Presenter

Philip R Holland, Consultant, Holland Numerics Limited, UK

Philip R Holland has been working with SAS software for over 30 years, providing SAS technical consultancy and training through his own company, Holland Numerics Limited, in the financial, retail and pharmaceutical sectors in the UK, Belgium, Holland, Germany and USA since 1992.

He is an enthusiastic software developer, and his love of graphics goes back to before he discovered SAS. His fourth book, “SAS Programming and Data Visualization Techniques”, was published in 2015.

Making Graphs Easier to Validate - The Benefits of ODS Graphics

Philip R Holland, Holland Numerics Limited, UK

Agenda

- Introduction
- Basic input data for basic graphs
- Using computed values
- An example of a complex graph
- TEXT statements
- Adding “Section” variables
- Validation of fonts, line patterns and colours
- Conclusions

Introduction

- Complex graphs can be validated simply by considering the combined individual SAS input data sets separately.
- ODS Graphics procedures require a single input data set, but sparse data is perfectly acceptable, so concatenating multiple data sets can create a suitable input data set.
- SAS/GRAPH annotations require a second data set, but in ODS Graphics you can use input data to add extra features.

Basic input data for basic graphs

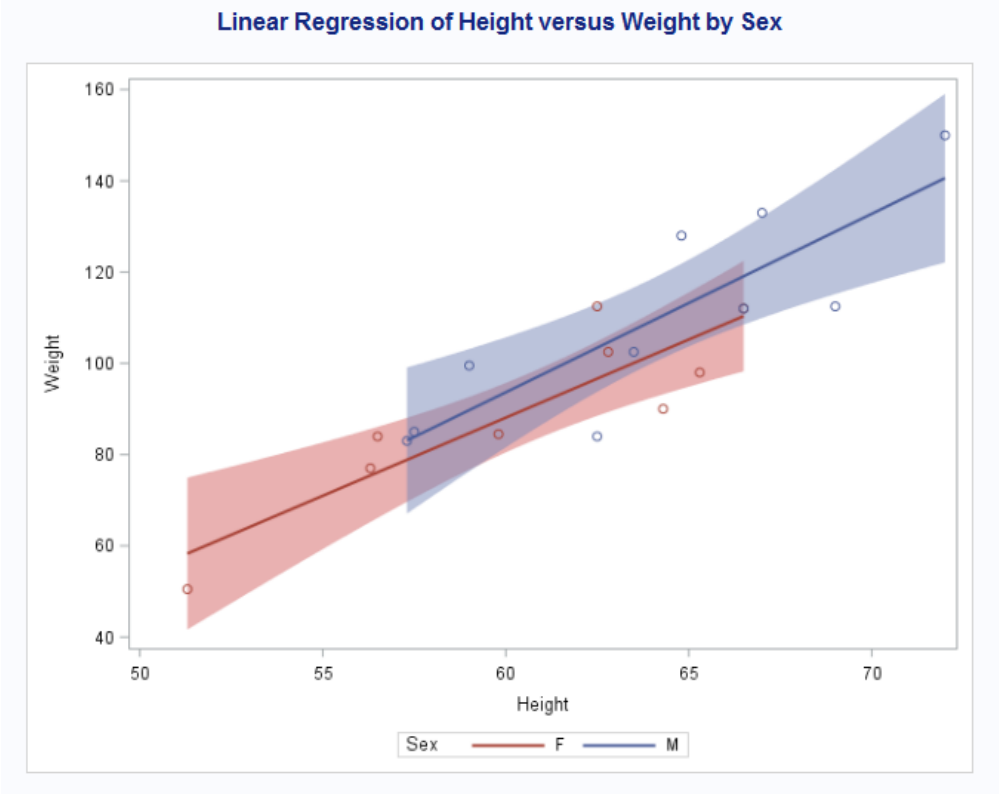
- Create a permanent SAS data set and use it to generate the graph.
- Only keep those variables relevant to the graph production or its validation.

