

## GTL (Graphics Template Language) in SAS 9.2

Philip R Holland, Holland Numerics Limited, Royston, Herts UK

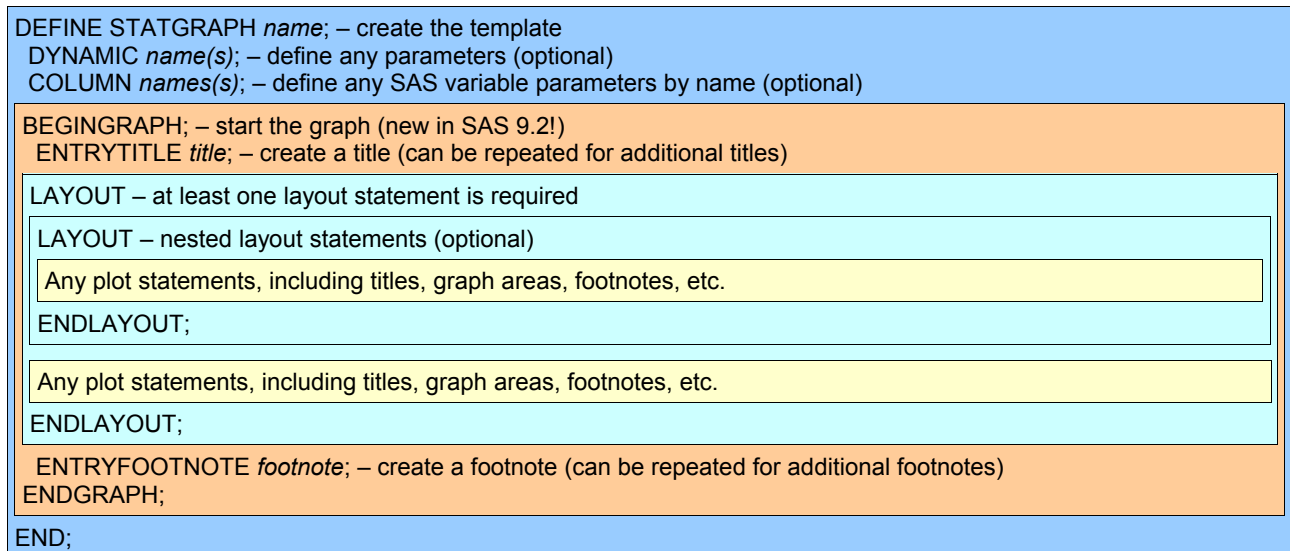
### ABSTRACT

The production version of the GTL (Graphics Template Language) was introduced in SAS 9.2. This new language gives all users the ability to create high quality graphs using simple ODS Graph templates and structured data. This paper describes how to get started with this new language by developing some typical graphs for publication.

### GTL BASICS

GTL is a subset of the PROC TEMPLATE statements used in the STATGRAPH templates. An experimental version was included in SAS 9.1.3, but this was not complete, and has been altered and updated for the production version in SAS 9.2, so that templates written in SAS 9.1.3 are not compatible with SAS 9.2.

The basic STATGRAPH template is made up of nested structures:



Layout statements include:

- LAYOUT GRIDDED – create a grid of graph cells with the same dimensions and properties.
- LAYOUT LATTICE – create a grid of graph cells with different dimensions and properties.
- LAYOUT OVERLAY – create a single graph cell with one or more overlaid plots.

Plot statements include:

- SERIESPLOT – create a plot of connected points.
- SCATTERPLOT – create a plot of symbols at specified points.
- NEEDLEPLOT – create a plot of vertical lines joining a horizontal axis line to specified points.
- LINEPARM – draw a line on the graph with a specified starting position and slope.
- DISCRETELEGEND – create a legend.

