Getting Started with the Graph Template Language

Sanjay Matange
SAS Institute, Inc.

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.

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 the system used to create analytical graphs in SAS®

- First released with SAS 9.2 as part of SAS/GRAPH®
- From SAS 9.3 onwards, this is part of Base.
- 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 specially 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.
- Additionally, you can write your own templates to create your custom graphs.

The Graph Template Language

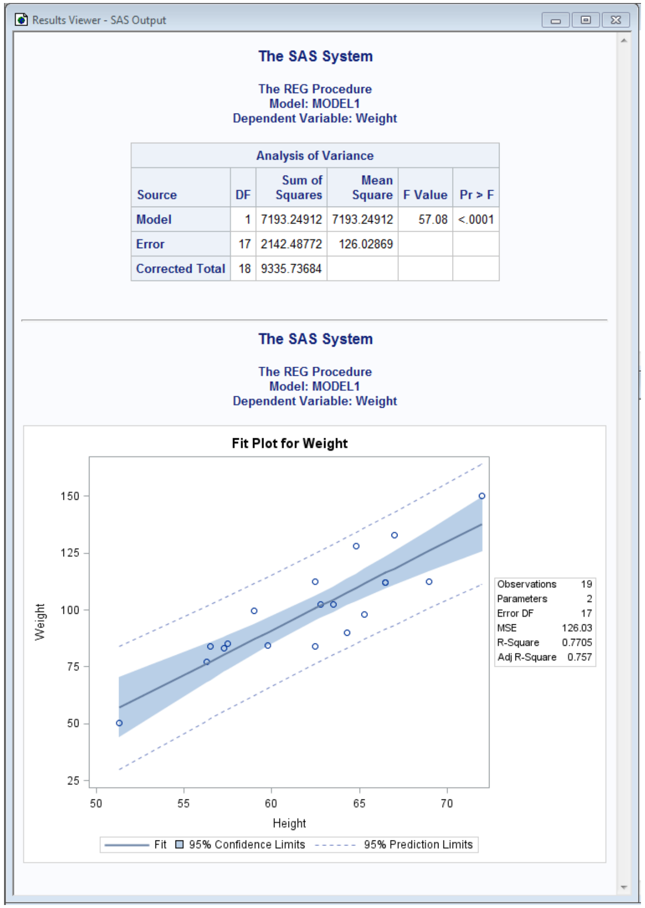
- The Graph Template Language (GTL) is the syntax used to define the statgraph or GTL template.
- For custom graphs, you can create your own statgraph template.
- To create the graph, you can associate your data with your template using the SGRENDER procedure.

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 DMS mode.



The SAS System

The REG Procedure
Model: MODEL1
Dependent Variable: Weight

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	7193.24912	7193.24912	57.08	<.0001
Error	17	2142.48772	126.02869		
Corrected Total	18	9335.73684			

The SAS System

The REG Procedure
Model: MODEL1
Dependent Variable: Weight

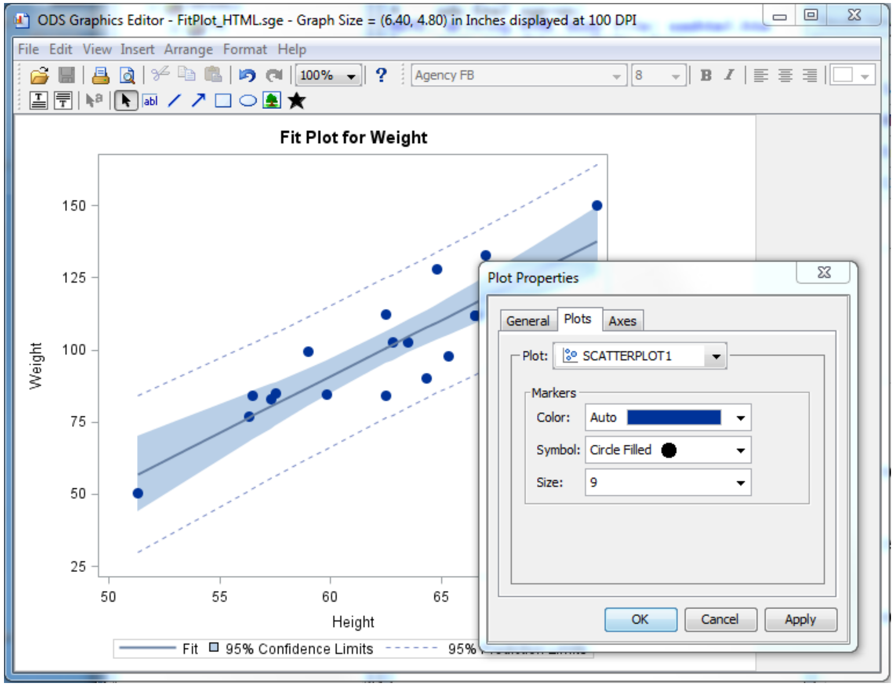
Fit Plot for Weight

Observations	19
Parameters	2
Error DF	17
MSE	126.03
R-Square	0.7705
Adj R-Square	0.757

