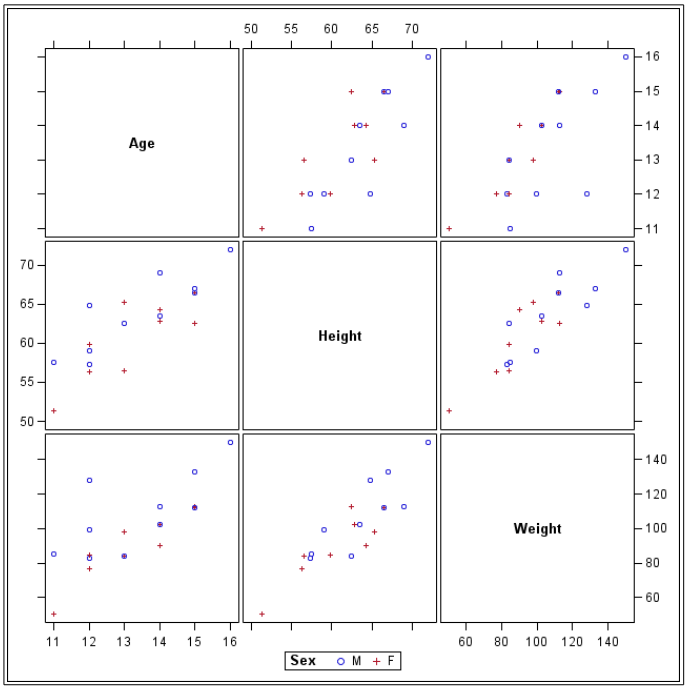
```
PROC SGPANEL DATA=plotdata_ods;
  PANELBY product / LAYOUT = PANEL;
  SERIES X = visitnum Y = value1 /
         MARKERATTRS = (SIZE = 10PX)
         LINEATTRS = (THICKNESS = 3PX PATTERN = SOLID)
         GROUP = product;
  SCATTER X = visitnum Y = value1 /
          YERRORUPPER = value1_upper YERRORLOWER = value1_lower
          MARKERATTRS = (SIZE = 10PX)
          GROUP = product;
  REFLINE 1000 / AXIS = Y LINEATTRS = (PATTERN = DOT);
  REFLINE 1200 / AXIS = Y LINEATTRS = (PATTERN = DOT);
RUN;
```

Similar graphs using PROC GPLOT would require PROC GREPLAY and careful template design and sizing to achieve.

PROC SGSCATTER

PROC SGSCATTER was introduced in SAS 9.2, and has a number of different plot statements. MATRIX creates an N x N grid of sub-graphs where each variable is plotted against the each of the other variables, with either variable labels, or graphs of each variable, along the diagonal. COMPARE creates a row or column of sub-graphs of different variables with a common axis. PLOT creates one or more scatter sub-graphs of pairs of specified variables.

The MATRIX statement can include just the variable labels along the diagonal and tick marks on the axes:



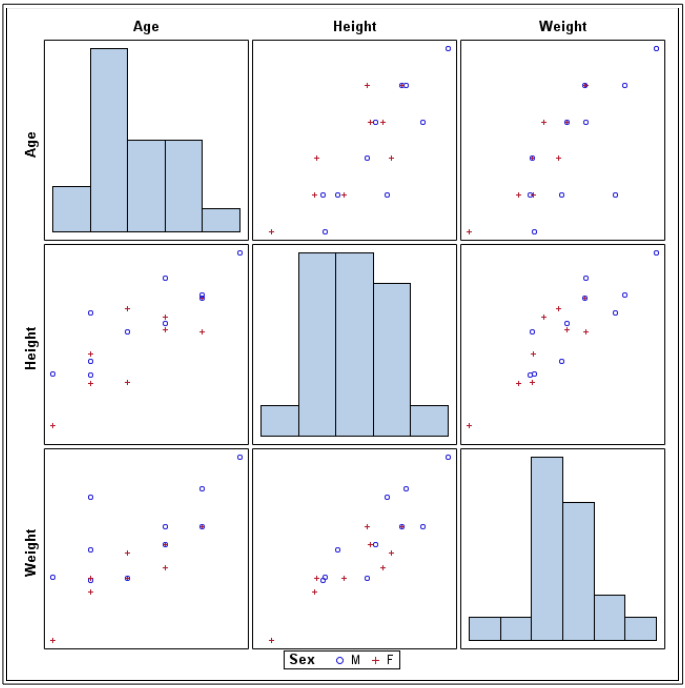
The following code was used to generate the graph above. Note that a SAS program containing PROC TEMPLATE code to recreate the graph is saved to **sgscatter_matrix_template1.sas** using the TMPLOUT= option:

```
PROC SGSCATTER DATA = sashelp.class
               TMPLOUT = "sgscatter_matrix_template1.sas";
  MATRIX age height weight / GROUP = sex;
RUN;
```

The generated Graph Template created by this PROC SGSCATTER example is given below:

```
proc template;
define statgraph sgscatter;
begingraph / designwidth=640 designheight=640;
layout gridded;
   layout lattice;
       ScatterPlotMatrix Age Height Weight / NAME="MATRIX" Group=Sex;
   endlayout;
   DiscreteLegend "MATRIX" / order=rowmajor title="Sex";
endlayout;
endgraph;
end;
run;
```

The MATRIX statement can also include any combination of histograms, normal density curves, or kernel density estimates of each variable along the diagonal by using the DIAGONAL= option, but this option removes the tick marks from the axes. The graph below shows only the histograms:



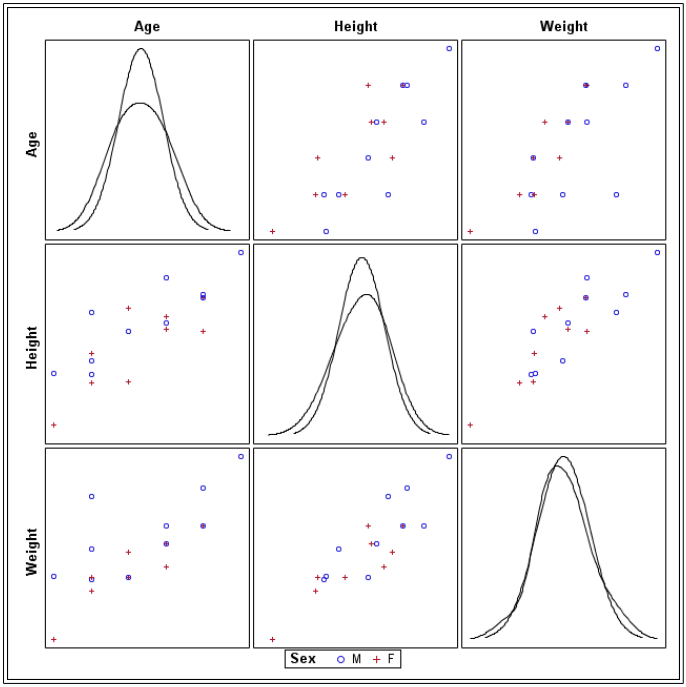
The following code was used to generate the graph above, with the PROC TEMPLATE code saved to **sgscatter_matrix_template2.sas** using the TMPLOUT= option:

```
PROC SGSCATTER DATA = sashelp.class
               TMPLOUT = "sgscatter_matrix_template2.sas";
  MATRIX age height weight / GROUP = sex DIAGONAL = (HISTOGRAM);
RUN;
```

The generated Graph Template is given below:

```
proc template;
define statgraph sgscatter;
begingraph / designwidth=640 designheight=640;
layout gridded;
   layout lattice;
       ScatterPlotMatrix Age Height Weight /
                         NAME="MATRIX" Group=Sex
                         diagonal=( histogram );
   endlayout;
   DiscreteLegend "MATRIX" / order=rowmajor title="Sex";
endlayout;
endgraph;
end;
run;
```

The graph below shows normal density curves and kernel density estimates together, but could have included histograms too:



The following code was used to generate the graph above, with the PROC TEMPLATE code saved to **sgscatter_matrix_template3.sas** using the TMPLOUT= option:

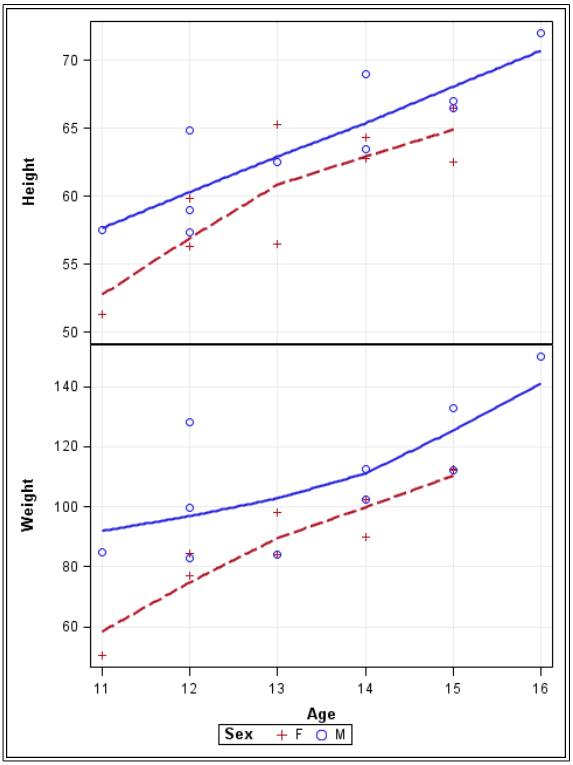
```
PROC SGSCATTER DATA = sashelp.class
               TMPLOUT = "sgscatter_matrix_template3.sas";
  MATRIX age height weight / GROUP = sex DIAGONAL = (KERNEL NORMAL);
RUN;
```

The generated Graph Template is given below:

```
proc template;
define statgraph sgscatter;
begingraph / designwidth=640 designheight=640;
layout gridded;
   layout lattice;
       ScatterPlotMatrix Age Height Weight /
                         NAME="MATRIX" Group=Sex
                         diagonal=( normal kernel );
   endlayout;
   DiscreteLegend "MATRIX" / order=rowmajor title="Sex";
endlayout;
endgraph;
end;
run;
```

As for PROC SGPANEL, similar graphs to those generated by PROC SGSCATTER, but using PROC GPLOT instead, would require PROC GREPLAY and careful template design and sizing to achieve.