Basic input data for basic graphs


```
DATA sasuser.graph1;
  SET sashelp.class;
  KEEP sex height weight;
RUN;

PROC SGPLOT DATA = sasuser.graph1;
  TITLE "Linear Regression of Height versus Weight by Sex";
  REG X = height Y = weight
      / GROUP = sex CLM CLMTRANSPARENCY = 0.5;
RUN;
```

Basic input data for basic graphs



Using computed values

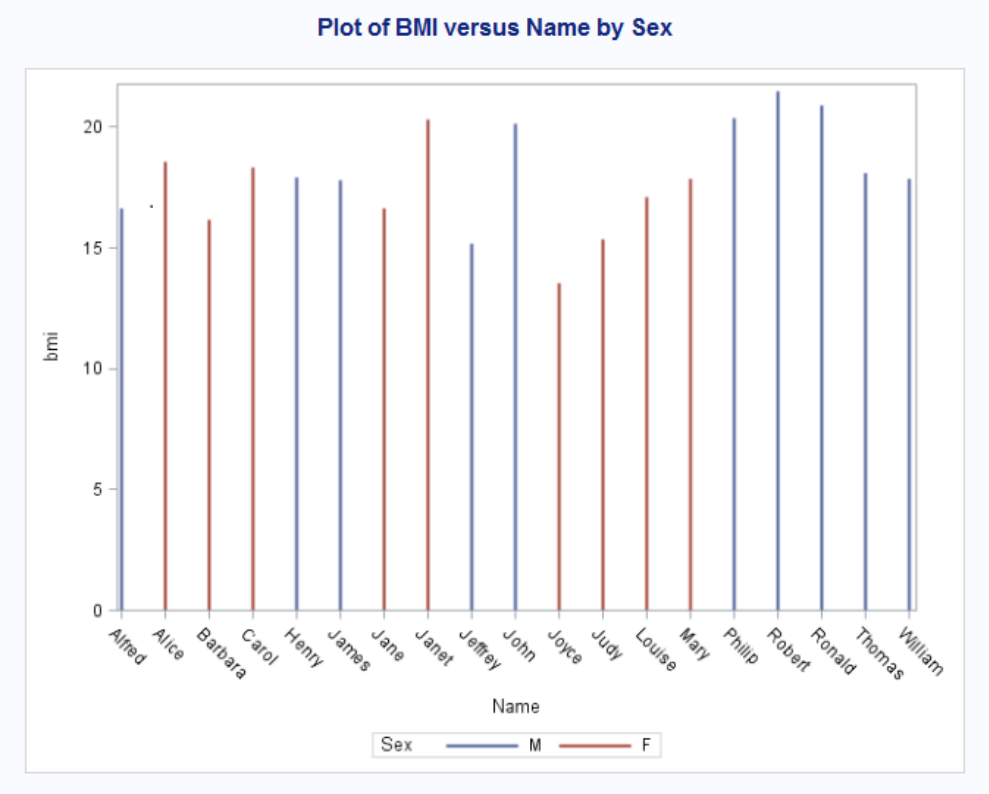
When using the sashelp.class data set, a new value of BMI must be calculated from height and weight variables, therefore keep bmi (computed), height and weight (validation), name and sex variables in the input data set.

Using computed values


```
DATA sasuser.graph2;
  SET sashelp.class;
  bmi = (weight / 2.2) / ((height * 2.54 / 100) ** 2);
  KEEP name sex height weight bmi;
RUN;

PROC SGPLOT DATA = sasuser.graph2;
  TITLE "Plot of BMI versus Name by Sex";
  NEEDLE X = name Y = bmi
         / GROUP = sex LINEATTRS = (THICKNESS=2);
RUN;
```