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;
```



ODS Graphics Editor - FitPlot_HTML.sge - Graph Size = (6.40, 4.80) in Inches displayed at 100 DPI

Plot Properties

General | Plots | Axes

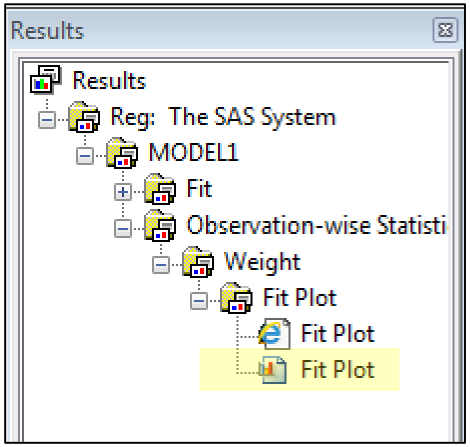
Plot: SCATTERPLOT1

Markers

Color: Auto

Symbol: Circle Filled ●

Size: 9



Results

- Results
  - Reg: The SAS System
    - MODEL1
      - Fit
      - Observation-wise Statisti
        - Weight
          - Fit Plot
            - Fit Plot
            - Fit Plot

Creating Custom Graphs

In SAS, you can create custom analytical graphs using one of the following 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 the features of GTL.

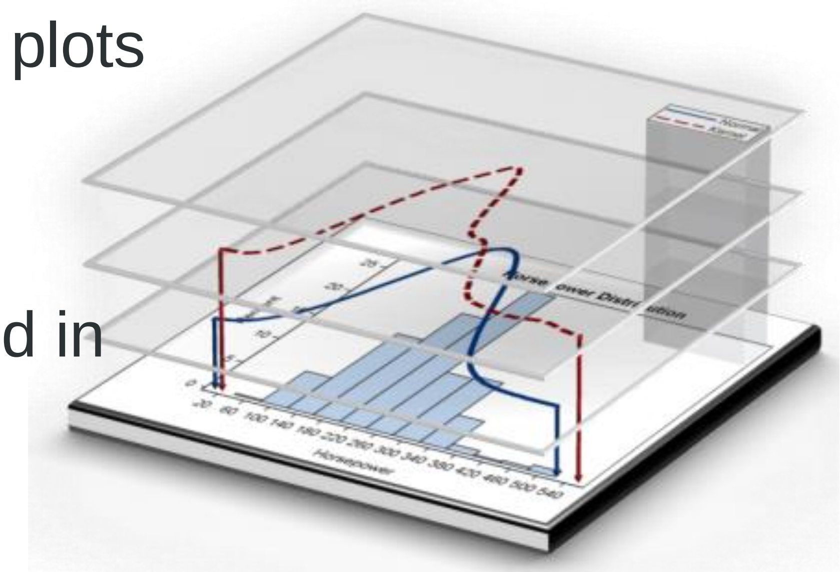
You Can Use GTL Directly

You can use GTL directly to create your custom graphs.

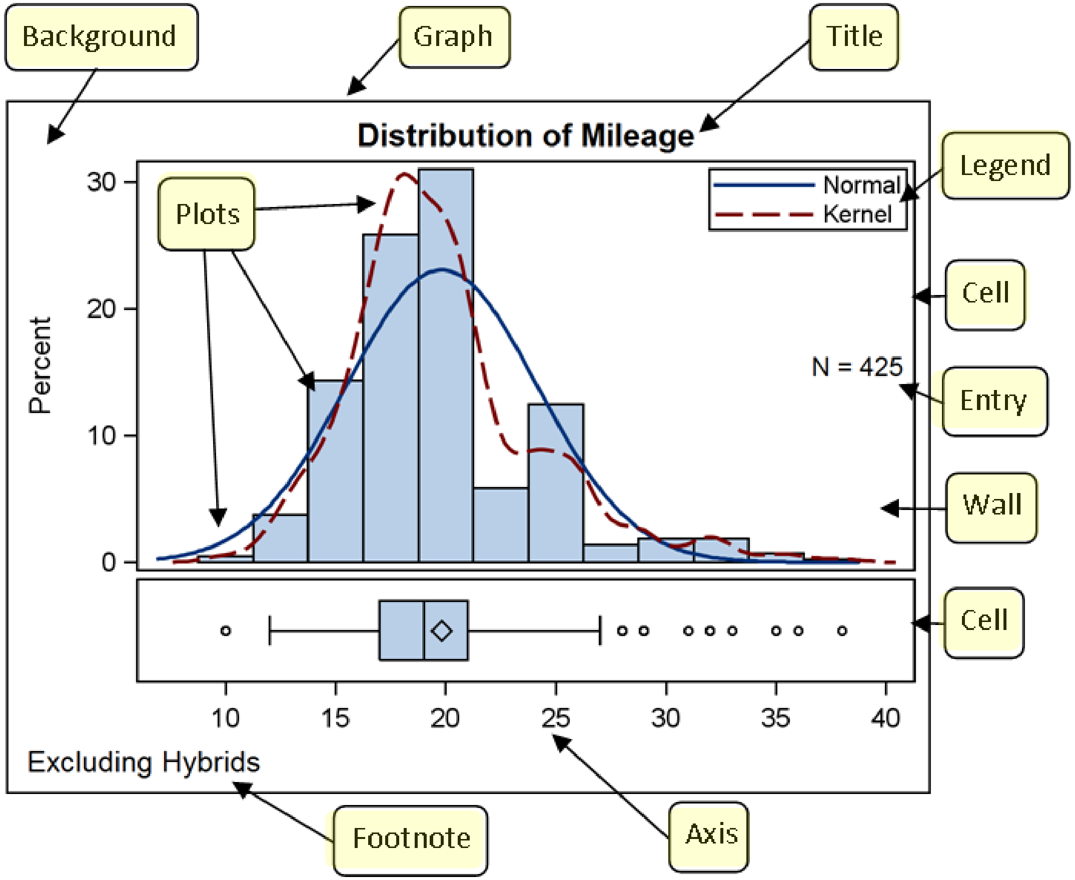
- You may need graph features beyond what are offered by the SG Procedures or Designer.
- You may want to customize the automatic graphs created by SAS analytical procedures.
- You may want to learn GTL because it is there.

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

So, without further ado, let us get started.

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.

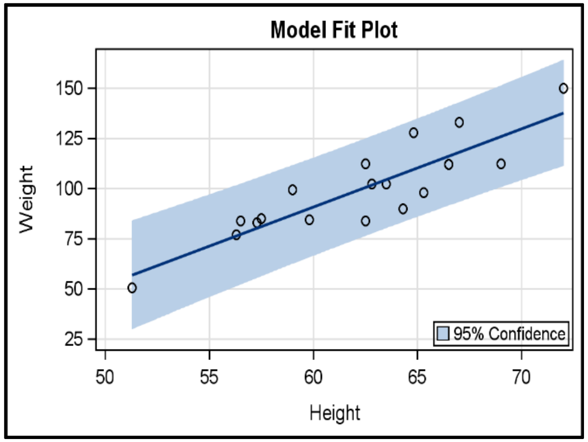
```
proc sgrender data=data-set-name
              template=template-name;
  <other optional statements>
run;
```

Now, the graph is created.

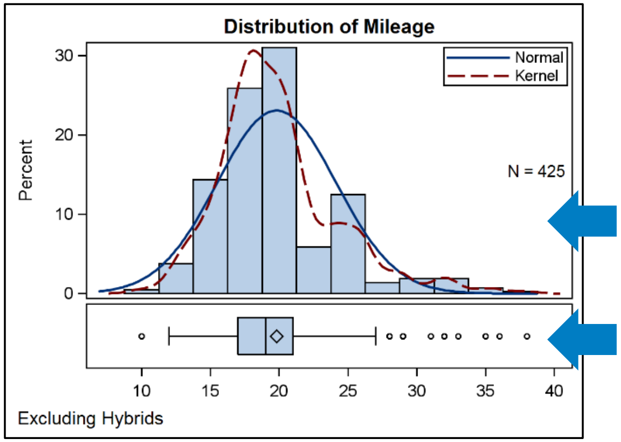
The same template can be used with different data sets to create graphs.

Types of Graphs

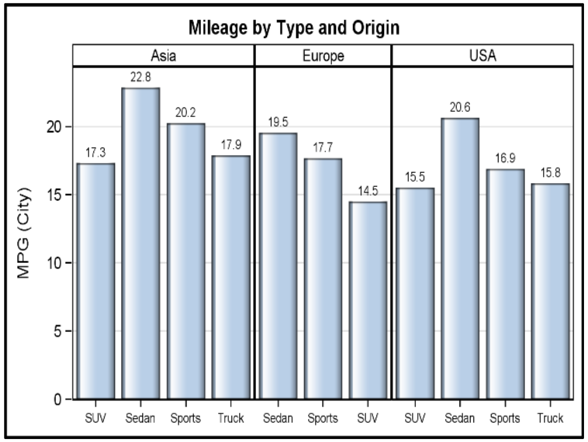
With GTL, you can create many type of graphs, such as the following.



Single-Cell Graph



Multi-Cell Graph

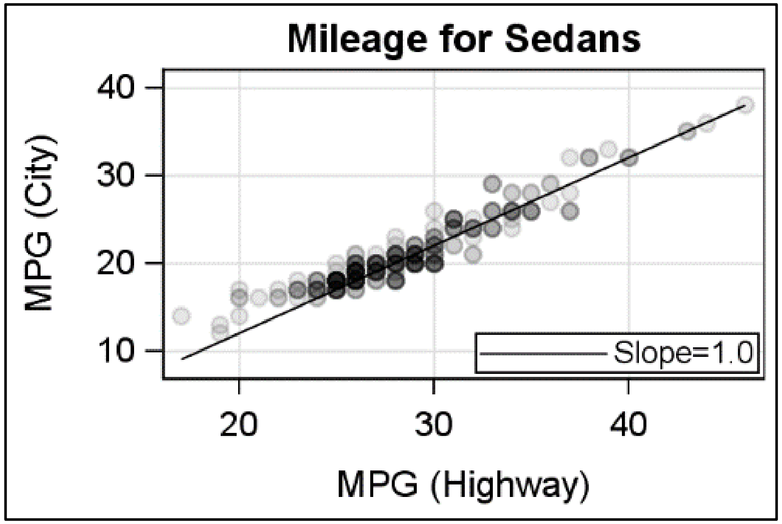


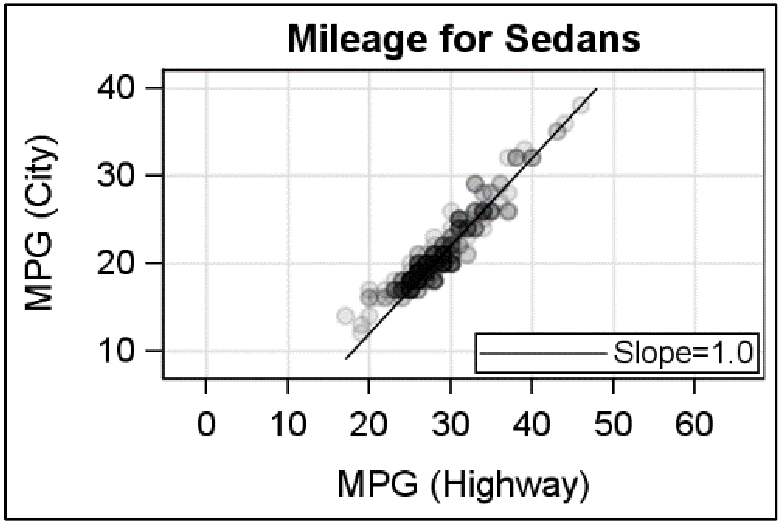
Classification Panel

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.

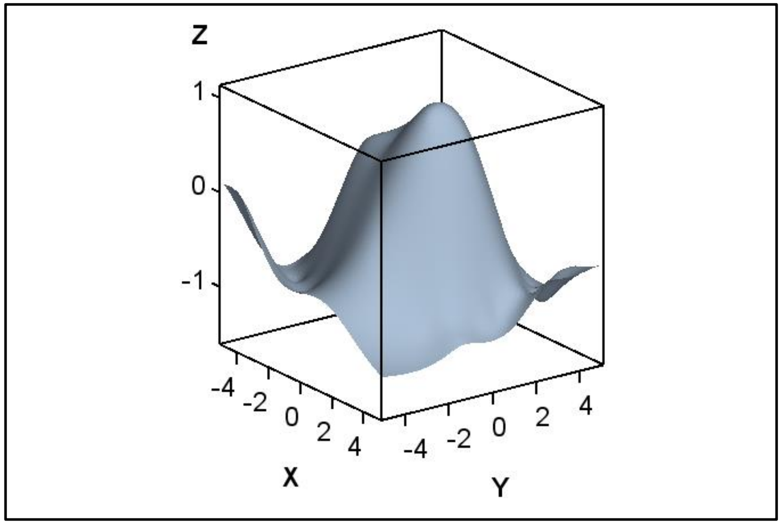


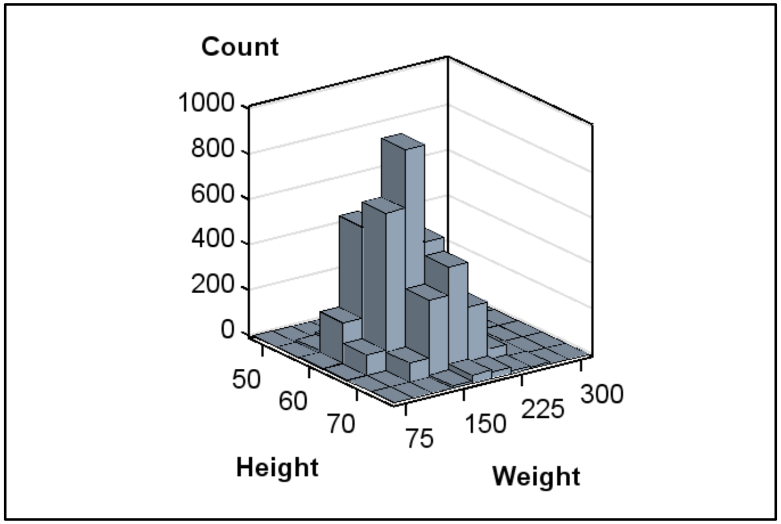


Single-Cell Graphs

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

- Layout Overlay3D – creates layered 3D graphs.

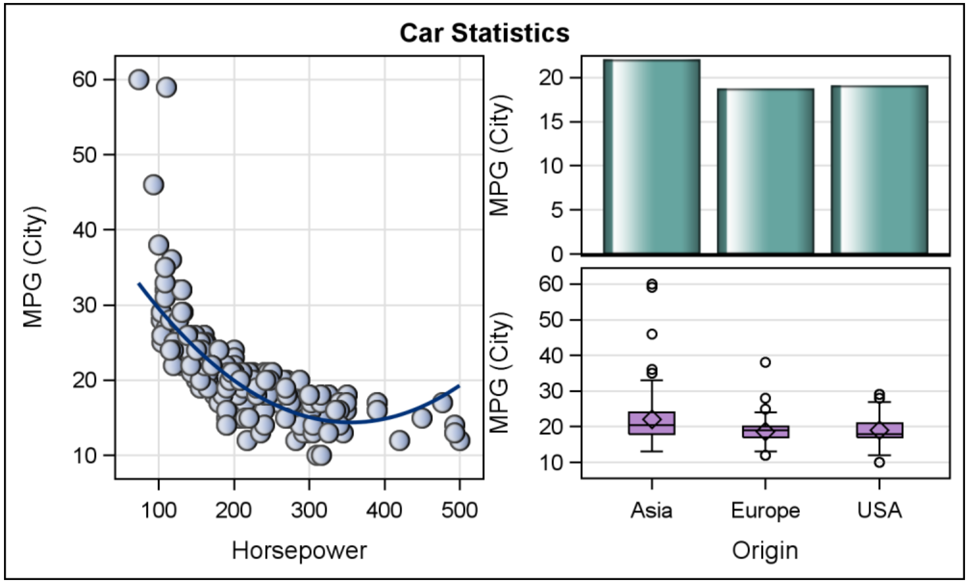




Multi-Cell (AdHoc) Graphs

Multi-Cell graphs are created using the following:

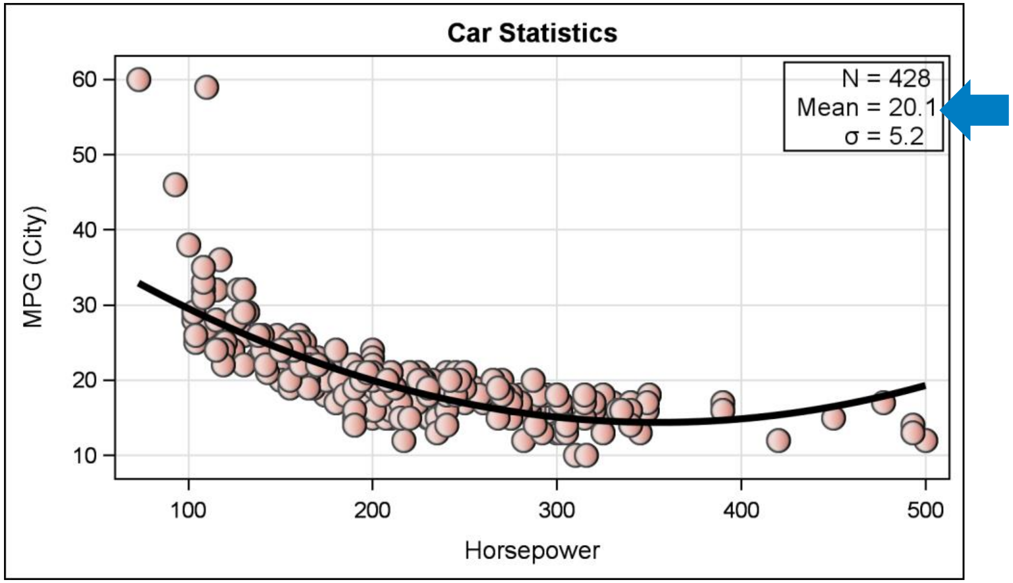
- Layout Lattice – divides the graph area into a regular grid of cells as specified by you. Each row and column can have different weights. Each cell can have nested layouts.
- In this example, the cell on the right uses a nested Layout Lattice.
- Multiple compatible plot statements can be added to each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



Multi-Cell (AdHoc) Layout

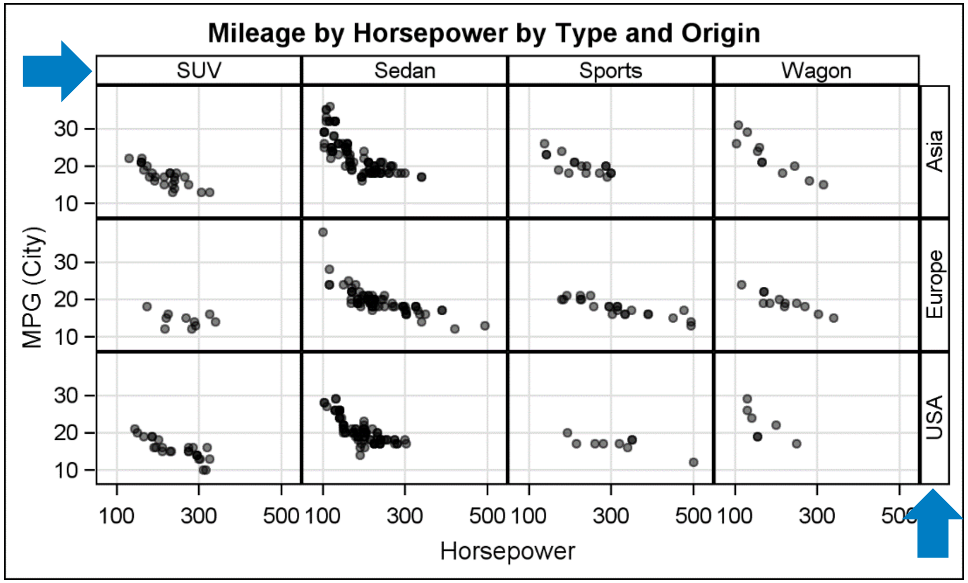
- Layout Gridded– divides the graph area in to a regular grid of cells. This layout is useful for creating statistics tables with text values.

- A Layout Gridded is nested in the top right corner of the cell to create the statistics table.



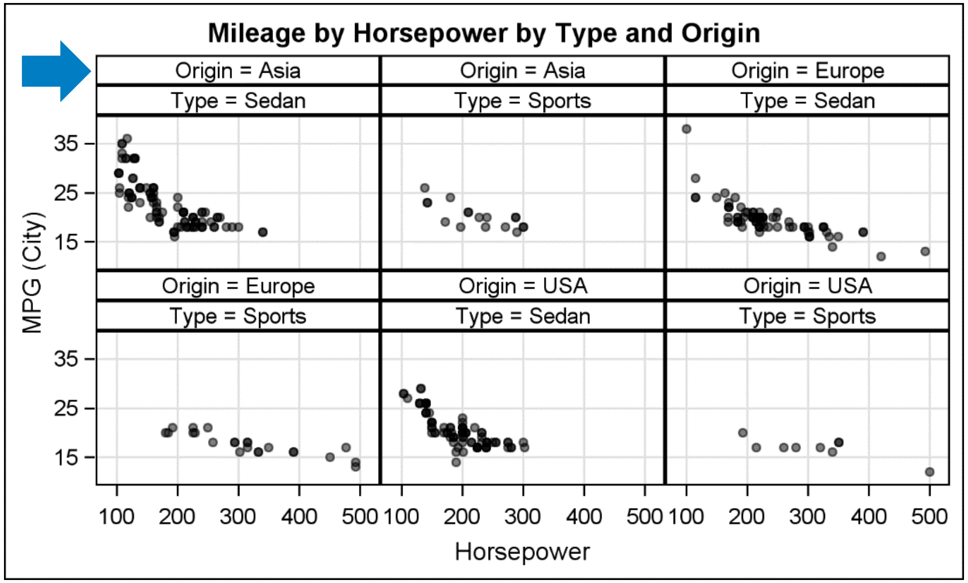
Multi-Cell Classification Lattice

- 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

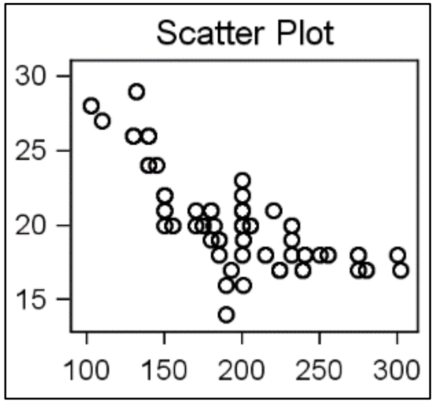
- Layout DataPanel– divides the graph area into a regular grid of cells based on multiple class variables.
- Each cell gets multiple headers.

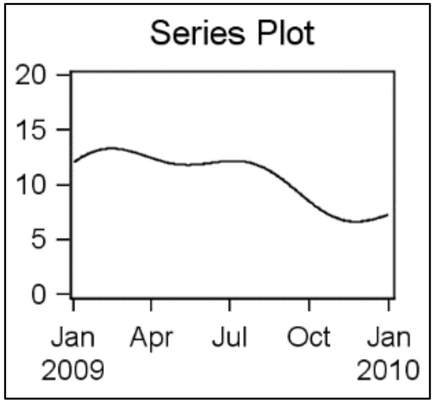


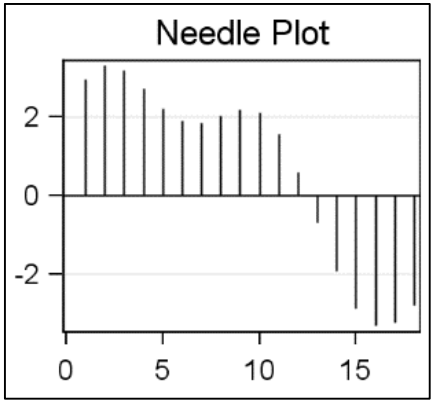
Plot Types

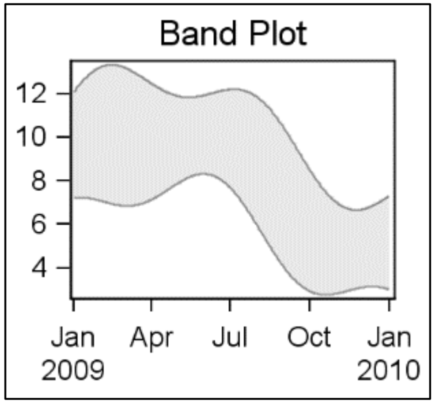
- Basic Plots: ScatterPlot, SeriesPlot, NeedlePlot,

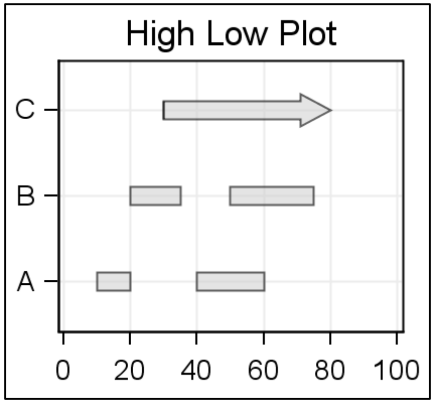
  BandPlot, HighLowPlot, BubblePlot.

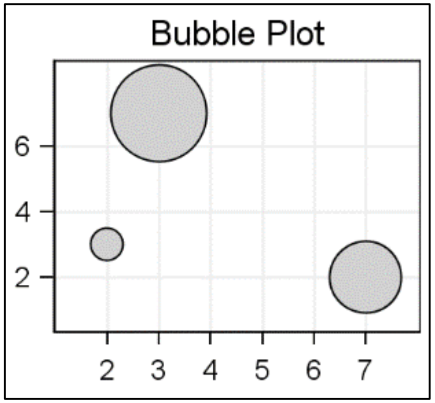






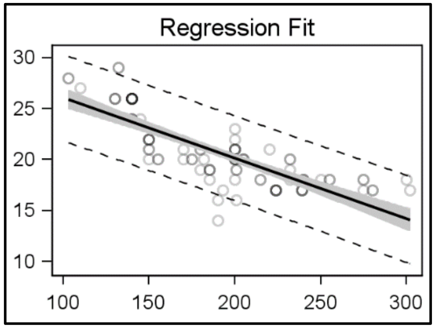


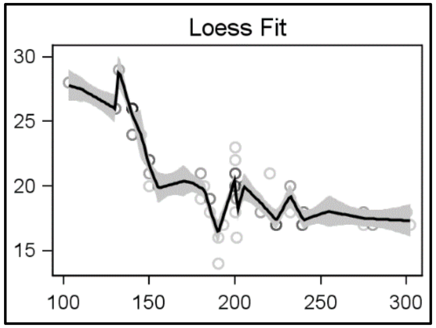


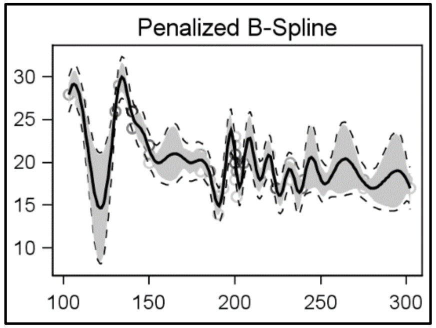


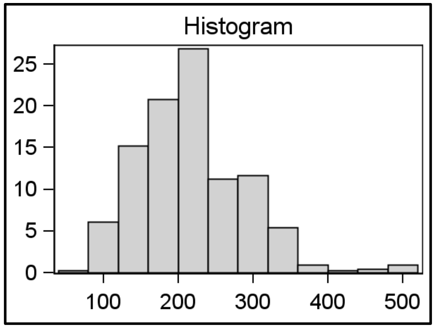
Plot Types

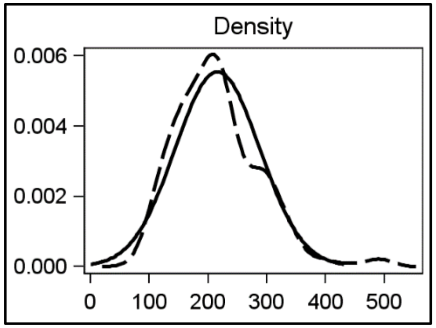
- Fit Plots: RegressionPlot, LoessPlot, PBSplinePlot.
- Distribution Plots: Histogram, DensityPlot, BoxPlot

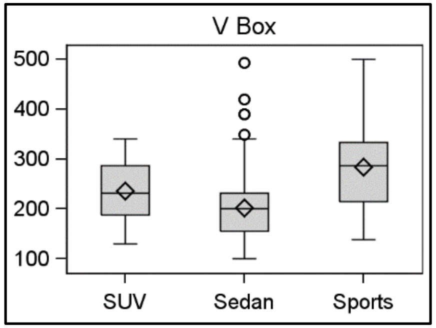








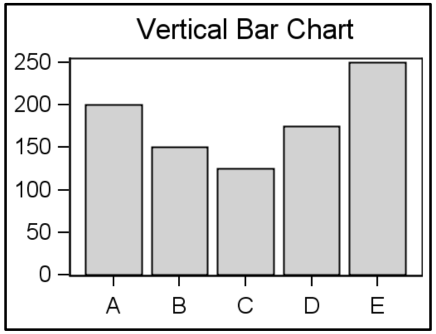


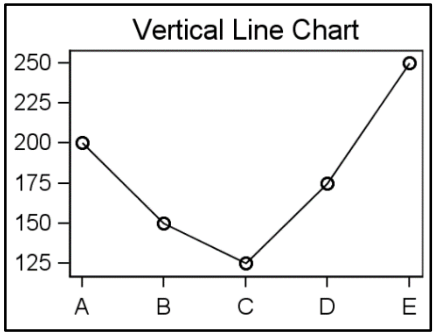


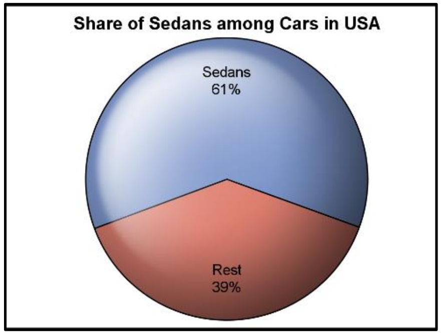
Plot Types

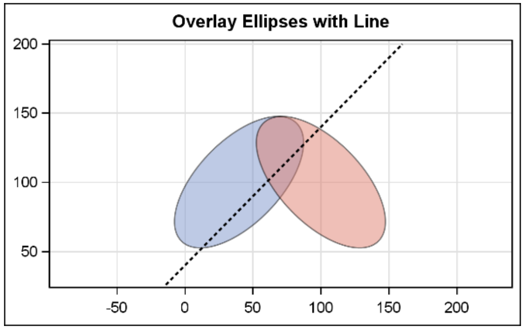
- Categorical Plots: BarChart, LineChart, PieChart.
- Parametric Plots: EllipseParm, LineParm, ContourParm,

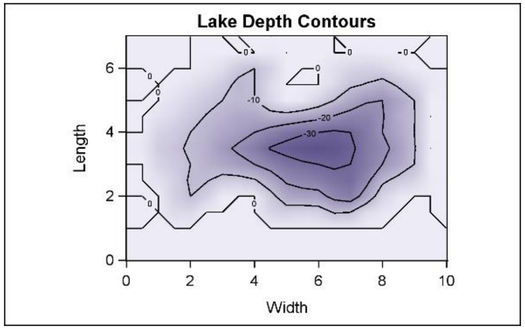
  BarChartParm, BoxPlotParm, HistogramParm.





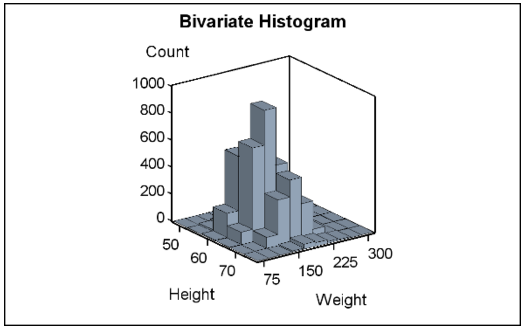


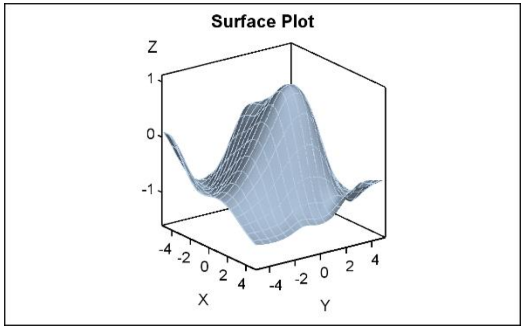


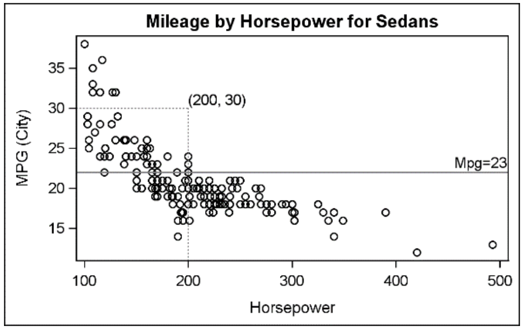


Plot Types

- 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.

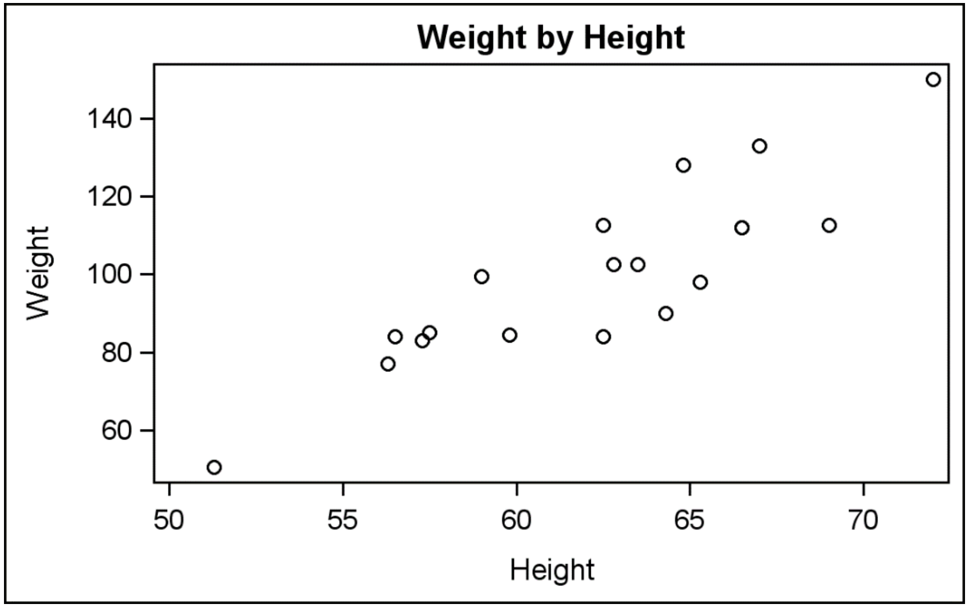
This is also the principle followed by the book “Getting Started with the Graph Template Language”.

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

Single Cell Graphs – Simple Example

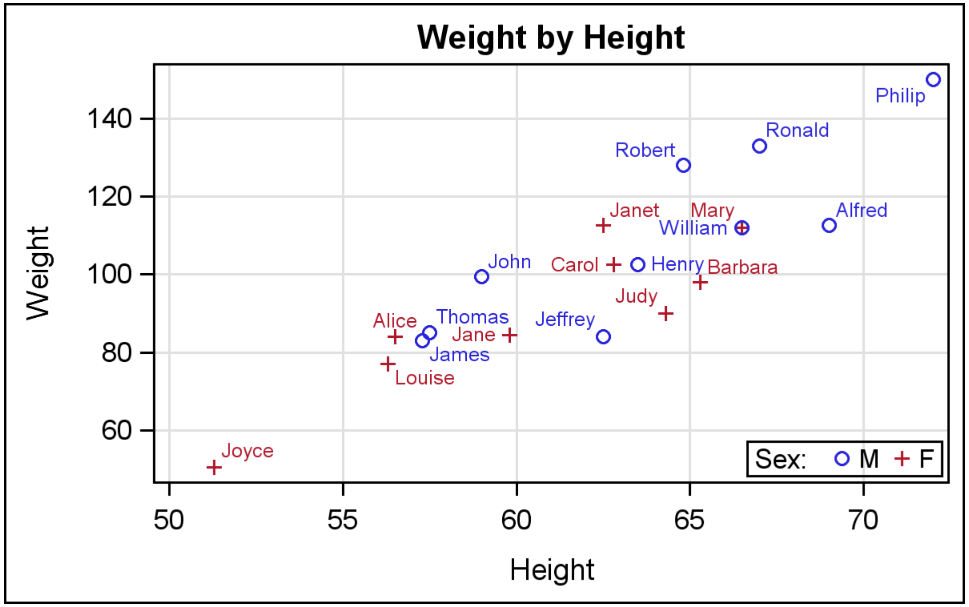
```
proc template;
  define statgraph scatter;
    begingraph;
      entrytitle 'Weight by Height';
      layout overlay;
        scatterplot x=height y=weight;
      endlayout;
    endgraph;
  end;
run;
```

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



Single Cell Graphs – Simple Example +

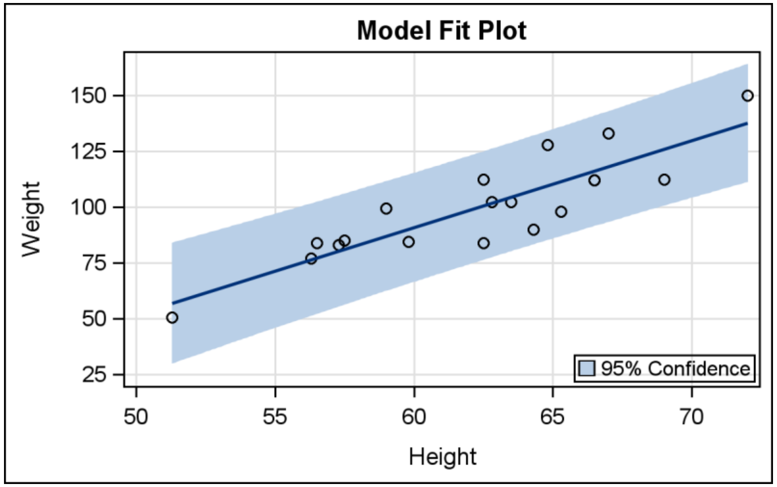
```
proc template;
  define statgraph Fig_4_2_1;
    begingraph;
      entrytitle 'Weight by Height';
      layout overlay / xaxisopts=(griddisplay=on)
                       yaxisopts=(griddisplay=on);
        scatterplot x=height y=weight / group=sex datalabel=name name='a';
        discretelegend '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

