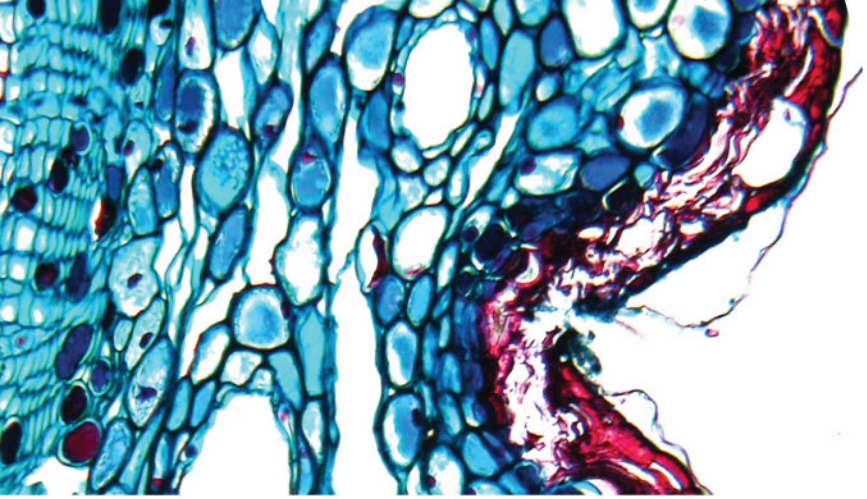
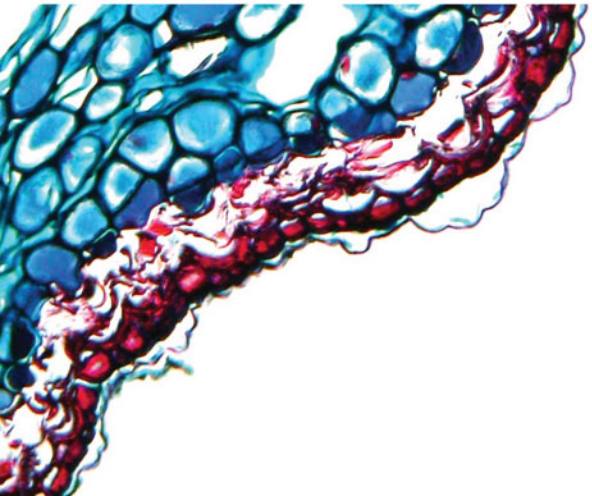


QUINTILES®

Navigating the new health

TS02
SAS GTL - Injecting New Life into Graphs

Lawrence Heaton-Wright, Quintiles



clinical | commercial | consulting | capital

QUINTILES

SAS GTL - Injecting New Life into Graphs

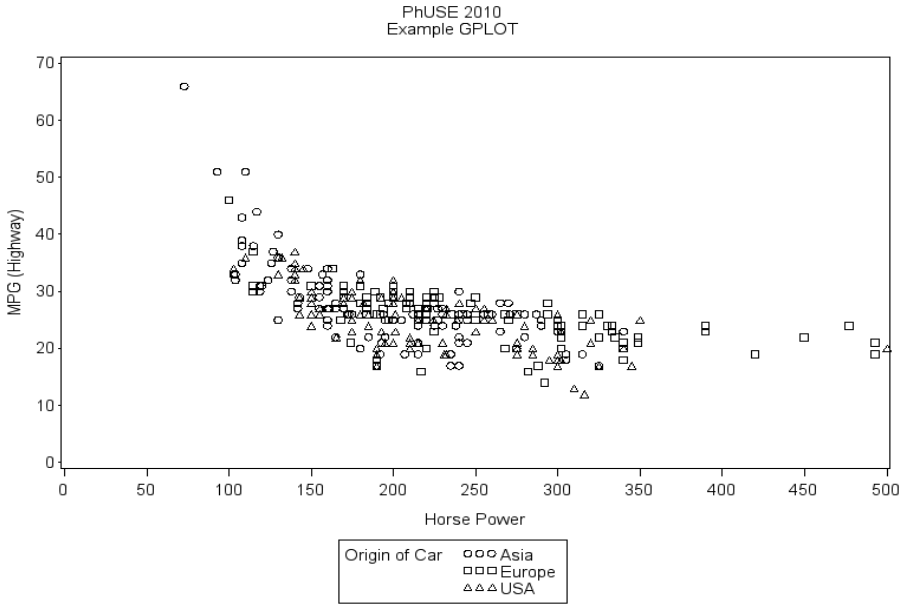
- What is GTL?
- Comparison of Traditional and GTL
  - Simple Plots
  - Common Tasks
  - Multiple Plots On One Page
- Dynamic Templates
- ODS Graphics
- Conclusion
- Questions

QUINTILES

What is GTL?

