An Introduction to the ODS Graphics Designer and the Graph Template Language (GTL)

Shannon J. Moore
SAS Systems Engineer

Overview of this Presentation

In this presentation we will discuss the following topics:

- A brief overview of the ODS Graphics System, including different ways to get analytical graphs in SAS
- The Graph Template Language (GTL), how it is used, and how you can use it
- Examples of using GTL to create custom graphs
- An introduction to the ODS Graphics Designer

The ODS Graphics System

The Graph Template Language (GTL) is a key part of the ODS Graphics system

Before we get into the details of GTL, let us take a brief tour of ODS Graphics

The ODS Graphics System

ODS Graphics is one of the systems used to create analytical graphs in SAS

- First released with SAS 9.2 as part of SAS/GRAPH®
- From SAS 9.3 onwards, ODS Graphics is part of Base SAS
- With this system, many SAS analytical procedures create graphics output automatically
- This system also provides other ways to create custom graphs

The *Statgraph* Template

- Statgraph templates can be created using the TEMPLATE procedure
- We often refer to a statgraph template as a GTL template
- Automatic graphs from SAS analytical procedures are created using predefined statgraph templates created for each procedure
- If you want to customize the automatic graphs create by SAS analytical procedures, you will need to know how to update the associated template
- You can write your own templates to create your custom graphs

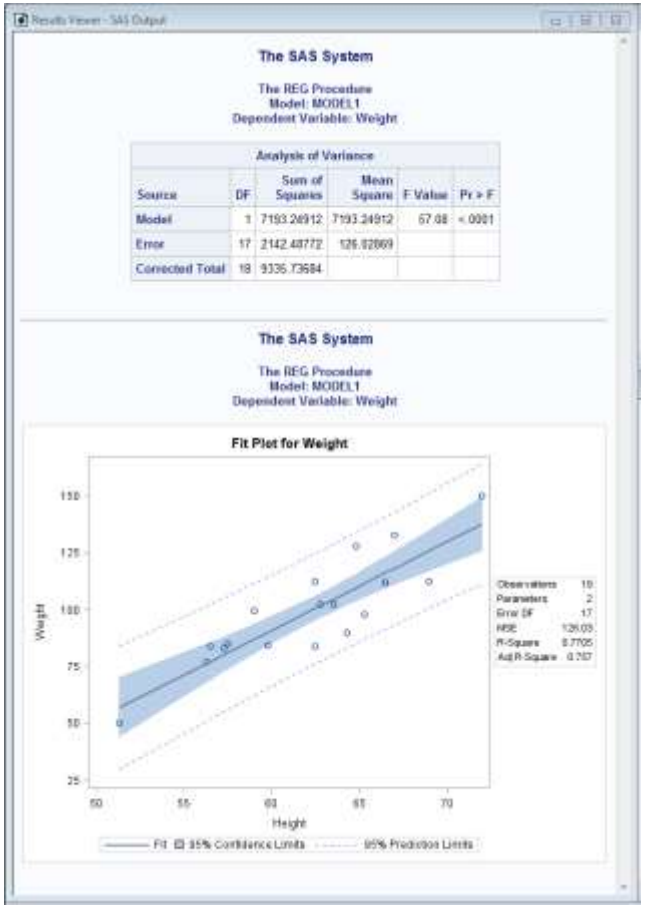
The Graph Template Language

- The Graph Template Language (GTL) is the *syntax* used to define the statgraph or GTL template
- For custom graphs, you can create your own statgraph template
- To create the graph, you associate your data with your template using the SGRENDER procedure

Example of Automatic Graphs

The ODS Graphics statement is used to create automatic graphs from SAS analytical procedures

```
ods select 'Analysis of Variance';
ods select 'Fit Plot';
ods graphics on;
proc reg data=sashelp.class;
  model weight=height;
  quit;
```

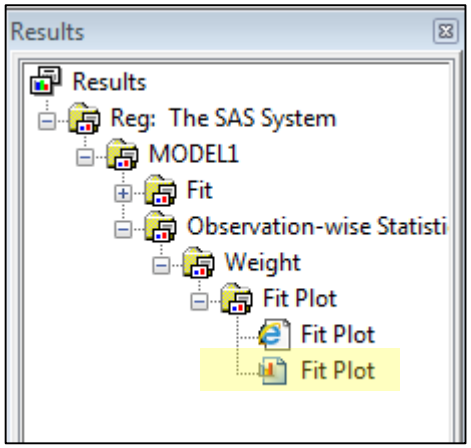


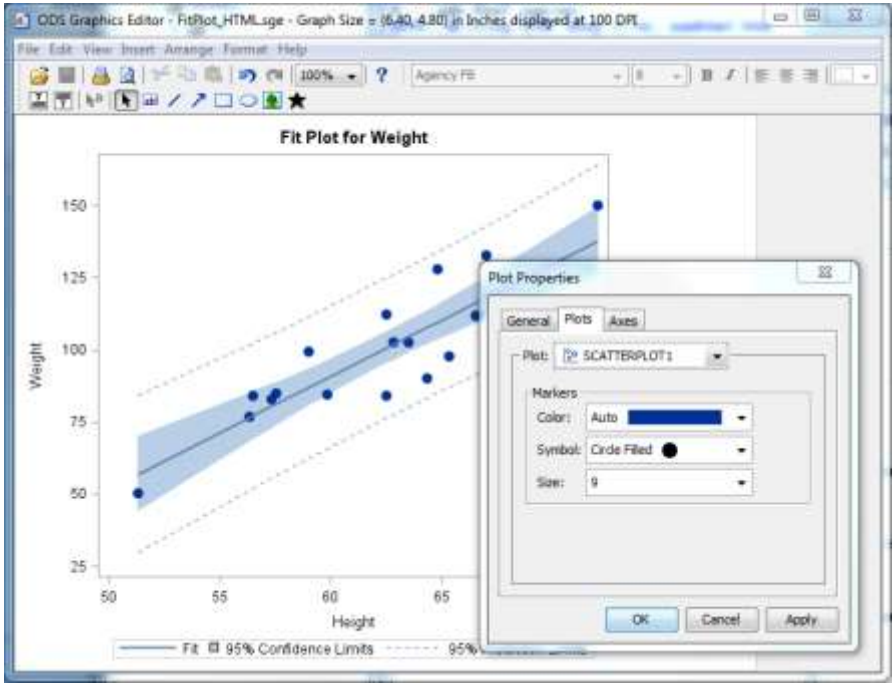
With SAS 9.3, ODS Graphics is "on" by default in Display Manager (DMS) mode

Editing Automatic Graphs

The ODS Graphics Editor can be used to make small customizations to graphs created by the SAS analytical procedures

```
ods select 'Analysis of Variance';
ods select 'Fit Plot';
ods graphics on;
ods html sge=on;
proc reg data=sashelp.class;
  model weight=height;
  quit;
```





Creating Custom Graphs

In SAS, you can create custom analytical graphs using one of the following ODS methods

- Use the interactive ODS Graphics Designer application to create custom graphs
- Use the SG Procedures to create custom graphs
- Both of these methods surface a simple and easy-to-use interface to create graphs and use GTL behind the scenes
- However, they do not cover all of the features of GTL

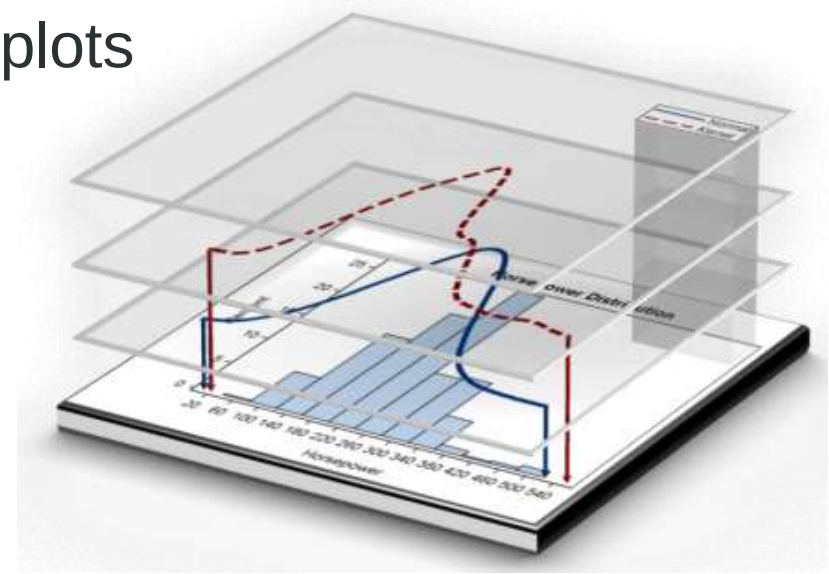
You Can Use GTL Directly

You can use GTL directly to create your custom graphs

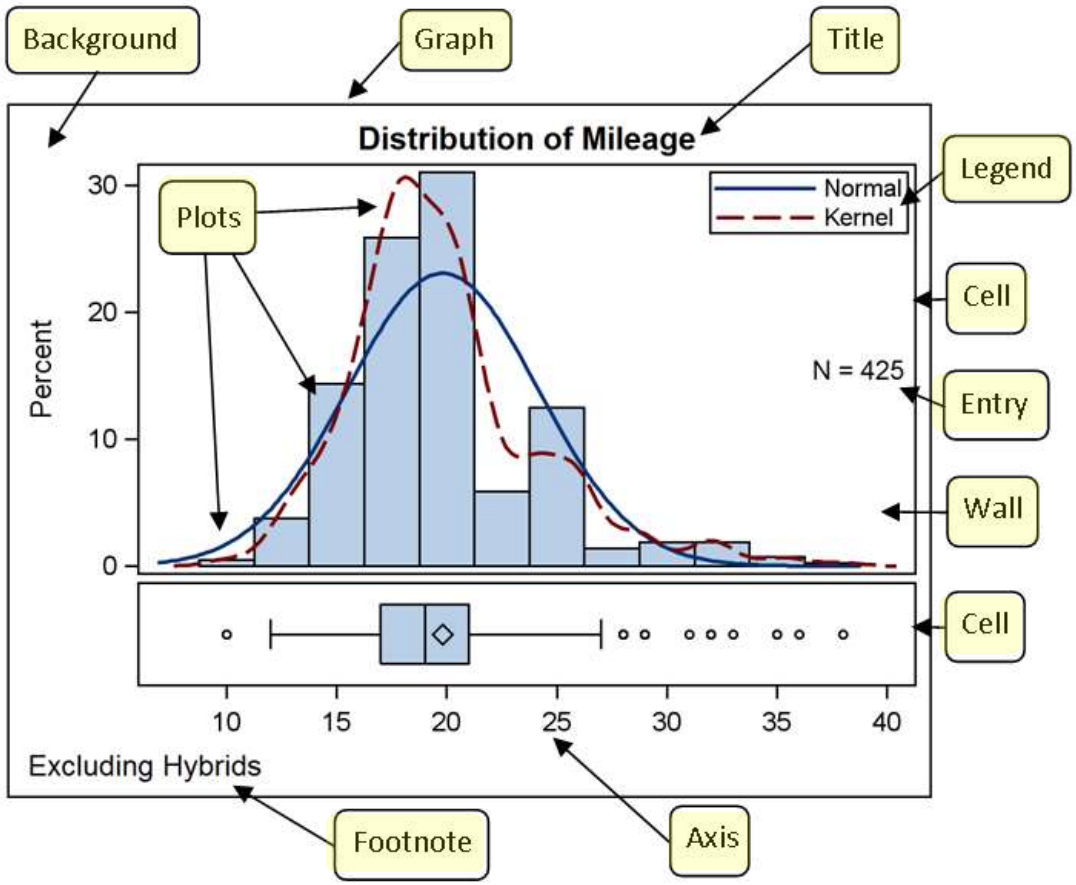
- You may need graph features beyond what are offered by the SG Procedures or the Designer
- You may want to customize the automatic graphs created by SAS analytical procedures
- You may want to learn GTL because it is there

About GTL Graphs

- Graphs are built using the “Building-Block” approach
- Graphs are made up of *Plots*, *Layouts* and other components like *Titles*, *Legends*, *Entries*, etc.
- Plots determine how the data is represented in the graph
- Layouts determine where the plots are drawn
- In this example, multiple compatible plots are combined in a *Layout Overlay* container