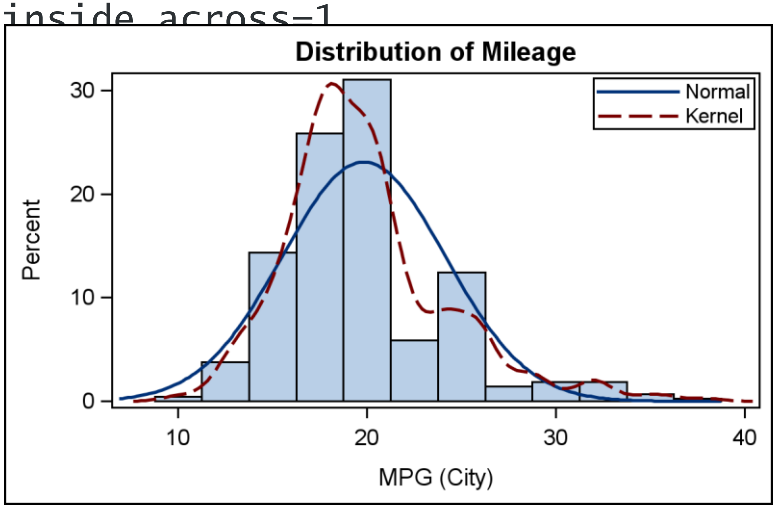
```
proc template;
  define statgraph Fig_4_7_1;
    begingraph;
      entrytitle 'Model Fit Plot';
      layout overlay / xaxisopts=(griddisplay=on)
                       yaxisopts=(griddisplay=on);
        modelband 'band' / display=(fill) name='b'
          legendlabel='95% Confidence';
        scatterplot 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;
```



