Paper TS02

SAS GTL – Injecting New Life into Graphs

Lawrence Heaton-Wright, Quintiles

ABSTRACT

The SAS/Graph® Graph Template Language (GTL) was introduced into SAS® version 9 as an extension to ODS and is a different way of creating graphs. GTL is at the heart of ODS GRAPHICS (Statistical Graphics associated with some SAS/Stat procedures) which enables some sophisticated analytical modelling to be produced with simple commands.

Traditional SAS/Graph produces some excellent graphics, especially if utilising the annotate functionality, but is often seen as complicated (or even "clunky") as well as requiring specialist expertise from your resident "Graph guru".

This paper will examine some graphical tasks that can be streamlined using GTL compared to using traditional SAS/Graph code.

WHAT IS GTL?

SAS/Graph Graph Template Language (GTL) is an addition to Version 9. The aim of this paper is to compare some common graphical programming tasks using the new GTL and traditional SAS/Graph techniques. This paper will also demonstrate some of the more complex graphics can be produced by GTL but which would require a significant programming effort using traditional SAS/Graph.

Traditional SAS/Graph code is seen by many programmers as a complex, specialist area and it can be, especially when the annotate system is used to enhance the basic graphs produced by SAS/Graph.

GTL allows many of these complex tasks to be broken down into simpler graphical tasks and combined into one graphical output. The area of combining multiple graphs onto one page is one of the complex areas of traditional SAS/Graph coding requiring extensive knowledge of the basic graph procedures alongside the annotate system, GSLIDE, GREPLAY and the use of graphics catalogs.

SIMPLE PLOTS

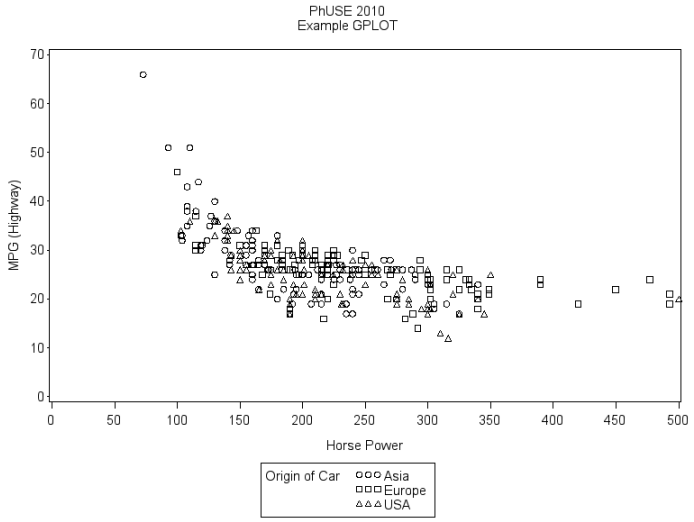
How does GTL differ from traditional SAS/Graph?

In traditional code you generate the data you want to plot, then you define graphics options, axes, legends and symbols. Then the procedure is built (GPLOT, GCHART, etc.) to display the data set in graphical form.

```
GOPTIONS RESET=GOPTIONS
         DEVICE=WIN TARGETDEVICE=PNG
         FTEXT="Arial" HTEXT=11pt HBY=0 HTITLE=11pt CPATTERN=GREY;

AXIS1 ORDER=(0 TO 500 BY 50) MINOR=NONE LABEL=("Horse Power");
AXIS2 ORDER=(0 TO 70 BY 10) MINOR=NONE LABEL=(A=90 "MPG (Highway)");

SYMBOL1 COLOR=BLACK VALUE=CIRCLE;
SYMBOL2 COLOR=BLACK VALUE=SQUARE;
SYMBOL3 COLOR=BLACK VALUE=TRIANGLE;

LEGEND1 FRAME ACROSS=1 LABEL=("Origin of Car");

TITLE1 "PhUSE 2010";
TITLE2 "Example GPLOT";
```