## PhUSE 2008

### PREPARING THE GRAPH DATA

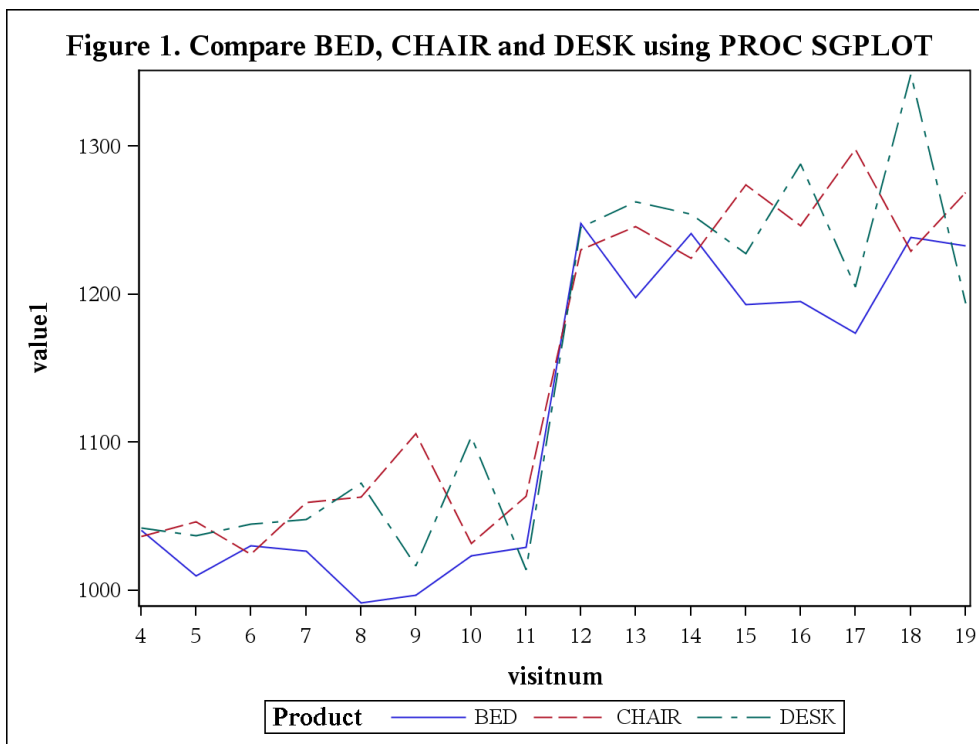
The following code generates simulated clinical data with visit numbers, products, an absolute value with a standard error (for the simple line plot), and a relative value (for the forest plot). The original data in `sashelp.prdsal2` is very uniform, so a filter is used to make the value counts less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

```
PROC SQL;
  CREATE TABLE plotdata AS
    SELECT INTCK('QTR', '01jan1994'd, monyr) AS visitnum
      ,product
      ,MEAN(predict) AS value1 /* used for the simple line plots */
      ,STDERR(predict) AS value1_se /* used for the simple line plots */
      ,MEAN(predict) - 1200 AS value2 /* used for the forest plot */
      ,COUNT(*) AS count
    FROM   sashelp.prdsal2 (WHERE = (product IN ('BED','CHAIR','DESK')
                                      AND predict > 400))

    GROUP BY
      visitnum
      ,product
  ;
QUIT;
```

### DRAWING GRAPHS WITH 'SG' PROCEDURES

It is possible to draw simple line plots using the 'SG' procedures introduced in SAS 9.2.



```
PROC SGPLOT DATA = plotdata;
  TITLE 'Figure 1. Compare BED, CHAIR and DESK using PROC SGPLOT';
  XAXIS TYPE = DISCRETE;
  SERIES X = visitnum Y = value1 / GROUP = product;
RUN;
```