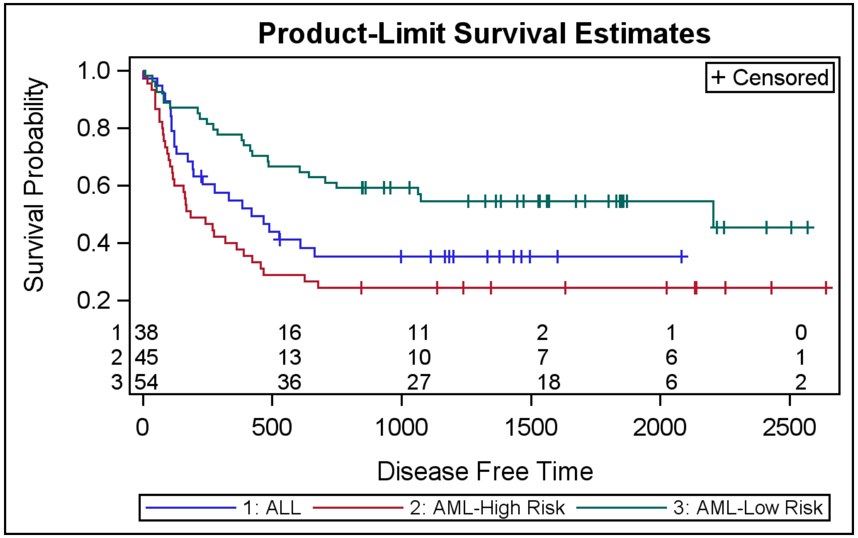
Single Cell Graphs – Distribution Plot

```
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;
```