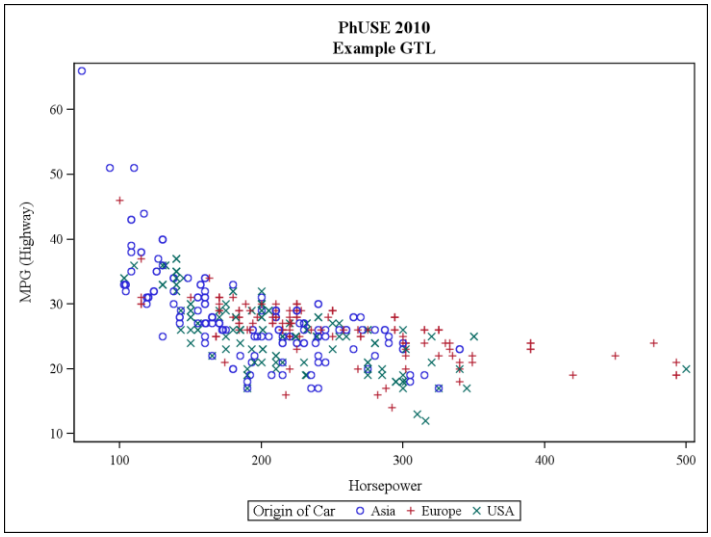
The A=90 option for the AXIS2 LABEL statement is required to rotate the label through 90 degrees.

```
PROC GPLOT DATA=cars;
  PLOT mpg_highway * horsepower = origin / HAXIS=AXIS1 VAXIS=AXIS2 LEGEND=LEGEND1;
RUN;QUIT;
```



In GTL, again you generate the data you want to plot but then you develop a template graph and then render the data into the template you've created. The template contains the plot type, titles and footnotes, axes, legends and symbols. There are some other procedures (SGPLOT, SGPANEL) but TEMPLATE and SGRENDER have been used because there are more options available to programmers and the templates themselves can be stored and reused.

```
PROC TEMPLATE;
  DEFINE STATGRAPH scatter;
    BeginGraph;
      EntryTitle "PhUSE 2010";
      EntryTitle "Example GTL";
      Layout OVERLAY;
        ScatterPlot X=horsepower Y=mpg_highway / GROUP=origin NAME="scatter";
        DiscreteLegend "scatter" / ACROSS=3 TITLE="Origin of Car";
      EndLayout;
    EndGraph;
  END;
RUN;

PROC SGRENDER DATA=cars TEMPLATE=scatter;
RUN;
```



Note that the Y-axis label has been rotated through 90 degrees without programming as this is a default action. The symbols have been cycled through both colours and shape, again without user interaction.
There are similarities between PROC TEMPLATE and traditional SAS/Graph code. Note how the legend statement for PROC GPLOT is constructed in a similar manner to the DISCRETELEGEND statement in PROC TEMPLATE.

The differences are improved clarity of code for inexperienced coders. Consider the SCATTERPLOT statement. We know that this will produce a scatter plot (other types of graph produced are histogram, barchart, boxplot). We know exactly what variable is to be plotted on the X and Y axes (X=horsepower, Y=mpg_highway). This makes many parts of GTL far easier for the novice graphics programmer to understand.

COMMON TASKS

One of the most common graphs we have to produce is a series plot with error bars at each time point for multiple groups.