The example below uses the COMPARE statement to plot a comparison of height and weight by age, with the different values where sex=”M” and sex=”F” shown using a Loess curve:



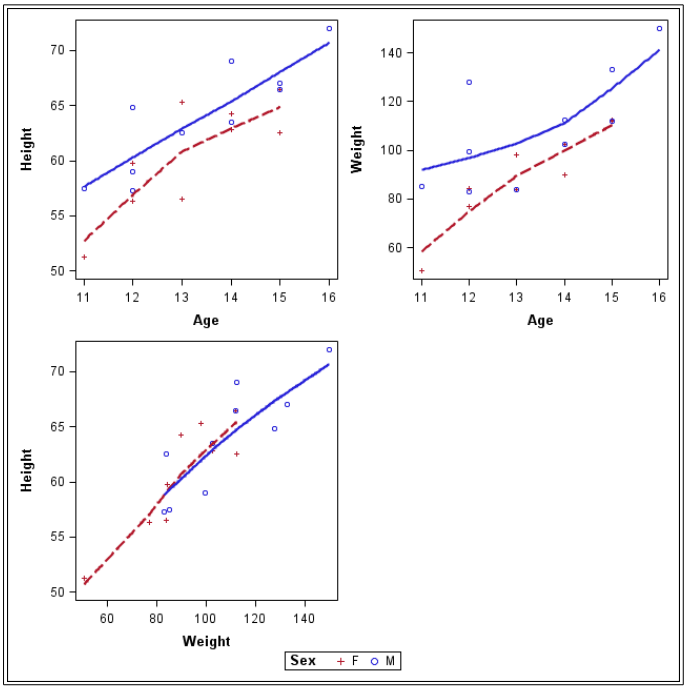
The following code was used to generate the graph:

```
PROC SGSCATTER DATA = sashelp.class
               TMPLOUT = "sgscatter_template.sas";
  COMPARE Y = (height weight) X = age /
          GROUP = sex
          MARKERATTRS = (SIZE = 10)
          LOESS = (ALPHA=0.05)
          GRID;
RUN;
```

The generated Graph Template created by the above PROC SGSCATTER example is given below:

```
proc template;
define statgraph sgscatter;
begingraph / designwidth=480 designheight=640;
DiscreteAttrVar attrvar=__ATTRVAR1__
                var=Sex attrmap="__ATTRMAP__";
DiscreteAttrVar attrvar=__ATTRVAR1__
                var=eval(sort(Sex, RETAIN=ALL)) attrmap="__ATTRMAP__";
DiscreteAttrMap name="__ATTRMAP__" / autocycleattrs=1;
Value "M";
Value "F";
EndDiscreteAttrMap;
layout gridded;
   layout lattice / columnDataRange=union;
      ColumnAxes;
      ColumnAxis / griddisplay=on;
      EndColumnAxes;
      layout overlay /
             xaxisopts=( griddisplay=on) yaxisopts=( griddisplay=on);
         ScatterPlot X=Age Y=Height /
                     primary=true Group=__ATTRVAR1__
                     Markerattrs=( Size=10) NAME="COMPARE";
         LoessPlot X=Age Y=Height / Group=__ATTRVAR1__ Alpha=0.05;
      endlayout;
      layout overlay /
             xaxisopts=( griddisplay=on) yaxisopts=( griddisplay=on);
         ScatterPlot X=Age Y=Weight /
                     primary=true Group=__ATTRVAR1__
                     Markerattrs=( Size=10);
         LoessPlot X=Age Y=Weight / Group=__ATTRVAR1__ Alpha=0.05;
      endlayout;
   endlayout;
   DiscreteLegend "COMPARE" / order=rowmajor title="Sex";
endlayout;
endgraph;
end;
run;
```

Finally the PLOT statement generates a grid of specified graphs, all with the same options, in this case each with a Loess curve:



The following code was used to generate this graph, with the PROC TEMPLATE code saved to **sgscatter_plot_template.sas** using the TMPLOUT= option:

```
PROC SGSCATTER DATA = sashelp.class
               TMPLOUT = "sgscatter_plot_template.sas";
  PLOT (height weight)*age height*weight /
       GROUP = sex LOESS = (ALPHA=0.05);
RUN;
```

The generated Graph Template is given below:

```
proc template;
define statgraph sgscatter;
begingraph / designwidth=640 designheight=640;
DiscreteAttrVar attrvar=__ATTRVAR1__
                var=Sex attrmap="__ATTRMAP__";
DiscreteAttrVar attrvar=__ATTRVAR1__
                var=eval(sort(Sex, RETAIN=ALL)) attrmap="__ATTRMAP__";
DiscreteAttrMap name="__ATTRMAP__" / autocycleattrs=1;
Value "M";
Value "F";
EndDiscreteAttrMap;
layout gridded;
   layout lattice / rowgutter=10 columngutter=10 columns=2;
      layout overlay;
         ScatterPlot X=Age Y=Height /
                     primary=true Group=__ATTRVAR1__ NAME="PLOT";
         LoessPlot X=Age Y=Height / Group=__ATTRVAR1__ Alpha=0.05;
      endlayout;
      layout overlay;
         ScatterPlot X=Age Y=Weight / primary=true Group=__ATTRVAR1__;
         LoessPlot X=Age Y=Weight / Group=__ATTRVAR1__ Alpha=0.05;
      endlayout;
      layout overlay;
         ScatterPlot X=Weight Y=Height /
                     primary=true Group=__ATTRVAR1__;
         LoessPlot X=Weight Y=Height / Group=__ATTRVAR1__ Alpha=0.05;
      endlayout;
   endlayout;
   DiscreteLegend "PLOT" / order=rowmajor title="Sex";
endlayout;
endgraph;
end;
run;
```

Customizing Graph Templates (for Advanced Users)

Structure and Syntax

Structure

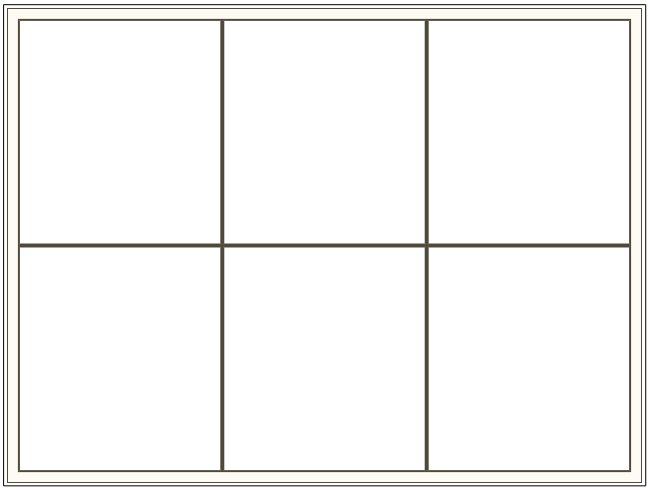
The basic Graph Template Language is made up of nested structures, including Layout statements and Plot statements:

The nested structure of a Graph Template is illustrated below:

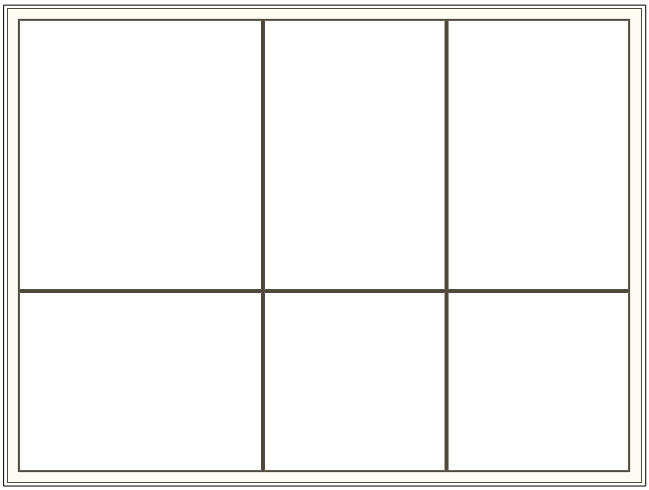
```
PROC TEMPLATE;
 DEFINE STATGRAPH name; ** create the template
  DYNAMIC name(s); ** define any parameters (optional)
  MVAR name(s); ** define character macro variables used by
                   name (optional)
  NMVAR name(s); ** define numeric macro variables used by
                    name (optional)
 BEGINGRAPH; ** start the graph (new in SAS 9.2!)
  ENTRYTITLE title; ** create a title (repeated for
                       additional titles)
  LAYOUT ** at least one layout statement is required
   LAYOUT ** nested layout statements (optional)
    Any plot statements, including titles, graph areas, footnotes,
    etc.
   ENDLAYOUT;
   Any plot statements, including titles, graph areas, footnotes,
   etc.
  ENDLAYOUT;
 ENTRYFOOTNOTE footnote; ** create a footnote (repeated for
                           additional footnotes)
 ENDGRAPH;
 END;
RUN;
```

Layout statements include:

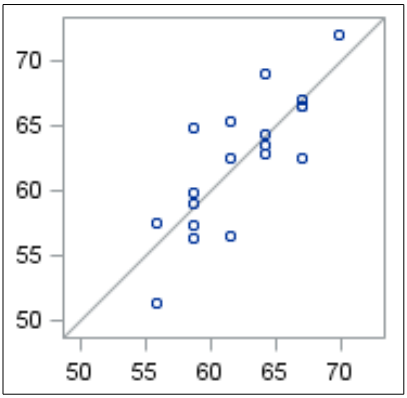
- LAYOUT GRIDDED and LAYOUT DATAPANEL – create a grid of graph cells with the same dimensions and properties:



- LAYOUT LATTICE and LAYOUT DATALATTICE – create a grid of graph cells with different dimensions and properties:

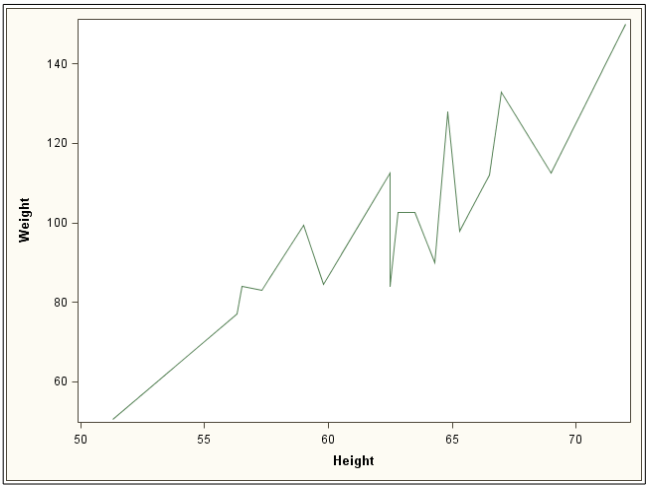


- LAYOUT OVERLAY (or LAYOUT PROTOTYPE inside LAYOUT DATALATTICE and LAYOUT DATAPANEL) – create a single graph cell with one or more overlaid plots:

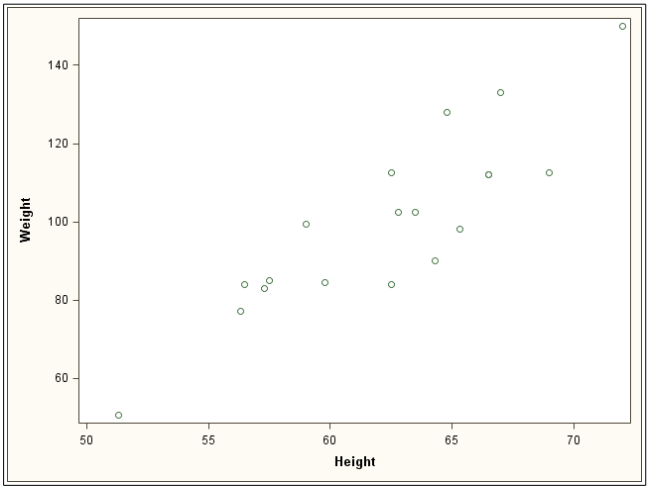


Plot statements include:

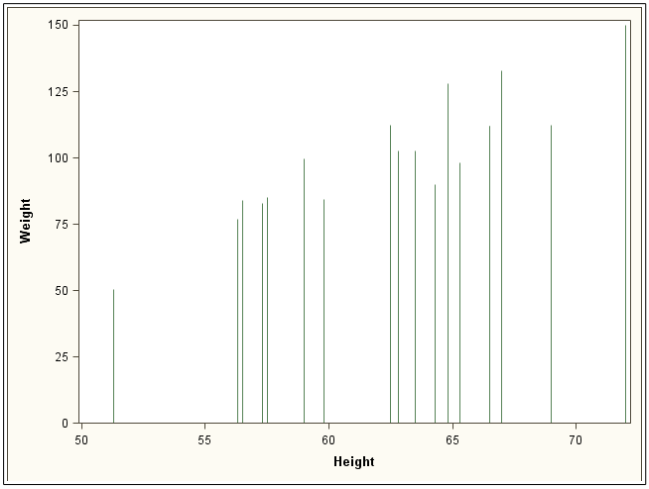
- SERIESPLOT – create a plot of connected points.