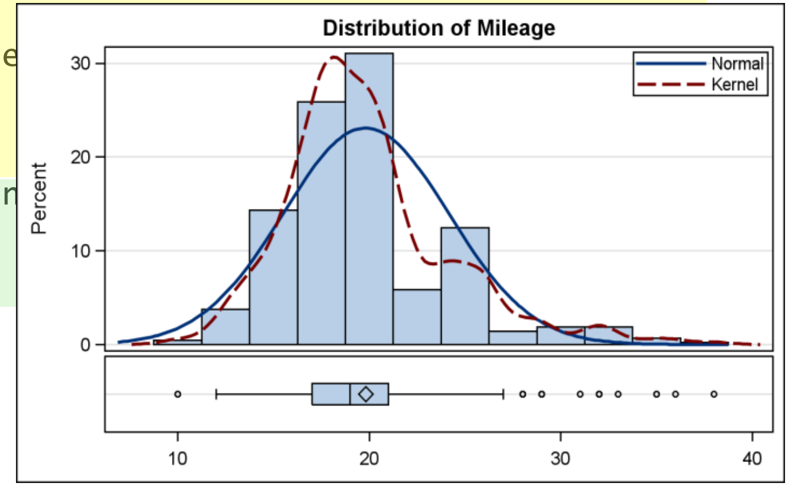
Survival Plot

```
proc template;
 define statgraph Fig_4_7_3;
  begingraph;
   entrytitle 'Product-Limit Survival Estimates';
   layout overlay;
    stepplot x=time y=survival / group=stratum lineattrs=(pattern=solid)  name='s';
    scatterplot x=time y=censored /  markerattrs=graphdatadefault(symbol=plus) name='c';
    scatterplot x=time y=censored / group=stratum markerattrs=(symbol=plus);
    innermargin;
      blockplot x=tatrisk block=atrisk / class=stratumnum
           display=(values label) valueattrs=(size=8)
           labelattrs=(size=8);
    endinnermargin;
    discretelegend 'c'/ location=inside halign=right valign=top;
    discretelegend 's' / valueattrs=(size=8);
   endlayout;
  endgraph;
 end;
run;

proc sgrender data=GTL_GS_SurvivalPlotData
                    template=Fig_4_7_3;
run;
```