Terminology of a GTL Graph



Getting Started with GTL

Creating a graph with GTL is a two-step process:

1. Define the structure of the graph in a statgraph template with GTL using the TEMPLATE procedure. Compile and save the template. No graph is created in this step.
2. Associate compatible data with a pre-compiled template to create the graph using the SGRENDER procedure

Note: Step 2 can be repeated with other *compatible* data sets to create more graphs

Step 1: Define the Graph Template

Define the statgraph template using the TEMPLATE procedure

Compile the template.

```
proc template;
  define statgraph template-name;
    begingraph / <options>;
      <gtl statements to define the graph>
    endgraph;
  end;
run;
```

Submitting the code above will compile and save the template

By default, the template is saved in SASUSER.TEMPLAT item store

No graph is created at this time

Step 2: Create the Graph

Create the graph by associating data with the template using the SGRENDER procedure

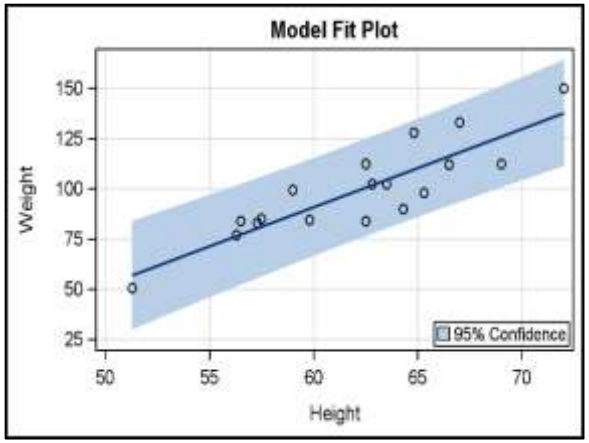
```
proc sgrender data=data-set-name
              template=template-name;
  <other optional statements>
run;
```

Now, the graph is created

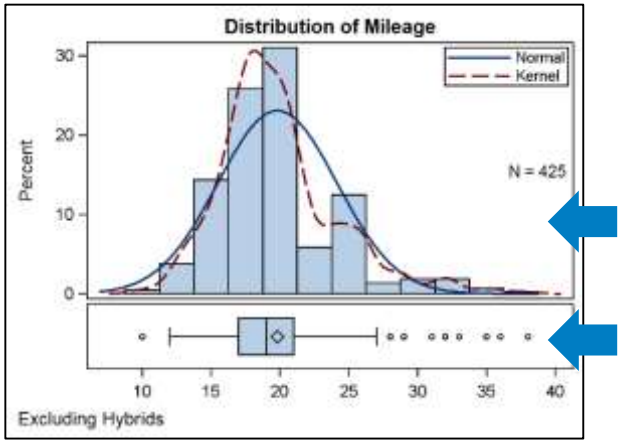
The same template can be used with different data sets to create graphs

Types of Graphs

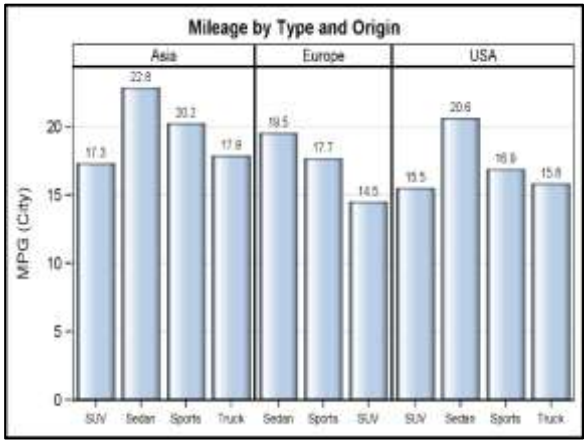
With GTL, you can create many type of graphs, such as the following



Single-Cell Graph



Multi-Cell Graph

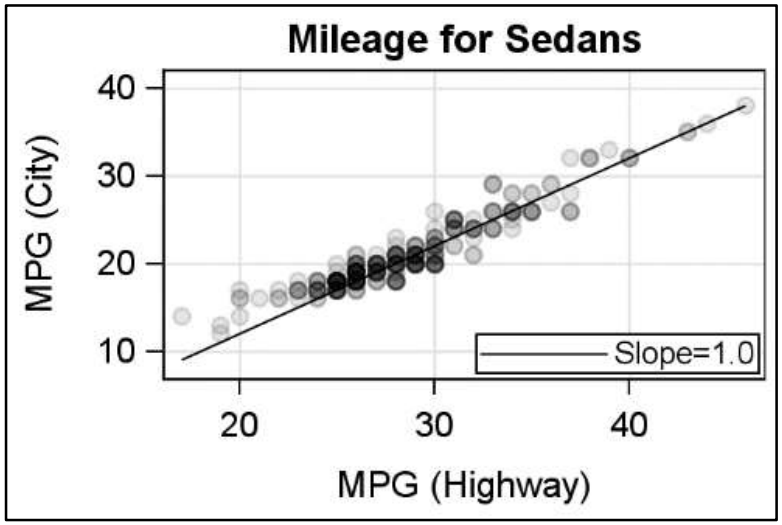


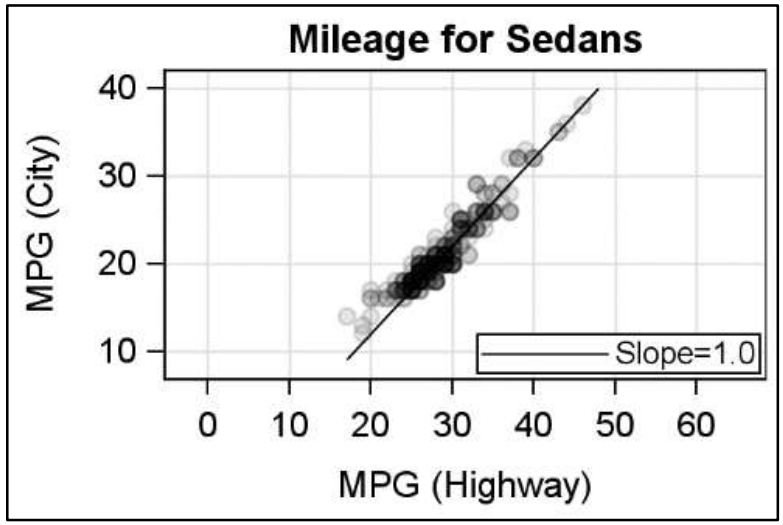
Classification Panel

Single-Cell Graphs

Single-Cell graphs have only one data area. These are created using one of these layouts

- Layout Overlay – creates layered graphs
- Layout OverlayEquated – creates layered graphs with equated axes
- Multiple compatible plot statements can be added to the layouts to create the graphs

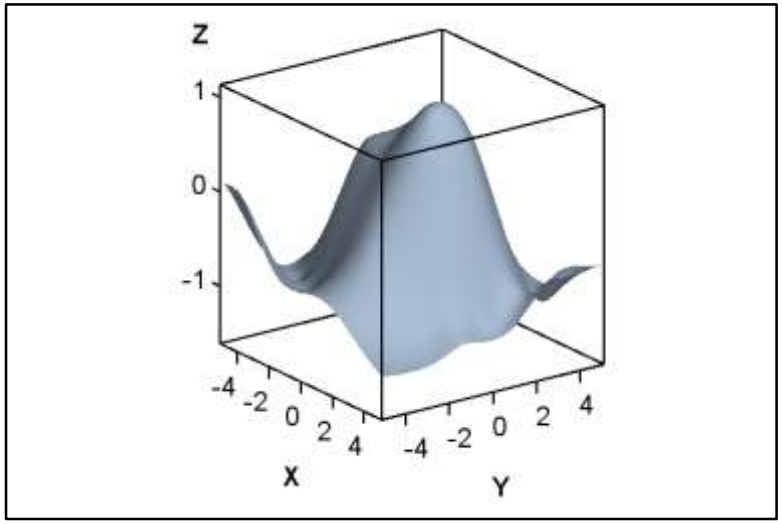


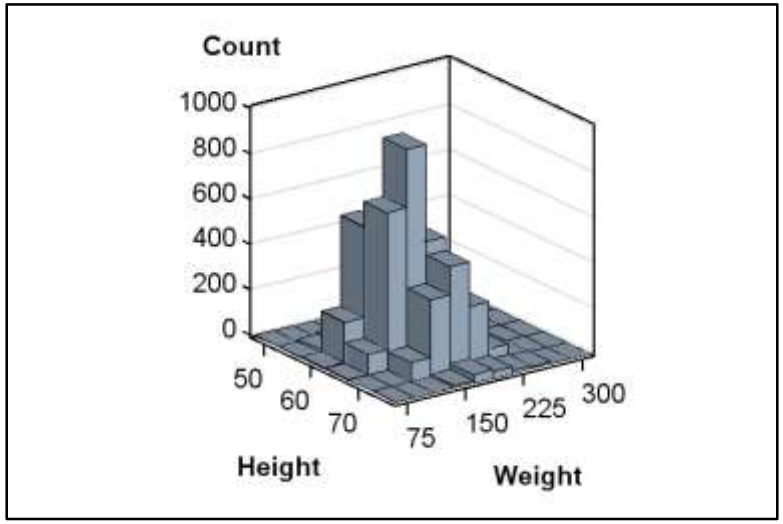


Single-Cell Graphs

3D Single-Cell graphs are created using the following:

- Layout Overlay3D – creates layered 3D graphs

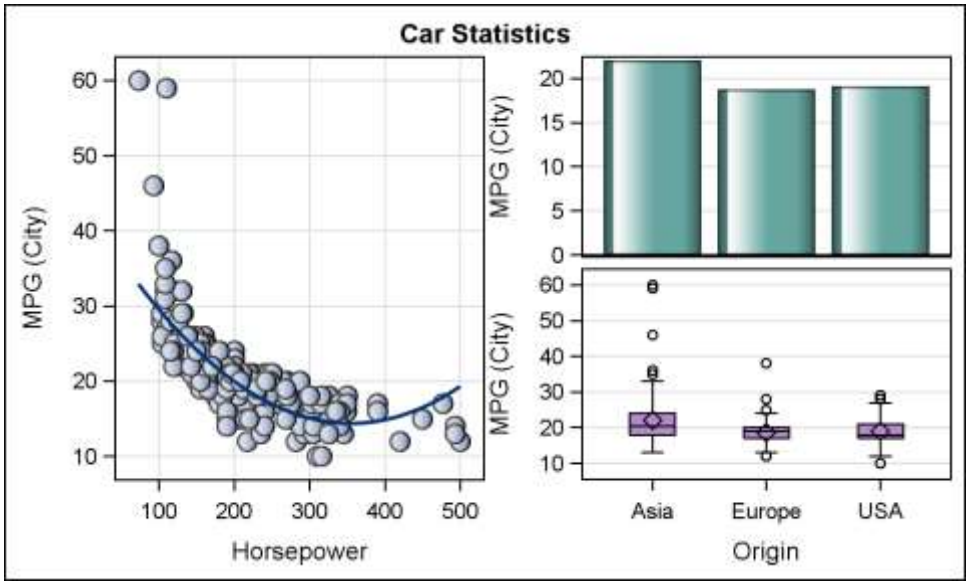




Multi-Cell Graphs

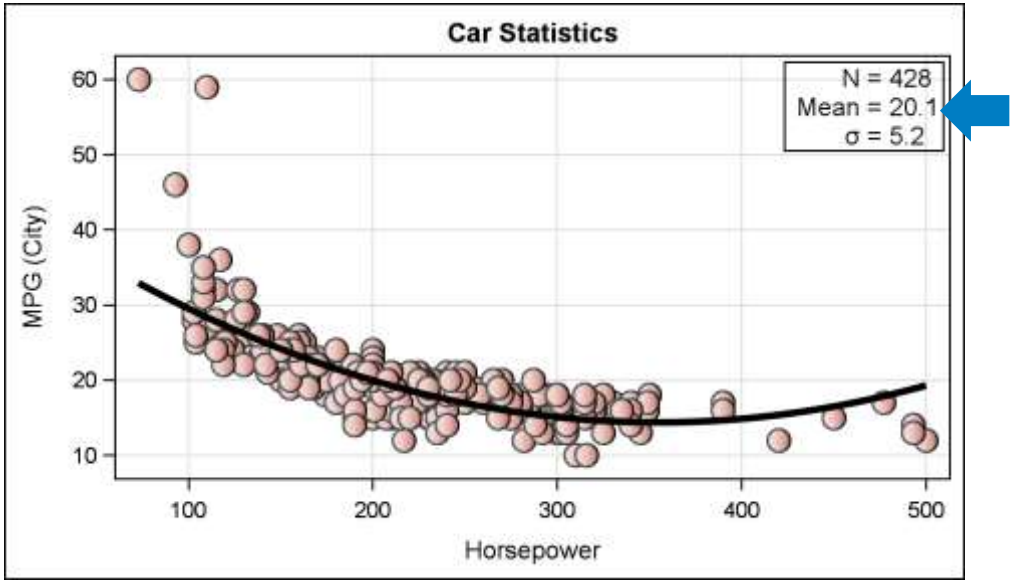
Multi-Cell graphs are created using the following:

- Layout Lattice – divides the graph area into a regular grid of cells as specified by you. Each row and column can have different weights Each cell can have nested layouts
- In this example, the cell on the right uses a nested Layout Lattice
- Multiple compatible plot statements can be added to each cell to create the graphs



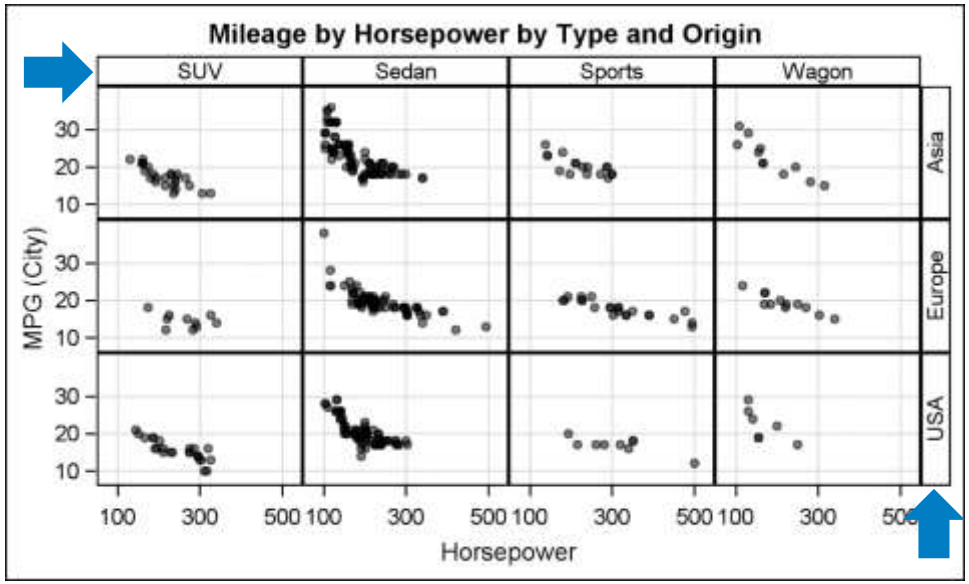
Multi-Cell Layout

- Layout Gridded– divides the graph area in to a regular grid of cells This layout is useful for creating statistics tables with text values
- A Layout Gridded is nested in the top right corner of the cell to create the statistics table



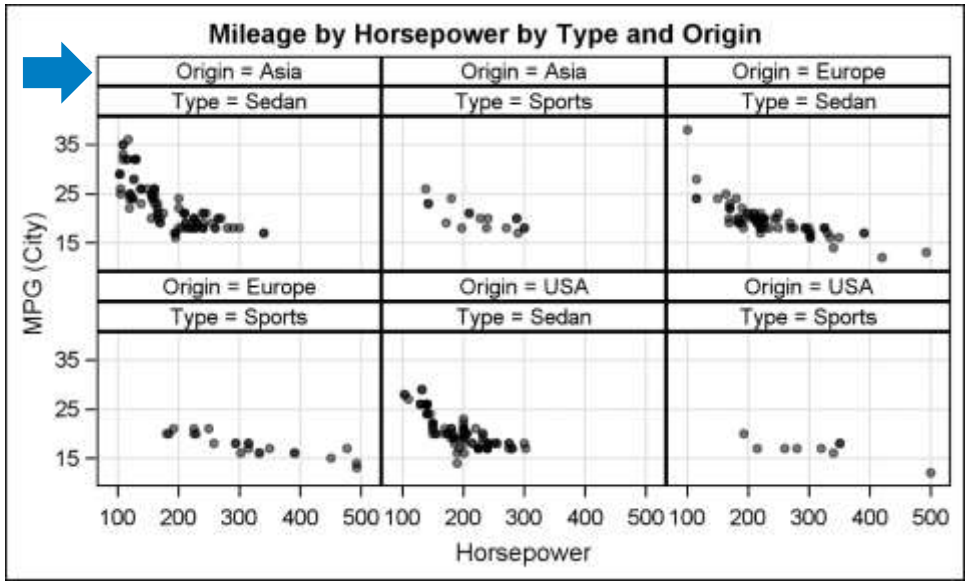
Multi-Cell Classification Lattice

- Layout DataLattice – divides the graph area into a regular grid of cells based on the unique values of the row and column class variables
- Each row and column gets a header



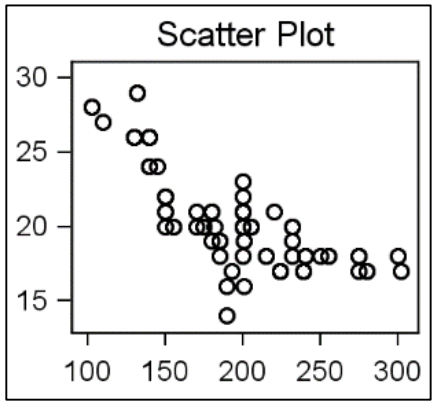
Multi-Cell Classification Panel

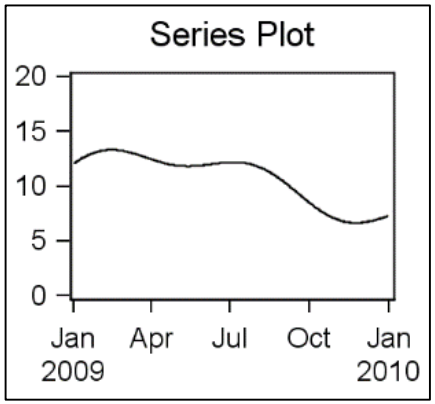
- Layout DataPanel– divides the graph area into a regular grid of cells based on multiple class variables.
- Each cell gets multiple headers.

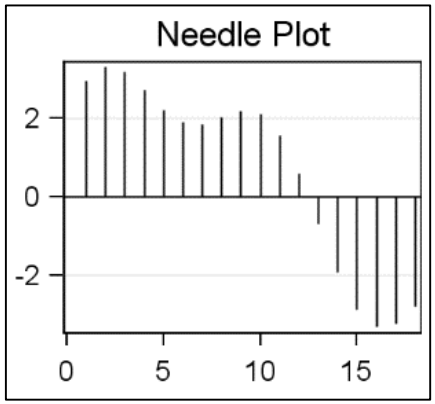


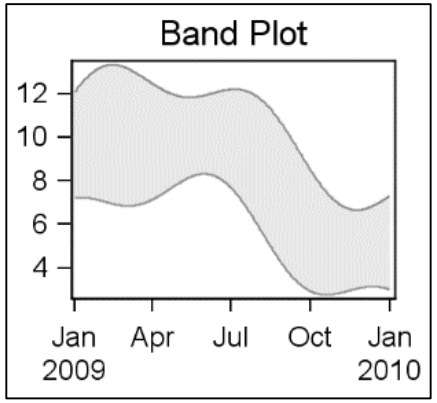
Plot Types

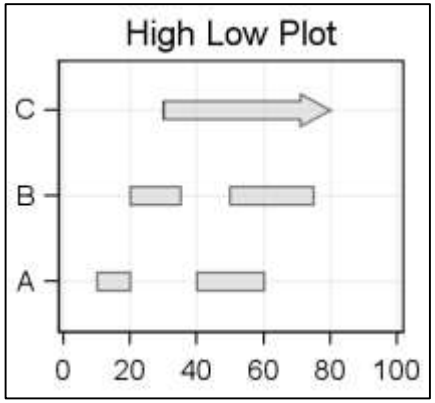
- Basic Plots: ScatterPlot, SeriesPlot, NeedlePlot, BandPlot, HighLowPlot, BubblePlot

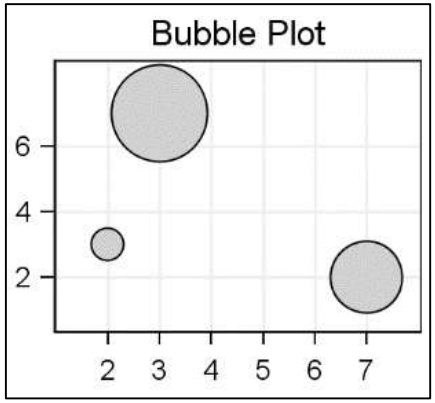






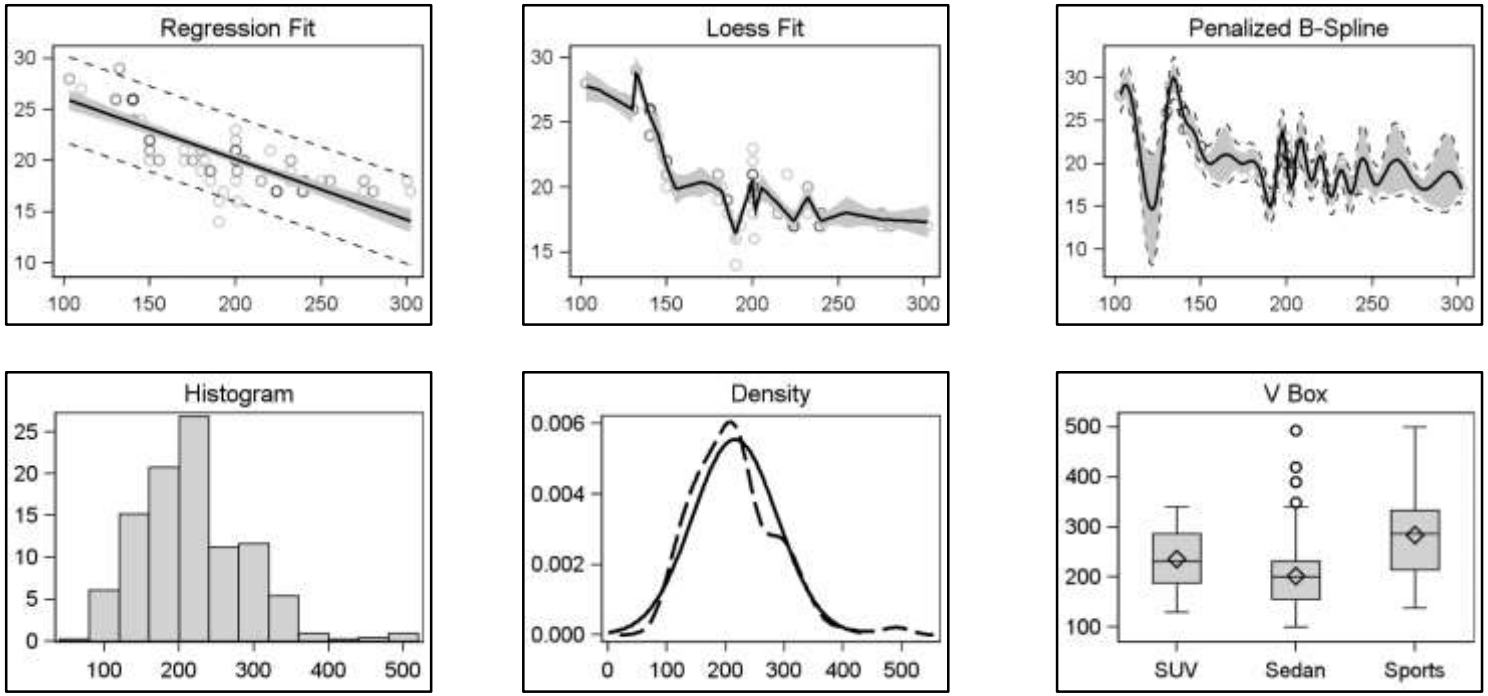






Plot Types

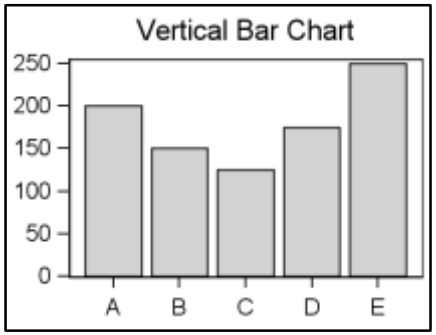
- Fit Plots: RegressionPlot, LoessPlot, PBSplinePlot
- Distribution Plots: Histogram, DensityPlot, BoxPlot