- Graph Template Language (GTL) is an addition to SAS/Graph for Version 9.
- Allows analytical graphics to be produced that are not available with traditional SAS/Graph procedures
- GTL graphics are produced by rendering data into a graph format template
- This allows the user a lot of flexibility for using these templates with different data sources
- The aim of this presentation is to
  - Compare tasks undertaken using traditional SAS/Graph techniques and GTL
  - Provide an introduction to using dynamic templates

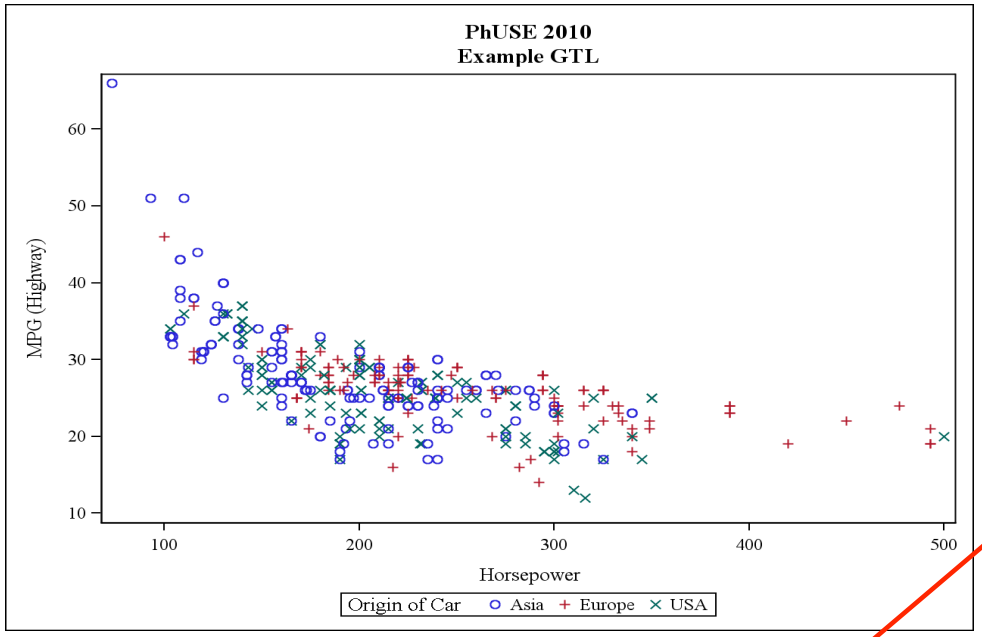
Simple Plots - Traditional




```
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";

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

- This is a fairly simple scatter plot
- The traditional code layout defines the axes, symbol attributes, legend and titles
- The procedure then creates the graph based on the input data and applies the previously defined axes, symbols, legend and titles to the final output

Simple Plots – GTL



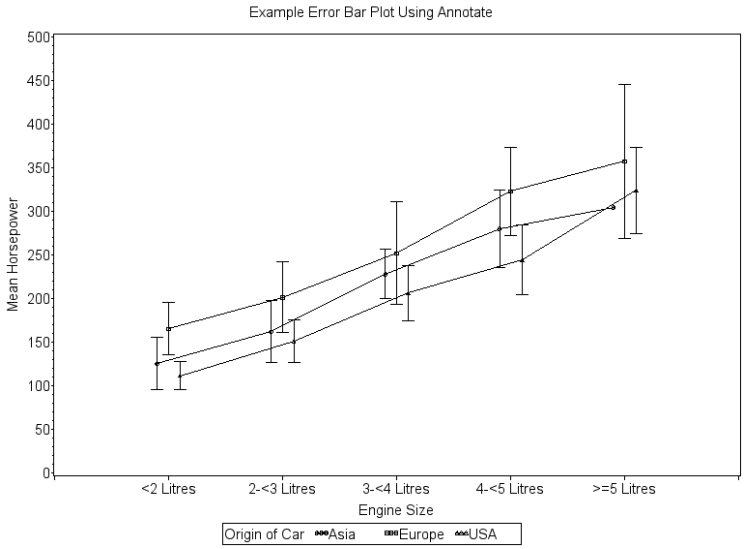
- Using GTL we define the type of graph, titles and legend
- The SGRENDER procedure then renders the data into the template defined
- GTL defines the symbols and axes definitions without programmer input (although this can be programmed)

```
PROC TEMPLATE;
  DEFINE STATGRAPH scatter;
    BeginGraph;
      EntryTitle "PhUSE 2010";
      EntryTitle "Example GTL";
```

The Layout OVERLAY statement builds a composite of one or more graphics statements

```
      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;
```

Common Tasks - Traditional



- Create summary data using PROC MEANS
- Offset the x-axis values to ensure that the plotted symbols do not overwrite each other
  - a common device used to “fool” SAS/ Graph