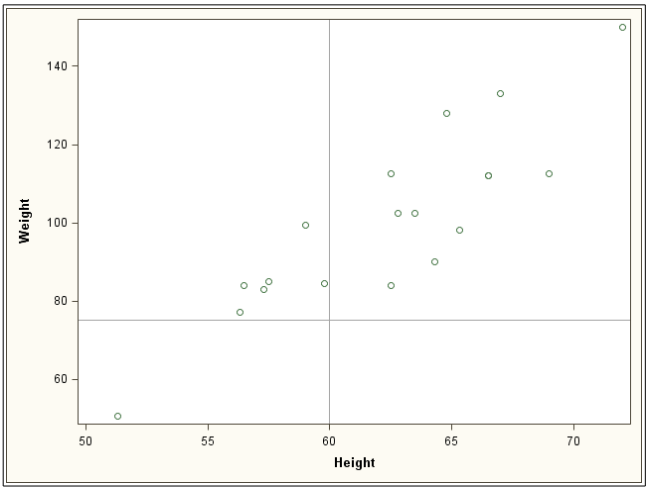
- SCATTERPLOT – create a plot of symbols at specified points.



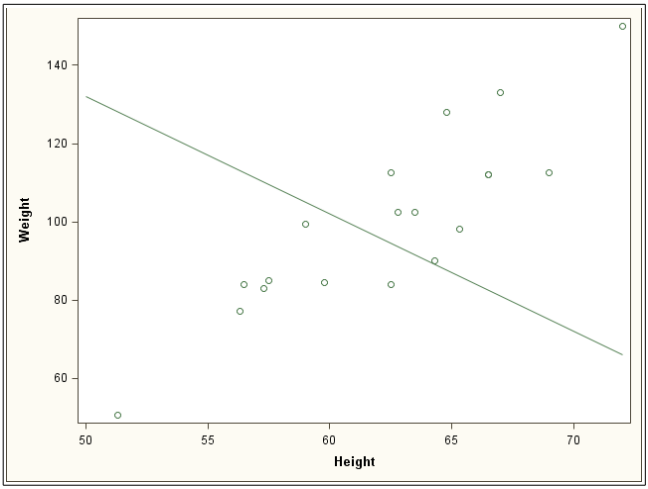
- NEEDLEPLOT – create a plot of vertical lines joining the horizontal zero axis line to each point.



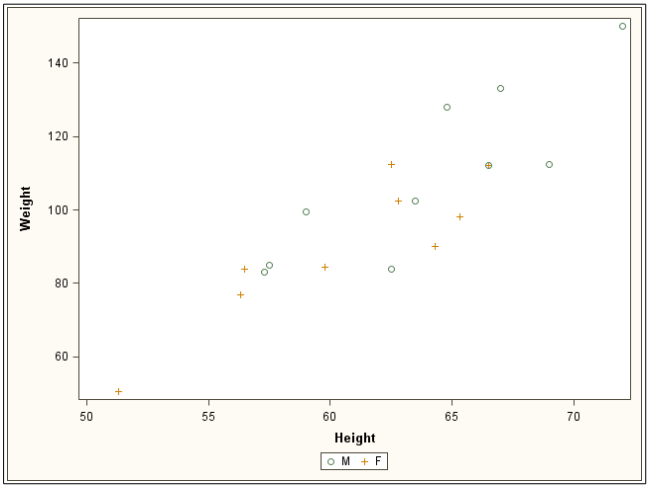
- REFERENCELINE – draw a line on the graph parallel with the x-axis or y-axis.



- LINEPARM – draw a line on the graph with a specified starting position and slope.



- DISCRETELEGEND – create a legend.



Graph Template Contents

The following templates were generated by the ODS Graphics Designer earlier in this book.

Template A

Note that in SAS 9.2 there is no CLUSTERWIDTH= parameter and the generated code does not include the PROC SGRENDER step.

```
proc template;
define statgraph sgdesign;
dynamic _AGE;
begingraph;
  entrytitle halign=center 'Vertical bar title';
  entryfootnote halign=left 'Vertical bar footnote';
  layout lattice / rowdatarange=data columndatarange=data
                   rowgutter=10 columngutter=10;
    layout overlay;
      barchart x=_AGE / name='bar' stat=freq clusterwidth=1.0;
    endlayout;
  endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.CLASS template=sgdesign;
dynamic _AGE="AGE";
run;
```

Template B

Comparing this new generated template with the original graph Template A, you will see that the multiple panels have been achieve using LAYOUT DATALATTICE around LAYOUT PROTOTYPE, instead of LAYOUT LATTICE around LAYOUT OVERLAY.

```
proc template;
define statgraph sgdesign;
dynamic _AGE _SEX;
dynamic _panelnumber_;
begingraph;
  entrytitle halign=center 'Vertical bar title';
  entryfootnote halign=left 'Vertical bar footnote';
  layout datalattice columnvar=_SEX / cellwidthmin=1 cellheightmin=1
                     rowgutter=3 columngutter=3 rowdatarange=unionall
                     row2datarange=unionall columndatarange=unionall
                     column2datarange=unionall
                     headerlabeldisplay=value;
    layout prototype;
      barchart x=_AGE / name='bar' stat=freq barwidth=0.85
               clusterwidth=0.85;
    endlayout;
  endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.CLASS template=sgdesign;
dynamic _AGE="AGE" _SEX="SEX";
run;
```

Template C

Comparing this new generated template with the original graph Template B, you will see that the multiple panels have been achieve using LAYOUT DATAPANEL around LAYOUT PROTOTYPE, instead of LAYOUT LATTICE around LAYOUT OVERLAY.

```
proc template;
define statgraph sgdesign;
dynamic _AGE _SEX2;
dynamic _panelnumber_;
begingraph;
  entrytitle halign=center 'Vertical bar title';
  entryfootnote halign=left 'Vertical bar footnote';
  layout datapanel classvars=(_SEX2) / cellwidthmin=1 rowgutter=3
                   columngutter=3 rowdatarange=unionall
                   row2datarange=unionall columndatarange=unionall
                   column2datarange=unionall headerlabeldisplay=value
                   rows=1 columns=2;
    layout prototype / ;
      barchart x=_AGE / name='bar' stat=freq barwidth=0.85
               clusterwidth=0.85;
    endlayout;
  endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.CLASS template=sgdesign;
dynamic _AGE="AGE" _SEX2="SEX";
run;
```

Template D

The generated template includes a LAYOUT LATTICE around two LAYOUT OVERLAY sections. Note that the GROUPDISPLAY= and CLUSTERWIDTH= parameters, and the PROC SGRENDER step are not available in SAS 9.2.

```
proc template;
define statgraph Graph;
dynamic _AGE _AGE2 _HEIGHT;
begingraph;
  entrytitle halign=center 'Panel title';
  entryfootnote halign=left 'Panel footnote';
  layout lattice / rowdatarange=data columndatarange=data rows=2
                   rowgutter=10 columngutter=10;
    layout overlay;
      barchart x=_AGE / name='bar' stat=freq groupdisplay=Cluster
               clusterwidth=1.0;
    endlayout;
    layout overlay;
      scatterplot x=_AGE2 y=_HEIGHT / name='scatter';
    endlayout;
  endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.CLASS template=Graph;
dynamic _AGE="AGE" _AGE2="AGE" _HEIGHT="HEIGHT";
run;
```

Template Syntax

What does DYNAMIC do?

If you are familiar with SAS macro programming, then you will notice that the DYNAMIC statement in a template provides a similar function to the list of parameters in parentheses supplied to a macro:

For example:

```
DYNAMIC _var1 _var2 _var3;
```

This is referenced in PROC SGRENDER as follows:

```
DYNAMIC _var1 = "age" _var2 = "height" _var3 = "sex";
```

The only major difference between macro parameters and DYNAMIC variables is that it is not possible to set default values for DYNAMIC variables.

How do you specify Titles and Footnotes?

Title and footnote parameters are a necessity for flexible templates. The generated template includes a single ENTRYTITLE statement.

For example:

```
ENTRYTITLE 'This is a title';
ENTRYFOOTNOTE 'This is a footnote';
```

This may not be sufficient in all cases, but, fortunately, any ENTRYTITLE and the corresponding ENTRYFOOTNOTE statement with no associated text are ignored, so a template can include the maximum reasonable number of both statements.

The following code will show two titles:

```
ENTRYTITLE 'Title A';
ENTRYTITLE;
ENTRYTITLE 'Title C';
```

It is still possible to insert a blank title by using an empty string, so the following code will show three titles, with the second title blank:

```
ENTRYTITLE 'Title A';
ENTRYTITLE ' ';
ENTRYTITLE 'Title C';
```

The default formatting of each of the text from these statements is controlled by the current ODS Style.

How do you specify Axes and Legends?

Unlike in traditional SAS/GRAPH, it is not usually necessary to specify axes when plotting with ODS Graphics, as the default settings based on the data are fairly good. However, when the graphics include more than one graph panel with a common axis, there may be a need to remove one of the axes and to keep the sizes of the graph panels consistent.

There are options for the LAYOUT statement to only show the common axes on the outside edge:

```
LAYOUT LATTICE / ROWDATARANGE = UNION;
                 /* common row axes */
LAYOUT LATTICE / COLUMNDATARANGE = UNION;
                 /* common column axes */
```

For secondary axes too:

```
LAYOUT LATTICE / ROW2DATARANGE = UNION;
                 /* common secondary row axes */
LAYOUT LATTICE / COLUMN2DATARANGE = UNION;
                 /* common secondary column axes */
```

COLUMNAXES and ROWAXES

COLUMNAXES and ROWAXES have the same function to replace column and row axes, respectively, from the existing layouts. They act like additional LAYOUT statements to hold COLUMNAXIS or ROWAXIS statements which specify the properties of the common axis to be displayed. Note that if there are multiple columns of panels, then a separate COLUMNAXIS statement can be specified for each column.

