PhUSE 2015

Paper CS 08

Enhance and enrich your plots using Graph Template Language (GTL)

Mathias Lebreton, Novartis, Basel, Switzerland

The opinions expressed in this presentation and on the following slides are solely those of the presenter and not necessarily those of Novartis. Novartis does not guarantee the accuracy or reliability of the information provided herein.

ABSTRACT

The need to visually display complex data is increasingly important, particularly when conducting exploratory analyses. Graphs and plots are therefore getting more and more detailed.
Using SAS 9.4, simple versions of graphs or plots can be produced easily, however the purpose of this paper is to raise that plot to the next level.
We show how to enhance plots by controlling the axes, or adding colors and patterns, amongst other features. And how to enrich plots, increasing data interpretability, by (for example) incorporating multiple types of graphs..

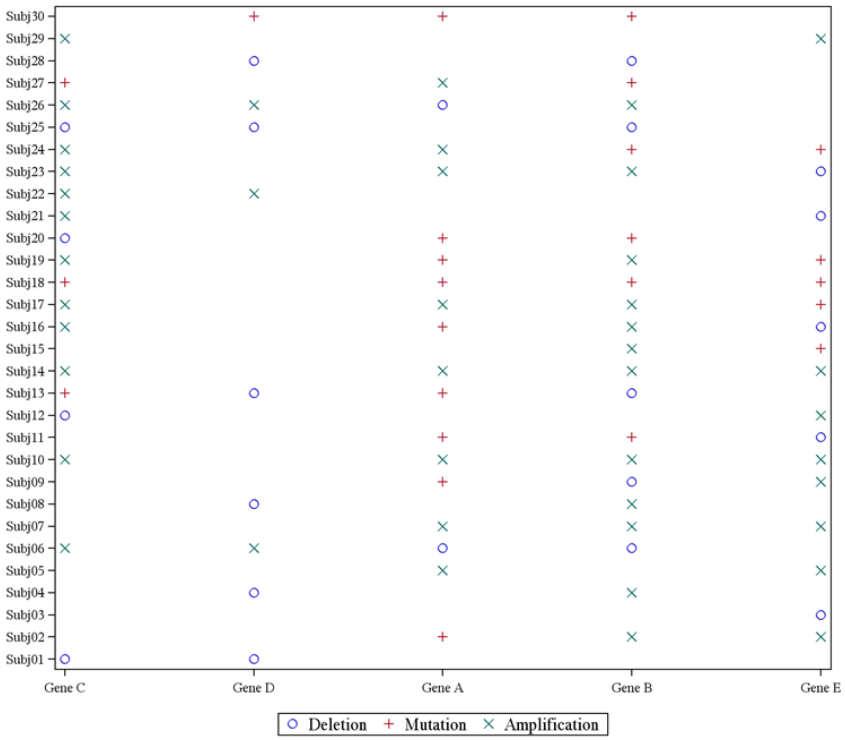
INTRODUCTION

The first part of this paper is focusing on graphs displaying one single plot. Using basic scatterplot and swimming plot as examples, we will reveal the lack of control resulting from the use of default SAS GTL options.
Why is the sorting of the axes and/or of the legend not the expected one? What is the best approach to control colors and symbols?
We will give some methods, from the input dataset to the graph template, in order to enhance those graphs.

The second part justifies the use of GTL by explaining how to combine multiple plots within the same graph. This implies some preprocesses such as the creation of a unique input dataset and the understanding of the layout lattice and overlay in the template.

Finally, we will demonstrate that in a controlled setting, we can start customizing graphs and make them more meaningful.

SINGLE GRAPH EXAMPLE (1/2): SCATTERPLOT



idpat	Gene	alterat
Subj01	Gene C	Deletion
Subj01	Gene D	Deletion
Subj02	Gene A	Mutation
Subj02	Gene B	Amplification
Subj02	Gene E	Amplification
Subj03	Gene E	Deletion
Subj04	Gene B	Amplification
Subj04	Gene D	Deletion

sample of input dataset

The above scatterplot describes the genetic alterations on five different genes by subject identifiers (subject ID). Both axes are expected to be sorted alphabetically.