```
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;
```

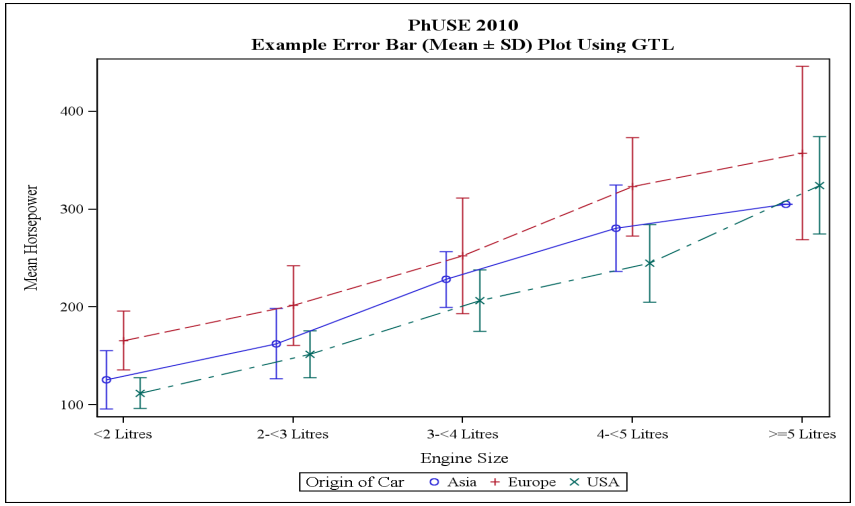
- Create an annotate data set for the vertical bars and tick marks

```
%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;
```

- Define axes, symbols, legend as usual
- Apply the annotate data set using the ANNOTATE option

```
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;
```

Common Tasks – GTL




```
PROC TEMPLATE;
  DEFINE STATGRAPH errorbar;

BeginGraph;
      EntryTitle "PhUSE 2010";
```

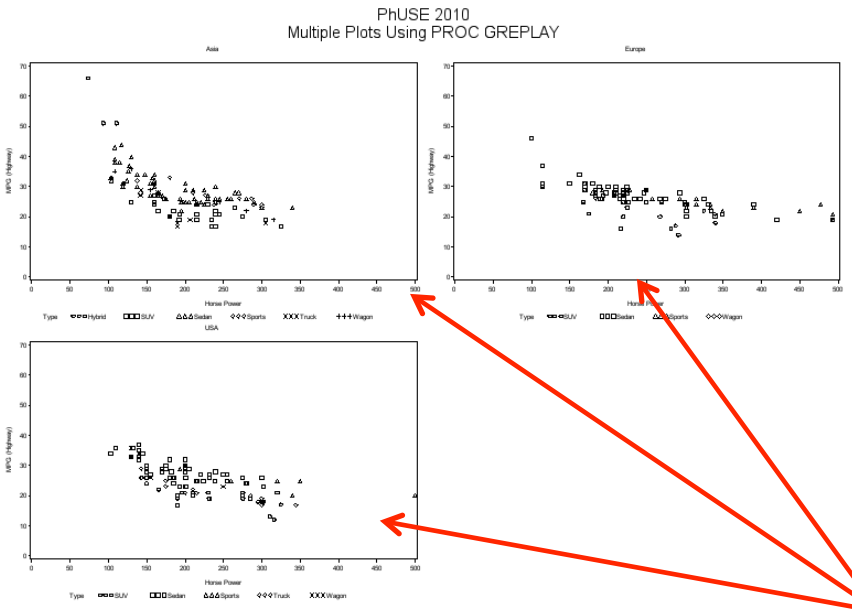
- The UNICODE statement inserts ± into the title line

```
   EntryTitle "Example Error Bar (Mean
" {UNICODE '00B1'X}
               " SD) Plot Using GTL";

   Layout OVERLAY /
       XAxisOpts=(LABEL="Engine Size")
YAxisOpts=(LABEL="Mean Horsepower");
```

- The scatter plot plots the mean values as well as the “error” bars (the same data is used, offsetting the X axis values)
- 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

```
    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;
```

Multiple Plots On One Page - Traditional

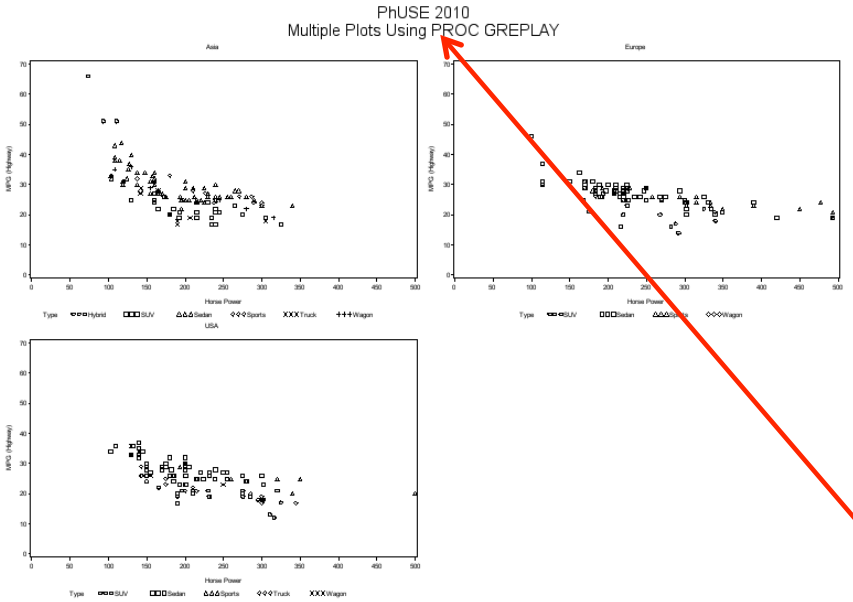


Again, SAS/Graph needs to be manipulated into achieving the results we want:

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;
```

