Paper 038-2013

Creating Graph Collections with Consistent Colors using ODS Graphics?

Philip R Holland, Holland Numerics Ltd, Royston, Hertfordshire, United Kingdom

ABSTRACT

Collections of line graphs or bar charts, where the graph data is grouped by the same value, are frequently used to identify differences and similarities in behavior. Unfortunately, by default, the colors used for each line can change across the graph collection if some group values are not present in every graph. In SAS/GRAPH™ this problem has been solved by generating SYMBOL or PATTERN statements based on the data, or using annotation to create all of the graph lines, bars and legends. Neither of these solutions is readily available in ODS Graphics, so this paper will explore how to solve this problem using macros to implement the graphic "layer" approach in PROC SGPLOT and PROC TEMPLATE, so that the user has complete control over the data colors and styles applied to every graph.

INTRODUCTION

The aim of this paper is to create a macro that draws up to 4 graphs:

- A combined graph of **Type A** values, plus optional values from **Type B**, with matching colors for data from specific **id**s, including color-coded footnotes for the graph legend.
- Up to 3 individual graphs with **Type A** values, plus optional values from **Type B**, for each **id**, but with colors that match the corresponding lines in the combined graph.

The sample data includes the data points for **Type A** and **Type B** for 2 **group**s of 3 **id**s each on 3 **day**s, where each **id** has a symbol **A**, **B** or **C**. The data points are then copied into columns specific to each **id** to simplify separation when plotting.

```
%LET seed = 1234;

%LET maxgroup = 2;
%LET maxid = 3;
%LET maxday = 3;

DATA grafdata;
    LENGTH group type $10 fullid $16 sym $1;
    ARRAY aval {*} avalue1-avalue&maxid.;
    ARRAY bval {*} bvalue1-bvalue&maxid.;
    LABEL day = 'Day';
    DO i = 1 TO &maxgroup.;
        group = 'Group ' || STRIP(PUT(i, BEST.));
        DO j = 1 TO &maxid.;
            id = i * 1000 + j;
            sym = BYTE(MOD(id, 1000) + 64);
            DO day = 1 TO &maxday.;
                DO k = 1 TO DIM(aval);
                    aval[k] = .;
                    bval[k] = .;
                END;
                DO type = 'Type A', 'Type B';
                    fullid = PUT(id, z4.) || ' ' || type;
                    value = RANUNI(&seed.) * 1000;
                    IF type = 'Type A' THEN DO;
                        avalue = value;
                        bvalue = .;
                        aval[j] = value;
                        bval[j] = .;
                    END;
                    ELSE DO;
                        bvalue = value;
                        avalue = .;
                        bval[j] = value;
                        aval[j] = .;
                    END;
```

```
                    OUTPUT;
                END;
            END;
        END;
    END;
RUN;
```

	group	id	value	day	type	fullid	sym	avalue1	avalue2	avalue3	bvalue1	bvalue2	bvalue3
1	Group 1	1001	243.811160...	1	Type A	1001 Type A	A	243.81116...	.	.	.	.	.
2	Group 1	1001	89.4751134...	1	Type B	1001 Type B	A	.	.	.	89.475113...	.	.
3	Group 1	1001	383.193714...	2	Type A	1001 Type A	A	383.19371...	.	.	.	.	.
4	Group 1	1001	97.9283946...	2	Type B	1001 Type B	A	.	.	.	97.928394...	.	.
5	Group 1	1001	257.581086...	3	Type A	1001 Type A	A	257.58108...	.	.	.	.	.
6	Group 1	1001	88.2495940...	3	Type B	1001 Type B	A	.	.	.	88.249594...	.	.
7	Group 1	1002	36.2996664...	1	Type A	1002 Type A	B	.	36.299666...	.	.	.	.
8	Group 1	1002	107.585959...	1	Type B	1002 Type B	B	.	.	.	.	107.585959...	.
9	Group 1	1002	445.828340...	2	Type A	1002 Type A	B	.	445.82834...	.	.	.	.
10	Group 1	1002	143.124675...	2	Type B	1002 Type B	B	.	.	.	.	143.124675...	.
11	Group 1	1002	39.1160934...	3	Type A	1002 Type A	B	.	39.116093...	.	.	.	.
12	Group 1	1002	456.223492...	3	Type B	1002 Type B	B	.	.	.	.	456.223492...	.
13	Group 1	1003	88.5795928...	1	Type A	1003 Type A	C	.	.	88.579592...	.	.	.
14	Group 1	1003	905.767766...	1	Type B	1003 Type B	C	.	.	.	.	.	905.767766...
15	Group 1	1003	967.4402871	2	Type A	1003 Type A	C	.	.	967.44028...	.	.	.
16	Group 1	1003	737.351761...	2	Type B	1003 Type B	C	.	.	.	.	.	737.351761...
17	Group 1	1003	402.947708...	3	Type A	1003 Type A	C	.	.	402.94770...	.	.	.
18	Group 1	1003	373.164565...	3	Type B	1003 Type B	C	.	.	.	.	.	373.164565...
19	Group 2	2001	327.840219...	1	Type A	2001 Type A	A	327.84021...	.	.	.	.	.
20	Group 2	2001	510.277885...	1	Type B	2001 Type B	A	.	.	.	510.27788...	.	.
21	Group 2	2001	283.903048...	2	Type A	2001 Type A	A	283.90304...	.	.	.	.	.
22	Group 2	2001	350.611041...	2	Type B	2001 Type B	A	.	.	.	350.61104...	.	.
23	Group 2	2001	255.775078...	3	Type A	2001 Type A	A	255.77507...	.	.	.	.	.
24	Group 2	2001	436.403616...	3	Type B	2001 Type B	A	.	.	.	436.40361...	.	.