Using computed values



An example of a complex graph

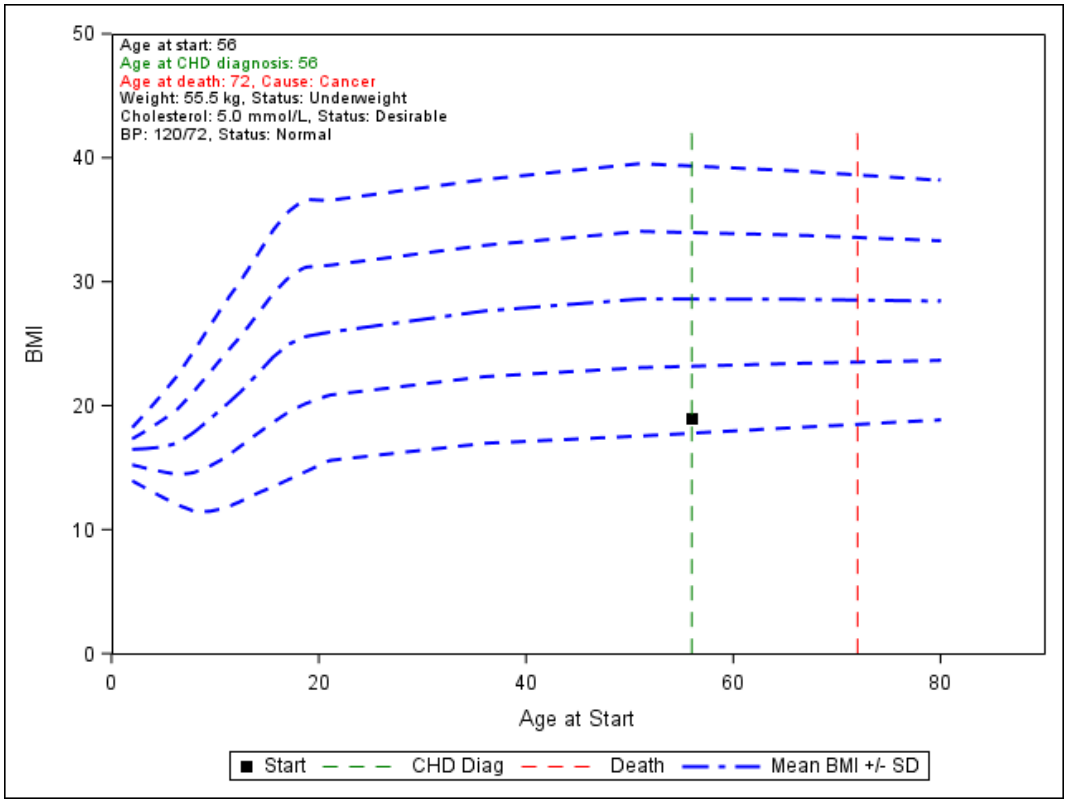
This example uses 2 SAS-supplied data sets:

- sashelp.heart
- sashelp.bmimen

There data sets are combined, after generating subject numbers and calculating BMI at different ages, to create 3 separate graphs in the same axes:

- Heart data for each subject
- BMI mean values by age with 1 & 2 x SD lines above and below.
- A text box containing any non-missing subject data.

An example of a complex graph



HOLLAND NUMERICS LIMITED

TEXT statements

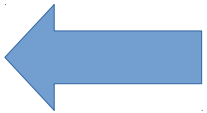
This example uses four variables to add the text:

- x_text
- y_text
- text
- textgroup (for formatting selection)

There are, however, more variables used the generate the text values, which are saved in the heart_data data set, some of which are used elsewhere in the graph too.

TEXT statements


```
DATA heart_text (KEEP = subject textgroup text x_text y_text);
  SET heart_data;
  BY subject;
  LENGTH textgroup $30 text $100;
  x = 0; y = 100; lineskip = -3;
  IF NMISS(ageatstart) = 0 THEN DO;
    textgroup = 'Start';
    x_text = x;
    y_text = y;
    text = 'Age at start: ' || STRIP(PUT(ageatstart, 8.));
    OUTPUT;
    y + lineskip;
  END;
  . . .
  . . .
RUN;
```