Multiple Plots On One Page - Traditional

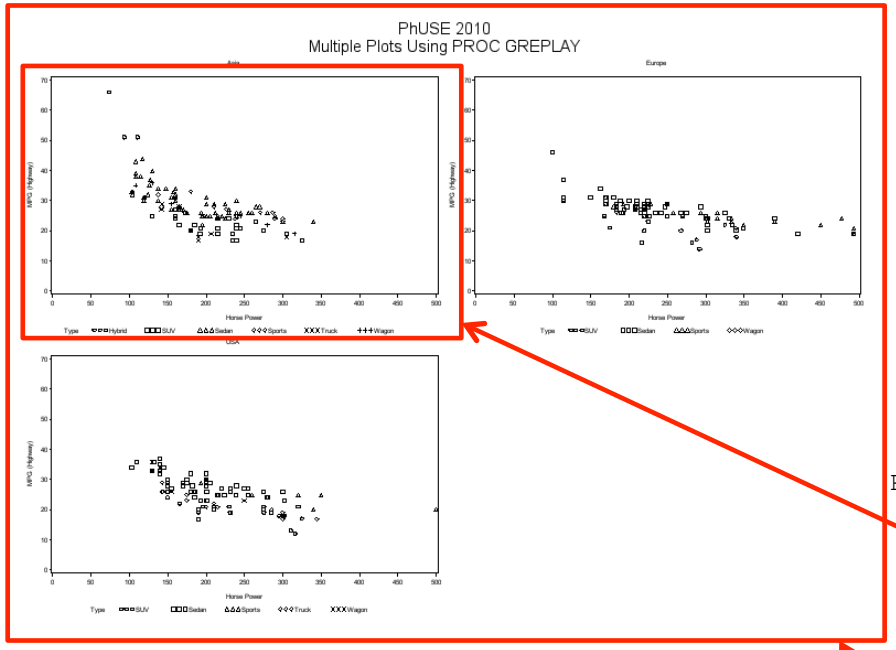


Again, SAS/Graph needs to be manipulated into achieving the results we want:

1. Produce plots and send to a graphics catalog [GRAPHT]
2. Produce a title and footnote output using GSLIDE and send to a graphics catalog [GRAPHT]

```
PROC GSLIDE GOUT=grapht;
  TITLE1 "PhUSE 2010";
  TITLE2 "Multiple Plots Using PROC
  GREPLAY";
RUN;QUIT;

TITLE;
```

Multiple Plots On One Page - Traditional

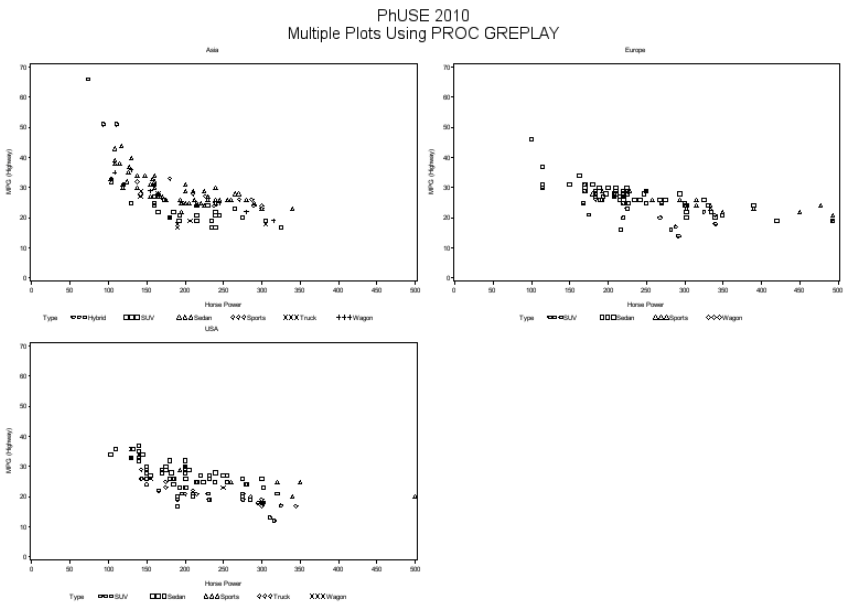


Again, SAS/Graph needs to be manipulated into achieving the results we want:

1. Produce plots and send to a graphics catalog [GRAPHT]
2. Produce a title and footnote output using GSLIDE and send to a graphics catalog [GRAPHT]
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;
```

Multiple Plots On One Page - Traditional

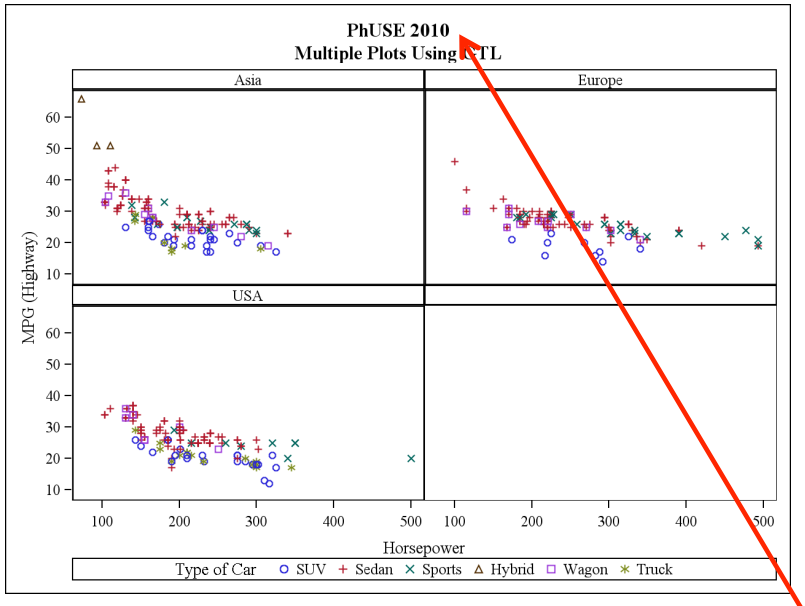


Again, SAS/Graph needs to be manipulated into achieving the results we want:

1. Produce plots and send to a graphics catalog [GRAPHT]
2. Produce a title and footnote output using GSLIDE and send to a graphics catalog [GRAPHT]
3. Produce a replay template and store in a template catalog [TEMPCAT]
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;
```

Multiple Plots On One Page – GTL

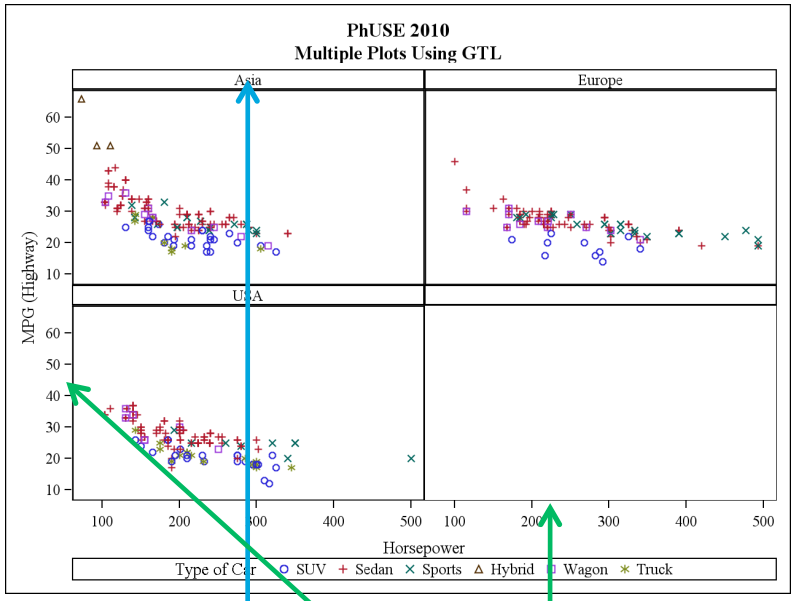


Multiple plots on page are far easier to achieve using GTL compared to traditional coding

1. Titles are defined for the entire graph after the BEGINGRAPH statement

```
PROC TEMPLATE;
  DEFINE STATGRAPH layoutdatapanel;
   BeginGraph;
     EntryTitle "PhUSE 2010";
     EntryTitle "Multiple Plots Using GTL";
   EndGraph;
  END;
RUN;
```

Multiple Plots On One Page – GTL

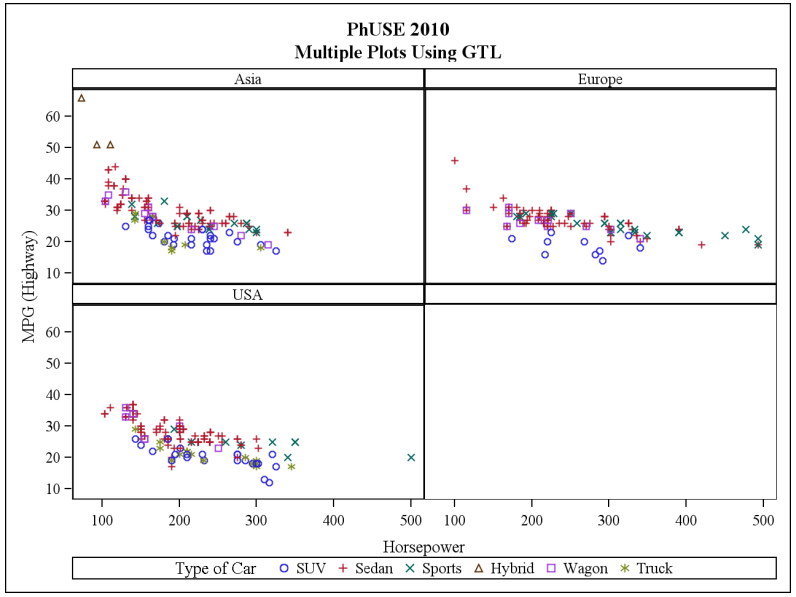


Multiple plots on page are far easier to achieve using GTL compared to traditional coding