For example the following code will display the x-axis below a single column of panels with all the axis ticks, tick values and labels shown:

```
COLUMNAXES;
  COLUMNAXIS / DISPLAY = (LABEL LINE TICKS TICKVALUES);
ENDCOLUMNAXES;
```

An examples of COLUMNAXES can be found in “Enhancing a Template: Adding Labels to Points” on page 92.

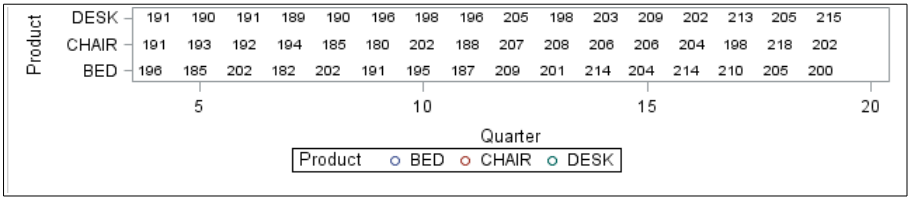
SIDEBAR

It is possible to specify SIDEBAR areas around the graph area using the ALIGN= option, with 4 available values TOP, LEFT, RIGHT and BOTTOM. These areas are located outside the axis areas. By default the contents are stretched to fill the entire width of the area, so the use of SPACEFILL=FALSE is recommended. A common use of these areas is to move DISCRETELEGEND statements out of the LAYOUT OVERLAY area to below all of the panels.

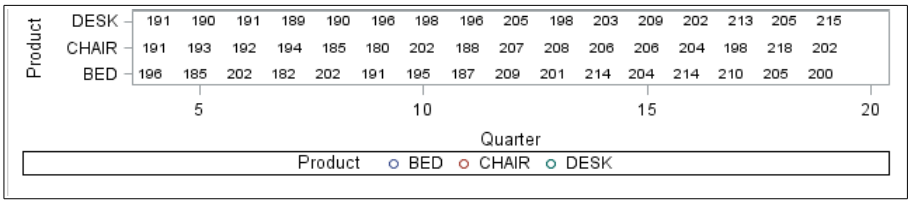
For example:

```
SIDEBAR / ALIGN = BOTTOM SPACEFILL = FALSE;
  DISCRETELEGEND "scatter" / TITLE = _grouplabel ORDER = ROWMAJOR
                 BORDER = TRUE BORDERATTRS = (COLOR = BLACK);
ENDSIDEBAR;
```

This generates the following legend:



If SPACEFILL=TRUE is used, which is the default setting, the generated legend will change to the following:



Another example of SIDEBAR can be found in “Enhancing a Template: Adding Labels to Points” on page 92.

What does IF do?

The IF statement can be used to conditionally include a single template statement, which can be used in this template, and is also followed by an ENDIF statement. The IF clause can include most Data Step functions, e.g. EXISTS() which checks whether a variable exists. Note that the IF clause is never followed by a semi-colon and utilizes standard SAS WHERE statement expressions:

```
IF (EXISTS(age) AND EXISTS(weight))
  SCATTERPLOT X = age Y = weight;
ENDIF;
```

IF can also be used to conditionally include multiple statements, like SCATTERPLOT and SERIESPLOT statements, where the first statement is effectively included in the IF construct. Both SCATTERPLOT and SERIESPLOT are included within the IF … ENDIF clause:

```
IF (EXISTS(age) AND EXISTS(weight))
  SCATTERPLOT X = age Y = weight;
  SERIESPLOT X = age Y = weight;
ENDIF;
```

There is also the option of an ELSE clause, which has a similar syntax to IF, but no ELSE IF is currently available. Here the statements used are either SCATTERPLOT and SERIESPLOT, or NEEDLEPLOT:

```
IF (EXISTS(_scatter))
  SCATTERPLOT X = age Y = weight;
  SERIESPLOT X = age Y = weight;
ELSE
  NEEDLEPLOT X = age Y = weight;
ENDIF;
```

More examples of IF clauses can be found in “Adding Conditional Features: Handling Missing Arguments” on page 81 and in “Adding Conditional Features: Optional Reference Lines” on page 95.

How to Create your own Templates

Graph Templates are not easy to simplify, so it is recommended that a Graph Template generator, e.g. ODS Graphics Designer, PROC SGPLOT or PROC SGSCATTER, be used to create a basic Graph Template, which can then be manually improved to fit your requirements.

It is important to remember when developing your own Graph Templates that they can only be applied to a single SAS data set, so the planning of the input data set can be almost as important as the design of the template itself. Therefore, if the Graph Template requires data from several different input data sets, then these data sets must be combined into a single data set before rendering. Note that missing values should not affect the graph's appearance, provided the data is sorted appropriately.

Graph templates allow the overlay of many different plots within the same axes. This helps to make the templates easier to generalize with parameters, rather than with hard-coded values.

Customizing PROC SGSCATTER Graphs

Generating a Simple Template with PROC SGSCATTER

PROC SGSCATTER is an ideal starting point to develop multi-panel graphs where the individual plots are similar, but not identical. The PLOT statement can be used to plot a series of different combinations of variables from the same data set as scatter plots in separate panels. The following example demonstrates the basic code:

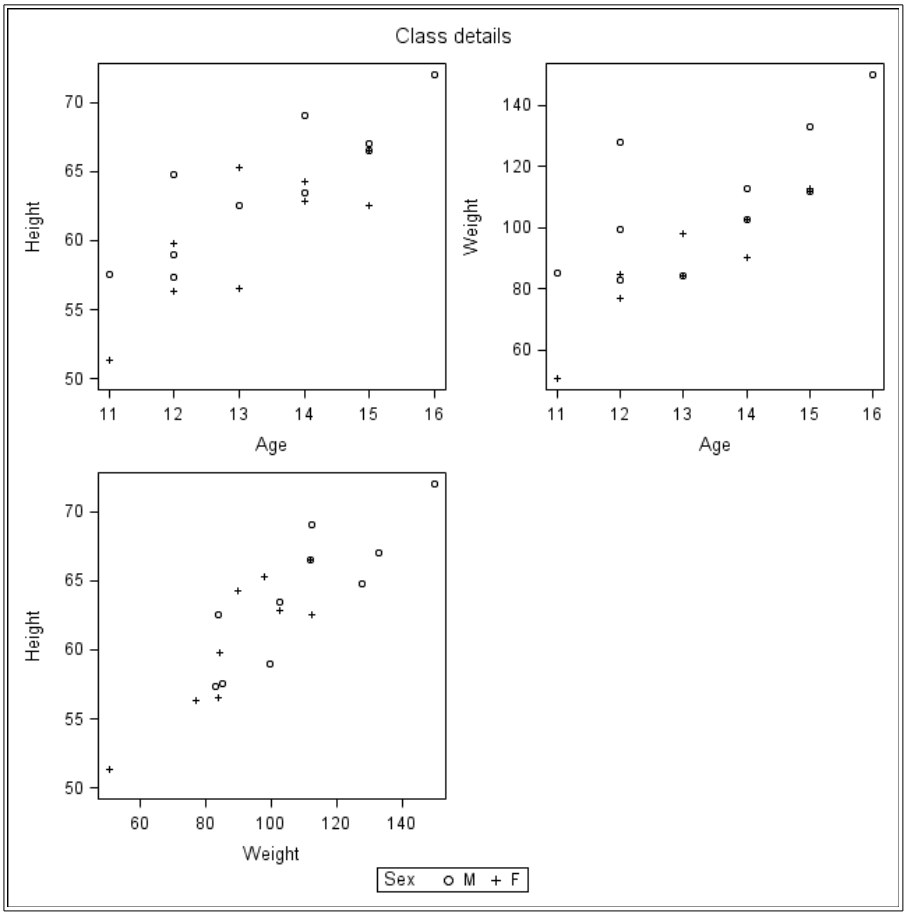
```
PROC SGSCATTER DATA = sashelp.class
               TMPLOUT = 'sgscatter.sas';
  TITLE 'Class details';
  PLOT (height weight) * age height * weight / GROUP = sex;
RUN;
```

Note that there will be three scatter plots, height*age, weight*age and height*weight. By default PROC SGSCATTER arranges plots in a 2x2 grid, so the three plots should appear in three out of the four cells. The GROUP= option is equivalent to the **y*x=group** syntax from PROC GPLOT, but applied to every plot.

The generated template code (from SAS 9.3, but only DESIGNWIDTH= and DESIGNHEIGHT= options are missing in SAS 9.2), stored in **sgscatter.sas**, is as follows:

```
proc template;
  define statgraph sgscatter;
    begingraph / designwidth=640 designheight=640;
      EntryTitle "Class details" /;
      layout gridded;
        layout lattice / pad=(top=5) rowgutter=10
                columngutter=10 columns=2;
          ScatterPlot X=Age Y=Height /
                      primary=true Group=Sex NAME="PLOT";
          ScatterPlot X=Age Y=Weight / primary=true Group=Sex;
          ScatterPlot X=Weight Y=Height /
                      primary=true Group=Sex;
        endlayout;
        DiscreteLegend "PLOT" / order=rowmajor title="Sex";
      endlayout;
    endgraph;
  end;
run;
```

The graph, generated with the SAS-supplied ODS style Journal, which has been designed for publications, is shown below:



Adding DYNAMIC Parameters to the Template

Having reformatted the generated template to match the other SAS code in this chapter, the generated template is as follows, with the hard-coded items to be converted to parameters underlined:

```
PROC TEMPLATE;
  DEFINE STATGRAPH sgscatter;
    BEGINGRAPH / DESIGNWIDTH = 640 DESIGNHEIGHT = 640;
      ENTRYTITLE "Class details" /;
      LAYOUT GRIDDED;
        LAYOUT LATTICE /
               PAD = (TOP = 5) ROWGUTTER = 10
               COLUMNGUTTER = 10 COLUMNS=2;
          SCATTERPLOT X = Age Y = Height /
                      PRIMARY = TRUE GROUP = Sex NAME = "plot";
          SCATTERPLOT X = Age Y = Weight /
                      PRIMARY = TRUE GROUP = Sex;
          SCATTERPLOT X = Weight Y = Height /
                      PRIMARY = TRUE GROUP = Sex;
        ENDLAYOUT;
        DISCRETELEGEND "plot" / ORDER = ROWMAJOR TITLE = "Sex";
      ENDLAYOUT;
    ENDGRAPH;
  END;
RUN;
```

The aim of adding parameters to a template is to allow it to be used with other data sets rather than the data set used during its development. The data set variables, Age, Height, Weight and Sex, can be replaced with **_var1**, **_var2**, **_var3** and **_group**. The legend title "Sex" can also be converted to a parameter **_grouplabel**. All these new parameters are specified on the DYNAMIC statement:

```
PROC TEMPLATE;
  DEFINE STATGRAPH sgscatter;
    DYNAMIC _var1 _var2 _var3 _group _grouplabel;
    BEGINGRAPH / DESIGNWIDTH = 640 DESIGNHEIGHT = 640;
      ENTRYTITLE "Class details" /;
      LAYOUT GRIDDED;
        LAYOUT LATTICE /
               PAD = (TOP = 5) ROWGUTTER = 10
               COLUMNGUTTER = 10 COLUMNS = 2;
          SCATTERPLOT X = _var1 Y = _var2 /
                      PRIMARY = TRUE GROUP = _group
                      NAME = "plot";
          SCATTERPLOT X = _var1 Y = _var3 /
                      PRIMARY = TRUE GROUP = _group;
          SCATTERPLOT X = _var3 Y = _var2 /
                      PRIMARY = TRUE GROUP = _group;
        ENDLAYOUT;
        DISCRETELEGEND "plot" / ORDER = ROWMAJOR
                        TITLE = _grouplabel;
      ENDLAYOUT;
    ENDGRAPH;
  END;
RUN;
```

Title and footnote parameters are a necessity for flexible templates. In this updated template below there are three of each of the ENTRYTITLE and ENTRYFOOTNOTE statements, with parameters for each one.