Figure 1: Sample of grafdata, showing values separated by id and type

MACROS AND PROC SGPLOT

The most obvious option would be to use PROC SGPLOT and overlay separate lines for each id. The macro can then select which lines and footnotes to include based on the parameters.

The following code is simplified version of the final PROC SGPLOT code, showing the points (plotted using the SCATTER statements) and the joining lines (plotted using the SERIES statements) overlaid on each other. The XAXIS, YAXIS and Y2AXIS statements are used to alter the x-axis, y-axis and secondary y-axis, if the default settings are not appropriate.

```
PROC SGPLOT DATA = grafdata NOAUTOLEGEND TMPLOUT = "sgplot.sas";
  BY group;
  SCATTER X = day Y = avalue1 /
          GROUP = fullid MARKERCHAR = sym MARKERCHARATTRS = (COLOR=blue)
          NAME = 'scatter';
  SCATTER X = day Y = bvalue1 /
          GROUP = fullid Y2AXIS MARKERCHAR = sym MARKERCHARATTRS = (COLOR=blue);
  SERIES X = day Y = avalue1 /
         GROUP = fullid LINEATTRS = (PATTERN=SOLID COLOR=blue);
  SERIES X = day Y = bvalue1 /
         GROUP = fullid Y2AXIS LINEATTRS = (PATTERN=DASH COLOR=blue);
  YAXIS LABEL = 'Type A' MIN = 0;
  Y2AXIS LABEL = 'Type B' MIN = 0;
  XAXIS VALUES = (1 to 3);
  FOOTNOTE1 COLOR = blue "A (solid) = 1001 Type A, A (dash) = 1001 Type B";
RUN;
```

The **%plot_all** macro statement needs to include all the parameters required for generating the SAS code, and with sensible defaults, so that most of the parameters can be omitted in normal usage:

```
%MACRO plot_all(plot=1, dsn=, by=, group=, x=day, xaxis=, ylabel1=, ylabel2=,
                a=avalue, b=bvalue, symbol=sym, ids=3,
                color1=blue, color2=red, color3=green, color4=black, color5=violet);
```

Next the macro variables &anote*n*, and optionally &bnote*n*, should be initialized to blank to prevent macro values being passed from previous macro runs. The macro variables &anote*n*, and optionally &bnote*n*, are then calculated from the input data:

```
%DO i = 1 %TO &ids.;
    %LET anote&i. =;
    %IF "&ylabel2. " NE " " %THEN %DO;
        %LET bnote&i. =;
    %END;
%END;

PROC SQL NOPRINT;
%DO i = 1 %TO &ids.;
    SELECT DISTINCT STRIP(&symbol.) || ' (solid) = ' || STRIP(&group.)
    INTO   :anote&i.
    FROM   &dsn.
    WHERE  &a&i. IS NOT MISSING;
    %IF "&ylabel2. " NE " " %THEN %DO;
        SELECT DISTINCT STRIP(&symbol.) || ' (dash) = ' || STRIP(&group.)
        INTO   :bnote&i.
        FROM   &dsn.
        WHERE  &b&i. IS NOT MISSING;
    %END;
%END;
QUIT;
```

The PROC SGPLOT code for the combined graph is then generated in a similar way to the sample SAS code above:

```
PROC SGPLOT DATA = &dsn. NOAUTOLEGEND TMPLOUT = "sgplot_group&plot..sas";
    BY &by.;
%DO i = 1 %TO &ids.;
    SCATTER X = &x. Y = &a.&i. /
            GROUP = &group.
            MARKERCHAR = &symbol. MARKERCHARATTRS = (COLOR=&&color&i.)
            NAME = 'scatter';
    %IF "&ylabel2. " NE " " %THEN %DO;
        SCATTER X = &x. Y = &b.&i. /
                GROUP = &group. Y2AXIS
                MARKERCHAR = &symbol. MARKERCHARATTRS = (COLOR=&&color&i.);
    %END;
    SERIES X = &x. Y = &a.&i. /
            GROUP = &group.
            LINEATTRS = (PATTERN=SOLID COLOR=&&color&i.);
    %IF "&ylabel2. " NE " " %THEN %DO;
        SERIES X = &x. Y = &b.&i. /
               GROUP = &group. Y2AXIS
               LINEATTRS = (PATTERN=DASH COLOR=&&color&i.);
    %END;
%END;
    YAXIS LABEL = &ylabel1. MIN = 0;
    %IF "&ylabel2. " NE " " %THEN %DO;
        Y2AXIS LABEL = &ylabel2. MIN = 0;
    %END;
%IF &xaxis. NE %THEN %DO;
    XAXIS VALUES = (&xaxis.) ;
%END;
```

```
    %DO i = 1 %TO &ids.;
        %IF "&ylabel2. " NE " " %THEN %DO;
            FOOTNOTE&i. COLOR = &&color&i.
                        "%sysfunc(strip(&&anote&i.)), %sysfunc(strip(&&bnote&i.))";
        %END;
        %ELSE %DO;
            FOOTNOTE&i. COLOR = &&color&i. "%sysfunc(strip(&&anote&i.))";
        %END;
    %END;
    RUN;
```

