PP24

Figures are Basic(ally Overlooked)

Kaleb Mead
MAC Clinical Research
kalebmead@macplc.com

1 INTRODUCTION

Figures play an important role in communicating clinical trial data. They convey complex information in a visually intuitive manner and enable anyone to understand critical insights effectively. But figures are often undervalued and there is a hesitancy among SAS® programmers to fully incorporate them.

We want to clearly explain the process of creating figures using SAS® and to empower SAS® programmers to adopt a more conscious and inclusive approach to figure development.

2 REASONS FOR AVOIDING FIGURES

- Perceived complexity
- Uncertainty in data interpretation
- Lack of training and knowledge
- Misconceptions about audience preference

3 BASIC FIGURE CREATION

PROC TEMPLATE

Within a PROC TEMPLATE procedure, there will be 4 sections to the figure structure.

1. DEFINE STATGRAPH – This is required to open a definition block for defining and naming a graphics template. The name should be a clear reflection of the figure (shown in Figure 1 as template_name).
2. BEGINGRAPH – This defines the outermost container for a graph template that is defined with GTL-statements. All template definitions in the Graphics Template Language must start with a BEGINGRAPH statement and end with an ENDGRAPH statement.
3. LAYOUT – Layout blocks always begin with the LAYOUT keyword followed by a keyword indicating the purpose of the layout. All layout blocks end with an ENDLAYOUT statement. These statements function like do/end blocks in SAS®.
4. PLOT – The Plot statement is where the desired plot type will be defined, this could be a SERIESPLOT, SCATTERPLOT, BOXPLOT, etc. (these would replace the plot_type in Figure 1).

```
proc template;
  define statgraph template_name;                 1
    begingraph;                                    2
      layout overlay;                              3
        plot_type x = xvariable y = yvariable;     4
      endlayout;
    endgraph;
  end;
run;
```

Example code of a PROC TEMPLATE procedure, labelled numerically.

Figure 1

PROC SGRENDER

To use the newly made template to output a figure - a PROC SGRENDER procedure is used. This takes both the template and the data we wish to use to create the figure. An example can be seen in Figure 2.

```
proc sgrender
  data = figure_data
  template = template_name;
run;
```

Example code of a PROC SGRENDER procedure.code.

Figure 2

First the data that is to be used for the figure is defined (shown as figure_data). Then the template used for the figure is defined (shown as template_name). These together will generate our figure in the output window.

4 PLOT TYPES

SCATTER PLOT

A scatter plot is a type of graph that displays individual data points on a two-dimensional plane, often used to show the relationship between two continuous variables. The data used for this example is the pre-packaged SAS® dataset detailing car information from the SASHELP library within SAS®. It is a dataset listing models of car, and their observed miles per gallon in a city (MPG_CITY), as well as their horsepower (HORSEPOWER), and body type (TYPE).

The code in Figure 5 is again, very similar to the example code we initially saw in Figure 1 and Figure 2, the additions/modifications are highlighted. We will look at the modifications made that weren't previously covered.

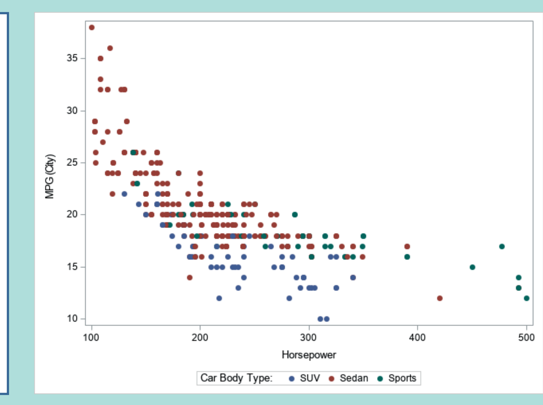
- (scatterplot) - specified that we will be using a scatter plot.
- (markerattrs) - opens the attributes for the markers used on the figure.
- (symbol = circlefilled) - swaps the symbol used for the markers out for a filled circle.