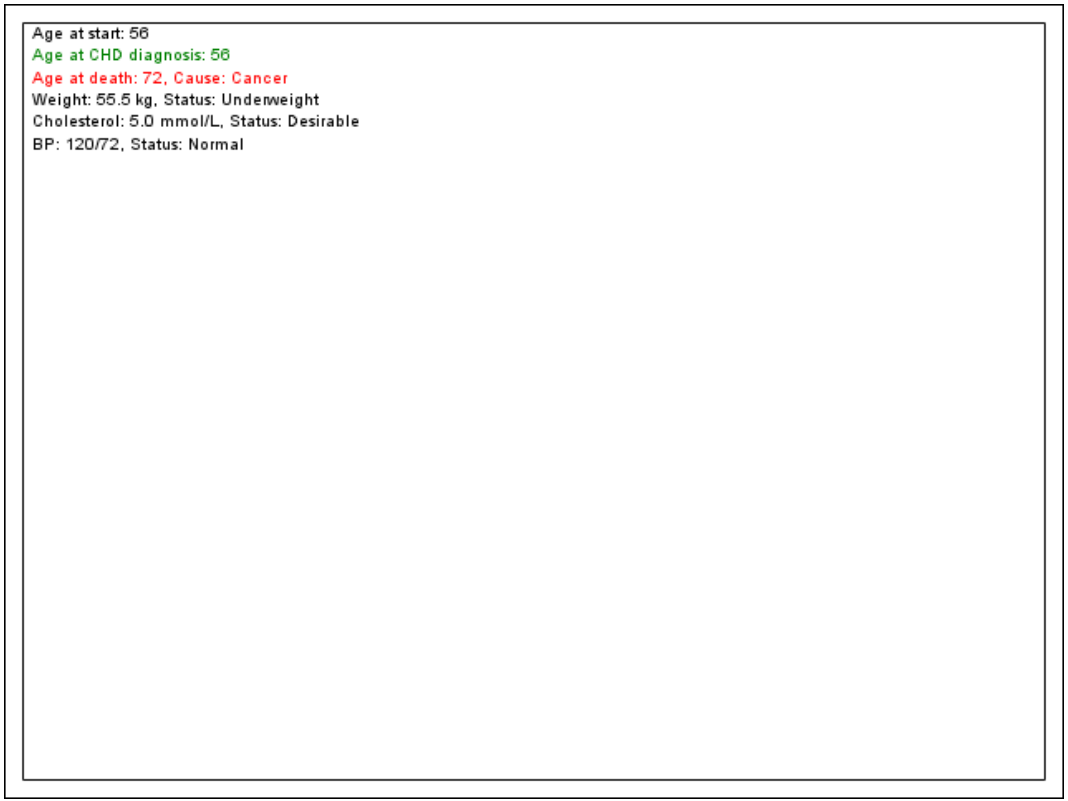
More tests can be added here using the textgroup variable to control the text colour.

TEXT statements


```
** text box - BOTTOMRIGHT is position of text relative
   to x_text and y_text **;
TEXT X = x_text Y = y_text TEXT = text
     / ATTRID = colors GROUP = textgroup
       TEXTATTRS = (SIZE = 7pt)
       POSITION = BOTTOMRIGHT VCENTER = BBOX
       X2AXIS Y2AXIS;
```

This code includes green text, which relates to formatting information, rather than data. This will be discussed later.

TEXT statements



Adding “Section” variables

The section variable created below is intended only for validation, as it identifies the source of the data.