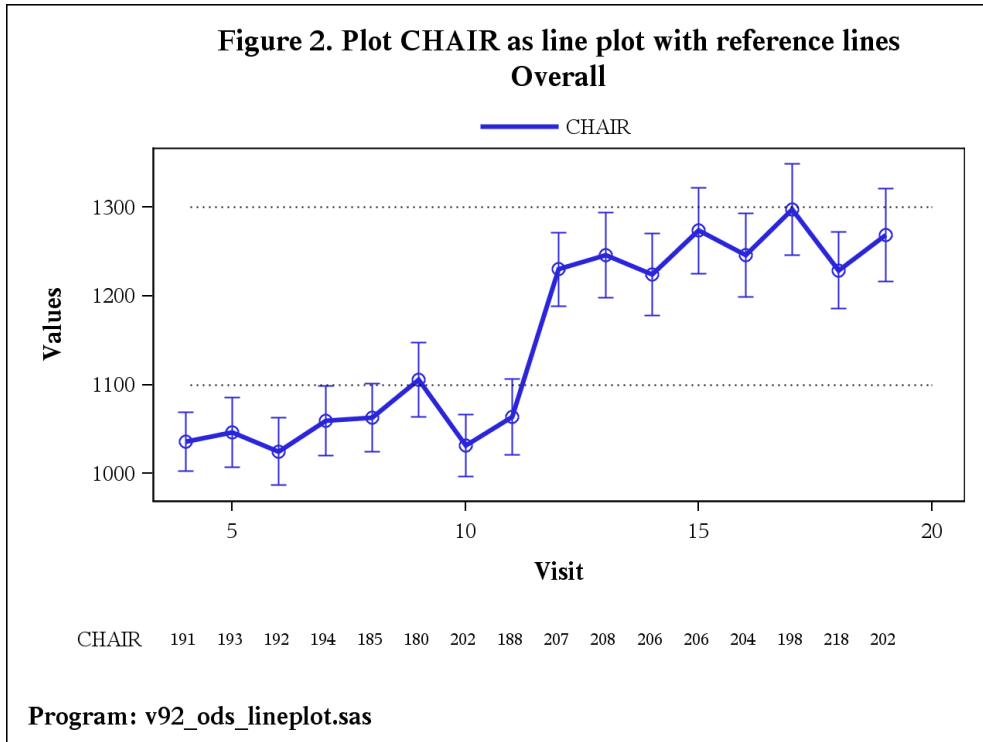
# PhUSE 2008

## SIMPLE LINE PLOTS

For more flexibility, we will be using GTL, below, to create more complex graphs.

### CREATE A LINE PLOT OF A SINGLE GROUP OF DATA POINTS WITH REFERENCE LINES

Generate the graph using a DATA step, with horizontal reference lines, converting the numeric counts to character values, and calculating the upper and lower values of the error bars:



### DEFINE THE TEMPLATE

Create the STATGRAPH template in work.mypath, then define the parameters and titles:

```
ODS PATH work.mypath(UPDATE) sashelp.templmst(READ);
PROC TEMPLATE;
  DEFINE STATGRAPH Graphics.LinePlot;
    DYNAMIC _title _title2 _title3
            _footnote _footnote2 _footnote3
            _xvar _xlabel _ylabel _yintercepta _yinterceptb
            _yvar1 _yupper1 _ylower1 _nvar1 _group
            ;
  BEGINGRAPH;
    ENTRYTITLE _title;
    ENTRYTITLE _title2;
    ENTRYTITLE _title3;
```

Define the layout as 2 full-width areas with common horizontal axes, with the top area occupying 85% of the height and the bottom area 15%. This split provides enough space to show 2 lines of counts, but the proportion allocated to the bottom area should be increased to show more than 2 lines:

```
LAYOUT LATTICE /
  COLUMNS = 1 ROWS = 2
  ROWWEIGHTS = (.85 .15) COLUMNDATARANGE = UNIONALL
  ;
```

Define the top area and fill with an overlay of a series plot of connecting lines and a scatter plot of error bars:

```
LAYOUT OVERLAY /
  PAD = (TOP = 2% BOTTOM = 2% LEFT = 2% RIGHT = 2%)
  XAXISOPTS = (LABEL = _xlabel)
  YAXISOPTS = (LABEL = _ylabel)
  OPAQUE = FALSE
  ;
```

## PhUSE 2008

```
SERIESPLOT X = _xvar Y = _yvar1 /
  MARKERATTRS = (SIZE = 10PX)
  LINEATTRS = (THICKNESS = 3PX)
  NAME = 'series'
  GROUP = _group
;
SCATTERPLOT X = _xvar Y = _yvar1 /
  YERRORUPPER = _yupper1
  YERRORLOWER = _ylower1
  MARKERATTRS = (SIZE = 10PX)
  GROUP = _group
;
```

Add horizontal reference lines, if requested:

```
IF (_yintercepta)
  LINEPARM X = 0 Y = _yintercepta SLOPE = 0 / LINEATTRS = (PATTERN = DOT);
ENDIF;
IF (_YINTERCEPTB)
  LINEPARM X = 0 Y = _yinterceptb SLOPE = 0 / LINEATTRS = (PATTERN = DOT);
ENDIF;
```

Add a legend for the connecting lines:

```
DISCRETELEGEND 'series' /
  ACROSS = 4
  BORDER = FALSE
  VALIGN = TOP
;
```

```
ENDLAYOUT;
```