The output from these modifications is shown in Figure 6. More plot type examples are shown in the paper associated.

```
proc template;
  define statgraph scatter_template;

    begingraph;

      layout overlay;

        scatterplot x = horsepower y = mpg_city /
                                   group = type
                                   name = "scatter1"
                                   markerattrs = (symbol = circlefilled);

        discretelegend "scatter1" / title = "Car Body Type:";

      endlayout;

    endgraph;

  end;
run;

proc sgrender
  data = scatter_plot_data
  template = scatter_template;
run;
```

Example code for a Scatter Plot, with notable sections highlighted.

Figure 5



The Scatter Plot generated from the provided code and data

Figure 6

5 LAYOUTS

LAYOUT OPTIONS

The format for adding options is the same, however, there is the addition of determining how you want your output displayed and, if there is more than one option, how they interact together. This is shown in Figure 10.

In Figure 11 we see the layout types. Although there are differences between the options LATTICE and GRIDDED, as well as DATAPANEL and DATALATTICE, these present similarly so will be shown as the same in Figure 12.

```
layout LAYOUT TYPE / OPTION1
                     OPTION2 ;
```

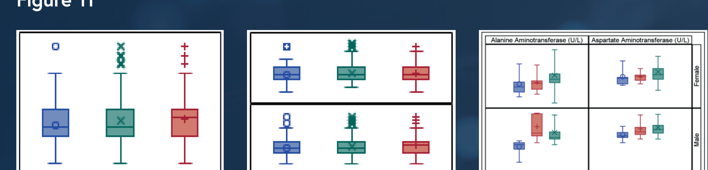
Example code for LAYOUT options

Figure 10

OVERLAY	One plot per page
LATTICE GRIDDED	Multiple Grouped Plots per page independent of data values
DATAPANEL DATALATTICE	Multiple Grouped Plots per page with shared axes

On left is the LAYOUT OPTION code and on the right the description of each option.

Figure 11



Example of an OVERLAY layout option. Example of a GRIDDED layout option. Example of a DATALATTICE layout option.

Figure 12

6 AXIS AND STYLE OPTIONS

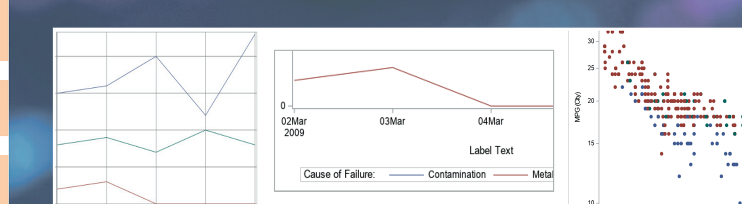
AXIS OPTIONS

In Figures 13 and 14 we will look at some of the options available within the layout section. The LAYOUT options we will use is xaxisopts = (OPTION) which opens options for the x-axis, there is also the y-axis version yaxisopts = (OPTION). These axis options will introduce a lot more flexibility into our figures.

GRIDDISPLAY = XX	Main inputs for XX here is ON or OFF. This will specify whether the axis lines are displayed.
GRIDATTRS = (XX = YY)	Controls the style of the axis lines, if displayed. This follows the general rule for attributes, which will be looked at momentarily.
LABEL = "XX"	Specifies the axis label. Can either a string or work dynamically.
LABELATTRS = (XX = YY)	Specifies the colour and font attributes of the axis label. Follows the general rule for attributes, which will be looked at after.
TYPE = XX	Specifies the type of axis wanted. Default is AUTO, which automatically determines the axis type (best practice is to select the axis type manually). Most of the time options LINEAR which uses the linear axis, and LOG which uses the log axis are used. More are available.

On left is the AXIS OPTION code and on the right the description of each option.

Figure 13



Example output with GRIDDISPLAY = ON. Example output with LABEL = "Label Text". Example output with TYPE = LOG.