This template is being updated to increase its flexibility, so the removal of any hard-coded values is encouraged. Therefore the DESIGNWIDTH= and DESIGNHEIGHT= should be removed from the BEGINGRAPH statement.

Replacing the Template Name

Finally the ODS Graphics Designer and each of the procedures give their generated templates a name. The procedures each use their own specific name, which, without alteration would restrict the number of different templates available for use. It is, therefore, strongly recommended that the default template name, in this case **sgscatter**, be changed to something a little more descriptive, e.g. **scatter_2x2**.

```
PROC TEMPLATE;
  DEFINE STATGRAPH scatter_2x2;
    DYNAMIC _var1 _var2 _var3 _group _grouplabel
            _title1 _title2 _title3
            _footnote1 _footnote2 _footnote3;
    BEGINGRAPH /;
      ENTRYTITLE _title1 /;
      ENTRYTITLE _title2 /;
      ENTRYTITLE _title3 /;
      LAYOUT GRIDDED;
        LAYOUT LATTICE /
               PAD = (TOP = 5) ROWGUTTER = 10
               COLUMNGUTTER = 10 COLUMNS = 2;
          SCATTERPLOT X = _var1 Y = _var2 /
                      PRIMARY = TRUE GROUP = _group
                      NAME = "plot";
          SCATTERPLOT X = _var1 Y = _var3 /
                      PRIMARY = TRUE GROUP = _group;
          SCATTERPLOT X = _var3 Y = _var2 /
                      PRIMARY = TRUE GROUP = _group;
        ENDLAYOUT;
        DISCRETELEGEND "plot" / ORDER = ROWMAJOR
                        TITLE = _grouplabel;
      ENDLAYOUT;
      ENTRYFOOTNOTE _footnote1 /;
      ENTRYFOOTNOTE _footnote2 /;
      ENTRYFOOTNOTE _footnote3 /;
    ENDGRAPH;
  END;
RUN;
```

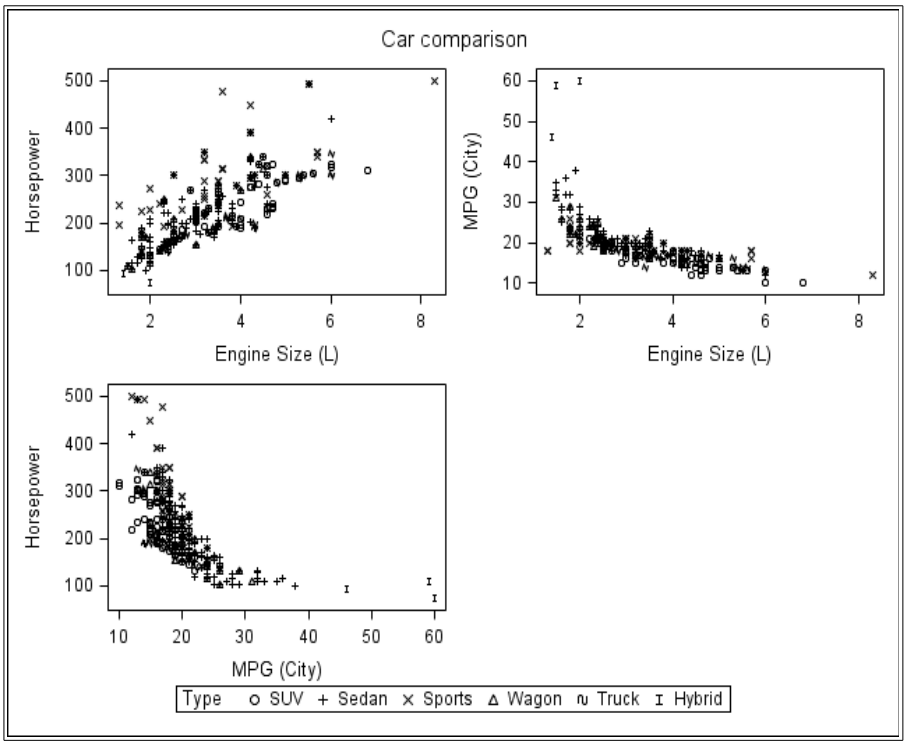
Having converted the generated template to use parameters, it is a good idea to test it to make sure it will still create the intended graph from **sashelp.class**.

```
PROC SGRENDER DATA = sashelp.class
              TEMPLATE = 'scatter_2x2';
  DYNAMIC _var1 = 'Age' _var2 = 'Height' _var3 = 'Weight'
          _group = 'Sex' _grouplabel = 'Sex'
          _title1 = 'Class details';
RUN;
```

The template is now ready to be used with another SAS data set, e.g. **sashelp.cars**:

```
PROC SGRENDER DATA = sashelp.cars
              TEMPLATE = 'scatter_2x2';
  DYNAMIC _var1 = 'EngineSize' _var2 = 'Horsepower'
          _var3 = 'MPG_City'
          _group = 'Type' _grouplabel = 'Type'
          _title1 = 'Car comparison';
RUN;
```

The graph, generated with the ODS style Journal, is shown below. Note that the variable labels in **sashelp.cars** are automatically used as the individual plot axis labels:



Enhancing a Template: Adding a New Graph

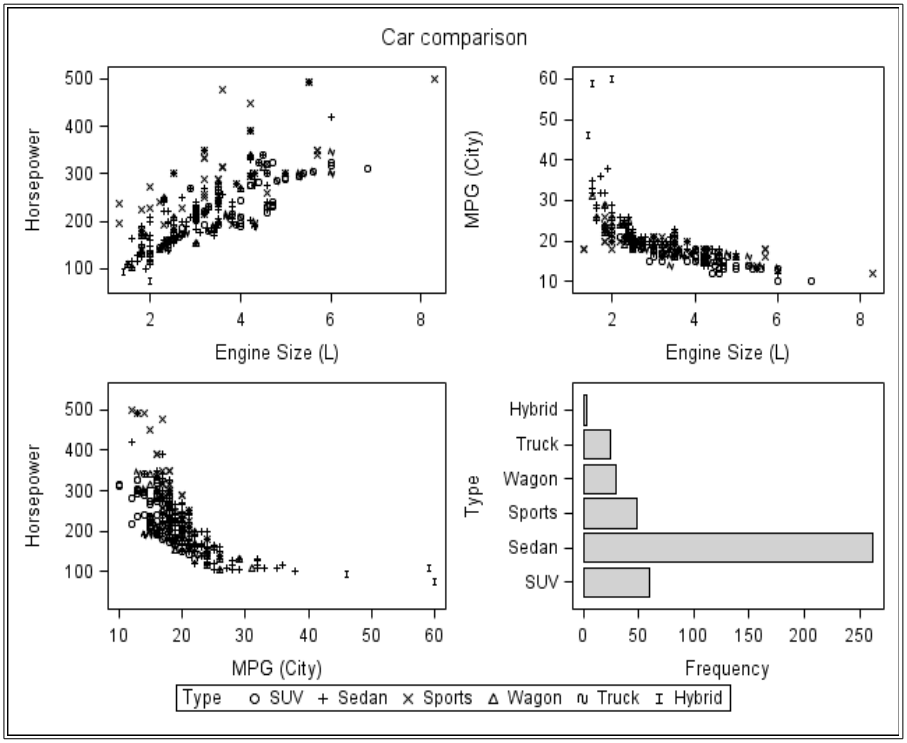
There are currently four panels, four variables, but only three plots, in this template. There is only a list of values in the legend for the group variable, so a useful enhancement could be to include a bar chart of the group frequencies:

```
PROC TEMPLATE;
  DEFINE STATGRAPH scatter_2x2_bar;
    DYNAMIC _var1 _var2 _var3 _group _grouplabel
            _title1 _title2 _title3
            _footnote1 _footnote2 _footnote3;
    BEGINGRAPH /;
      ENTRYTITLE _title1 /;
      ENTRYTITLE _title2 /;
      ENTRYTITLE _title3 /;
      LAYOUT GRIDDED;
        LAYOUT LATTICE /
               PAD = (TOP = 5) ROWGUTTER = 10
               COLUMNGUTTER = 10 COLUMNS = 2;
          SCATTERPLOT X = _var1 Y = _var2 /
                      PRIMARY = TRUE GROUP = _group NAME = "plot";
          SCATTERPLOT X = _var1 Y = _var3 /
                      PRIMARY = TRUE GROUP = _group;
          SCATTERPLOT X = _var3 Y = _var2 /
                      PRIMARY = TRUE GROUP = _group;
          BARCHART X = _group /
                   PRIMARY = TRUE STAT = FREQ ORIENT = HORIZONTAL;
        ENDLAYOUT;
        DISCRETELEGEND "plot" / ORDER = ROWMAJOR TITLE = _grouplabel;
      ENDLAYOUT;
      ENTRYFOOTNOTE _footnote1 /;
      ENTRYFOOTNOTE _footnote2 /;
      ENTRYFOOTNOTE _footnote3 /;
    ENDGRAPH;
  END;
RUN;
```

The template is now ready for use:

```
PROC SGRENDER DATA = sashelp.cars
              TEMPLATE = 'scatter_2x2_bar';
  DYNAMIC _var1 = 'EngineSize' _var2 = 'Horsepower' _var3 = 'MPG_City'
          _group = 'Type' _grouplabel = 'Type'
          _title1 = 'Car comparison';
RUN;
```

The graph, generated with the ODS style Journal, is shown below.