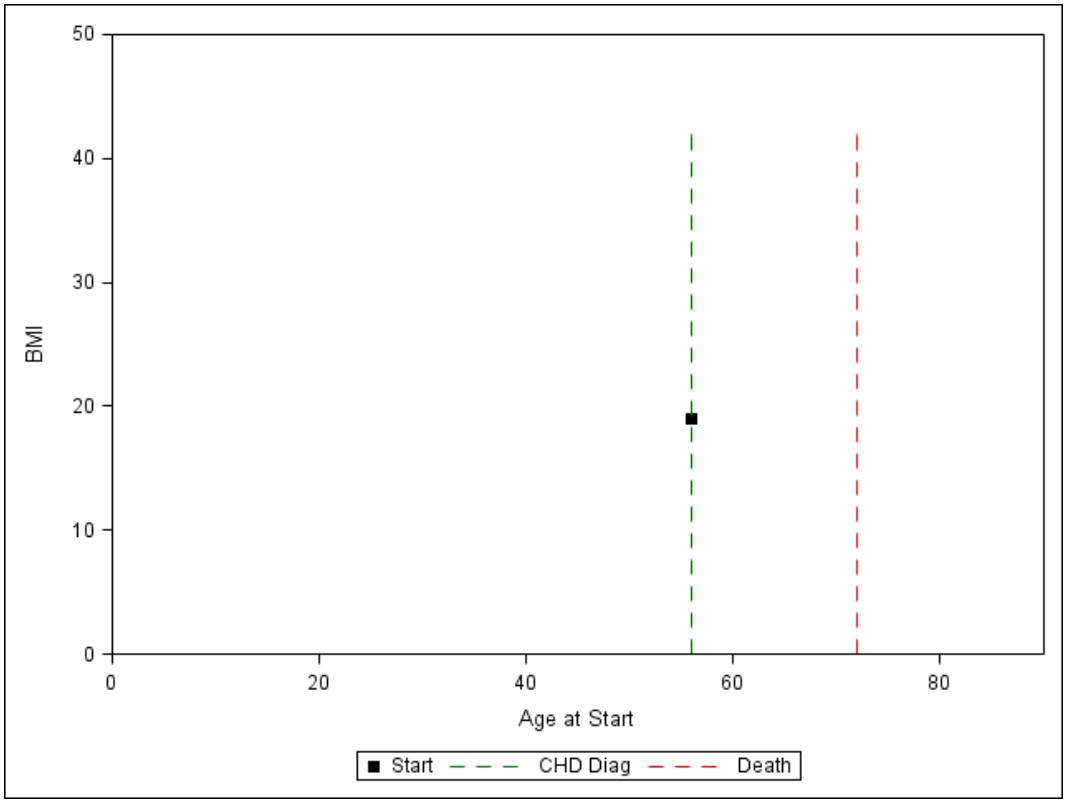
```
** combine heart data, text box and BMI lines **;
DATA sasuser.plot_data;
  SET heart_data (IN = a)
      heart_text (IN = b)
      bmimen_data (IN = c)
      ;
  BY subject;
  SELECT;
    WHEN (a) section = 'Health';
    WHEN (b) section = 'Text';
    WHEN (c) section = 'BMI lines';
    OTHERWISE;
  END;
RUN;
```

Adding “Section” variables

The data for this part of the graph is from heart_data, and identified by section=”Health”.

```
** subject BMI at start **;
SCATTER X = ageatstart Y = bmi
        / MARKERATTRS = (COLOR = BLACK SYMBOL = SQUAREFILLED)
          NAME = 'Start' LEGENDLABEL = 'Start';
** CHD diagnosis and death markers **;
DROPLINE X = agechddiag Y = 42
         / LINEATTRS = (COLOR = GREEN PATTERN = 20)
           NAME = 'CHD Diag';
DROPLINE X = ageatdeath Y = 42
         / LINEATTRS = (COLOR = RED PATTERN = 20)
           NAME = 'Death';
```