Define the bottom area and fill with a scatter plot of the counts as character values:

```
LAYOUT OVERLAY /
  PAD = (BOTTOM = 2% LEFT = 2% RIGHT = 2%)
  BORDER = FALSE
  WALLDISPLAY = NONE
  XAXISOPTS = (DISPLAY = NONE)
  X2AXISOPTS = (DISPLAY = NONE)
  Y2AXISOPTS = (DISPLAY = NONE)
  YAXISOPTS = (DISPLAY = (TICKVALUES))
;
SCATTERPLOT X = _xvar Y = _group /
  MARKERCHARACTERATTRS = (COLOR = BLACK)
  MARKERCHARACTER = _nvar1
;
ENDLAYOUT;
ENDLAYOUT; /* lattice*/
```

Add the footnotes:

```
ENTRYFOOTNOTE HALIGN = LEFT _footnote;
ENTRYFOOTNOTE HALIGN = LEFT _footnote2;
ENTRYFOOTNOTE HALIGN = LEFT _footnote3;
ENDGRAPH;
END;
RUN;
```

### CREATE THE PLOT

Generate the graph, requesting the horizontal reference lines:

```
%LET pgm=v92_ods_lineplot;
OPTIONS NODATE NONUMBER ORIENTATION = LANDSCAPE;
/*--- Figure 2: Plot CHAIR as line plot with reference lines ---*/
TITLE ' ';
ODS RTF FILE = "&pgm._chair.rtf" STYLE = serifprinter;
ODS GRAPHICS ON;
```

## PhUSE 2008

```

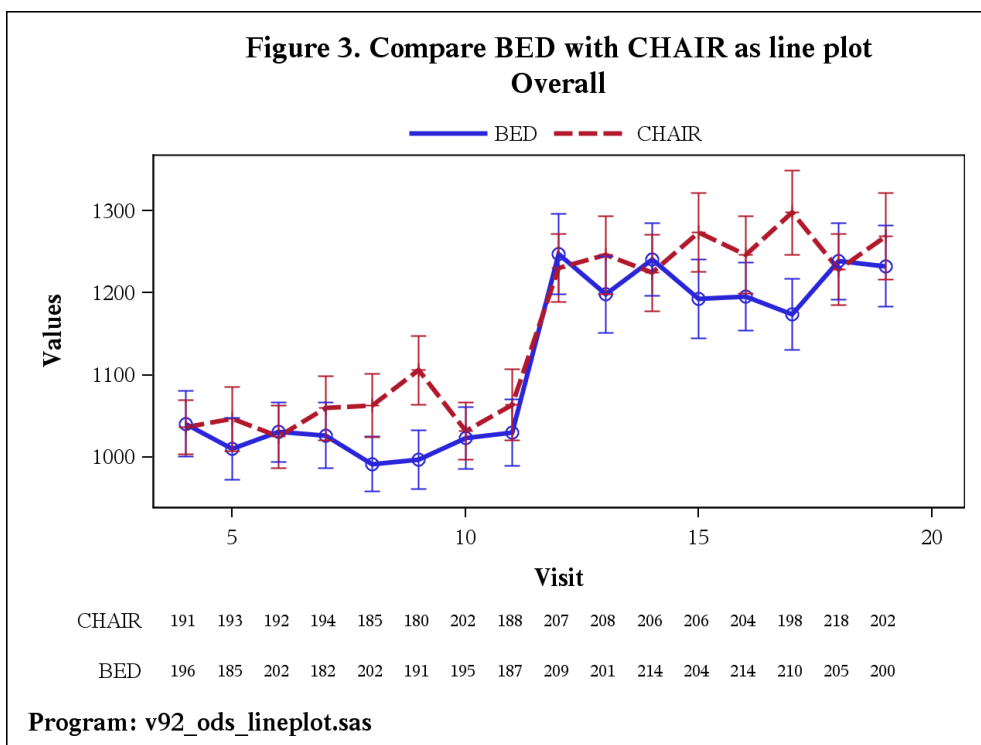
DATA _NULL_;
  LENGTH ccount $4;
  SET plotdata (WHERE = (product = 'CHAIR'));
  value1_upper = value1 + value1_se;
  value1_lower = value1 - value1_se;
  ccount = STRIP(PUT(count, 4.));
  FILE PRINT ODS = (TEMPLATE = 'Graphics.LinePlot'
    DYNAMIC = (_title =
      "Figure 2. Plot CHAIR as line plot with reference lines"
      _title2 = "Overall"
      _footnote = "Program: &pgm..sas"
      _xvar = "visitnum"
      _xlabel = "Visit"
      _ylabel = "Values"
      _yintercepta = 1300
      _yinterceptb = 1100
      _yvar1 = "value1"
      _yupper1 = "value1_upper"
      _ylower1 = "value1_lower"
      _nvar1 = "ccount"
      _group = "product"
    )
  );

  PUT _ODS_;
RUN;
ODS GRAPHICS OFF;
ODS RTF CLOSE;