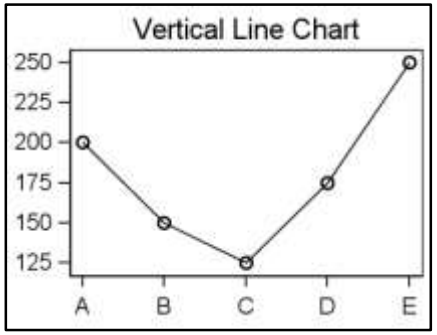


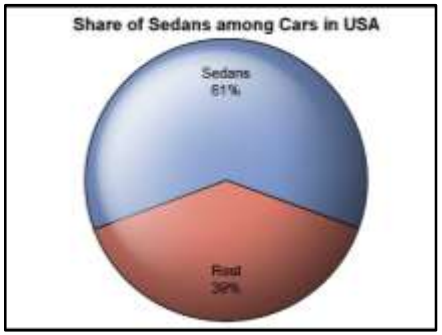
Plot Types

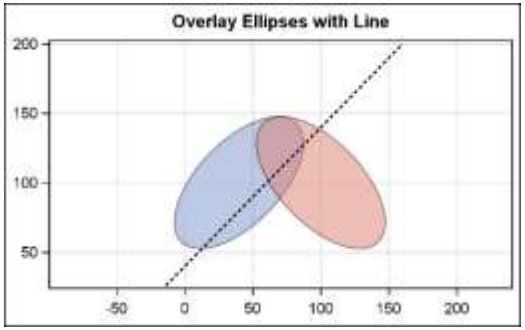
- Categorical Plots: BarChart, LineChart, PieChart
- Parametric Plots: EllipseParm, LineParm, ContourParm,

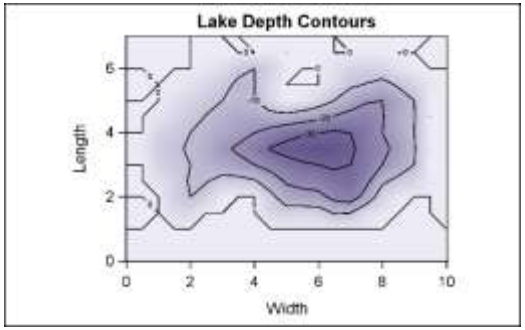
  BarChartParm, BoxPlotParm, HistogramParm





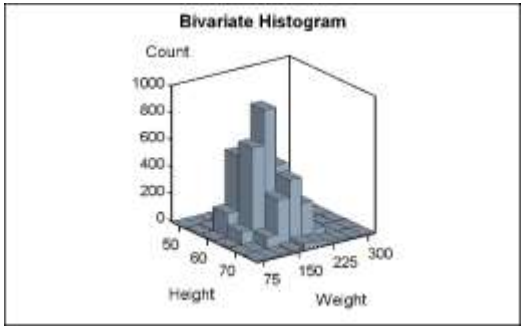


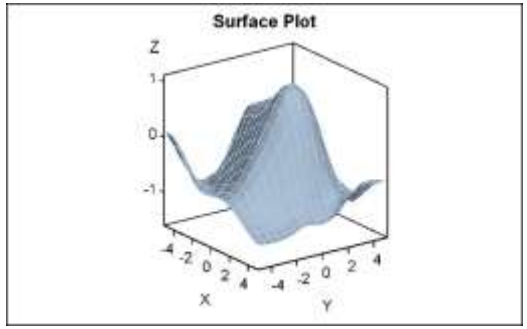


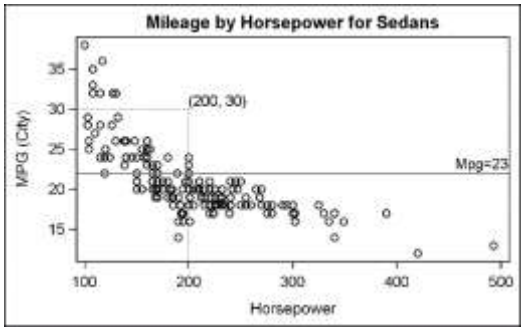


Plot Types

- 3D Plots: BiHistogram3DParm, SurfacePlotParm
- Other Plots: DropLine, ReferenceLine, ScatterPlotMatrix







Other Features

- Titles, footnotes, entries
- Discrete and continuous legends
- Discrete and range attribute maps
- Settings for group attributes like color, symbol, etc.
- Define image and font symbols
- Define custom legend entries
- Dynamics, macro and numeric macro variables
- Function evaluation and conditional logic

GTL Structure

Let us take a look at the structure of all GTL templates

All GTL statements must be inside the BeginGraph – EndGraph block

All plot statements must be inside some layout container

Layouts can be nested within other layouts

Title and footnote statements must be outside the layout tree

Other global statements like attribute maps etc., can be placed in the graph block outside the layouts

```
proc template;
  define statgraph dist;
    begingraph;
      layout lattice;
        layout overlay;
          histogram var;
        endlayout;
        layout overlay;
          boxplot var;
        endlayout;
      endlayout;
    endgraph;
  end;
run;
```

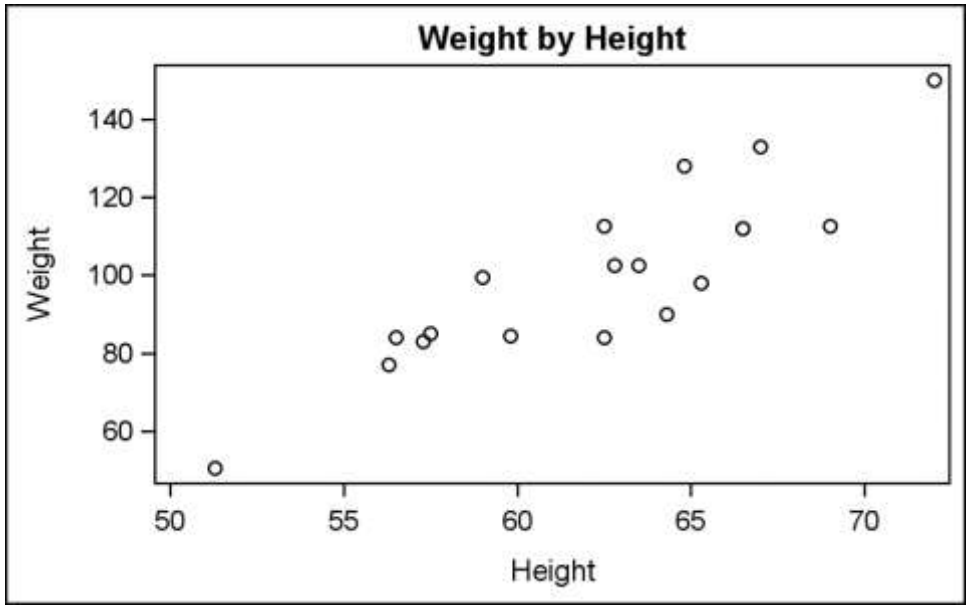
Learning by Example

The best way to learn GTL is by example

Let us now look at some examples that will illustrate the process

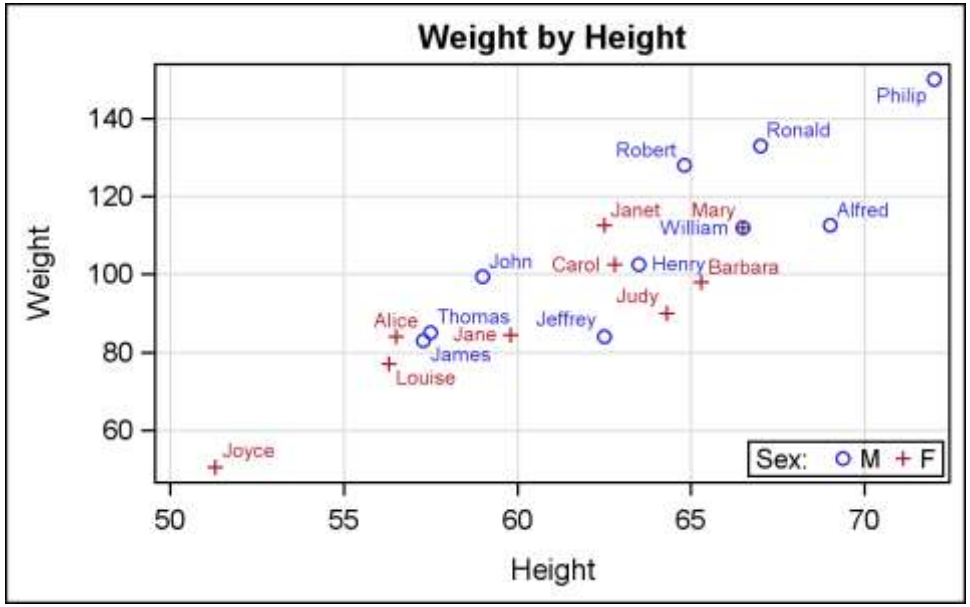
Single Cell Graphs – Simple Example

```
proc template;
  define statgraph scatter;
    begingraph;
      entrytitle 'Weight by Height';
      layout overlay;
        scatterplot x=height y=weight;
      endlayout;
    endgraph;
  end;
run;

proc sgrender data=sashelp.class
              template= scatter;
  run;
```



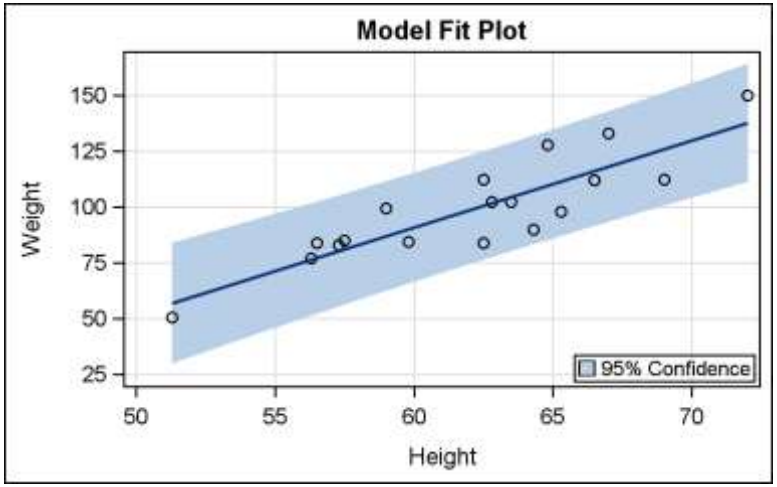
Single Cell Graphs – Simple Example +

```
proc template;
  define statgraph Fig_4_2_1;
    begingraph;
      entrytitle 'Weight by Height';
      layout overlay / xaxisopts=(griddisplay=on)
                       yaxisopts=(griddisplay=on);
        scatterplot x=height y=weight / group=sex datalabel=name name='a';
        discretelegend 'a' / location=inside title='Sex:'
                             halign=right valign=bottom ;
      endlayout;
    endgraph;
  end;
run;

proc sgrender data=sashelp.class
              template= Fig_4_2_1;
run;
```