Figure 14

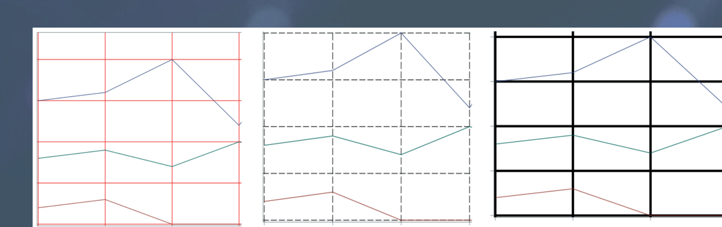
STYLE OPTIONS

For attribute options - it will depend on the category it would fall into. There are several but in Figures 15 and 16 we will look at the GRIDATTRS. GRIDATTRS falls into Line Options so we can specify the lines colour, pattern and thickness.

LINEATTRS = (COLOR = XX)	Specifies colour. XX can be given as a valid colour name, such as RED, or a colour code, such as CXFF0000 or #FF0000.
LINEATTRS = (PATTERN = XX)	Specifies the line pattern. XX can be given as the pattern number or as the pattern name (Solid, ShortDash, LongDash, etc.)
LINEATTRS = (THICKNESS = XX)	Specifies the thickness of the line. XX must be given as the desired thickness and the associated dimension (0.2in, 3mm, 10pct, etc.)

On left is the STYLE OPTION code and on the right the description of each option.

Figure 15



Example of output with GRIDATTRS = (COLOR = RED). Example of output with GRIDATTRS = (PATTERN = 4). Example of output with GRIDATTRS = (THICKNESS = 1pct).

Figure 16

ACCESSIBILITY

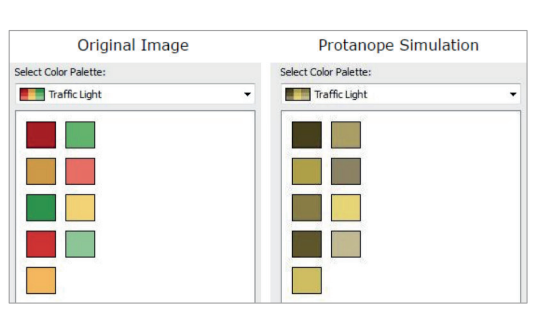
7 BASICS OF CVD

Colour vision deficiency (CVD), commonly known as colour-blindness, affects about 8% of men and 0.5% of women. There are many types of CVD but 98% people living with it have "red-green colour-blindness. They have difficulty telling the difference between red and green. This is by far the largest group we can easily improve accessibility for.

There is a lack of comprehensive resources and guidelines from industry regulators and leaders to address the needs of people with CVD. CDISC, the primary standards organisation in the clinical trials industry, provides minimal accommodation for CVD, with only a brief mention of colour considerations in their guidelines. The FDA only offers general guidance on electronic submissions, cautioning against the use of colour due to potential issues with printing or photocopying.

8 CVD ILLUSTRATED

In Figure 25, on the left a colour palette is shown containing greens, reds and oranges. On the right is a simulation of what that may look to someone with protanope CVD. Here we see that for someone with strong protanope CVD - their reds, greens, and oranges seem to blend together into shades of brown.



Comparison of colours for normal vision vs a protanope simulation.

Figure 25

9 AN EXAMPLE IN SAS

Now we will look at a scatter plot which is similar to the one covered earlier. We will be displaying several car makes and their respective MSRP (manufacturer's suggested retail price) vs MPG in the city.