```

### CREATE A LINE PLOT OF 2 GROUPS OF DATA POINTS

Generate a similar graph as above using the same template, but removing the horizontal reference lines and adding another group of data points:



```

/*--- Figure 3: Compare BED with CHAIR as line plot ---*/
TITLE ' ';
ODS RTF FILE = "&pgm._bedchair.rtf" STYLE = serifprinter;
ODS GRAPHICS ON;

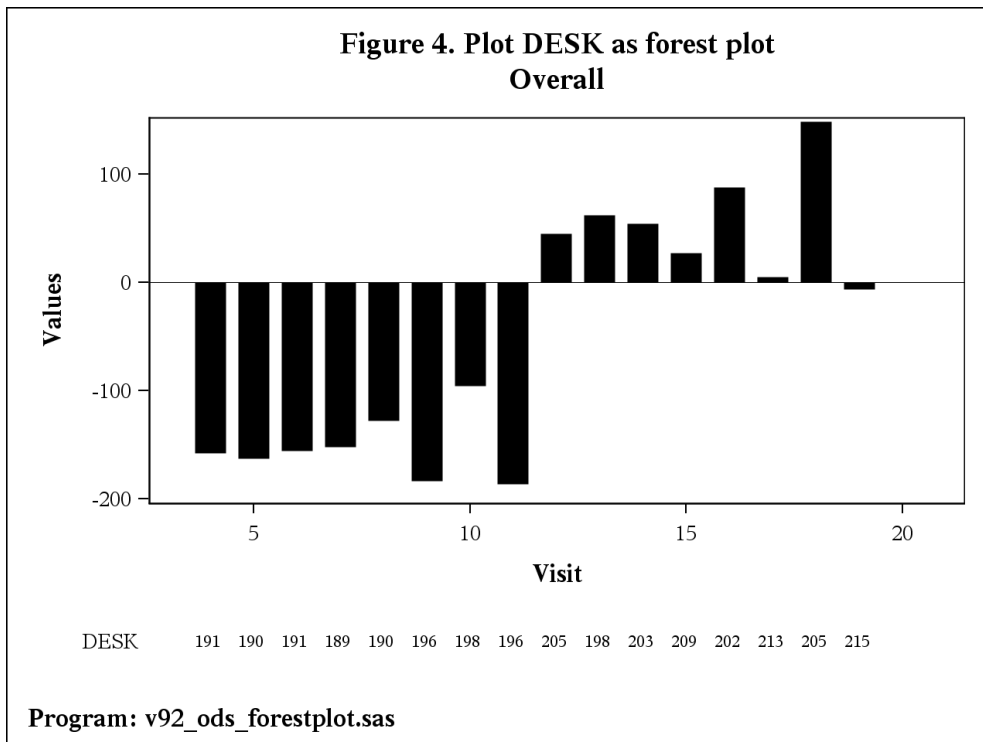
```

## PhUSE 2008

```
DATA _NULL_;
  LENGTH ccount $4;
  SET plotdata (WHERE = (product IN ('BED','CHAIR')));
  value1_upper = value1 + value1_se;
  value1_lower = value1 - value1_se;
  ccount = STRIP(PUT(count, 4.));
  FILE PRINT ODS = (TEMPLATE = 'Graphics.LinePlot'
    DYNAMIC = (_title =
      "Figure 3. Compare BED with CHAIR as line plot"
      _title2 = "Overall"
      _footnote = "Program: &pgm..sas"
      _xvar = "visitnum"
      _xlabel = "Visit"
      _ylabel = "Values"
      _yvar1 = "value1"
      _yupper1 = "value1_upper"
      _ylower1 = "value1_lower"
      _nvar1 = "ccount"
      _group = "product"
    )
  );
  PUT _ODS_;
RUN;
ODS GRAPHICS OFF;
ODS RTF CLOSE;
```

### FOREST PLOTS

To create a Forest Plot, instead of the Simple Line Plot above, the STATGRAPH templates are very similar.