The individual graphs are then added, using a loop, for each id value, again in a similar way to the sample SAS code above:

```
    %DO i = 1 %TO &ids.;
    PROC SGPLOT DATA = &dsn. NOAUTOLEGEND TMPLOUT = "sgplot_indiv&plot._&i..sas";
        BY &by.;
        SCATTER X = &x. Y = &a.&i. /
                GROUP = &group.
                MARKERCHAR = &symbol. MARKERCHARATTRS = (COLOR=&&color&i.);
        %IF "&ylabel2. " NE " " %THEN %DO;
            SCATTER X = &x. Y = &b.&i. /
                    GROUP = &group. Y2AXIS
                    MARKERCHAR = &symbol. MARKERCHARATTRS = (COLOR=&&color&i.);
        %END;
        SERIES X = &x. Y = &a.&i. /
               GROUP = &group.
               LINEATTRS = (PATTERN=SOLID COLOR=&&color&i.);
        %IF "&ylabel2. " NE " " %THEN %DO;
            SERIES X = &x. Y = &b.&i. /
                   GROUP = &group. Y2AXIS
                   LINEATTRS = (PATTERN=DASH COLOR=&&color&i.);
        %END;
        YAXIS LABEL = &ylabel1. MIN = 0;
        %IF "&ylabel2. " NE " " %THEN %DO;
            Y2AXIS LABEL = &ylabel2. MIN = 0;
        %END;
    %IF &xaxis. NE %THEN %DO;
        XAXIS VALUES = (&xaxis.);
    %END;
        %IF "&ylabel2. " NE " " %THEN %DO;
            FOOTNOTE1 COLOR = &&color&i.
                      "%sysfunc(strip(&&anote&i.)), %sysfunc(strip(&&bnote&i.))";
        %END;
        %ELSE %DO;
            FOOTNOTE1 COLOR = &&color&i. "%sysfunc(strip(&&anote&i.))";
        %END;
    RUN;
    %END;
%MEND plot_all;
```

This method also allows the modification of the tick marks on the x-axis in a similar way to that used on AXIS statements for PROC GPLOT or PROC GCHART.

Note that the PROC SGPLOT statements all include TMPLOUT= options, which will save Graph Templates to replay these graphs later.

```
TITLE 'Testing graphs';

%plot_all(plot=1, dsn=grafdata(where=(group='Group 1')), by=group, group=fullid,
          xaxis=1 to 3, ylabel1='Type A', ylabel2='Type B')

%plot_all(plot=2, dsn=grafdata(where=(group='Group 2')), by=group, group=fullid,
          ylabel1='Type A', ylabel2='Type B')
```

```
%plot_all(plot=3, dsn=grafdata(where=(group='Group 1')), by=group, group=fullid,
          ylabel1='Type A')
```