Adding “Section” variables

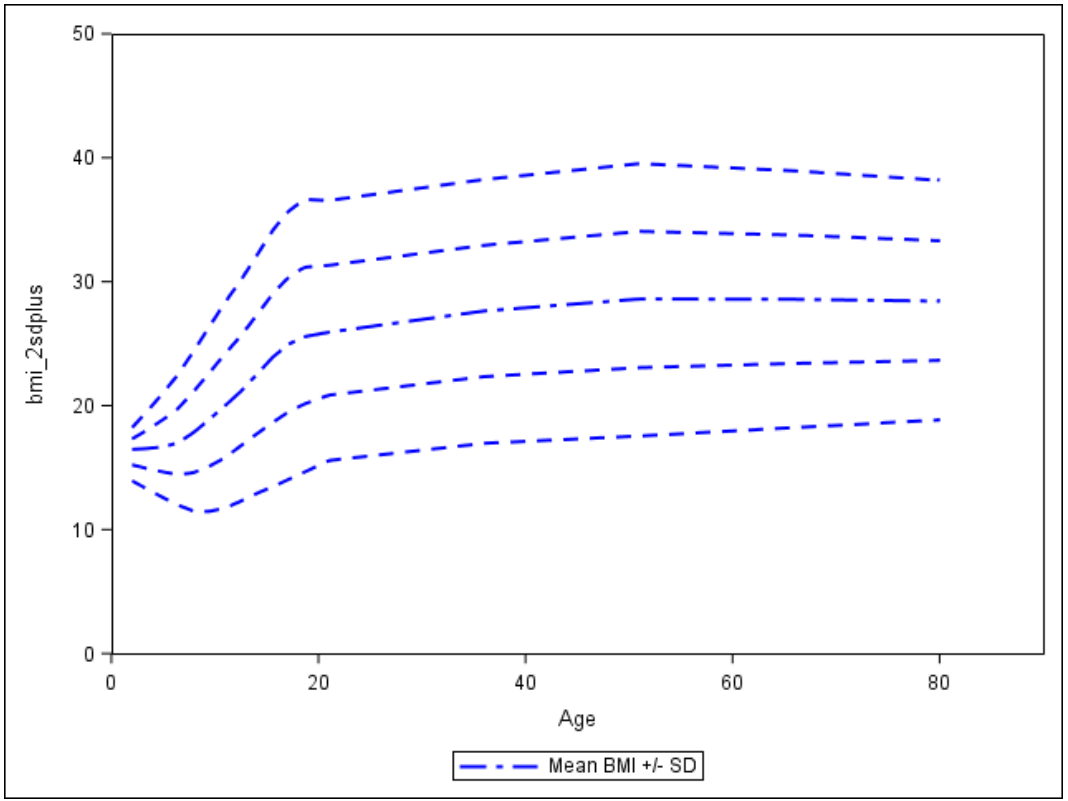


Adding “Section” variables

The data for this part of the graph is from bmimen_data, and identified by section=”BMI lines”.

```
** BMI lines **;
LOESS X = age Y = bmi_2sdplus
      / NOMARKERS LINEATTRS = (COLOR = BLUE PATTERN = 20);
LOESS X = age Y = bmi_sdplus
      / NOMARKERS LINEATTRS = (COLOR = BLUE PATTERN = 20);
LOESS X = age Y = bmi_mean
      / NOMARKERS LINEATTRS = (COLOR = BLUE)
        NAME = 'BMI' LEGENDLABEL = 'Mean BMI +/- SD';
LOESS X = age Y = bmi_sdless
      / NOMARKERS  LINEATTRS = (COLOR = BLUE PATTERN = 20);
LOESS X = age Y = bmi_2sdless
      / NOMARKERS LINEATTRS = (COLOR = BLUE PATTERN = 20);
```