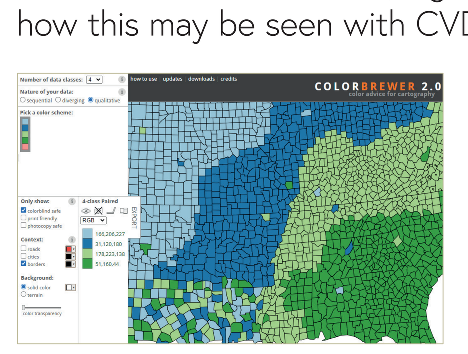
We will use ColorBrewer2.0 to decide on a colour palette. There are four data classes (Audi, Jaguar, Suzuki, and Volvo), the data we are displaying is qualitative and we need it to be usable for people with CVD. With these options selected we are given the colour palette we need to move forward. This process is shown in Figure 29.

To use the colours we've now found, some modifications must be made to the "standard" scatter plot code.

- (%colormac) - this macro utility would be put before the PROC TEMPLATE. It was created to help SAS® users use alternatives to SAS® inbuilt colour codes. We will be using the RGB colour codes now available.

The next two modifications would be made as BEGINGRAPH options:

- (attrpriority = none) - this changes the priority list that SAS® automatically selects off and changes to NONE.
- (datacontrastcolors=(...)) - this allows us to change the contrast colours of the graphics element, such as lines and in this case markers.
- (%rgb(166,206,227)) - this macro will be called inside DATACONTRASTCOLORS, and allows us to use the RGB codes we found in ColorBrewer2.0, in SAS®. This is repeated for all four colour codes.



ColorBrewer2.0 for this specific instance.

Figure 29

The output, shown in Figure 31, from our SAS® code is effectively displaying our data while also ensuring it is accessible to people with CVD.

To illustrate the difference this could make, Figure 32 has the same output as Figure 31, but we have used colours that may hinder someone with CVD. To simulate its effect to wider audiences', Figure 33 shows how this may be seen with CVD.

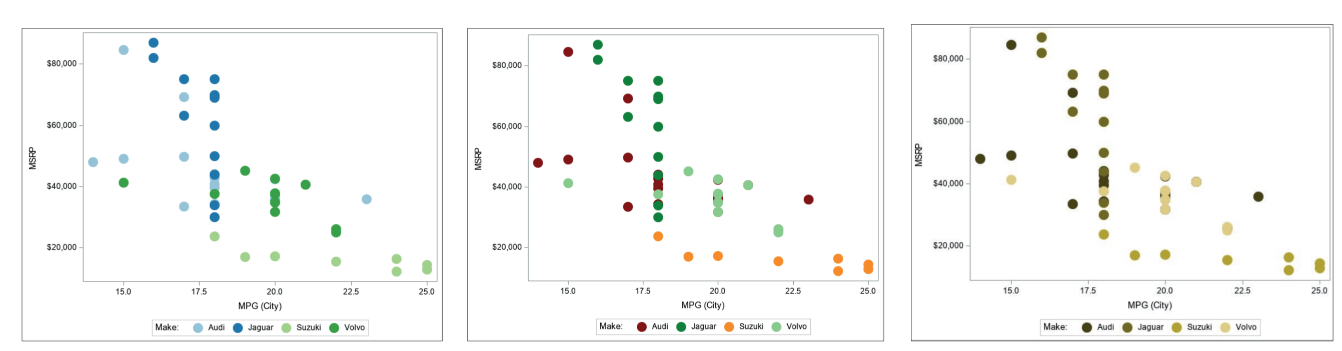


Figure output using the given colours and code. Figure output using colours that may confuse someone with CVD. Simulation of how Figure 32 may look for someone with CVD.

Figure 31 **Figure 32** **Figure 33**

10 CONCLUSION

Figures are incredible mediums for communicating complex results from data and can be adopted by newcomers with ease. However, even if the use of figures becomes widespread, this will be a hollow achievement if they are inaccessible to a large percentage of the population.

By shedding light on the challenges this presents to individuals with CVD and demonstrating how easily these issues can be resolved - we hope to encourage collaborative efforts within the industry to establish future guidelines that will ensure figures are produced and utilized in a manner that is accessible to all.

Scan QR code to read full paper