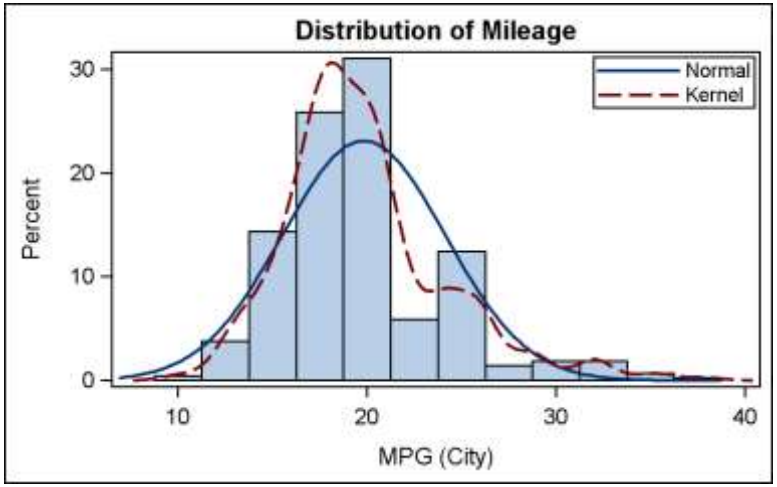
Single Cell Graphs – Model Fit

```
proc template;
  define statgraph Fig_4_7_1;
    begingraph;
      entrytitle 'Model Fit Plot';
      layout overlay / xaxisopts=(griddisplay=on)
                       yaxisopts=(griddisplay=on);
        modelband 'band' / display=(fill) name='b'
            legendlabel='95% Confidence';
        scatterplot x=height y=weight / name='a';
        regressionplot x=height y=weight / cli='band';
        discretelegend 'b' / location=inside
            halign=right valign=bottom;
      endlayout;
    endgraph;
  end;
run;

proc sgrender data=sashelp.class
              template=Fig_4_7_1;
run;
```



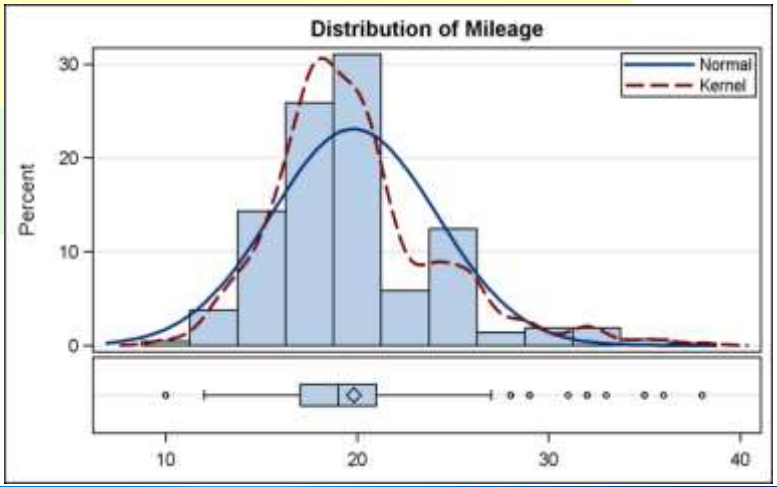
Single Cell Graphs – Distribution Plot

```
proc template;
  define statgraph Fig_4_7_2;
    begingraph;
      entrytitle 'Distribution of Mileage';
      layout overlay;
        histogram mpg_city / binaxis=false;
        densityplot mpg_city / name='n' lineattrs=graphfit
           legendlabel='Normal';
       densityplot mpg_city / kernel() name='k' lineattrs=graphfit2
           legendlabel='Kernel';
        discretelegend 'n' 'k' / location=inside across=1
           halign=right valign=top;
      endlayout;
    endgraph;
  end;
run;

proc sgrender data=sashelp.cars
     template=Fig_4_7_2;
run;
```



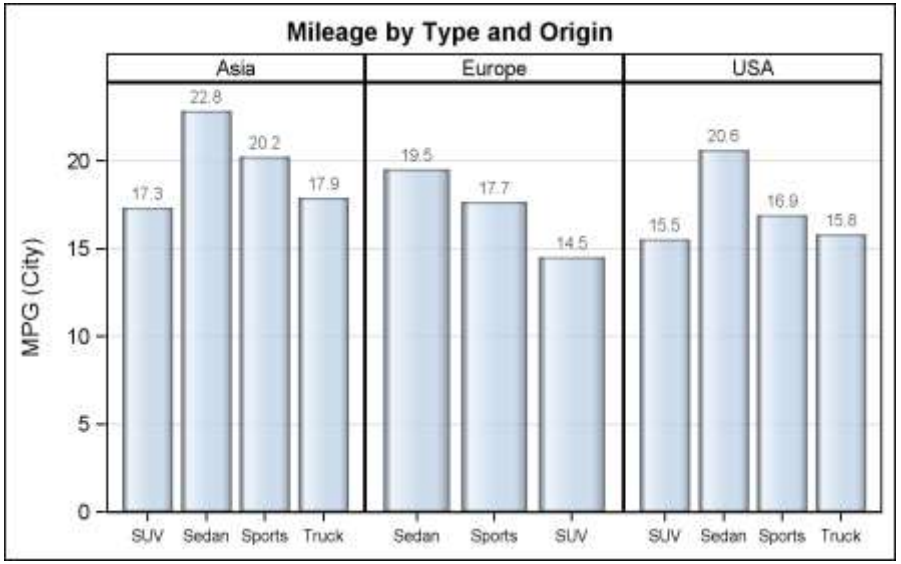
Multi Cell Graphs – Distribution Plot

```
proc template;
  define statgraph Fig_5_4_3;
    begingraph;
      entrytitle 'Distribution of Mileage';
      layout lattice / columndatarange=union rowweights=(0.8 0.2);
        columnaxes;
          columnaxis / display=(ticks tickvalues);
        endcolumnaxes;
        layout overlay / yaxisopts=(griddisplay=on);
          histogram mpg_city / binaxis=false;
          densityplot mpg_city / lineattrs=graphfit name='n'legendlabel='Normal';
          densityplot mpg_city / kernel() lineattrs=graphfit2 name='k‘
              legendlabel='Kernel';
          discretelegend 'n' 'k' / location=inside across=1
               halign=right valign=top ;
        endlayout;
        layout overlay / yaxisopts=(griddisplay=on);
          boxplot y=mpg_city / orient=horizontal;
        endlayout;
      endlayout;
    endgraph;
  end;
run;
```



Classification Panel

```
proc template;
  define statgraph Fig_5_2_3;
    begingraph;
      entrytitle 'Mileage by Type and Origin';
      layout datalattice columnvar=origin / headerlabeldisplay=value
             columndatarange=union rowaxisopts=(griddisplay=on offsetmin=0)
             columnaxisopts=(display=(ticks tickvalues) tickvalueattrs=(size=7));
        layout prototype;
          barchart x=type y=mpg_city / stat=mean dataskin=gloss
                   datatransparency=0.3 barlabel=true;
        endlayout;
      endlayout;
    endgraph;
  end;
run;
```

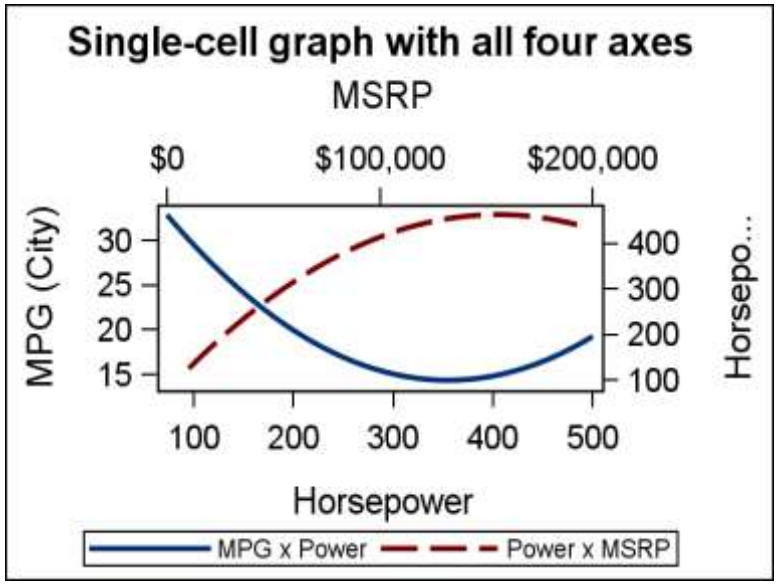


Axis Features

Each cell can have up to 4 axes – X(b), X2(t), Y(l), Y2(r)

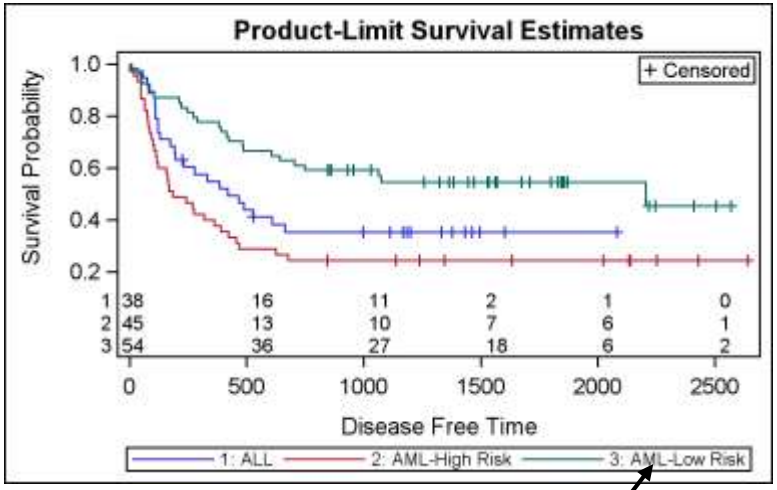
Each plot in the cell can associate with one X and one Y axis

GTL supports multiple axis types such as Linear, Discrete, Time and Log. Multiple options are supported for each