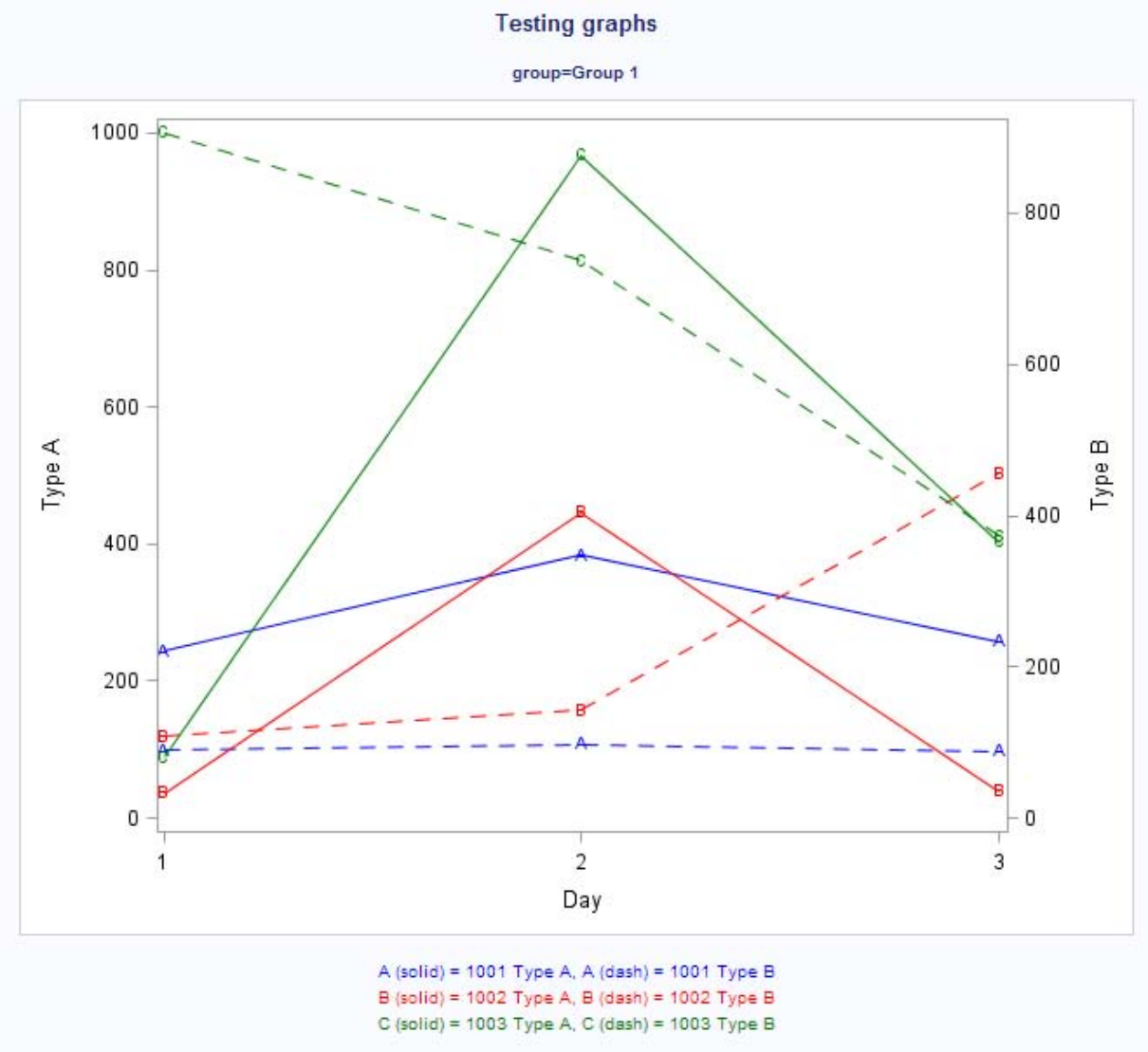

Figure 2: Combined graph from first macro call created by PROC SGPLOT (plot=1, id=1001, 1002, 1003)

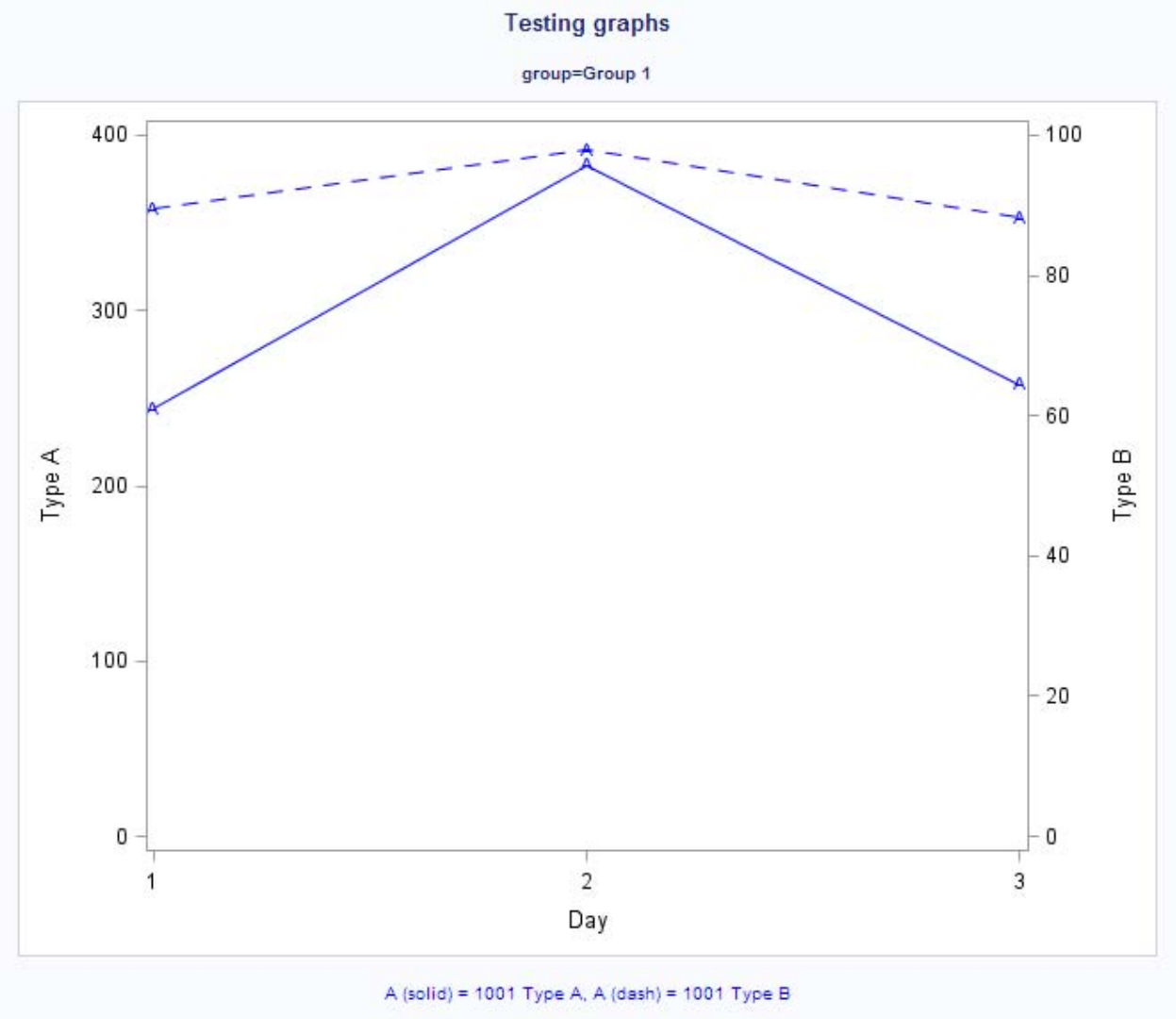

Figure 3: Individual graph from first macro call created by PROC SGPLOT (plot=1, id=1001)