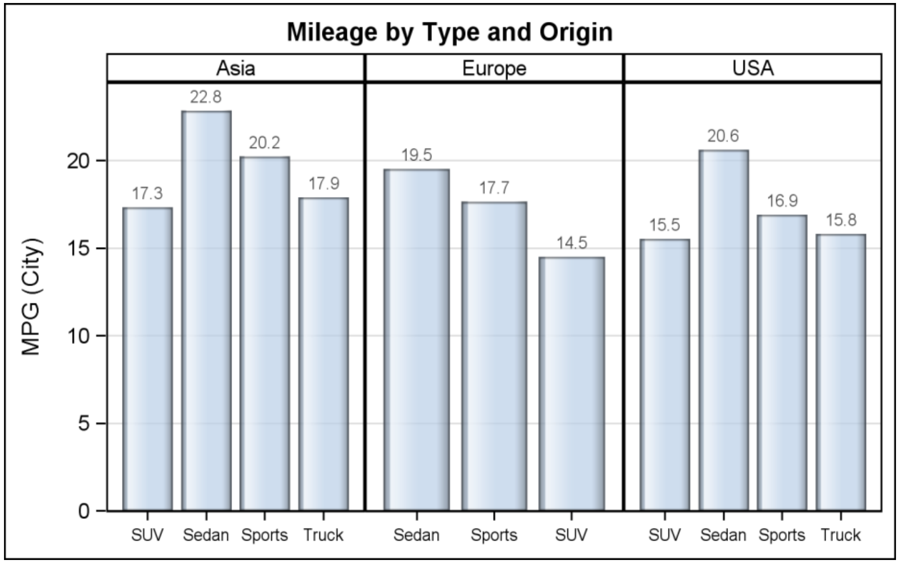
Multi Cell Graphs – Distribution Plot

```
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
              halign=right valign=top ;
        endlayout;
        layout overlay / yaxisopts=(griddisplay=on
          boxplot y=mpg_city / orient=horizontal;
        endlayout;
      endlayout;
    endgraph;
  end;
run;
```



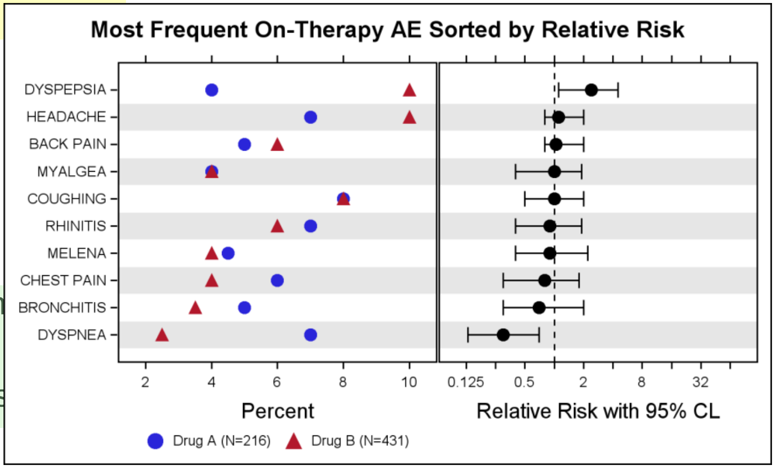
Classification Panel

```
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;
```