Adding “Section” variables

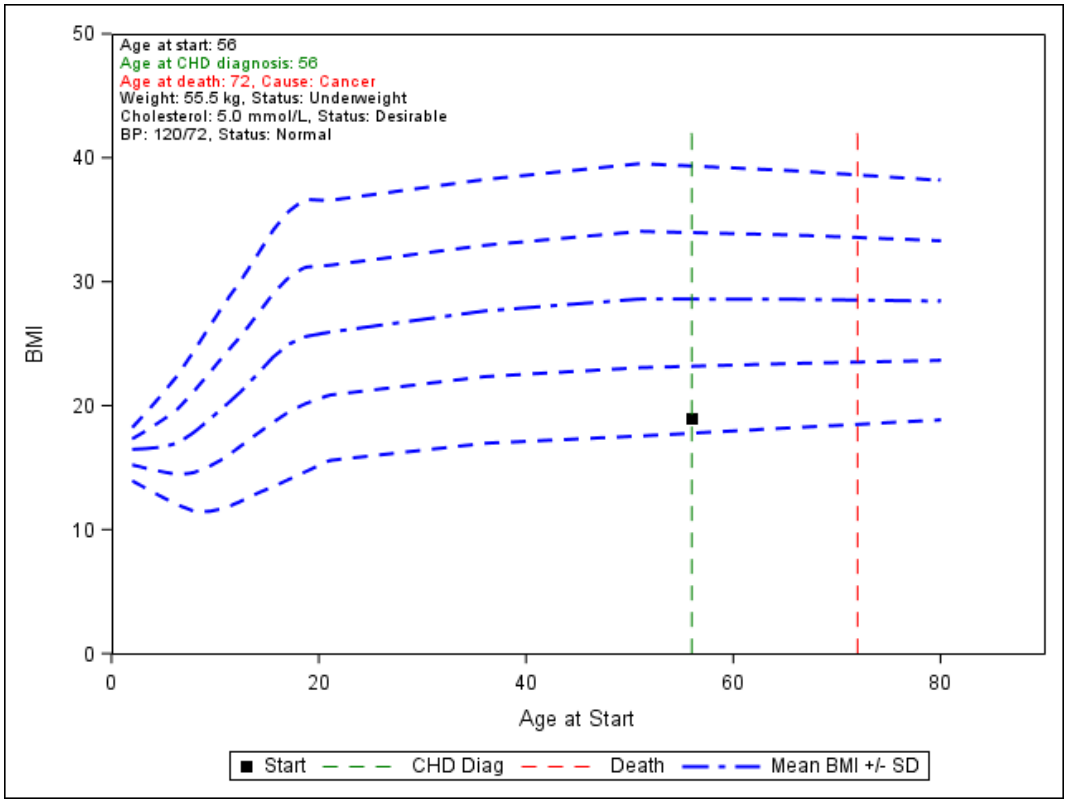


Adding “Section” variables

The primary axes are used with the main graph data for “Health” and “BMI lines”. The secondary axes are used to position the text in the top-left corner of the graph based on a fixed percentage of the y-axis.

```
** legend **;
KEYLEGEND 'Start' 'CHD Diag' 'Death' 'BMI';
** primary axes **;
XAXIS MIN = 0 MAX = 90 OFFSETMIN = 0 OFFSETMAX = 0;
YAXIS MIN = 0 MAX = 50 OFFSETMIN = 0 OFFSETMAX = 0;
** secondary axes, used to position the text box **;
X2AXIS MIN = 0 VALUES = (0 TO 1000 BY 1000) DISPLAY = NONE
       OFFSETMIN = 0.01 OFFSETMAX = 0;
Y2AXIS MIN = 0 VALUES = (0 TO 100 BY 100) DISPLAY = NONE
       OFFSETMAX = 0;
```

Adding “Section” variables



HOLLAND NUMERICS LIMITED

Validation of fonts, line patterns and colours

- So far I have deliberately avoided discussing fonts, line patterns and colours (highlighted above in green), because I firmly believe that they are outside the scope of validation.
- ODS Graphics even includes options to create editable graphics files (*.sge) as well, which can be edited using the ODS Graphics Editor to change fonts, line patterns and colours, but not the data displayed in the graph.
- Therefore I avoid any validation of fonts, line patterns and colours, preferring to focus on those features in the graphs that cannot be altered later.

Conclusions

Take full advantage of the ODS Graphics requirement for a single input data set:

- Include SAS variables that are required to create the graph.
- Include any variables necessary to calculate computed values in the graph, particularly for those variable used to generate TEXT statement values, or to subset the data to simplify validation.
- Do not include any variables that are not needed for the graph production or for validation.

Contact Details

Philip R Holland, SAS Consultant

Holland Numerics Limited

94 Green Drift, Royston, Herts SG8 5BT, UK

tel/fax: +44-7714-279085

email: phil@hollandnumerics.org.uk

web: blog.hollandnumerics.org.uk