1. Titles are defined for the entire graph after the BEGINGRAPH statement
2. Define number of panels required (2 x 2)
   - RowDataRange ensures that the same axis range is used
   - HeaderLabelDisplay ensures individual graph headings are produced

```
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;
      EndLayout;
    EndGraph;
  END;
RUN;
```

Multiple Plots On One Page – GTL



Multiple plots on page are far easier to achieve using GTL compared to traditional coding

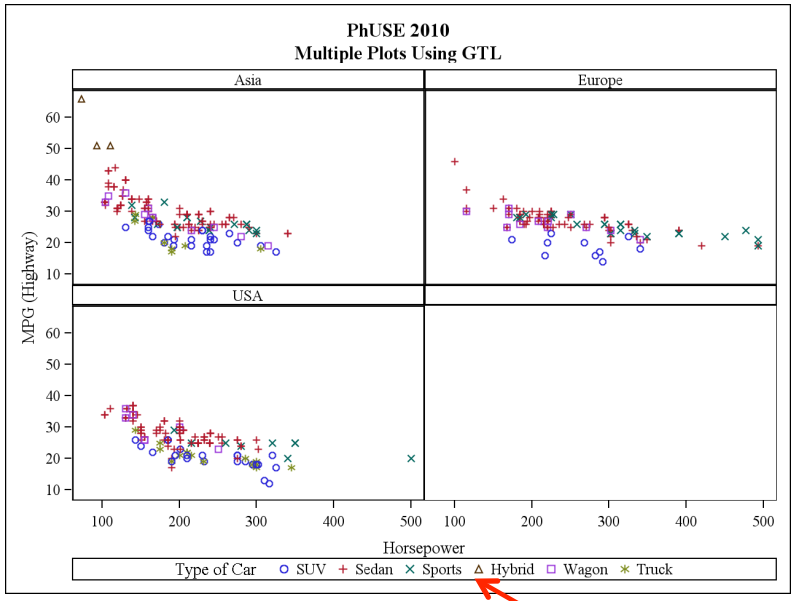
1. Titles are defined for the entire graph after the BEGINGRAPH statement
2. Define number of panels required (2 x 2)
3. Layout PROTOTYPE builds a composite of plot statements
   - This repeats for each cell defined in a parent DATAPANEL statement

```
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;
      EndLayout;
    EndGraph;
  END;
RUN;
```

Parent

Child

Multiple Plots On One Page – GTL

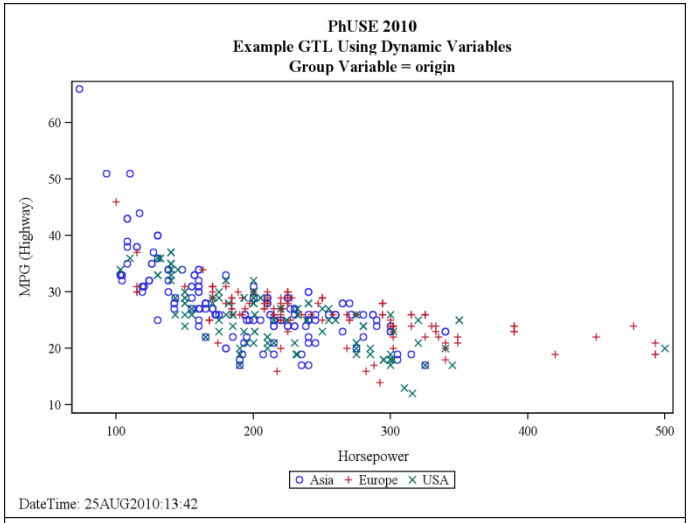


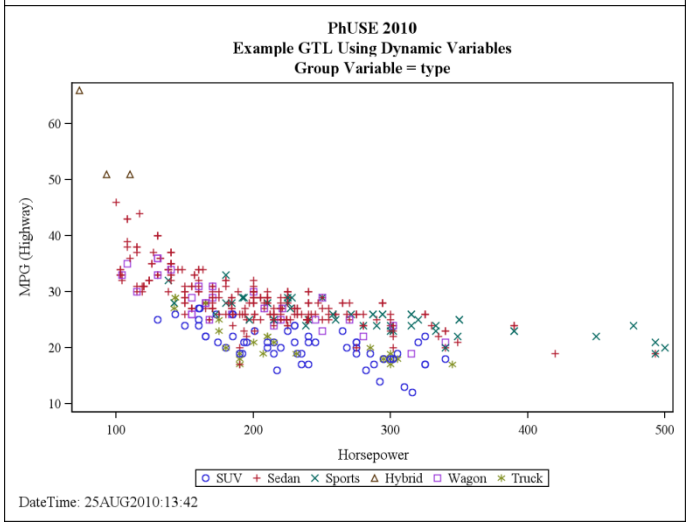
Multiple plots on page are far easier to achieve using GTL compared to traditional coding