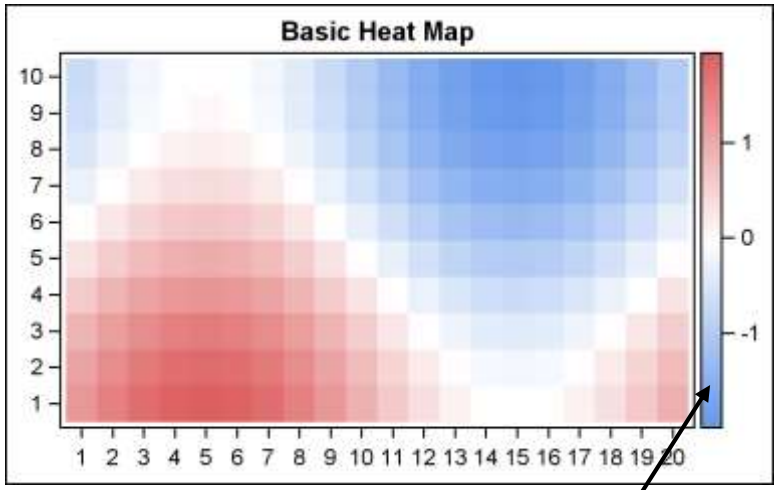


Legend Features

GTL supports discrete and continuous legends



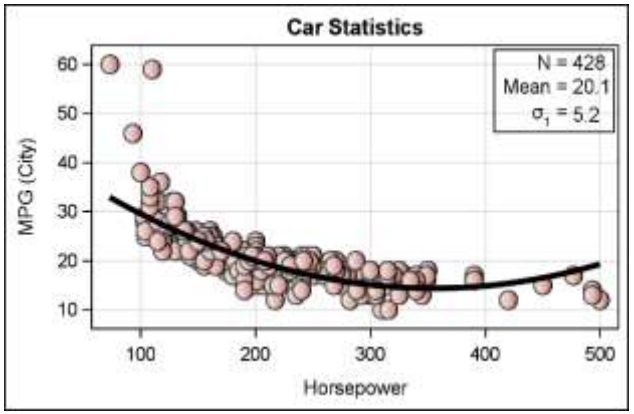
Discrete Legend



Continuous Legend

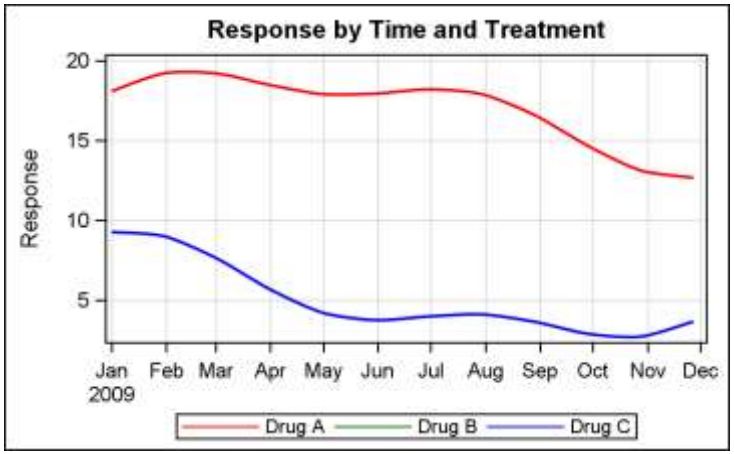
Entries, Unicode and Function Evaluation

```
proc template;
  define statgraph Fig_6_4_2;
    begingraph / subpixel=on;
      entrytitle 'Car Statistics';
      layout overlay / yaxisopts=(griddisplay=on) xaxisopts=(griddisplay=on);
        layout gridded / rows=3 order=columnmajor autoalign=(topright) border=true;
          entry textattrs=(size=10) halign=right 'N = ';
          entry textattrs=(size=10) halign=right 'Mean = ';
          entry textattrs=(size=10) halign=right {unicode SIGMA} {sub '1'} ' =';
          entry textattrs=(size=10) halign=left eval(strip(put(n(mpg_city),4.0)));
          entry textattrs=(size=10) halign=left eval(strip(put(mean(mpg_city),4.1)));
          entry textattrs=(size=10) halign=left eval(strip(put(stddev(mpg_city),4.1)));
        endlayout;

        scatterplot x=horsepower y=mpg_city / filledoutlinedmarkers=true
            dataskin=gloss markerattrs=(size=13 symbol=circlefilled)
            markerfillattrs=graphdata2;
        regressionplot x=horsepower y=mpg_city / degree=2
            lineattrs=(color=black thickness=4);
      endlayout;
    endgraph;
  end;
run;
```



Discrete Attributes Map

```
proc template;
  define statgraph Fig_7_2_1;
    dynamic _title;
    begingraph / subpixel=on;
      discreteattrmap name='drug';
        value 'Drug A' / lineattrs=(color=red pattern=solid);
        value 'Drug B' / lineattrs=(color=green pattern=solid);
        value 'Drug C' / lineattrs=(color=blue pattern=solid);
      enddiscreteattrmap;

      discreteattrvar attrvar=drugname var=drug attrmap='drug';

      entrytitle 'Response by Time and Treatment';
      layout overlay / yaxisopts=(griddisplay=on label='Response')
                       xaxisopts=(griddisplay=on display=(ticks tickvalues));
        seriesplot x=date y=val / group=drugname
              name='a' smoothconnect=true
              lineattrs=(thickness=2);
        discretelegend 'drug' / type=line
      endlayout;
    endgraph;
  end;
run;
```

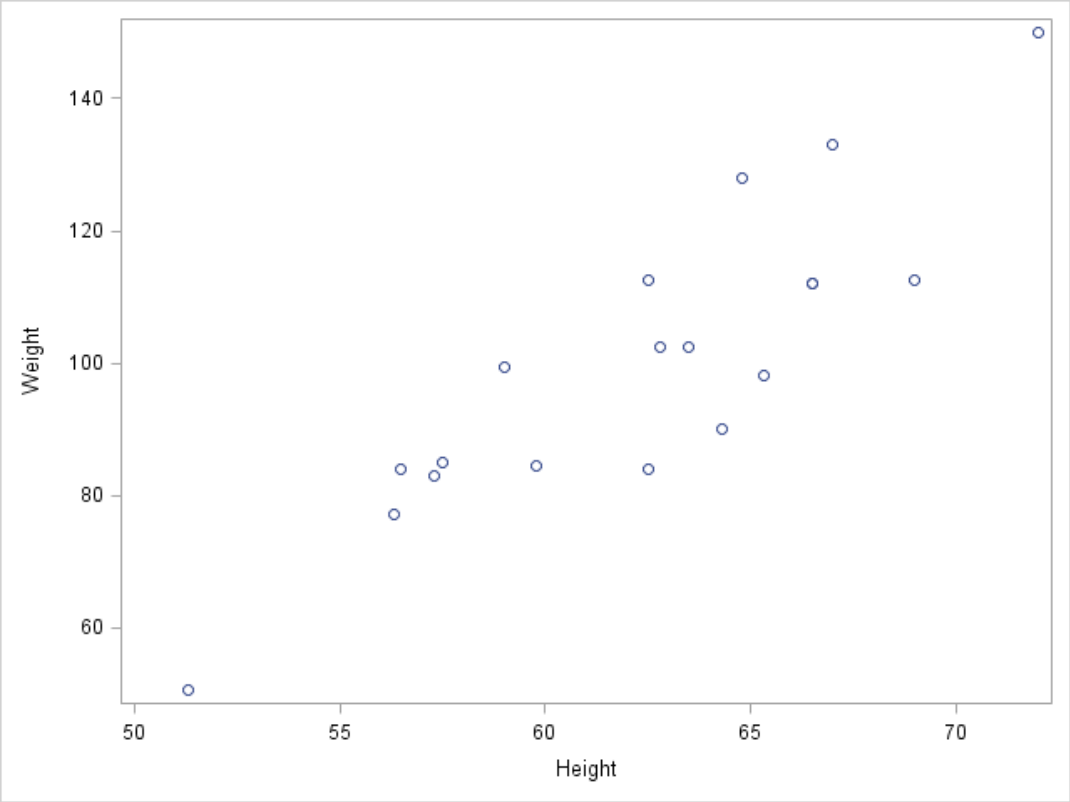


MAKING YOUR TEMPLATE FLEXIBLE

```
proc template;
      define statgraph gtl.dyn;
            begingraph;
                  dynamic _var1 _var2;
                  layout overlay;
                        scatterplot x=_var1 y=_var2;
                  endlayout;
            endgraph;
      end;
run;

proc sgrender data=sashelp.class template=gtl.dyn;
      dynamic _var1='height' _var2='weight';
run;
```

MAKING YOUR TEMPLATE FLEXIBLE

```
proc sgrender data=sashelp.class template=gtl.dyn;
       dynamic _var1='height' _var2='weight';
run;
```

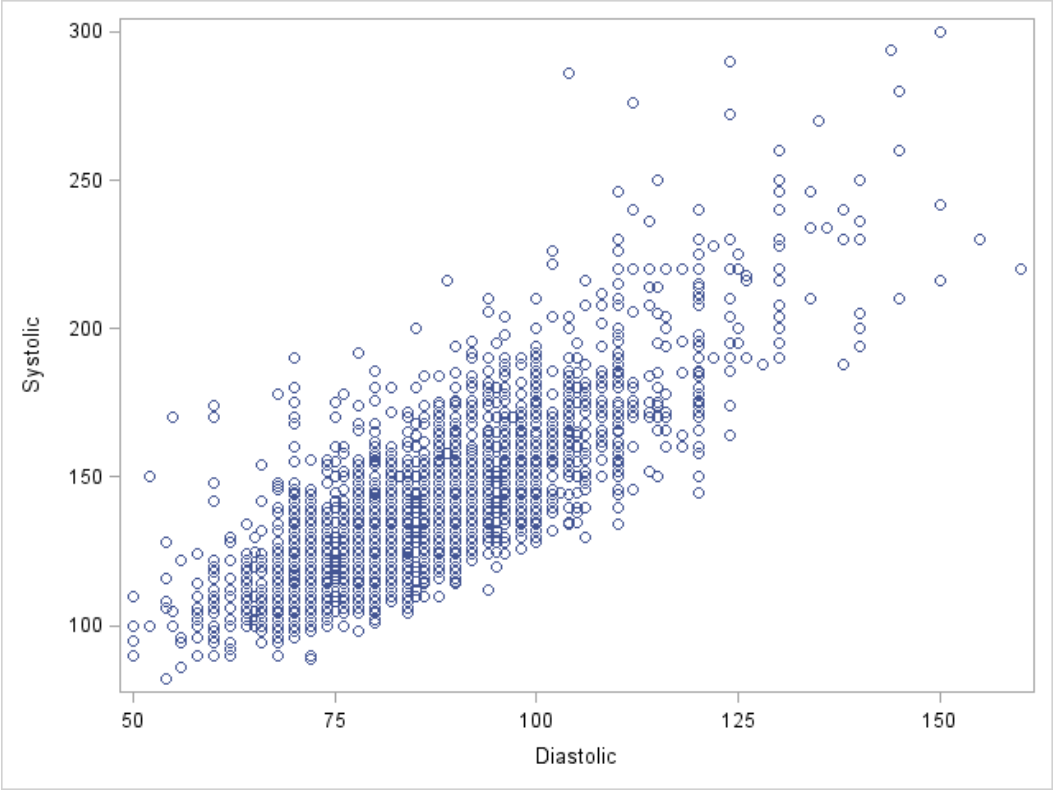


MAKING YOUR TEMPLATE FLEXIBLE

```
proc template;
      define statgraph gtl.dyn;
            begingraph;
                  dynamic _var1 _var2;
                  layout overlay;
                        scatterplot x=_var1 y=_var2;
                  endlayout;
            endgraph;
      end;
run;

proc sgrender data=sashelp.heart template=gtl.dyn;
      dynamic _var1='diastolic' _var2='systolic';
run;
```

MAKING YOUR TEMPLATE FLEXIBLE

```
proc sgrender data=sashelp.heart template=gtl.dyn;
      dynamic _var1='diastolic' _var2='systolic';
run;
```