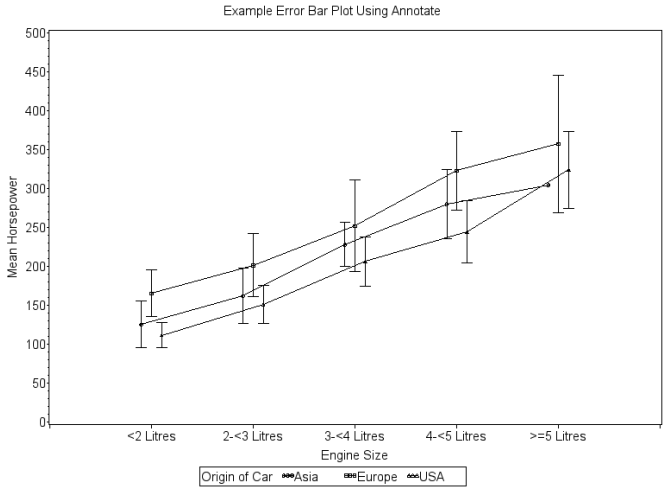
In traditional code this can be achieved using the annotate system and a Y x X = Z plot.

```
ODS LISTING CLOSE;
PROC MEANS DATA=cars;
  CLASS origin engine;
  VAR horsepower;
  OUTPUT OUT=summary (WHERE=(_type_ EQ 3)) N=n MEAN=mean STDDEV=stddev MIN=min
MAX=max;
RUN;
ODS LISTING;

DATA summary;
  SET summary;
  IF UPCASE(origin) EQ 'ASIA' THEN engine = engine-0.1;
  IF UPCASE(origin) EQ 'USA' THEN engine = engine+0.1;
RUN;

%annomac;

DATA anno;
  SET summary;
  %SYSTEM(2, 2);
  IF stddev GT 0 THEN DO;
    %LINE(engine, mean-stddev, engine, mean+stddev, black, 1, 1);
    %LINE(engine-0.05, mean-stddev, engine+0.05, mean-stddev, black, 1, 1);
    %LINE(engine-0.05, mean+stddev, engine+0.05, mean+stddev, black, 1, 1);
  END;
RUN;

AXIS1 ORDER=(0 TO 6) MINOR=NONE LABEL=("Engine Size");
AXIS2 ORDER=(0 TO 500 BY 50) LABEL=(A=90 "Mean Horsepower");

SYMBOL1 COLOR=BLACK I=JOIN VALUE=CIRCLE;
SYMBOL2 COLOR=BLACK I=JOIN VALUE=SQUARE;
SYMBOL3 COLOR=BLACK I=JOIN VALUE=TRIANGLE;

LEGEND1 FRAME LABEL=("Origin of Car");

TITLE2 "Example Error Bar Plot Using Annotate";
PROC GPLOT DATA=summary;
  PLOT mean * engine = origin /
           HAXIS=AXIS1 VAXIS=AXIS2 LEGEND=LEGEND1 ANNOTATE=anno;
RUN;QUIT;
```

This off-setting of the X-axis variable is to ensure that the plotted symbols and error bars do not overwrite each other. This same data is used in the GTL code.

Invoke the annotate system macros supplied with SAS. These macros enable several common tasks using the annotate system.

This annotation data set produces the vertical lines between the top and bottom limits, as well as the horizontal tick marks at either end.



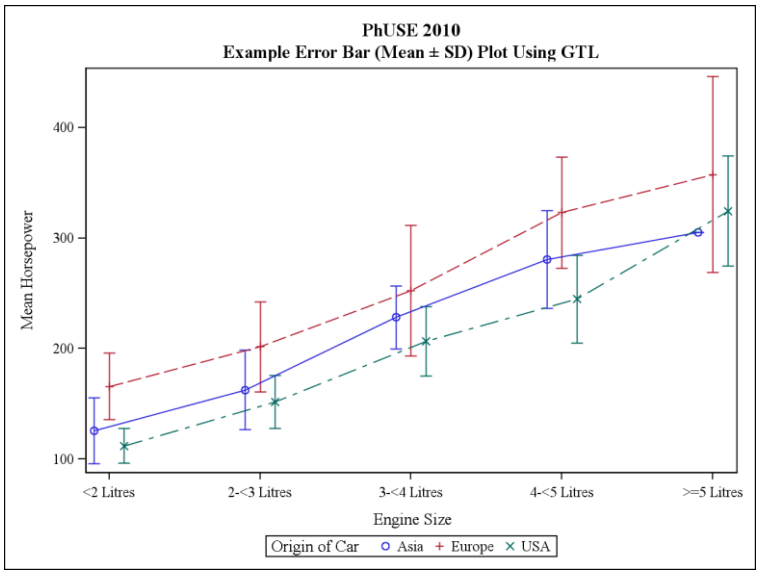
Using the new, funky GTL this can be produced with 2 procedures, PROC TEMPLATE and SGRENDER. This is because you can add calculations within the template of the figure.

```
PROC TEMPLATE;
  DEFINE STATGRAPH errorbar;
    BeginGraph;
      EntryTitle "PhUSE 2010";
      EntryTitle "Example Error Bar (Mean " {UNICODE '00B1'X}
                 " SD) Plot Using GTL";
      Layout OVERLAY /
          XAxisOpts=(LABEL="Engine Size")
          YAxisOpts=(LABEL="Mean Horsepower");

        ScatterPlot X=engine Y=mean /
            GROUP=origin NAME="scatter"
            YErrorLower=EVAL(mean-stddev)
            YErrorUpper=EVAL(mean+stddev);

        SeriesPlot X=engine Y=mean / GROUP=origin;

        DiscreteLegend "scatter" / ACROSS=3 TITLE="Origin of Car";

      EndLayout;
    EndGraph;
  END;
RUN;

PROC SGRENDER DATA=summary TEMPLATE=errorbar;
RUN;
```