1. Titles are defined for the entire graph after the BEGINGRAPH statement
2. Define number of panels required (2 x 2)
3. Layout PROTOTYPE builds a composite of plot statements
4. Use the SIDEBAR statement to define a legend for the entire DATAPANEL

```
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;
```

Dynamic Templates





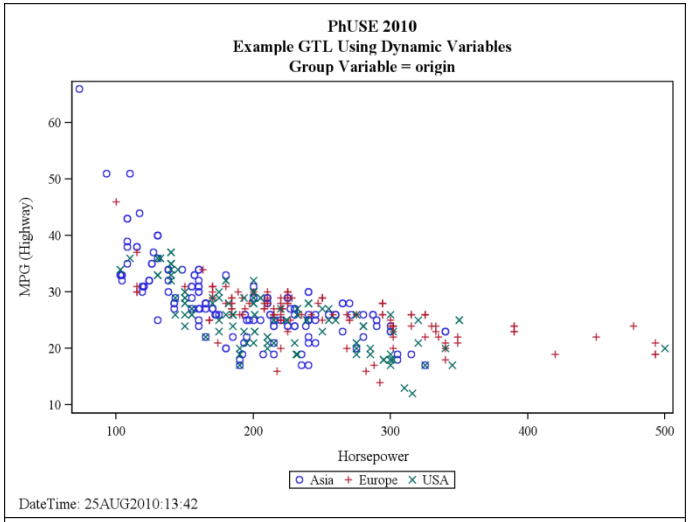
Dynamic templates are analogous to macros

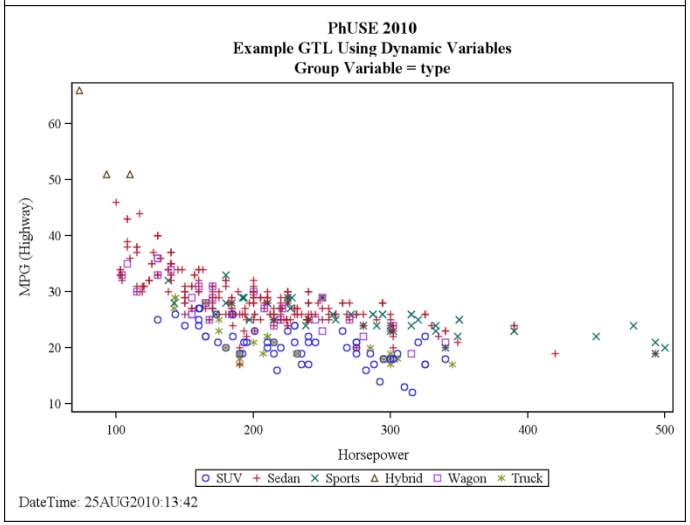
The same basic graph is being produced but the programmer can alter the input variables

1. MVAR defines macro variables used in the template
   - Note how the macro variables are not referenced using &
2. DYNAMIC defines variables that the user can define at run-time dependent on the input data set

```
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;
    EndGraph;
  END;
RUN;
```

Dynamic Templates





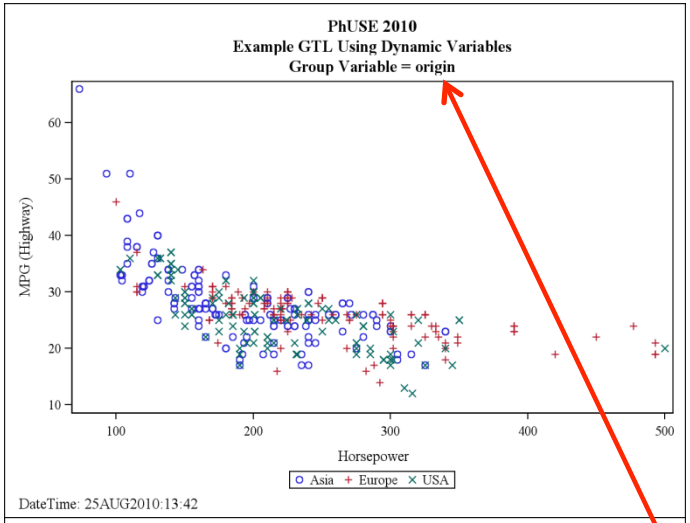
Dynamic templates are analogous to macros

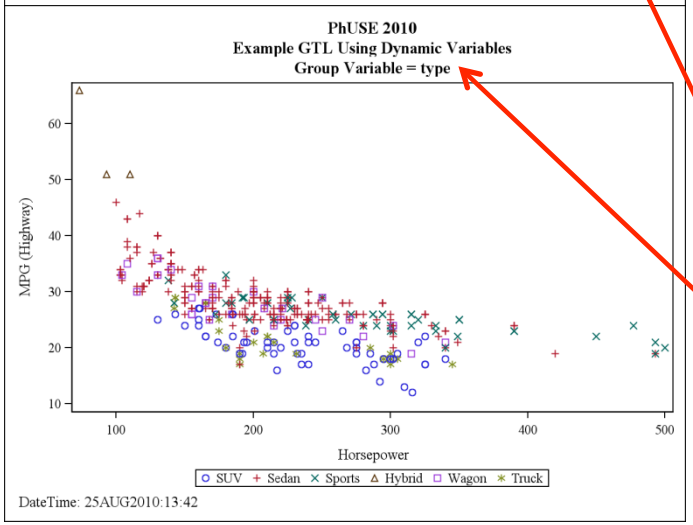
The same basic graph is being produced but the programmer can alter the input variables