MACROS AND GRAPH TEMPLATES

Converting PROC SGPLOT code to Graph Templates ought to be fairly straightforward using the generated Graph Templates.

```
proc template;
  define statgraph sgplot;
    dynamic __BYLINE__;
    begingraph /;
      EntryTitle "Testing graphs" /;
      EntryTitle __BYLINE__ /
                 textattrs=(size=GraphLabelText:fontsize);
      layout overlay /
             xaxisopts=( type=linear
                         linearopts=( tickvaluelist=( 1 2 3 ) viewmin=1
                                      viewmax=3 ) )
             yaxisopts=( Label="Type A" type=linear linearopts=( viewmin=0 ) )
             y2axisopts=( Label="Type B" type=linear linearopts=( viewmin=0 ) );
```

```
        ScatterPlot X=day Y=avalue1 /
                    primary=true Group=fullid MarkerCharacter=sym
                    MarkerCharacterAttrs=( Color=CX0000FF) LegendLabel="avalue1"
                    NAME="scatter";
        ScatterPlot X=day Y=bvalue1 /
                    yaxis=y2 Group=fullid MarkerCharacter=sym
                    MarkerCharacterAttrs=( Color=CX0000FF) LegendLabel="bvalue1"
                    NAME="SCATTER";
        SeriesPlot X=day Y=avalue1 /
                   Group=fullid Lineattrs=( Color=CX0000FF Pattern=1)
                   LegendLabel="avalue1" NAME="SERIES";
        SeriesPlot X=day Y=bvalue1 /
                   yaxis=y2 Group=fullid Lineattrs=( Color=CX0000FF Pattern=20)
                   LegendLabel="bvalue1" NAME="SERIES1";
        ScatterPlot X=day Y=avalue2 /
                    Group=fullid MarkerCharacter=sym
                    MarkerCharacterAttrs=( Color=CXFF0000) LegendLabel="avalue2"
                    NAME="scatter";
        ScatterPlot X=day Y=bvalue2 /
                    yaxis=y2 Group=fullid MarkerCharacter=sym
                    MarkerCharacterAttrs=( Color=CXFF0000) LegendLabel="bvalue2"
                    NAME="SCATTER1";
        SeriesPlot X=day Y=avalue2 /
                   Group=fullid Lineattrs=( Color=CXFF0000 Pattern=1)
                   LegendLabel="avalue2" NAME="SERIES2";
        SeriesPlot X=day Y=bvalue2 /
                   yaxis=y2 Group=fullid Lineattrs=( Color=CXFF0000 Pattern=20)
                   LegendLabel="bvalue2" NAME="SERIES3";
        ScatterPlot X=day Y=avalue3 /
                    Group=fullid MarkerCharacter=sym
                    MarkerCharacterAttrs=( Color=CX008000) LegendLabel="avalue3"
                    NAME="scatter";
        ScatterPlot X=day Y=bvalue3 /
                    yaxis=y2 Group=fullid MarkerCharacter=sym
                    MarkerCharacterAttrs=( Color=CX008000) LegendLabel="bvalue3"
                    NAME="SCATTER2";
        SeriesPlot X=day Y=avalue3 /
                   Group=fullid Lineattrs=( Color=CX008000 Pattern=1)
                   LegendLabel="avalue3" NAME="SERIES4";
        SeriesPlot X=day Y=bvalue3 /
                   yaxis=y2 Group=fullid Lineattrs=( Color=CX008000 Pattern=20)
                   LegendLabel="bvalue3" NAME="SERIES5";
      endlayout;
      EntryFootnote "A (solid) = 1001 Type A, A (dash) = 1001 Type B" /
                    textattrs=( color=CX0000FF);
      EntryFootnote "B (solid) = 1002 Type A, B (dash) = 1002 Type B" /
                    textattrs=( color=CXFF0000);
      EntryFootnote "C (solid) = 1003 Type A, C (dash) = 1003 Type B" /
                    textattrs=( color=CX008000);
    endgraph;
  end;
run;
```

Figure 4: Generated Graph Template from the combined graph

```
proc template;
  define statgraph sgplot;
    dynamic __BYLINE__;
    begingraph /;
      EntryTitle "Testing graphs" /;
      EntryTitle __BYLINE__ /
                 textattrs=(size=GraphLabelText:fontsize);
```

```
      layout overlay /
             xaxisopts=( type=linear
                         linearopts=( tickvaluelist=( 1 2 3 ) viewmin=1
                                      viewmax=3 ) )
             yaxisopts=( Label="Type A" type=linear linearopts=( viewmin=0 ) )
             y2axisopts=( Label="Type B" type=linear linearopts=( viewmin=0 ) );
        ScatterPlot X=day Y=avalue1 /
                    primary=true Group=fullid MarkerCharacter=sym
                    MarkerCharacterAttrs=( Color=CX0000FF) LegendLabel="avalue1"
                    NAME="SCATTER";
        ScatterPlot X=day Y=bvalue1 /
                    yaxis=y2 Group=fullid MarkerCharacter=sym
                    MarkerCharacterAttrs=( Color=CX0000FF) LegendLabel="bvalue1"
                    NAME="SCATTER1";
        SeriesPlot X=day Y=avalue1 /
                   Group=fullid Lineattrs=( Color=CX0000FF Pattern=1)
                   LegendLabel="avalue1" NAME="SERIES";
        SeriesPlot X=day Y=bvalue1 /
                   yaxis=y2 Group=fullid Lineattrs=( Color=CX0000FF Pattern=20)
                   LegendLabel="bvalue1" NAME="SERIES1";
      endlayout;
      EntryFootnote "A (solid) = 1001 Type A, A (dash) = 1001 Type B" /
                    textattrs=( color=CX0000FF);
    endgraph;
  end;
run;
```

Figure 5: Generated Graph Template from one of the individual graphs

The following **%template_make** macro includes the majority of these generated Graph Templates, but reduces and generalizes the number of code lines by using macro loops, so that the PROC TEMPLATE code can be generated with a user-specified maximum number of lines maxids=.

```
%MACRO template_make(maxids=);
    PROC TEMPLATE;
        DEFINE STATGRAPH colour_lineplot;
            DYNAMIC _group _x _symbol
                    _ylabel1 _ylabel2
    %DO i = 1 %TO &maxids.;
                    _a&i. _b&i. _color&i. _foot&i.
    %END;
                    ;
            BEGINGRAPH;
```