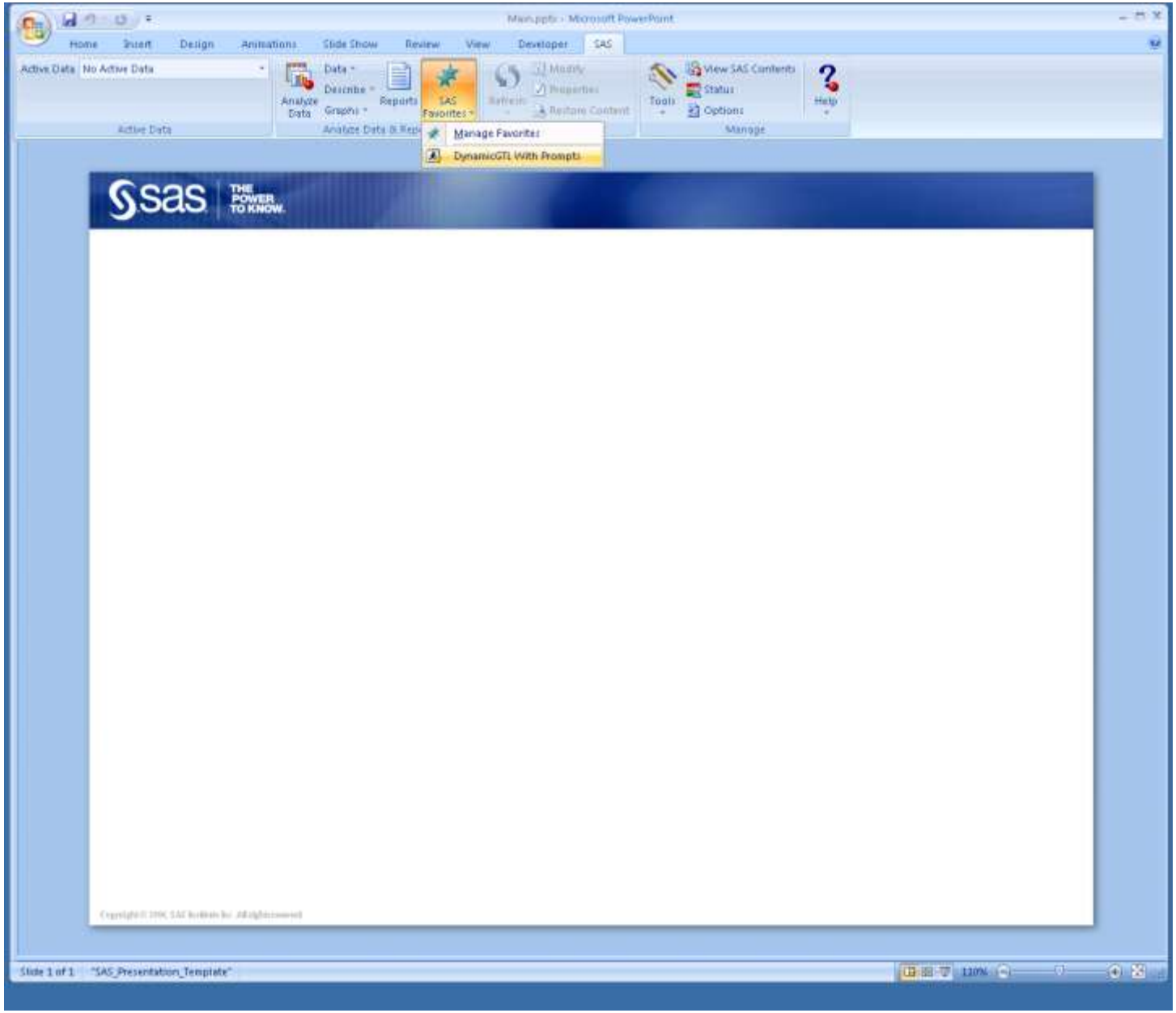


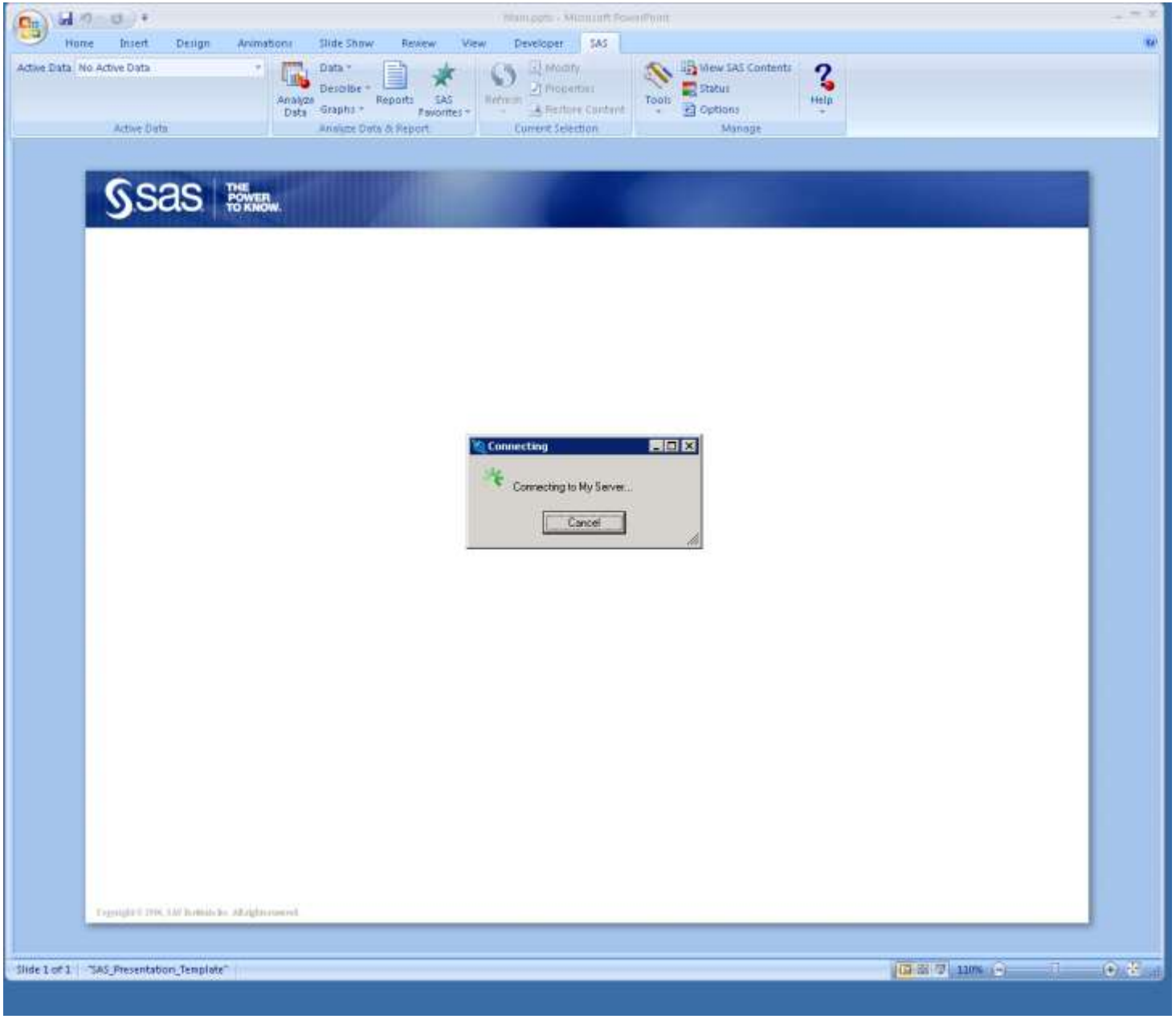
MAKING YOUR TEMPLATE FLEXIBLE

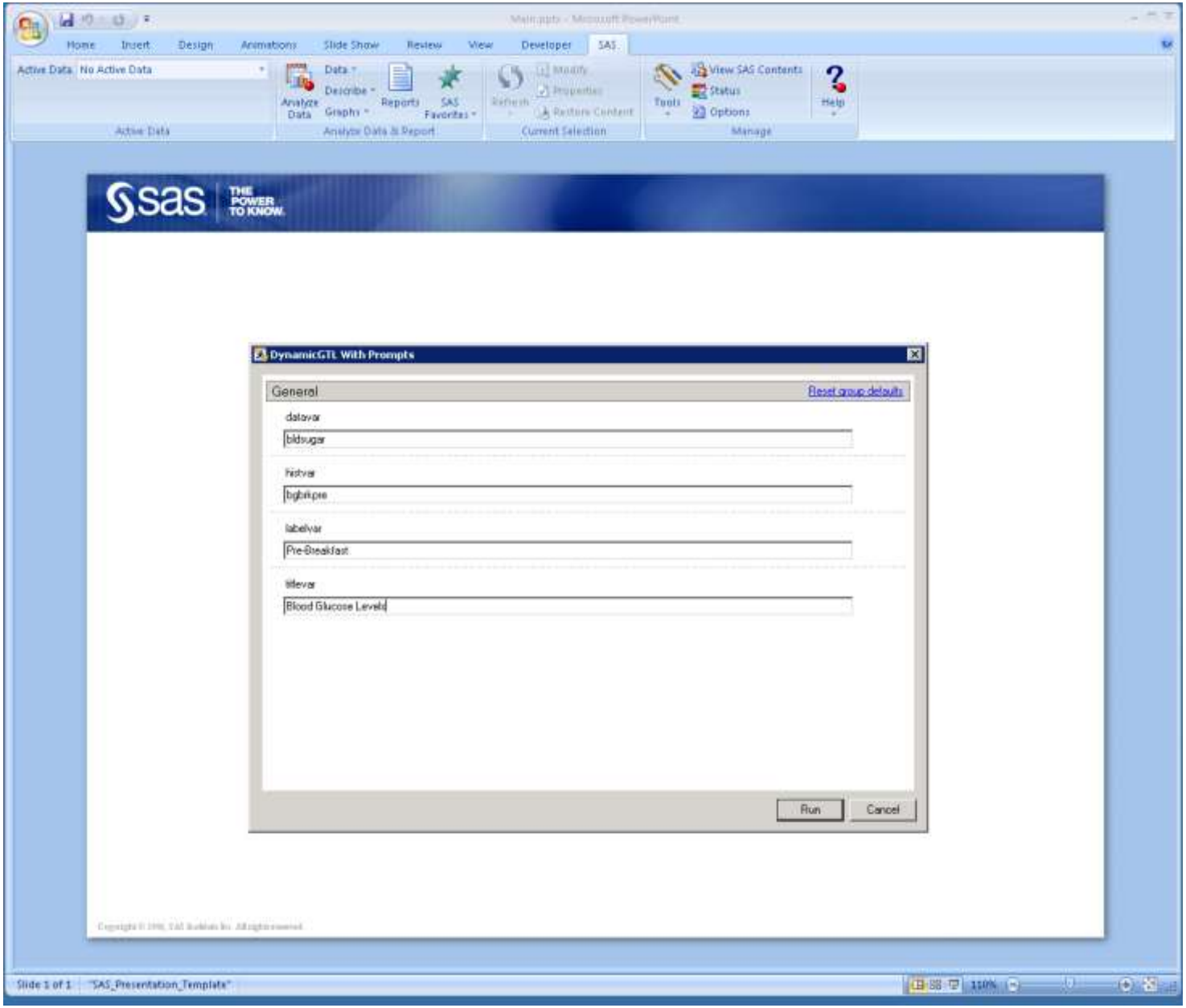
```
proc template;
            define statgraph distribution;
                        dynamic VAR VARLABEL TITLE NORMAL;
                        begingraph;
                        entrytitle TITLE;
                        layout lattice / columns=1 rows=2  rowgutter=2px rowweights=(.9 .1) columndatarange=union;
                                                columnaxes;
                                                columnaxis / label=VARLABEL;
                                                endcolumnaxes;
                                    layout overlay / yaxisopts=(offsetmin=.035);
                                    layout gridded / columns=2 border=true autoalign=(topleft topright);
                                                entry halign=left "Nobs";
                                                entry halign=left eval(strip(put(n(VAR),8.)));
                                                entry halign=left "Mean";
                                                entry halign=left eval(strip(put(mean(VAR),8.2)));
                                                entry halign=left "StdDev";
                                                entry halign=left eval(strip(put(stddev(VAR),8.3)));
                                                endlayout;
                                    histogram VAR / scale=percent;
                                    if (exists(NORMAL)) densityplot VAR / normal( );
                                    endif;
                                    fringeplot VAR / datatransparency=.7;
                                    endlayout;
                                    boxplot y=VAR / orient=horizontal;
                                    endlayout;
                        endgraph;
            end;
run;
```