1. MVAR defines macro variables used in the template
2. DYNAMIC defines variables that the user can define at run-time dependent on the input data set
3. The dynamic variables have been used instead of static pre-defined variables in the scatter plot and title

```
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;
```

Dynamic Templates





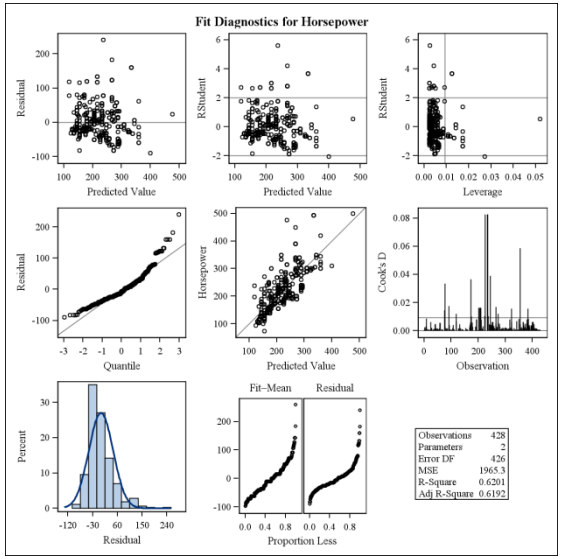
Dynamic templates are analogous to macros

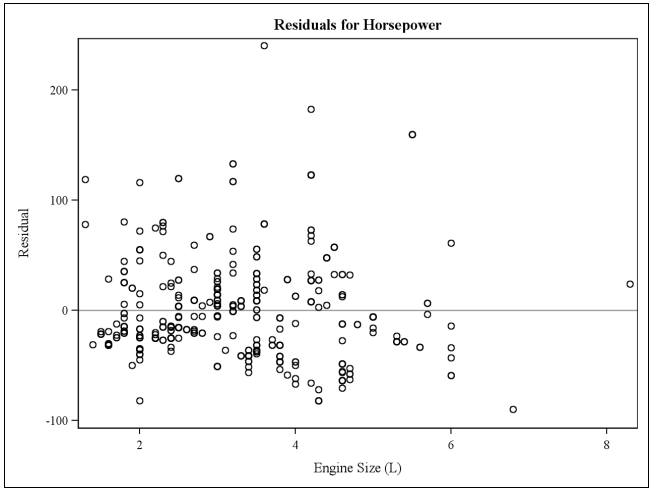
The same basic graph is being produced but the programmer can alter the input variables

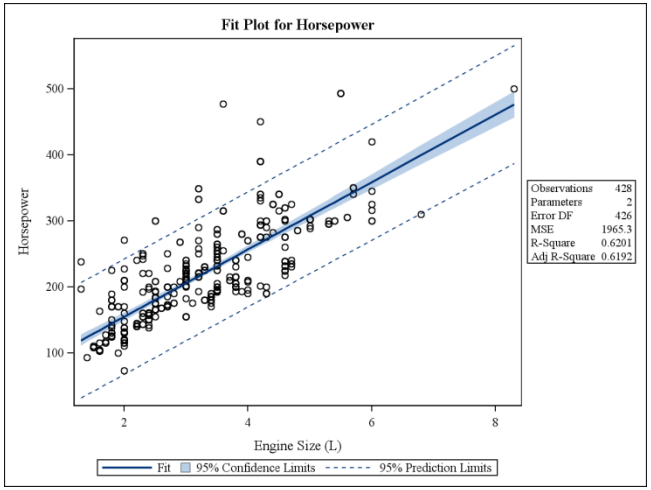
1. MVAR defines macro variables used in the template
2. DYNAMIC defines variables that the user can define at run-time dependent on the input data set
3. The dynamic variables have been used instead of static pre-defined variables in the scatter plot and title
4. Using PROC SGRENDER with the same data set [CARS], referencing the previously defined template [SCATTER_DYN], the DYNAMIC statement is used to produce 2 different plots by specifying a different GRPVAR
   - Note that variables are referenced within “”

```
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;
```

ODS Graphics







ODS GRAPHICS uses pre-defined GTL templates (these can be modified by the user if required) to produce statistical model checking output

It's available for a wide range of SAS/STAT procedures and is easy to invoke

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

QUINTILES

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

Questions/Contact Details

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

Phone: +44 1344 708320, Fax: +44 1344 708106

Email: lawrence.heaton-wright@quintiles.com
Web: www.quintiles.com