The UNICODE statement inserts ± into the title line

The scatter plot plots the mean values as well as the error bars. The EVAL functions are performing the same calculation as in the annotate macros previously.

The series plot is overlaid to apply the joined mean points.



One of the important things to notice with this error bar plot is the ease with which the different types of plot are overlaid. This ease of use means many different types of GTL plot statements can be added including horizontal and vertical reference lines, histograms, curves of best fit, etc.

MULTIPLE PLOTS ON ONE PAGE

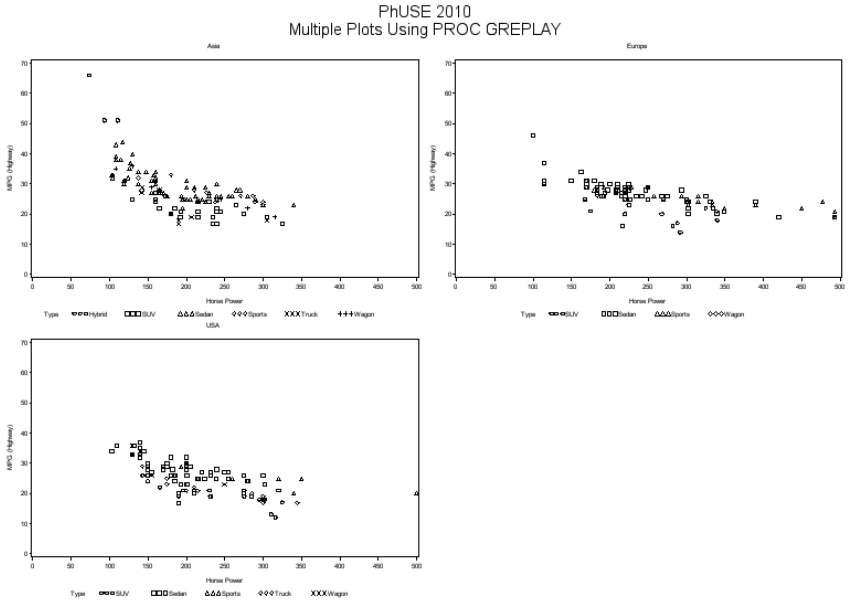
Experienced SAS/Graph users are familiar with PROC GREPLAY. This allows users to replay previously produced graphs into a template. Typically this is used to produce an output with multiple figures on one page. Producing such complex graphical displays involves several steps as well as a certain amount of "fooling" SAS. The use of GSLIDE to produce titles and footnotes valid for the entire graphical output rather than each individual graph is one such trick.