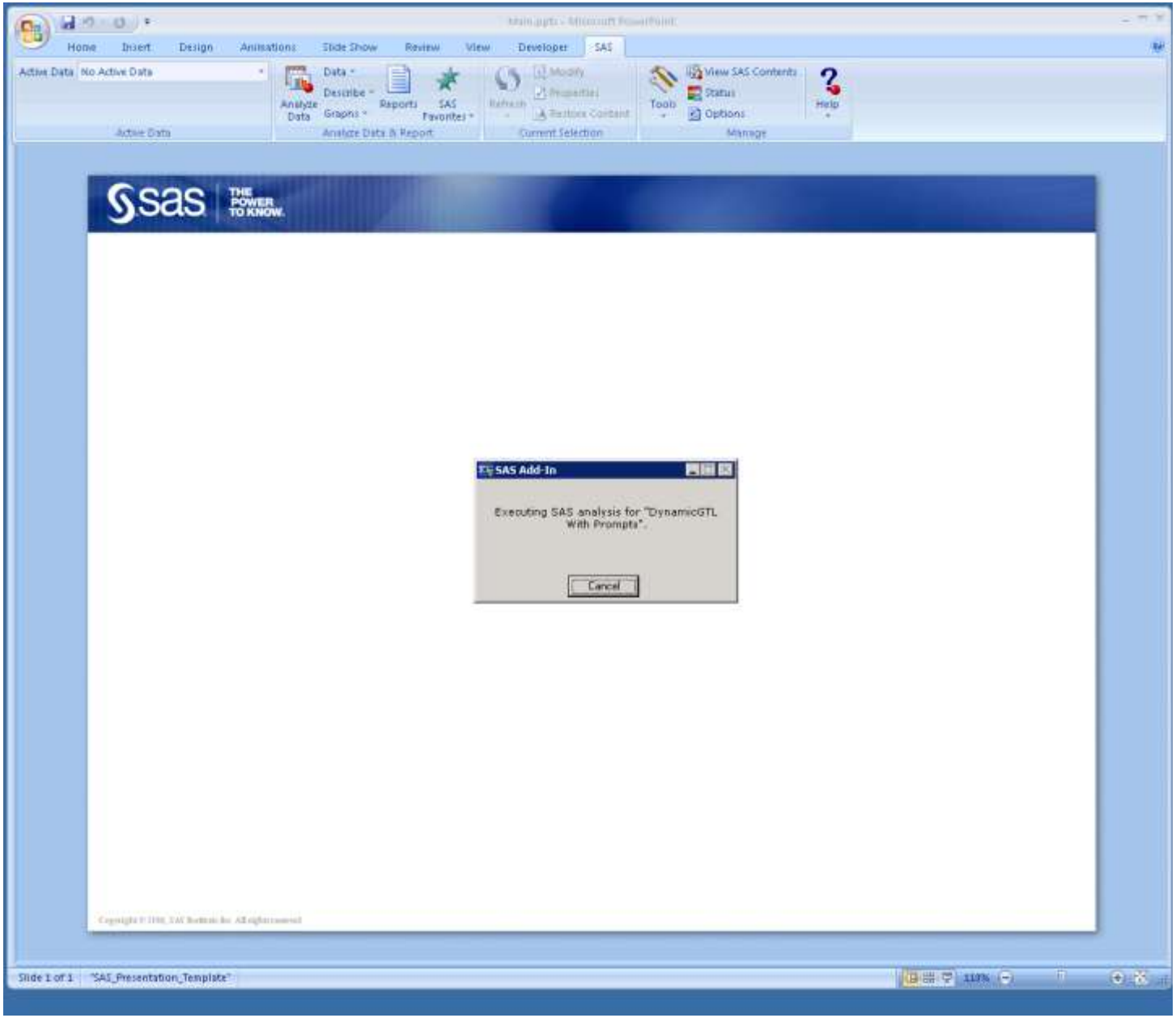
MAKING YOUR TEMPLATE FLEXIBLE

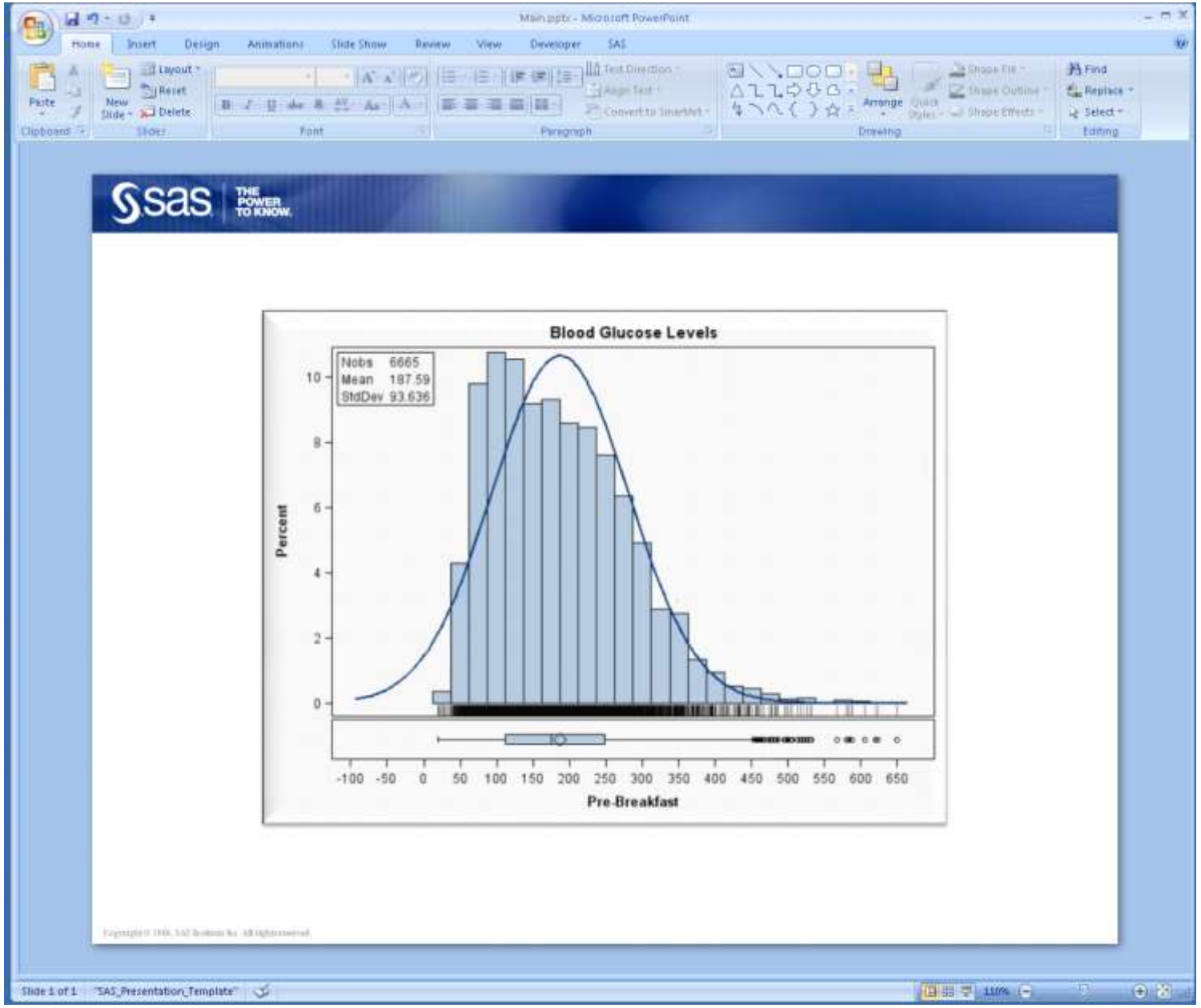
```
proc sgrender data=&datavar template=distribution;
 dynamic var="&histvar"
        varlabel="&labelvar"
        normal="yes"
        title="&titlevar";
run; quit;
```





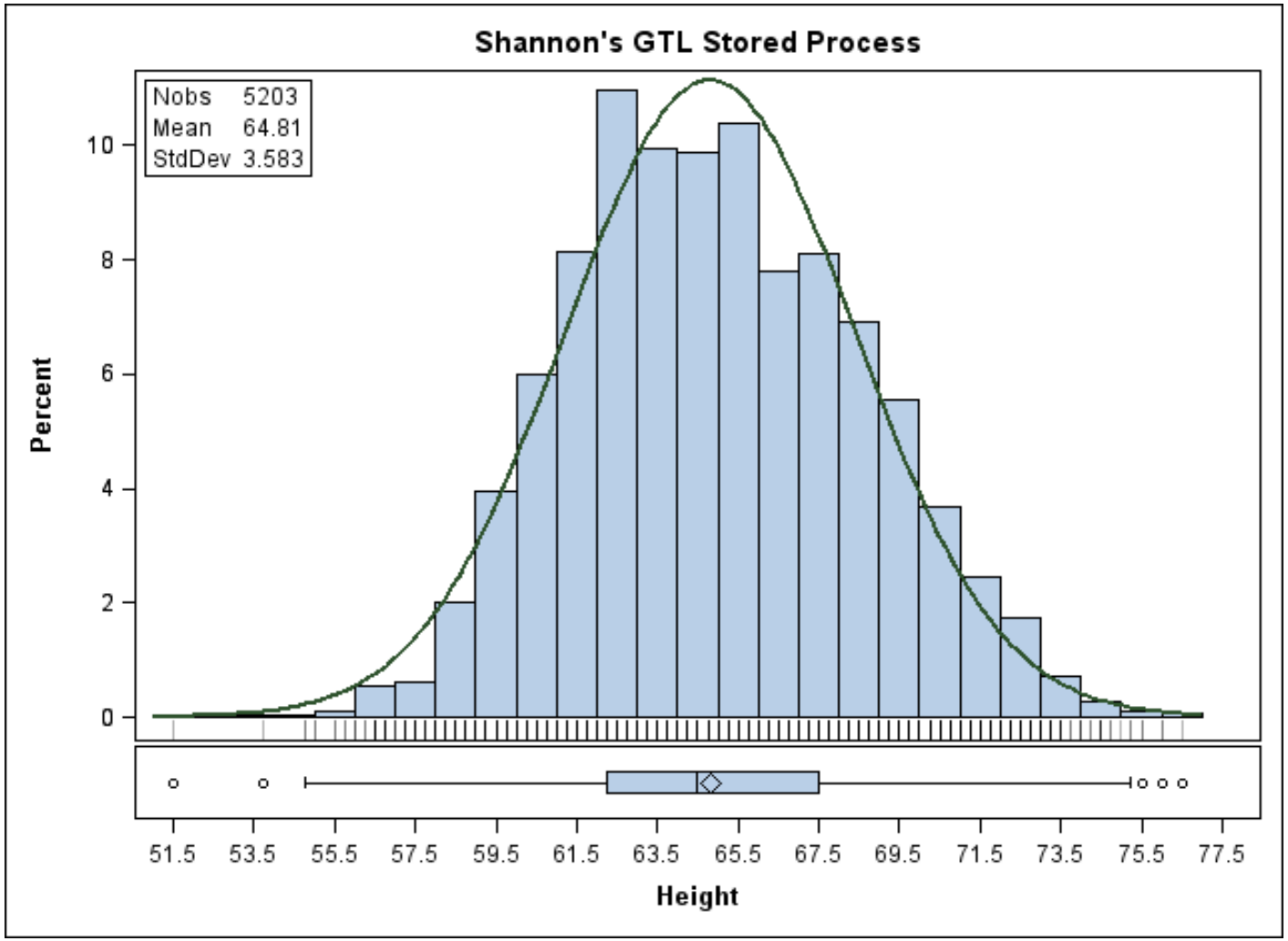






ODS GRAPHICS DESIGNER AND GTL

LIVE EXAMPLE



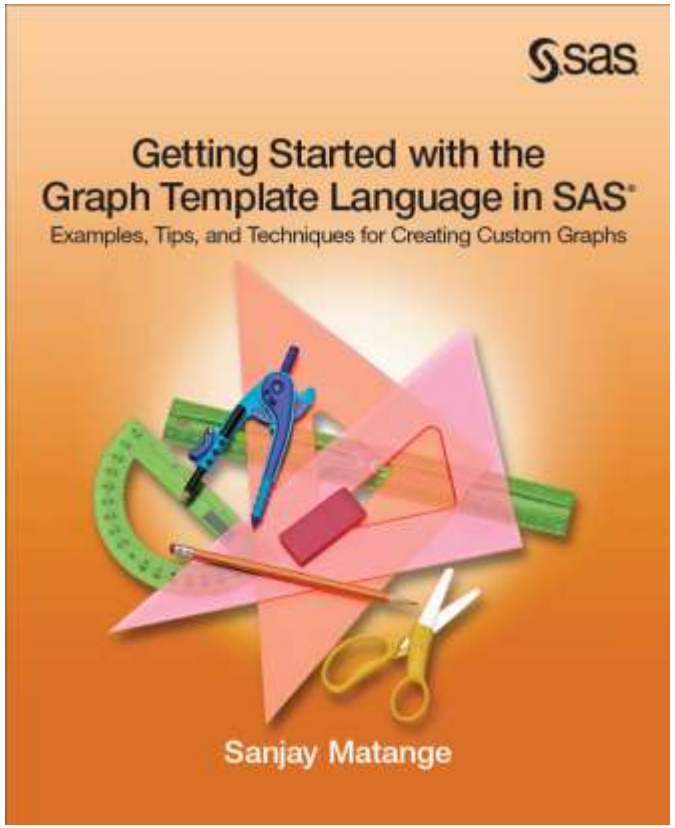
Review of this Presentation

In this presentation we discussed the following topics

- Brief overview of the ODS Graphics System, including different ways to get analytical graphs in SAS
- The Graph Template Language (GTL), how it is used, and how you can use it
- Examples of using GTL to create custom graph

Resources

- [Graph Focus Area Page on Support.sas.com]



MANY THANKS TO SANJAY MATANGE FOR HIS HELP WITH THIS PRESENTATION

THANK YOU FOR YOUR TIME AND FOR USING SAS!