Adding Conditional Features: Handling Missing Arguments

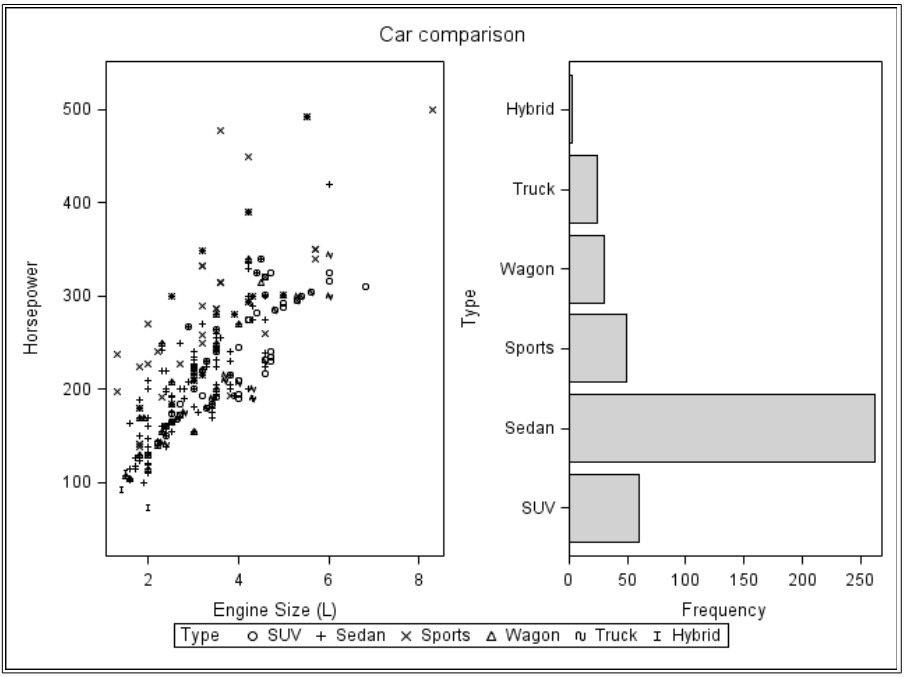
As it stands this template is designed to plot information from 4 variables in four plots, three scatter plots and one bar chart. However, there may not always be all three variables used in the scatter plots, so how does this template cope with just two scatter plot variables?

```
PROC SGRENDER DATA = sashelp.cars
              TEMPLATE = 'scatter_2x2_bar';
  DYNAMIC _var1 = 'EngineSize' _var2 = 'Horsepower'
          _group = 'Type' _grouplabel = 'Type'
          _title1 = 'Car comparison';
RUN;
```

Unfortunately, this gives the following log warnings:

```
WARNING: The SCATTERPLOT statement will not be drawn because one or
more of the required arguments were not supplied.
WARNING: The SCATTERPLOT statement will not be drawn because one or
more of the required arguments were not supplied.
```

However, the graph below is still created:



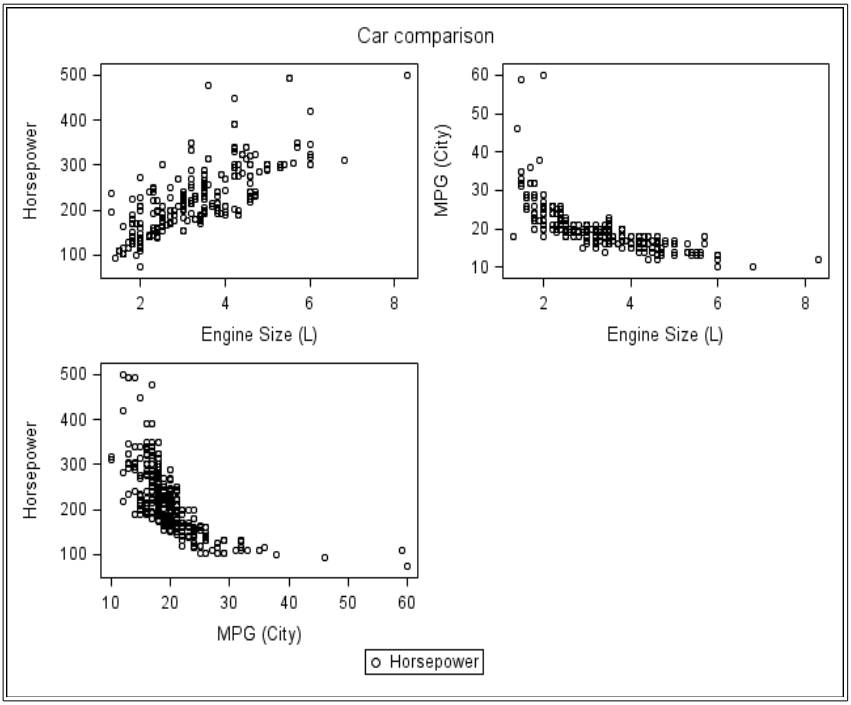
If you omit the _group= and _grouplabel= values:

```
PROC SGRENDER DATA = sashelp.cars
              TEMPLATE = 'scatter_2x2_bar';
  DYNAMIC _var1 = 'EngineSize' _var2 = 'Horsepower' _var3 = 'MPG_City'
          _title1 = 'Car comparison';
RUN;
```

which produces the following log warning:

```
WARNING: The BARCHART statement will not be drawn because one or more
of the required arguments were not supplied.
```

However, the graph below is created, with the unexpected legend label:



All these log warnings and unexpected legend labels can be prevented by using conditional processing statements in the template. The IF statement can be used to conditionally include a single template statement, which can be used in this template. However, IF cannot be used to conditionally include multiple statements, like LAYOUT clauses, so its use can be limited.

To prevent the warning messages, the plot statements can be included only when all the required variables are present. Note that the IF clause is never followed by a semi-colon and utilizes standard SAS WHERE statement expressions:

```
IF (EXISTS(_var1) AND EXISTS(_var2))
    SCATTERPLOT X = _var1 Y = _var2 /
                PRIMARY = TRUE GROUP = _group NAME = "plot";
ENDIF;
IF (EXISTS(_var1) AND EXISTS(_var3))
    SCATTERPLOT X = _var1 Y = _var3 /
                PRIMARY = TRUE GROUP = _group;
ENDIF;
IF (EXISTS(_var3) AND EXISTS(_var2))
    SCATTERPLOT X = _var3 Y = _var2 /
                PRIMARY = TRUE GROUP = _group;
ENDIF;
IF (EXISTS(_group))
    BARCHART X = _group /
             PRIMARY = TRUE STAT = FREQ ORIENT = HORIZONTAL;
ENDIF;
IF (EXISTS(_group))
    DISCRETELEGEND "plot" / ORDER = ROWMAJOR TITLE = _grouplabel;
ENDIF;
```

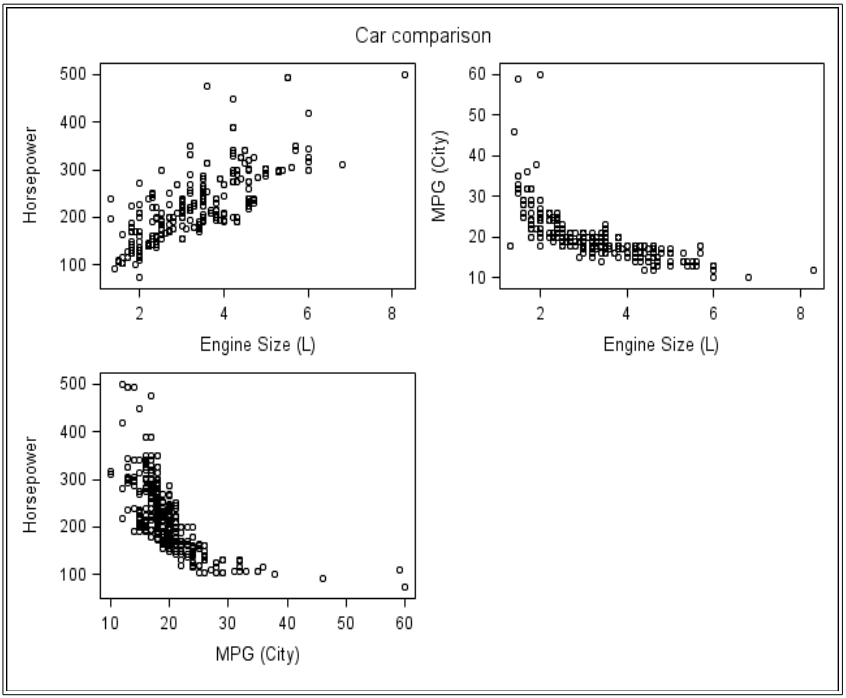
The EXISTS() function also works well here because when a DYNAMIC parameter is not populated by PROC SGRENDER it simply "drops out" and ceases to exist. This produces a more robust template:

```
PROC TEMPLATE;
  DEFINE STATGRAPH scatter_2x2_bar_robust;
    DYNAMIC _var1 _var2 _var3 _group _grouplabel
            _title1 _title2 _title3
            _footnote1 _footnote2 _footnote3;
    BEGINGRAPH /;
      ENTRYTITLE _title1 /;
      ENTRYTITLE _title2 /;
      ENTRYTITLE _title3 /;
      LAYOUT GRIDDED;
        LAYOUT LATTICE /
               PAD = (TOP = 5) ROWGUTTER = 10
               COLUMNGUTTER = 10 COLUMNS = 2;
          IF (EXISTS(_var1) AND EXISTS(_var2))
            SCATTERPLOT X = _var1 Y = _var2 /
                        PRIMARY = TRUE GROUP = _group
                        NAME = "plot";
          ENDIF;
          IF (EXISTS(_var1) AND EXISTS(_var3))
            SCATTERPLOT X = _var1 Y = _var3 /
                        PRIMARY = TRUE GROUP = _group;
          ENDIF;
          IF (EXISTS(_var3) AND EXISTS(_var2))
            SCATTERPLOT X=_var3 Y=_var2 /
                        PRIMARY = TRUE GROUP = _group;
          ENDIF;
          IF (EXISTS(_group))
            BARCHART X = _group /
                     PRIMARY = TRUE STAT = FREQ
                     ORIENT = HORIZONTAL;
          ENDIF;
        ENDLAYOUT;
        IF (EXISTS(_group))
          DISCRETELEGEND "plot" /
                         ORDER = ROWMAJOR TITLE = _grouplabel;
        ENDIF;
      ENDLAYOUT;
      ENTRYFOOTNOTE _footnote1 /;
      ENTRYFOOTNOTE _footnote2 /;
      ENTRYFOOTNOTE _footnote3 /;
    ENDGRAPH;
  END;
RUN;
```

The PROC SGRENDER code below uses the “robust” template:

```
PROC SGRENDER DATA = sashelp.cars
              TEMPLATE = 'scatter_2x2_bar_robust';
  DYNAMIC _var1 = 'EngineSize' _var2 = 'Horsepower' _var3 = 'MPG_City'
          _title1 = 'Car comparison';
RUN;
```

Now the following graph is created, without the unnecessary legend and without any log warnings:



However, the previous PROC SGRENDER code, with the full set of parameters will still produce the full graph, as before.