The next section will depend on whether the graph has just a primary y-axis, or both primary and secondary y-axes, depending on the presence of the **_ylabel2** parameter value, which is tested with the EXISTS() function. If the **_ylabel2** parameter is present, then the graph will have two y-axes:

```
                IF (EXISTS(_ylabel2))
                    LAYOUT OVERLAY /
                           YAXISOPTS = (LABEL=_ylabel1 LINEAROPTS=(VIEWMIN=0))
                           Y2AXISOPTS = (LABEL=_ylabel2 LINEAROPTS=(VIEWMIN=0));
    %DO i = 1 %TO &maxids.;
                    IF (EXISTS(_a&i.))
                        SCATTERPLOT X = _x Y = _a&i. /
                                    GROUP = _group MARKERCHARACTER = _symbol
                                    MARKERCHARACTERATTRS = (COLOR=_color&i.);
                        SERIESPLOT X = _x Y = _a&i. /
                                   GROUP = _group
                                   LINEATTRS = (PATTERN=SOLID COLOR=_color&i.);
                    ENDIF;
```

```
                    IF (EXISTS(_b&i.))
                        SCATTERPLOT X = _x Y = _b&i. /
                                    YAXIS = Y2 GROUP = _group
                                    MARKERCHARACTER = _symbol
                                    MARKERCHARACTERATTRS = (COLOR=_color&i.);
                        SERIESPLOT X = _x Y = _b&i. /
                                   YAXIS = Y2 GROUP = _group
                                   LINEATTRS = (PATTERN=DASH COLOR=_color&i.);
                    ENDIF;
    %END;
                    ENDLAYOUT;
```

If the **_ylabel2** parameter is not present, then the graph will only have one y-axis:

```
                ELSE LAYOUT OVERLAY /
                             YAXISOPTS = (LABEL=_ylabel1 LINEAROPTS=(VIEWMIN=0));
    %DO i = 1 %TO &maxids.;
                    IF (EXISTS(_a&i.))
                        SCATTERPLOT X = _x Y = _a&i. /
                                    GROUP = _group MARKERCHARACTER = _symbol
                                    MARKERCHARACTERATTRS = (COLOR=_color&i.);
                        SERIESPLOT X = _x Y = _a&i. /
                                   GROUP = _group
                                   LINEATTRS = (PATTERN=SOLID COLOR=_color&i.);
                    ENDIF;
    %END;
                    ENDLAYOUT;
                ENDIF;
    %DO i = 1 %TO &maxids.;
                ENTRYFOOTNOTE _foot&i. / TEXTATTRS = (COLOR=_color&i.);
    %END;
            ENDGRAPH;
        END;
    RUN;
%MEND template_make;
```

The **%template_make** macro is then submitted to set a maximum allowed lines at 5:

```
%template_make(maxids=5);
```

The **%template_all** macro is used to generate the PROC SGRENDER code to display the graph data using the Graph Templates. However, the plot= and xaxis= parameters from the **%plot_all** macro have been removed:

- plot= was used in **%plot_all** to label the TMPLOUT= program names, which will not be created with the **%template_all** macro.
- xaxis= was used in **%plot_all** to amend the x-axis tick marks, e.g. xaxis=1 to 3. However, **%template_all** would only have permitted xaxis=1 2 3, so has been removed to avoid confusion. PROC SGPLOT converts "1 to 3" into "1 2 3", but the Graph Template does not, so extra text manipulation would have been required in **%template_all**.

```
%MACRO template_all(dsn=, by=, group=, x=day, ylabel1=, ylabel2=,
                    a=avalue, b=bvalue, symbol=sym, ids=3,
                    color1=blue, color2=red, color3=green, color4=black,
                    color5=violet);
```

As in **%plot_all** the macro variables &anote*n*, and optionally &bnote*n*, should be initialized to blank to prevent macro values being passed from previous macro runs. The macro variables &anote*n*, and optionally &bnote*n*, are then calculated from the input data:

```
%DO i = 1 %TO &ids.;
    %LET anote&i. =;
    %IF "&ylabel2. " NE " " %THEN %DO;
        %LET bnote&i. =;
    %END;
%END;

PROC SQL NOPRINT;
%DO i = 1 %TO &ids.;
    SELECT DISTINCT STRIP(&symbol.) || ' (solid) = ' || STRIP(&group.)
    INTO   :anote&i.
    FROM   &dsn.
    WHERE  &a&i. IS NOT MISSING;
    %IF "&ylabel2. " NE " " %THEN %DO;
        SELECT DISTINCT STRIP(&symbol.) || ' (dash) = ' || STRIP(&group.)
        INTO   :bnote&i.
        FROM   &dsn.
        WHERE  &b&i. IS NOT MISSING;
    %END;
%END;
QUIT;
```

The PROC SGRENDER code for the combined graph is then generated to pass the macro parameters to the specified Graph Template:

```
PROC SGRENDER DATA = &dsn. TEMPLATE = 'colour_lineplot';
    BY &by.;
    DYNAMIC _group = "&group." _x = "&x." _symbol = "&symbol."
%DO i = 1 %TO &ids.;
            _a&i. = "&a.&i."
            _b&i. = "&b.&i."
            _color&i. = "&&color&i."
%END;
            _ylabel1 = "&ylabel1."
%IF "&ylabel2. " NE " " %THEN %DO;
            _ylabel2 = "&ylabel2."
    %DO i = 1 %TO &ids.;
            _foot&i. =
                "%sysfunc(strip(&&anote&i.)), %sysfunc(strip(&&bnote&i.))"
    %END;
%END;
%ELSE %DO;
    %DO i = 1 %TO &ids.;
            _foot&i. = "%sysfunc(strip(&&anote&i.))"
    %END;
%END;
            ;
RUN;
```