Most Frequent Adverse Events by Relative Risk

```
proc template;
  define statgraph Fig_5_5_1_MostFreqAE;
    begingraph / designwidth=8in designheight=4in;
      entrytitle "Most Frequent On-Therapy AE Sorted by Relative Risk";
      layout lattice / columns=2 rowdatarange=union;
         rowaxes;
           rowaxis / display=(ticks tickvalues line) tickvalueattrs=(size=7);
         endrowaxes;
         layout overlay / xaxisopts=(displaysecondary=(ticks);
           referenceline y=refae / lineattrs=(thickness=14) datatransparency=0.8;
          scatterplot x=a y=ae / name='a' legendlabel='Drug A (N=216)' ;
          scatterplot x=b y=ae / name='b' legendlabel='Drug B (N=431)‘;
           discretelegend 'a' 'b' / border=false valueattrs=(size=7);
         endlayout;
          layout overlay / xaxisopts=(label='Relative Risk with 95% CL'
                displaysecondary=(ticks) tickvalueattrs=(size=7)
                type=log logopts=(base=2 viewmin=0.125 viewmax=64
                tickintervalstyle=logexpand));
            referenceline y=refae / lineattrs=(thickness=16) datatransparen
           scatterplot x=mean y=ae / xerrorlower=low xerrorupper=high;
            referenceline x=1 / lineattrs=graphdatadefault(pattern=shortdas
           endlayout;
       endlayout;
     endgraph;
    end;   run;
```

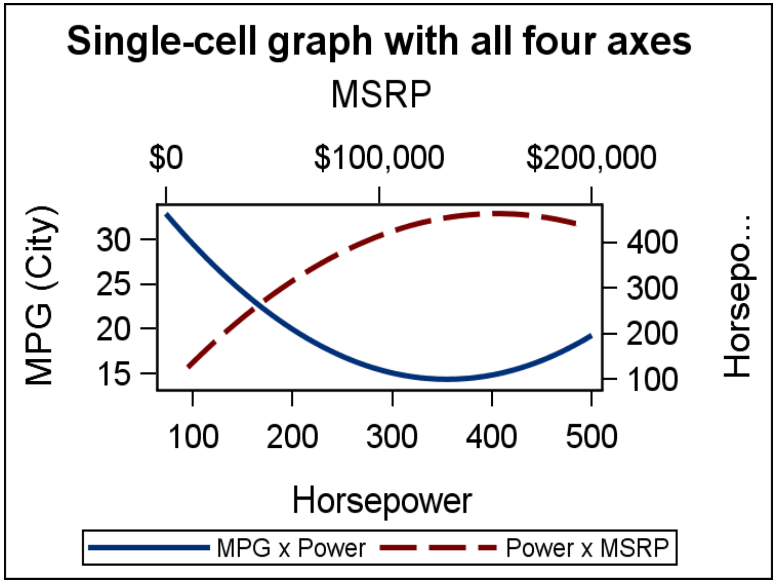


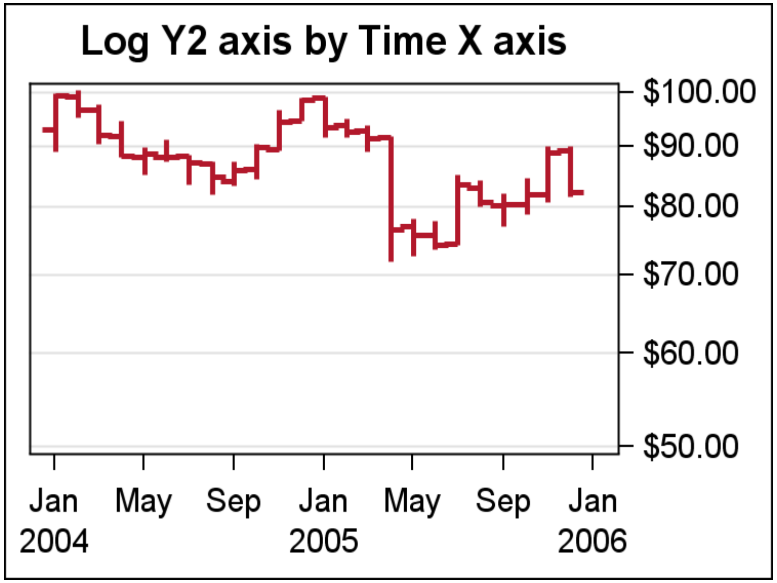
Axis Features

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.

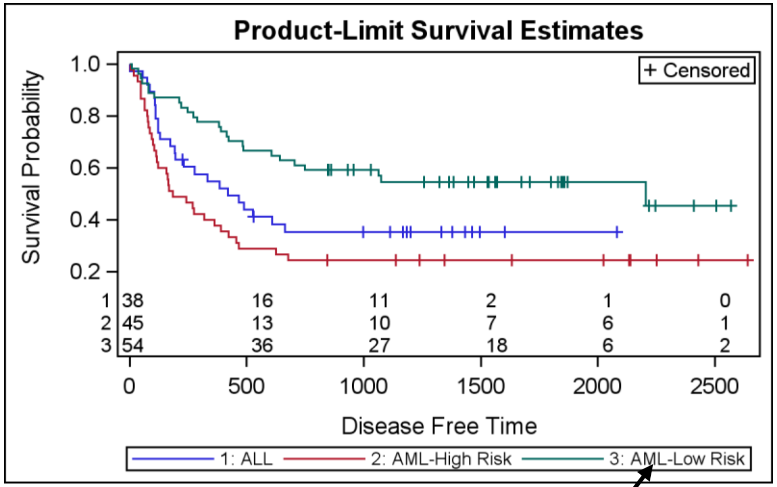
GTL supports multiple axis types such as Linear, Discrete, Time and Log. Multiple options are supported for each.



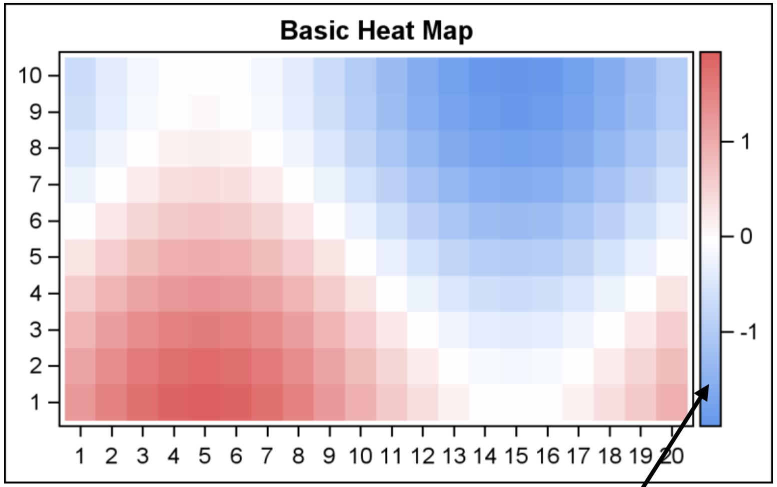


Legend Features

GTL supports discrete and continuous legends.



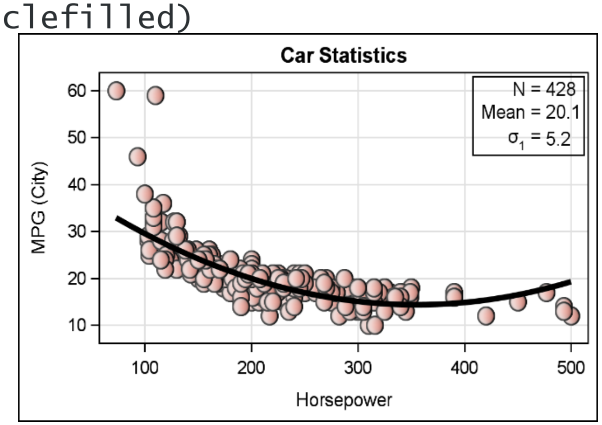
Discrete Legend



Continuous Legend

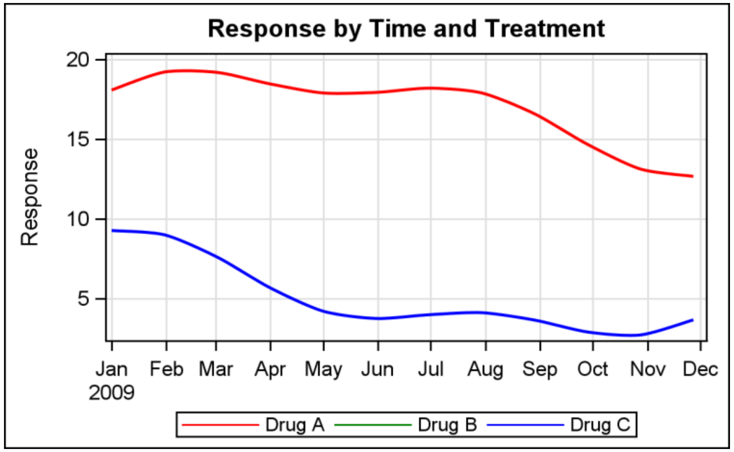
Entries, Unicode and Function Evaluation

```
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;
```