The (typical) steps used to produce a GREPLAY are as follows:

1. Produce plots and send to a graphics catalog
2. Produce a title and footnote output using GSLIDE
3. Produce a replay template and store in a template catalog
4. Replay plots and slide into previously designed template using GREPLAY

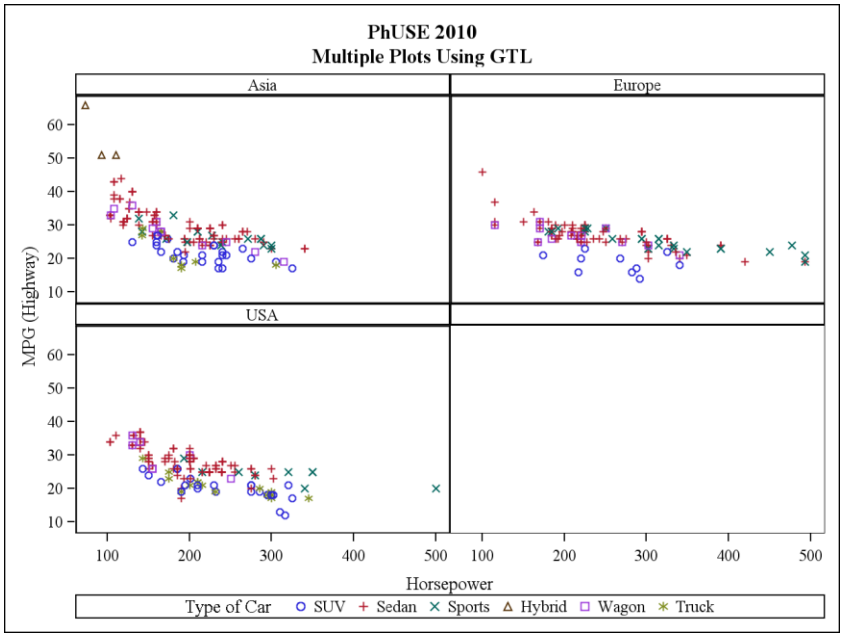
```
PROC SORT DATA=cars;
  BY origin;
RUN;

*** Step 1 Produce plots and send to a graphics catalog [GRAPHT] ***;
GOPTIONS NODISPLAY;
PROC GPLOT DATA=cars GOUT=grapht;
  TITLE1 "#BYVAL(ORIGIN)";
  BY origin;
  PLOT mpg_highway * horsepower = type / HAXIS=AXIS1 VAXIS=AXIS2;
RUN;QUIT;

*** Step 2 Produce a title and footnote output using GSLIDE ***;
PROC GSLIDE GOUT=grapht;
  TITLE1 "PhUSE 2010";
  TITLE2 "Multiple Plots Using PROC GREPLAY";
RUN;QUIT;
TITLE;
```

```
*** Step 3 Produce a replay template and store in a template catalog [TEMPCAT] ***;
PROC GREPLAY TC=tempcat NOFS;
  TDEF fourGS DES='Four Plots + GSLIDE'
  1 / LLX=0  LLY=50 ULX=0  ULY=94  LRX=50  LRY=50 URX=50  URY=94
  2 / LLX=50 LLY=50 ULX=50 ULY=94  LRX=100 LRY=50 URX=100 URY=94
  3 / LLX=0  LLY=6  ULX=0  ULY=50  LRX=50  LRY=6  URX=50  URY=50
  4 / LLX=50 LLY=6  ULX=50 ULY=50  LRX=100 LRY=6  URX=100 URY=50
  5 / DEF;
  TEMPLATE fourGS;
  LIST TEMPLATE;
RUN; QUIT;

GOPTIONS RESET=GOPTIONS DISPLAY
         DEVICE=PNG TARGETDEVICE=PNG
         FTEXT="Arial" HTEXT=11pt HBY=0 HTITLE=11pt CPATTERN=GREY;

*** Step 4 Replay plots and slide into previously designed template ***;
***        using GREPLAY ***;
PROC GREPLAY IGOUT=grapht GOUT=grapht TC=tempcat NOFS;
  LIST IGOUT;
  TEMPLATE=fourGS;
  TREPLAY
    1: gplot1
    2: gplot2
    3: gplot3
    5: gslide
  ;
RUN; QUIT;
```