The PROC SGRENDER code for each of the individual graphs is then generated to pass the macro parameters to the specified Graph Template:

```
%DO i = 1 %TO &ids.;
PROC SGRENDER DATA = &dsn. TEMPLATE = 'colour_lineplot';
    BY &by.;
```

```
        DYNAMIC _group = "&group." _x = "&x." _symbol = "&symbol."
                _a&i. = "&a.&i."
                _b&i. = "&b.&i."
                _color&i. = "&&color&i."
                _ylabel1 = "&ylabel1."
        %IF "&ylabel2. " NE " " %THEN %DO;
                _ylabel2 = "&ylabel2."
                _foot&i. =
                     "%sysfunc(strip(&&anote&i.)), %sysfunc(strip(&&bnote&i.))"
        %END;
        %ELSE %DO;
                _foot&i. = "%sysfunc(strip(&&anote&i.))"
        %END;
                ;
    RUN;
    %END;
%MEND template_all;
```

The macro is then called to generate the requested graph collections. Note that the footnotes are now inside the graphs, so a FOOTNOTE statement is used to reset any existing footnotes:

```
TITLE 'Testing templates';
FOOTNOTE;

%template_all(dsn=grafdata(where=(group='Group 1')), by=group, group=fullid,
              ylabel1=Type A, ylabel2=Type B)

%template_all(dsn=grafdata(where=(group='Group 2')), by=group, group=fullid,
              ylabel1=Type A, ylabel2=Type B)

%template_all(dsn=grafdata(where=(group='Group 1')), by=group, group=fullid,
              ylabel1=Type A)
```

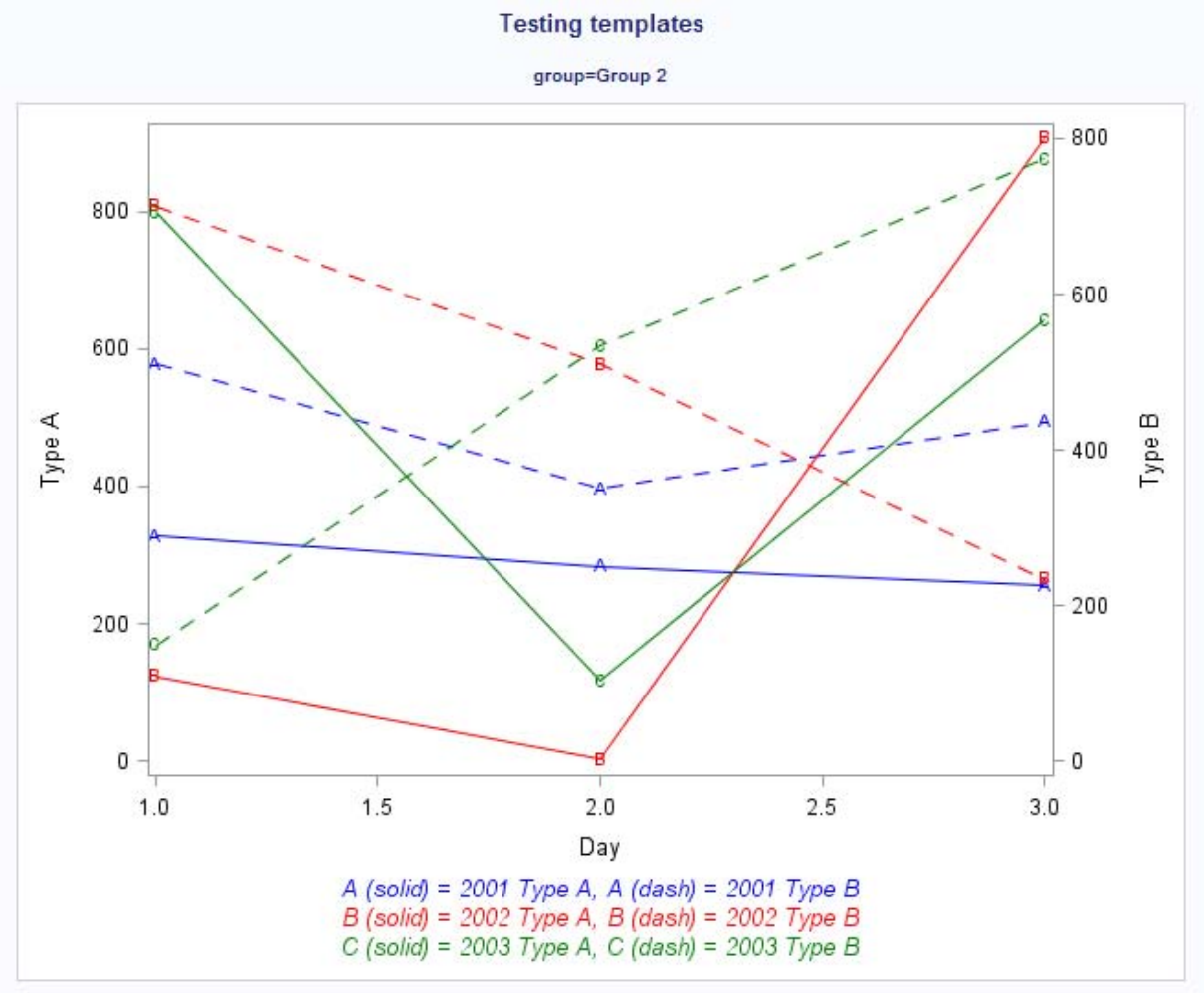

Figure 6: Combined graph from second macro call created by PROC SGPLOT (id=2001, 2002, 2003)