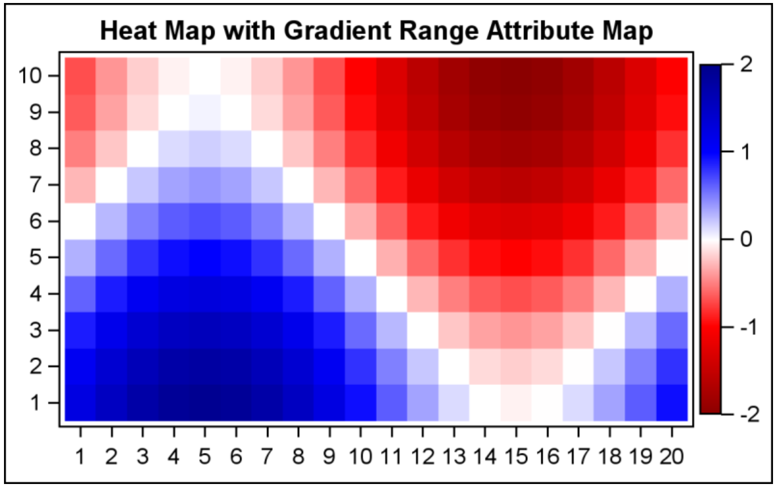
Discrete Attributes Map

```
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;
```



Range Attributes Map

```
proc template;
  define statgraph Fig_7_3_2;
    begingraph;
      rangeattrmap name='Resp';
        range -2 - -1 / rangecolormodel=(darkred red);
        range -1 -  0 / rangecolormodel=(red white);
        range  0 -  1 / rangecolormodel=(white blue);
        range  1 -  2 / rangecolormodel=(blue darkblue);
      endrangeattrmap;

      rangeattrvar attrvar=AttrValue var=value attrmap='Resp';

      entrytitle "Basic Heat Map";
      layout overlay /
           xaxisopts=(display=(ticks tickvalues))
           yaxisopts=(display=(ticks tickvalues));
        heatmapparm x=x y=y
           colorresponse=AttrValue / name='a';
        continuouslegend 'a';
      endlayout;
    endgraph;
  end;
run;
```



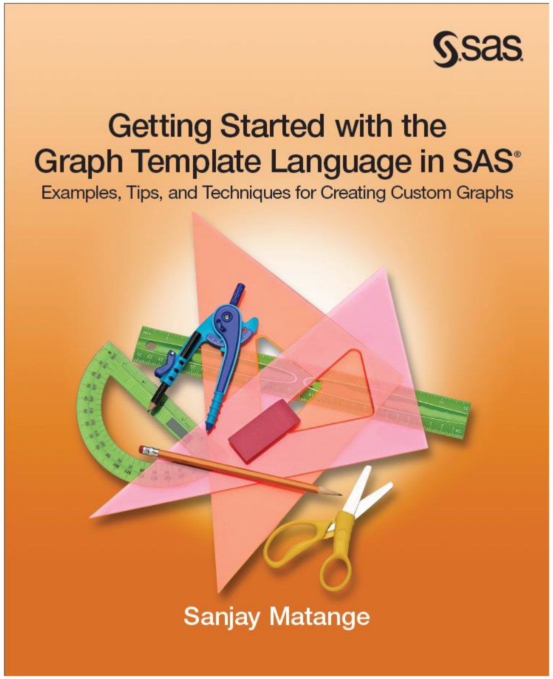
Review of this Presentation

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 graphs.

Helpful Resources

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



sas

Getting Started with the
Graph Template Language in SAS®

Examples, Tips, and Techniques for Creating Custom Graphs

Sanjay Matange

Graphically Speaking Blog, Visual Index

Stacked Bar with Labels 2013/09/20

Graphs with SAS Web Editor 2013/09/17

Bars w/ Stack & Cluster Groups 2013/09/07

Series Plot with a Twist 2013/08/24

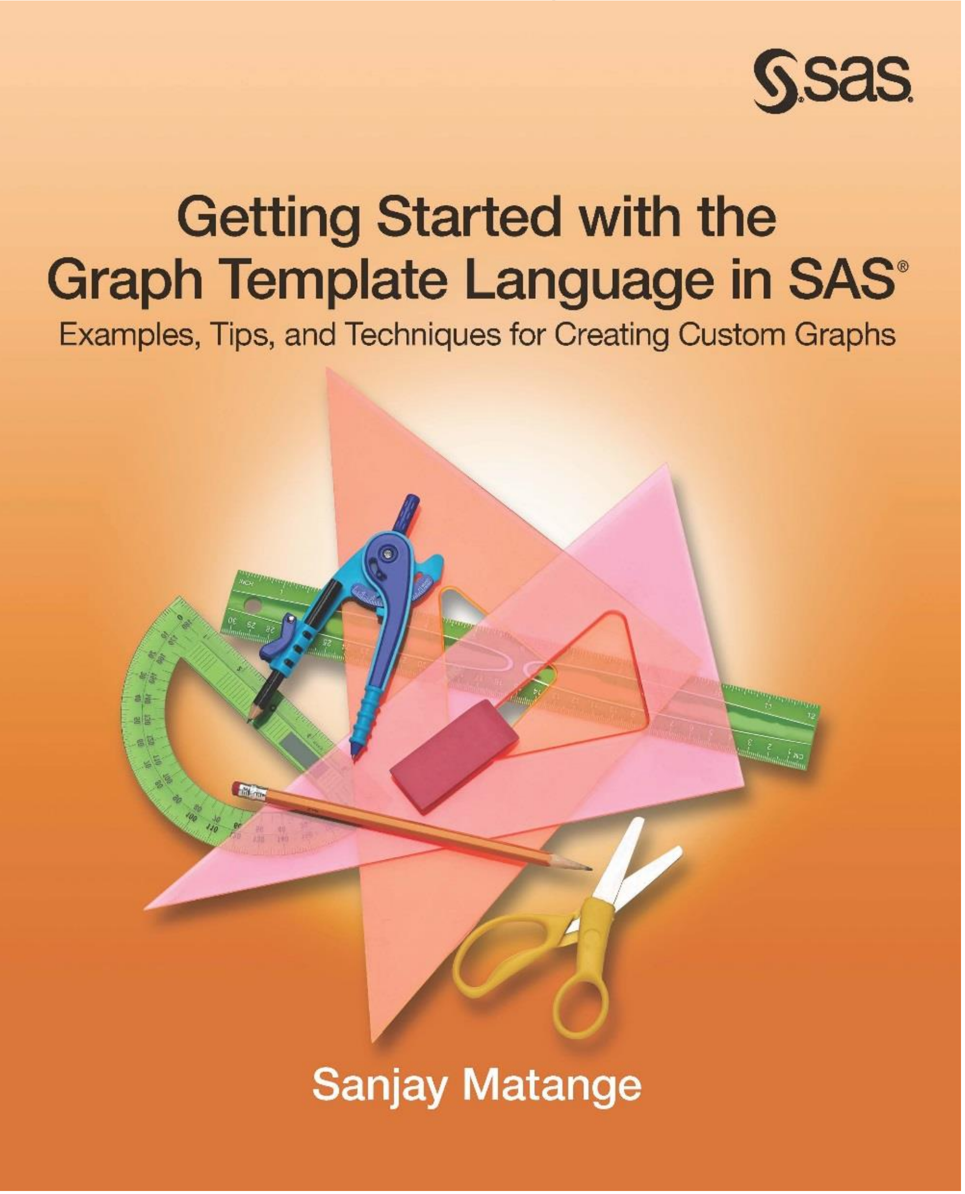
Video Says It Better 2013/08/19

Customize Plot Appearance 2013/08/11

A Better Combined AE - CM Graph 2013/08/05

A Combined AE - CM Graph 2013/07/31

Setting Group Colors 2013/07/14



Getting Started with the Graph Template Language in SAS®

Examples, Tips, and Techniques for Creating Custom Graphs

Sanjay Matange

- Book available November 1.
- Preorder from the SAS Bookstore from October 21st to 25th and use promo code “BOOK20” to receive a 20% discount.

SAS GLOBAL FORUM

SAS Global Forum 2014

Washington, D.C.

March 23-26, 2014
www.sasglobalforum.org