## PhUSE 2008

The first SERIESPLOT and SCATTERPLOT statements have been replaced with a single NEEDLEPLOT statement, and the DISCRETELEGEND statement has also been deleted:

```
ODS PATH work.mypath(UPDATE) sashelp.tmplmst(READ);
PROC TEMPLATE;
  DEFINE STATGRAPH Graphics.ForestPlot;
    DYNAMIC _title _title2 _title3
            _footnote _footnote2 _footnote3
            _xvar _xlabel _ylabel _yintercepta _yinterceptb
            _yvar1 _yupper1 _ylower1 _nvar1 _group
            ;
  BEGINGRAPH;
    :
    :
    NEEDLEPLOT X = _xvar Y = _yvar1 /
              DISPLAY = STANDARD
              LINEATTRS = (THICKNESS = 20PX)
              NAME = 'needle'
              ;
    :
    :
  ENDGRAPH;
END;
RUN;
```

The DATA step code for generating the graph is comparable to that used for the Simple Line Plots, but using a different template and a different variable:

```
%LET pgm = v92_ods_forestplot;
OPTIONS NODATE NONUMBER ORIENTATION = LANDSCAPE;
/*--- Figure 4: Plot DESK as forest plot ---*/
TITLE ' ';
ODS RTF FILE = "&pgm._desk.rtf" STYLE = serifprinter;
ODS GRAPHICS ON;
DATA _NULL_;
  LENGTH ccount $4;
  SET plotdata (WHERE = (product = 'DESK'));
  ccount = STRIP(PUT(count, 4.));
  FILE PRINT ODS = (TEMPLATE = 'Graphics.ForestPlot'
                   DYNAMIC = (_title = "Figure 4. Plot DESK as forest plot"
                               _title2 = "Overall"
                               _footnote = "Program: &pgm..sas"
                               _xvar = "visitnum"
                               _xlabel = "Visit"
                               _ylabel = "Values"
                               _yvar1 = "value2"
                               _nvar1 = "ccount"
                               _group = "product"
                               )
                   );
  PUT _ODS_;
RUN;
ODS GRAPHICS OFF;
ODS RTF CLOSE;
```

# PhUSE 2008

## PROS AND CONS

### DISADVANTAGES

- Yet another "SAS" language to learn!
- Template code can be difficult to debug.
- Templates are not necessary for simple graphs, as the 'SG' procedures can be used instead.

### ADVANTAGES

- High quality graphs can be created using templates.
- Templates are re-usable and produce the same output on all supported platforms.
- Templates can be used to reproduce complex layouts.

## CONCLUSIONS

- The time taken to learn GTL is likely to be repaid by the time saved in re-using the templates to create standard graphs.
- The ability to reproduce graphs in Windows or UNIX will save time and effort currently required when developing graphs on multiple platforms.
- The high quality of the graphs is a major improvement over that created by existing SAS/GRAPH procedures.

## REFERENCES

- Standard Graph Templates: Philip R Holland, PhUSE 2007, [www.hollandnumerics.com/SASPAPER.HTM](http://www.hollandnumerics.com/SASPAPER.HTM)

## CONTACT INFORMATION

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

Author Name	<b>Philip R Holland</b>
Company	<b>Holland Numerics Ltd</b>
Address	<b>94 Green Drift</b>
City / Postcode	<b>Royston, Herts SG8 5BT, UK</b>
Work Phone:	<b>+44 7714 279085</b>
Fax:	<b>+44 1763 244497</b>
Email:	<b>phil@hollandnumerics.com</b>
Web:	<b>www.hollandnumerics.com</b>

This paper and associated sample SAS code can be downloaded from the Holland Numerics Ltd web site at [www.hollandnumerics.com/SASPAPER.HTM](http://www.hollandnumerics.com/SASPAPER.HTM)

Brand and product names are trademarks of their respective companies.