GTL allows users to define layouts which allow for far more dynamic multiple plotting on several pages.

```
PROC TEMPLATE;
  DEFINE STATGRAPH layoutdatapanel;
    BeginGraph;
      EntryTitle "PhUSE 2010";
      EntryTitle "Multiple Plots Using GTL";
      Layout DataPanel ClassVars=(origin) /
          COLUMNS=2 ROWS=2 RowDataRange=UNIONALL HeaderLabelDisplay=VALUE;
        Layout Prototype / CycleAttrs=TRUE;
          ScatterPlot X=horsepower Y=mpg_highway / GROUP=type NAME="scatter";
        EndLayout;
        Sidebar;
          DiscreteLegend "scatter" / TITLE="Type of Car";
        EndSidebar;
      EndLayout;
    EndGraph;
  END;
RUN;

PROC SGRENDER DATA=cars TEMPLATE=layoutdatapanel;
RUN;
```



There are several differences between the 2 graphics produced but the GTL version has the advantage that if the number of plots produced are more than the number of “slots” available then a second (and subsequent) pages will be produced without user interaction.

This means that, for instance, if the group variable is changed to one with more than 4 different groups then the additional groups would be presented on a new page. Use GREPLAY a second PROC GREPLAY statement replaying the additional GPLOTs would be required. This means that the programmer would have to interrogate the graphics catalog to identify how many plots were produced and then build macro code around the replay section to display all the available graphs.

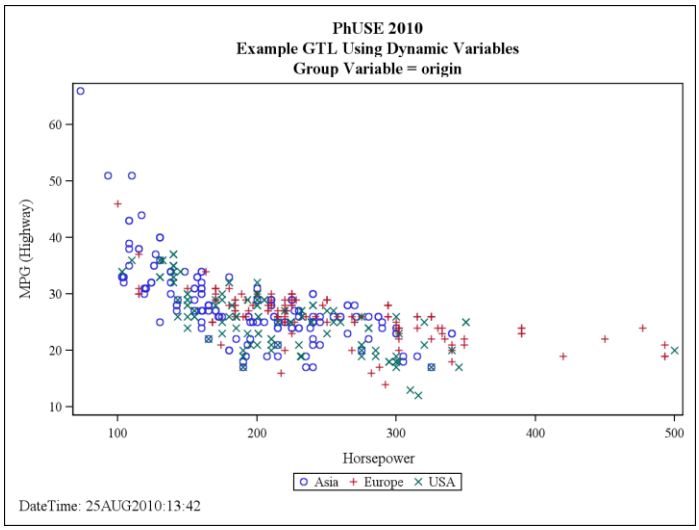
DYNAMIC TEMPLATES

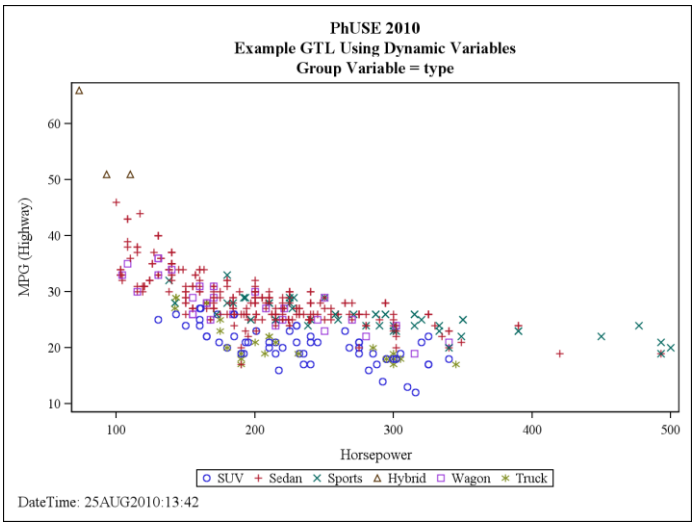
The templates produced for GTL allow users to create dynamic templates which can be stored and re-used across a variety of projects. Expressions and functions can be evaluated, dynamic and macro variables can be passed through and conditional logic can be applied, all at run time.

```
PROC TEMPLATE;
  DEFINE STATGRAPH scatter_dyn;
    BeginGraph;
      MVar SysDate9 SysTime;
      Dynamic XVar YVar GrpVar;
      EntryTitle "PhUSE 2010";
      EntryTitle "Example GTL Using Dynamic Variables";
      EntryTitle "Group Variable = " GrpVar;
      EntryFootnote HAlign=LEFT "DateTime: " SysDate9 ":" SysTime;
      Layout OVERLAY;
        ScatterPlot X=XVar Y=YVar / GROUP=GrpVar NAME="scatter";
        DiscreteLegend "scatter";
      EndLayout;
    EndGraph;
  END;
RUN;

PROC SGRENDER DATA=cars TEMPLATE=scatter_dyn;
  DYNAMIC XVAR="horsepower" YVAR="mpg_highway" GRPVAR="origin";
RUN;

PROC SGRENDER DATA=cars TEMPLATE=scatter_dyn;
  DYNAMIC XVAR="horsepower" YVAR="mpg_highway" GRPVAR="type";
RUN;
```