Customizing PROC SGPLOT Templates

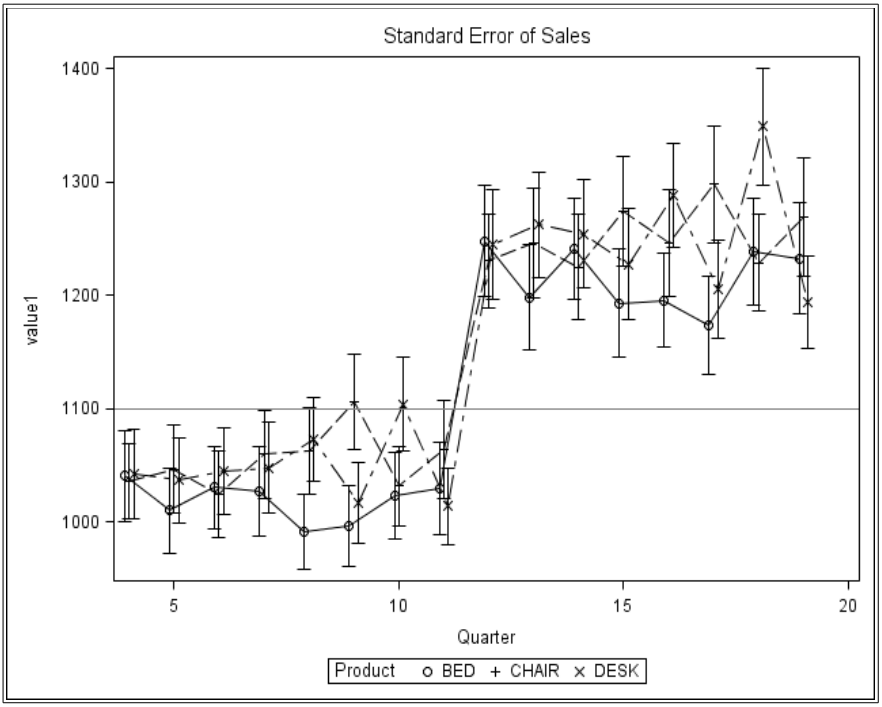
Generating a Simple Template with PROC SGPLOT

PROC SGPLOT is designed to produce single plots. Both PROC SGPANEL and PROC SGSCATTER can produce single plots, but that is not what they are designed to do, and the options available in these procedures are more limited. All three procedures use the current ODS Style to supply information on colour, fonts, sizes and patterns to display the plot being generated. This may not be exactly what is required, so any customization can override these defaults.

The code below uses the plotdata_ods data set created earlier to generate one or more overlaid lines with error bars at each point and a horizontal reference line:

```
PROC SGPLOT DATA = plotdata_ods
            TMPLOUT = 'sgplot.sas';
  TITLE 'Standard Error of Sales';
  SCATTER X = visitnum Y = value1 /
          GROUP = product YERRORLOWER = value1_lower
          YERRORUPPER = value1_upper;
  SERIES X = visitnum Y = value1 / GROUP = product;
  REFLINE 1100 / AXIS = Y;
RUN;
```

This produces the following graph:



The following template is also generated, which is what will be used as the starting point for the customization:

```
proc template;
  define statgraph sgplot;
    begingraph /;
      EntryTitle "Standard Error of Sales" /;
      layout overlay;
        ScatterPlot X=visitnum Y=value1 /
          primary=true Group=PRODUCT
          YErrorUpper=value1_upper YErrorLower=value1_lower
          LegendLabel="value1" NAME="SCATTER";
        SeriesPlot X=visitnum Y=value1 /
          Group=PRODUCT LegendLabel="value1"
          NAME="SERIES";
        ReferenceLine y=1100 / clip=true;
        DiscreteLegend "SCATTER"/ title="Product";
      endlayout;
    endgraph;
  end;
run;
```

Adding DYNAMIC Parameters to the Template

Having reformatted the generated template to match the other SAS code in this chapter, the generated template is as follows with the hard-coded items to be converted to parameters underlined:

```
PROC TEMPLATE;
  DEFINE STATGRAPH sgplot;
    BEGINGRAPH /;
      ENTRYTITLE "Standard Error of Sales" /;
      LAYOUT OVERLAY;
        SCATTERPLOT X = visitnum Y = value1 /
                    PRIMARY = TRUE GROUP = product
                    YERRORUPPER = value1_upper
                    YERRORLOWER = value1_lower
                    LEGENDLABEL = "value1" NAME = "scatter";
        SERIESPLOT X = visitnum Y = value1 /
                   GROUP = product LEGENDLABEL = "value1"
                   NAME = "series";
        REFERENCELINE Y = 1100 / CLIP = TRUE;
        DISCRETELEGEND "scatter"/ TITLE = "Product";
      ENDLAYOUT;
    ENDGRAPH;
  END;
RUN;
```

The aim of adding parameters to a template is to allow its use with different data sets to that used during its development. The data set variables, Visitnum, Value1, Value1_lower, Value1_upper and Product, can be replaced with **_xvar**, **_yvar1**, **_yvar1_lower**, **_yvar1_upper** and **_group** The legend title "Product" can also be converted to a parameter **_grouplabel**, and also the reference line value to **_yintercepta**. All these new parameters are specified on the DYNAMIC statement:

```
PROC TEMPLATE;
  DEFINE STATGRAPH sgplot;
    DYNAMIC _xvar _yvar1 _yvar1_lower _yvar1_upper
            _group _grouplabel
            _yintercepta;
    BEGINGRAPH /;
      ENTRYTITLE "Standard Error of Sales" /;
      LAYOUT OVERLAY;
        SCATTERPLOT X = _xvar Y = _yvar1 /
                    PRIMARY = TRUE GROUP = _group
                    YERRORUPPER = _yvar1_upper
                    YERRORLOWER = _yvar1_lower
                    NAME = "scatter";
        SERIESPLOT X = _ xvar Y = _yvar1 /
                   GROUP = _group NAME="series";
        REFERENCELINE Y = _yintercepta / CLIP = TRUE;
        DISCRETELEGEND "scatter" / TITLE = _grouplabel;
      ENDLAYOUT;
    ENDGRAPH;
  END;
RUN;
```

In the same way as the template from PROC SGSCATTER had ENTRYTITLE and ENTRYFOOTNOTE statements added, this template will need a similar number of titles and footnotes available to users. It is also reasonable to make this new template unique by updating the template name, e.g. to **sgplot_dynamic**.

```
PROC TEMPLATE;
  DEFINE STATGRAPH sgplot_dynamic;
    DYNAMIC _xvar _yvar1 _yvar1_lower _yvar1_upper
            _group _grouplabel
            _yintercepta
            _title1 _title2 _title3
            _footnote1 _footnote2 _footnote3;
    BEGINGRAPH /;
      ENTRYTITLE _title1 /;
      ENTRYTITLE _title2 /;
      ENTRYTITLE _title3 /;
      LAYOUT OVERLAY;
        SCATTERPLOT X = _xvar Y = _yvar1 /
                    PRIMARY = TRUE GROUP = _group
                    YERRORUPPER = _yvar1_upper
                    YERRORLOWER = _yvar1_lower
                    NAME = "scatter";
        SERIESPLOT X = _xvar Y = _yvar1 /
                   GROUP = _group NAME = "series";
        REFERENCELINE Y = _yintercepta / CLIP = TRUE;
        DISCRETELEGEND "scatter" / TITLE = _grouplabel;
      ENDLAYOUT;
      ENTRYFOOTNOTE _footnote1 /;
      ENTRYFOOTNOTE _footnote2 /;
      ENTRYFOOTNOTE _footnote3 /;
    ENDGRAPH;
  END;
RUN;
```

Having converted the generated template to use parameters, it is a good idea to test it to make sure it will still create the intended graph from **plotdata_ods**.

```
PROC SGRENDER DATA = plotdata_ods
              TEMPLATE = 'sgplot_dynamic';
  DYNAMIC _xvar = 'Visitnum' _yvar1 = 'Value1'
          _yvar1_lower = 'Value1_lower'
          _yvar1_upper = 'Value1_upper'
          _group = 'Product' _grouplabel = 'Product'
          _yintercepta = 1100
          _title1 = 'Standard Error of Sales';
RUN;
```

Enhancing a Template: Adding Labels to Points

There is currently no information about how many data values contribute to each point on the graph. There is, however, a variable in **plotdata_ods** (**ccount**) that contains this information in character format, so a useful enhancement could be to include these values somewhere.

So that the values are aligned with the points on the graph, a separate graph panel with a common x-axis scale can be drawn below the main graph, and the text values plotted as data points in a row per group. The MARKERCHARACTER= and MARKERCHARACTERATTRS= options of the SCATTERPLOT statement make this very easy to achieve. The LAYOUT LATTICE statement determines the relative height of each panel in the lattice, and the **_nvar1** parameter provides the value to be plotted in the new panel.

Finally, in order to position the legend below the new panel, the DISCRETELEGEND statement is moved into a SIDEBAR located at the bottom of the graph, with the SPACEFILL=FALSE option so the legend is not stretched across the whole width of the SIDEBAR:

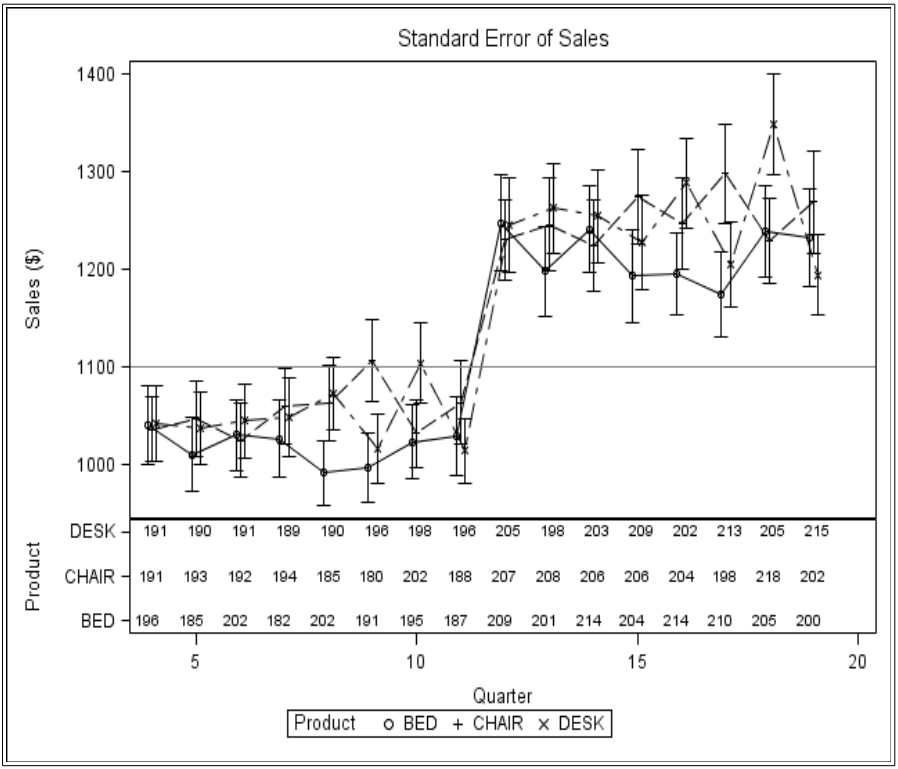
```
PROC TEMPLATE;
  DEFINE STATGRAPH sgplot_marker;
    DYNAMIC _xvar _yvar1 _yvar1_lower _yvar1_upper
            _group _grouplabel
            _yintercepta
            _nvar1
            _title1 _title2 _title3
            _footnote1 _footnote2 _footnote3;
    BEGINGRAPH /;
      ENTRYTITLE _title1 /;
      ENTRYTITLE _title2 /;
      ENTRYTITLE _title3 /;
      LAYOUT LATTICE /
             COLUMNS = 1 ROWS = 2 COLUMNDATARANGE = UNION
             ROWWEIGHTS = (0.85 0.15);
        LAYOUT OVERLAY;
          SCATTERPLOT X = _xvar Y = _yvar1 /
                      PRIMARY = TRUE GROUP = _group
                      YERRORUPPER = _yvar1_upper
                      YERRORLOWER = _yvar1_lower
                      NAME = "scatter";
          SERIESPLOT X = _xvar Y = _yvar1 /
                     GROUP = _group NAME = "series";
          REFERENCELINE Y = _yintercepta / CLIP = TRUE;
        ENDLAYOUT;
```

```
        LAYOUT OVERLAY;
          SCATTERPLOT X = _xvar Y = _group /
                      PRIMARY = TRUE GROUP = _group
                      MARKERCHARACTERATTRS = (COLOR = BLACK)
                      MARKERCHARACTER = _nvar1;
        ENDLAYOUT;
        COLUMNAXES;
          COLUMNAXIS / DISPLAY = (LABEL LINE TICKS TICKVALUES);
        ENDCOLUMNAXES;
        SIDEBAR / ALIGN = BOTTOM SPACEFILL = FALSE;
          DISCRETELEGEND "scatter" / TITLE = _grouplabel
          ORDER = ROWMAJOR;
        ENDSIDEBAR;
      ENDLAYOUT;
      ENTRYFOOTNOTE _footnote1 /;
      ENTRYFOOTNOTE _footnote2 /;
      ENTRYFOOTNOTE _footnote3 /;
    ENDGRAPH;
  END;
RUN;
```