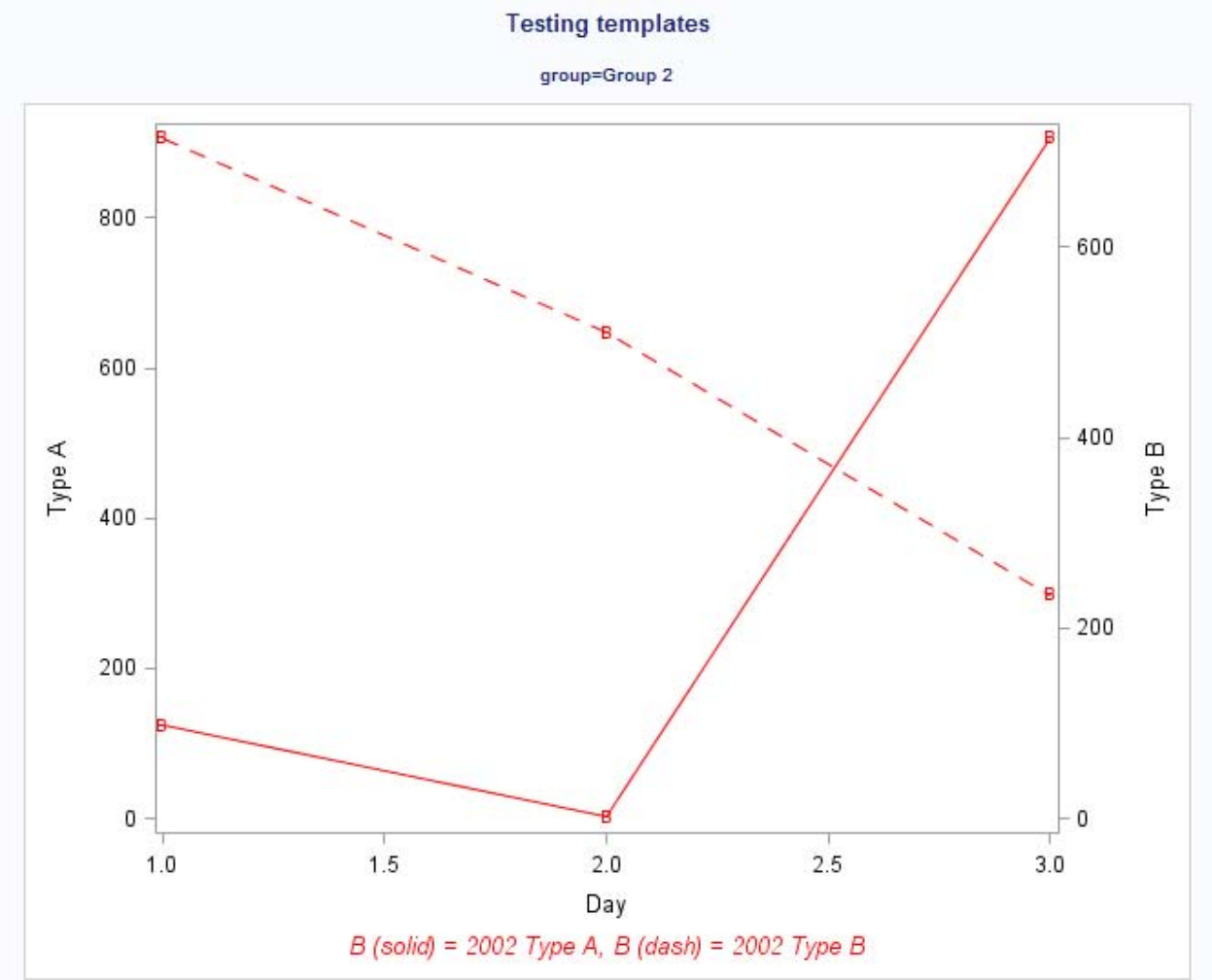

Figure 7: Individual graph from second macro call created by PROC SGPLOT (id=2002)

CONCLUSIONS

The conclusions that can be drawn from these techniques are:

- ODS Graphics cannot be used in the same way as traditional SAS/GRAPH, as the data to be displayed on any graph must be stored in a single input file.
- There are problems, like keeping consistent colors, that is easier to solve using PROC SGPLOT than using Graph Templates and SGRENDER, because PROC SGPLOT automatically converts some of its statements from familiar SAS/GRAPH syntax, whereas in Graph Templates this syntax needs to be specified and converted manually.

RECOMMENDED READING

- Philip R Holland, “Power User's Guide to SAS Graph Templates”, ISBN 978-1-291-31249-2, Lulu.com (2013)
- Philip R Holland, “Developing ODS Templates for Non-standard Graphs in SAS® 9.2”, 226-2010, SAS Global Forum (2010), Seattle, WA, USA

CONTACT INFORMATION

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

Name:	Philip R Holland
Enterprise:	Holland Numerics Ltd
Address:	94 Green Drift
City, State ZIP:	Royston, Hertfordshire SG8 5BT, United Kingdom
Work Phone:	+44 7714 279085
E-mail:	phil@hollandnumerics.com
Web:	www.hollandnumerics.com