MVAR defines macro variables used in the template.
DYNAMIC defines variables that the user can define at run-time.

The DYNAMIC statement is analogous to a macro call using keyword parameters. The same template is being used but the group variable is different which alters the legend used automatically.



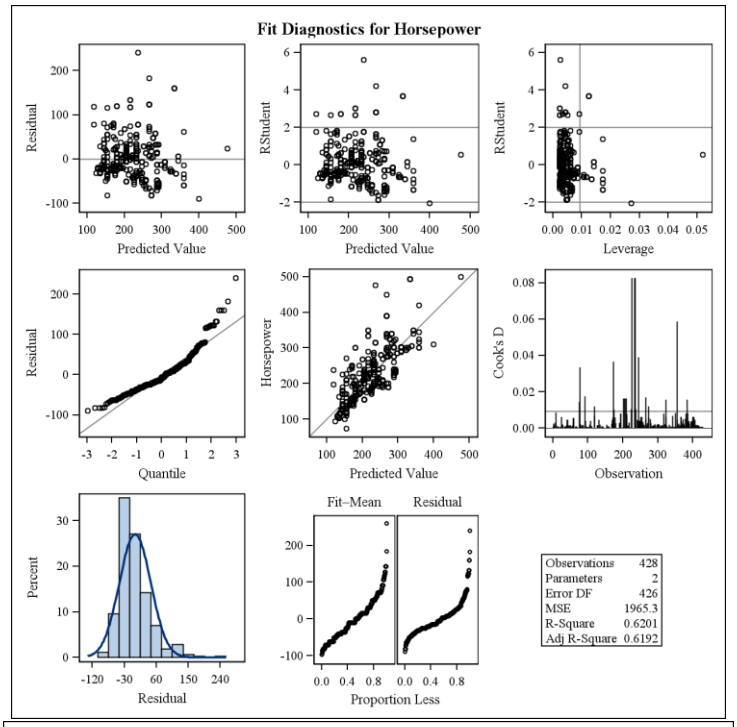


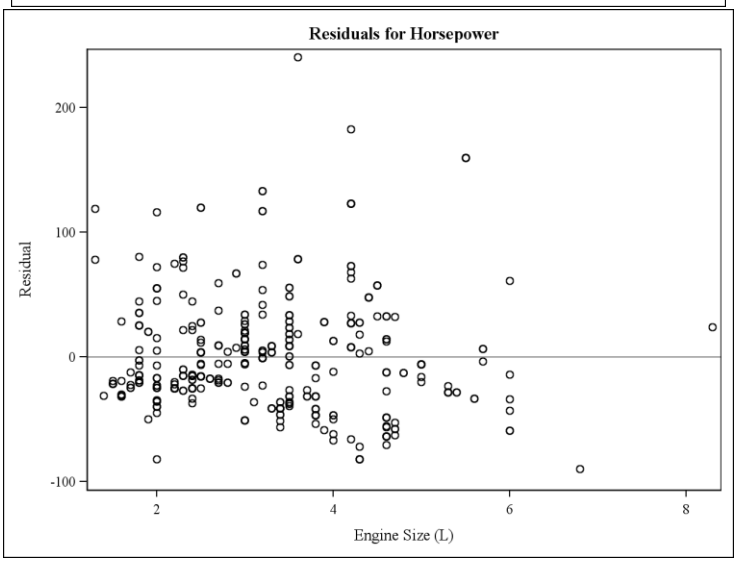
Conditional logic (IF statements) can also be applied to dynamic templates within the LAYOUT statement to execute statements based on satisfying a condition.

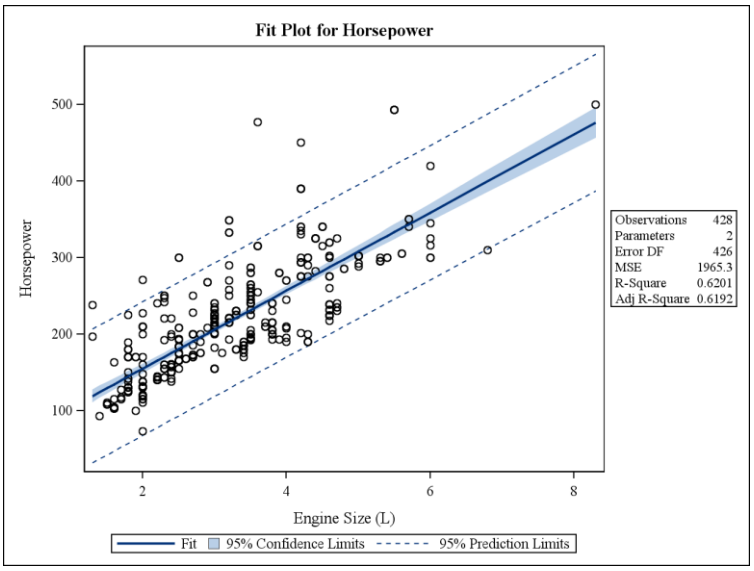
ODS GRAPHICS

If you use SAS/STAT then you might have already discovered the ODS GRAPHICS statement. This ODS statement uses GTL, applied to several statistical procedures, to produce a series of standard model checking plots. This is extremely useful as in traditional SAS, you would have had to send the output to several datasets and produce a whole series of SAS/Graph plots. Now "at the touch of a button" SAS produces these for you. See the SAS help for the individual statistical procedure to identify what statistical graphics are produced.

```
ODS GRAPHICS ON;
PROC REG DATA=cars;
  MODEL horsepower = enginesize;
RUN; QUIT;
ODS GRAPHICS OFF;
```







CONCLUSION

The addition of GTL has brought some exciting new tools to the SAS/Graph programmer's armoury. Traditional SAS/Graph will always have its place, especially with the extremely useful and versatile annotate system, but the addition of GTL to SAS/Graph means that programmers have gained some powerful and efficient graphics creation tools.

GTL is a different way of programming graphs but it can be used to produce some outputs that previously took many lines of code.

GTL can be used to produce dynamic multiple-plot displays efficiently and quickly that would previously take many procedures, data steps and some fairly complex macros.

GTL is a fairly recent introduction to the SAS/Graph programming environment and most of us have only scratched the surface of what can be accomplished with GTL.

REFERENCES

SAS/Graph® 9.2 Graph Template Language Reference
SAS/Graph® 9.2 Graph Template Language User's Guide, Second Edition

CONTACT INFORMATION

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

Lawrence Heaton-Wright
Quintiles Limited
Station House, Market Street
Bracknell, Berkshire, RG12 1HX
United Kingdom
Work Phone: +44 1344 708320
Fax: +44 1344 708106
Email: lawrence.heaton-wright@quintiles.com
Web: www.quintiles.com