The template is now ready for use:

```
PROC SGRENDER DATA = plotdata_ods
              TEMPLATE = 'sgplot_marker';
  DYNAMIC _xvar = 'Visitnum' _yvar1 = 'Value1'
          _yvar1_lower = 'Value1_lower'
          _yvar1_upper = 'Value1_upper'
          _group = 'Product' _grouplabel = 'Product'
          _yintercepta = 1100
          _nvar1 = 'ccount'
          _title1 = 'Standard Error of Sales';
RUN;
```

It generates the following graph:



Adding Conditional Features: Optional Reference Lines

As it stands, this template will always generate a single reference line. However, there may be requirements for two, three, or even no, reference lines, so the IF statement can be used to provide this flexibility by adding two more reference line parameters **_yinterceptb** and **_yinterceptc**, and making all the reference lines dependent on values being specified.

This produces a more robust template:

```
PROC TEMPLATE;
  DEFINE STATGRAPH sgplot_count;
    DYNAMIC _xvar _yvar1 _yvar1_lower _yvar1_upper
            _group _grouplabel
            _yintercepta _yinterceptb _yinterceptc
            _nvar1
            _title1 _title2 _title3
            _footnote1 _footnote2 _footnote3;
    BEGINGRAPH /;
      ENTRYTITLE _title1 /;
      ENTRYTITLE _title2 /;
      ENTRYTITLE _title3 /;
      LAYOUT LATTICE /
             COLUMNS = 1 ROWS = 2 COLUMNDATARANGE = UNION
             ROWWEIGHTS = (0.85 0.15);
        LAYOUT OVERLAY;
          SCATTERPLOT X = _xvar Y = _yvar1 /
                      PRIMARY = TRUE GROUP = _group
                      YERRORUPPER = _yvar1_upper
                      YERRORLOWER = _yvar1_lower
                      NAME = "scatter";
          SERIESPLOT X = _xvar Y = _yvar1 /
                     GROUP = _group NAME = "series";
          IF (EXISTS(_yintercepta))
            REFERENCELINE Y = _yintercepta / CLIP = TRUE;
          ENDIF;
          IF (EXISTS(_yintercepta))
            REFERENCELINE Y = _yinterceptb / CLIP = TRUE;
          ENDIF;
          IF (EXISTS(_yintercepta))
            REFERENCELINE Y = _yinterceptc / CLIP = TRUE;
          ENDIF;
        ENDLAYOUT;
        LAYOUT OVERLAY;
          SCATTERPLOT X = _xvar Y = _group /
                      PRIMARY = TRUE GROUP = _group
                      MARKERCHARACTERATTRS = (COLOR = BLACK)
                      MARKERCHARACTER = _nvar1;
        ENDLAYOUT;
```

```
        COLUMNAXES;
          COLUMNAXIS / DISPLAY = (LABEL LINE TICKS TICKVALUES);
        ENDCOLUMNAXES;
        SIDEBAR / ALIGN = BOTTOM SPACEFILL = FALSE;
          DISCRETELEGEND "scatter" / TITLE = _grouplabel
                         ORDER = ROWMAJOR;
        ENDSIDEBAR;
      ENDLAYOUT;
      ENTRYFOOTNOTE _footnote1 /;
      ENTRYFOOTNOTE _footnote2 /;
      ENTRYFOOTNOTE _footnote3 /;
    ENDGRAPH;
  END;
RUN;
```

Therefore, the previous PROC SGRENDER code, with the same set of parameters, will still produce the same graph.

Other Useful Features of ODS Graphics

The following features of ODS Graphics are not directly related to creating Graph Templates themselves, but they will be useful for producing high quality graphs.

ODS GRAPHICS Statement

The ODS GRAPHICS statement is used to switch the output of ODS Graphics on and off, but it can also be used to specify the default features of any graphs produced. The basic syntax is as follows:

```
ODS GRAPHICS {ON} {/ option(s)};

ODS GRAPHICS OFF;
```

Options include:

Option	***Usage***
IMAGEFMT Specify the output format used to generate image or vector graphic files, e.g. IMAGEFMT=PNG.	The default is STATIC. See below for version-specific information.
IMAGEFILE Specify the base image filename, e.g. IMAGEFILE="C:\temp\image".	The default name is the output object, the default folder is the current folder, and the suffix is set by IMAGEFMT.
HEIGHT Specify the height of any graph, e.g. HEIGHT=8cm.	The default is the value of the SAS registry entry "ODS > STATISTICAL GRAPHICS > Design Height" or the value of the DESIGNHEIGHT= option in a STATGRAPH template.
WIDTH Specify the width of any graph, e.g. WIDTH=10cm.	The default is the value of the SAS registry entry "ODS > STATISTICAL GRAPHICS > Design Width" or the value of the DESIGNWIDTH= option in a STATGRAPH template.

SAS 9.2

The ODS GRAPHICS ON statement is not required before any of the “SG” procedures, PROC SGPLOT, PROC SGPANEL, PROC SGSCATTER and PROC SGRENDER, but is required if the Data Step is used to render data with Graph Templates, or if you want to generate ODS Graphics from other SAS procedures.

Output Destination	***Supported Image File Types: IMAGEFMT=***
HTML	PNG (default), GIF, JPEG, JPG
LISTING	PNG (default), BMP, DIB, EMF, EPSI, GIF, JFIF, JPEG, JPG, PBM, PDF, PS, SASEMF, STATIC, TIFF, WMF
LATEX	PS (default), EPSI, GIF, PNG, PDF, JPG
PDF, PCL (PRINTER) and PS (PRINTER)	PNG (default), JPEG, JPG, GIF
RTF	PNG (default), JPEG, JPG, JFIF
Markup Tagsets	All markup family tagsets have the default value built in.

SAS 9.3

The ODS GRAPHICS ON statement is no longer required in the interactive environments of SAS 9.3 (including Enterprise Guide) on Windows, UNIX and Linux platforms, as it is automatically switched on by default whenever you start an interactive SAS session. All batch SAS sessions, as well as interactive SAS sessions on other platforms, still behave the same way as in SAS 9.2, unless the SAS options relating to the default behaviour have been changed.

The documentation only mentions the OUTPUTFMT= option for specifying the output image file format, but IMAGEFMT= will still work as an alternative, so there is no need to change your existing programs yet.

Output Destination	***Supported Image File Types: OUTPUTFMT=***
HTML	PNG (default), GIF, JPEG, JPG, PBM, SVG, EMF, BMP
LISTING	PNG (default), BMP, DIB, EMF, EPSI, GIF, JFIF, JPEG, JPG, PBM, PDF, PS, SASEMF, STATIC, TIFF, WMF, XBM, XPM, PSL, SVG
LATEX	PS (default), EPSI, GIF, PNG, PDF, JPG, PSL, EPS, EMF
PDF and PCL (PRINTER)	SVG (default), JPEG, JPG, GIF, PSL, EPS, EPSI, PDF, PCL, PNG, EMF
PS (PRINTER)	PNG (default), JPEG, JPG, GIF, PSL, EPS, EPSI, PDF, PCL, EMF
RTF	PNG (default), JPEG, JPG, JFIF, EMF
Markup Tagsets	All markup family tagsets have the default value built in.

Recommended Reading

Books

- “SAS(R) 9.2 Output Delivery System: User's Guide”, SAS Publishing
- “SAS(R) 9.3 Output Delivery System: User's Guide, Second Edition”, SAS Publishing
- “Base SAS(R) 9.2 Procedures Guide: Statistical Procedures, Third Edition”, SAS Publishing
- “SAS(R) 9.3 ODS Graphics: Procedures Guide, Third Edition”, SAS Publishing
- “SAS/GRAPH(R) 9.2: Graph Template Language Reference, Second Edition”, SAS Publishing
- “SAS(R) 9.3 Graph Template Language: Reference, Third Edition”, SAS Publishing
- “Saving Time and Money Using SAS”, Philip R Holland, SAS Publishing, 2007
- “Power User's Guide to SAS Programming”, Philip R Holland, Lulu.com, 2008
- “PROC TEMPLATE Made Easy”, Kevin D Smith, SAS Publishing, 2013

Web Links

- All SAS code used this book can be downloaded from:
  www.hollandnumerics.com/books/Power_Users_Guide_to_Templates.htm

- Graph Templates for Forest Plots:
  www.squidoo.com/sas92_graph_templates

- Graph Templates for Line Plots:
  www.squidoo.com/sas92_graph_templates2

- Updating SAS-related eBook apps for Android/webOS phones and tablets, and Chrome/Chromium browsers:
  www.hollandnumerics.com/apps

Conference Papers

- "Using Just SAS 9.1 to Draw Trellis Graphs", Philip R Holland, PhUSE (Dublin, Eire), 2006

- "Standard Graph Templates", Philip R Holland, PhUSE (Lisbon, Portugal), 2007

- "GTL (Graphics Template Language) in SAS 9.2", Philip R Holland, PhUSE (Manchester, UK), 2008

- "Using the ODS Graphics Designer to Create Your Own Templates in SAS 9.2", Philip R Holland, PhUSE (Basel, Switzerland), 2009

- "Developing ODS Templates for Non-Standard Graphs in SAS 9.2", Philip R Holland, SAS Global Forum (Seattle, WA, USA), 2010

- "Why Should You Be Using SG (Statistical Graphics) Procedures in SAS 9.2?", Philip R Holland, SAS Global Forum (Las Vegas, NV, USA), 2011

Download all of these papers from **www.lexjansen.com**.

Alphabetical Index

D

E

F

L

M

N

O

P

R

S

T

U

V

W

X

Y

_

.

%