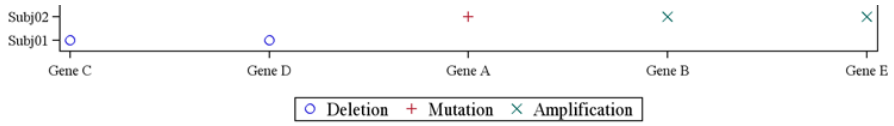
A sgplot procedure is usually more appropriate to produce this simple plot. However, as the aim of this paper is to program graphs with multiple plots on the same page, the Graphic Template Language (GTL) has been considered as the most appropriate approach.

Scatterplot GTL code:

```
proc template;
  define statgraph scatplot;
    begingraph /;
      layout overlay;
        scatterplot y=idpat x=Gene / group= alterat name='mutation';
        discretelegend  'mutation';
     endlayout;
    endgraph;
  end;
run;

proc sort data= scatdata;
  by idpat Gene;
run;


proc sgrender data=scatdata template=scatplot;
run;
```

DISPLAY ORDER VERSUS INPUT DATASET SORTING

As specified previously, we expect an alphabetical sorting on both axes which may not be the case by default. The axis order is data dependent and we can't always control the sorting on both axes with the sorting of the input dataset.



idpat	Gene	alterat
Subj01	Gene C	Deletion
Subj01	Gene D	Deletion
Subj02	Gene A	Mutation
Subj02	Gene B	Amplification
Subj02	Gene E	Amplification

As the input dataset is sorted by subject identifiers and by genes, the first value displayed on the X-axis is the first gene observed for the first patient (Subj01) which, in our case, is gene C and then Gene D.
By default, the type of axis is discrete for character variables. The proposal is to transform the X/Y-axis type to linear in order to define coordinates for any marker in the plot.

Example of X-axis transformation to linear:

idpat	Gene	alterat
Subj01	Gene C	Deletion
Subj01	Gene D	Deletion
Subj02	Gene A	Mutation
Subj02	Gene B	Amplification
Subj02	Gene E	Amplification

1

idpat	idx	Gene	alterat
Subj02	1	Gene A	Mutation
Subj02	2	Gene B	Amplification
Subj01	3	Gene C	Deletion
Subj01	4	Gene D	Deletion
Subj02	5	Gene E	Amplification

2

idpat	idx	alterat	Gene
Subj02	1	Mutation	Gene A
Subj02	2	Amplification	Gene B
Subj01	3	Deletion	Gene C
Subj01	4	Deletion	Gene D
Subj02	5	plification	Gene E

4

3

Format:	Gene_	
Start	End	Label
1	1	Gene A
2	2	Gene B
3	3	Gene C
4	4	Gene D
5	5	Gene E

1 – Sort by gene
2 – Assign a number in a new variable (idx) to each gene following the sorting
3 – Create a format (gene_) with start=idx and label=gene
4 – Assign the format to the numeric variable idx, gene can be dropped

The same logic is applied to the Y-axis (idpat). As a result, the final dataset is now containing two numerical variables idy and idx associated to specific formats, respectively idpat_ and gene_ as shown below:

Format:	Idpat_	
Start	End	Label
1	1	Subj01
2	2	Subj02
3	3	Subj03
4	4	Subj04
...	...	...
X	X	SubiX

idy	idx	alterat
1	3	Deletion
1	4	Deletion
2	1	Mutation
2	2	Amplification
2	5	Amplification
3	5	Deletion
4	2	Amplification
4	4	Deletion

Format:	Gene_	
Start	End	Label
1	1	Gene A
2	2	Gene B
3	3	Gene C
4	4	Gene D
5	5	Gene E

With linear axes, the marker (Gene C, subj01) is now defined by (x=3,y=1) and it will be plotted at the corresponding location regardless of the sorting of the input dataset.

Note: To ensure all tick values are being displayed, we need to defined X/Y statement in the template of linear type.
E.g. X-axis: Tickvaluesequence=(Start= *first value of idx* End= *last value of idx* Increment=*1 to display all values)*
Viewmin/Viewmax=*show corresponding area without displaying tick values*.

```
proc template;
  define statgraph scatplot;
    begingraph /;
      layout overlay/
       xaxisopts=( display=(line ticks tickvalues) type=linear GRIDDISPLAY=on
                   linearopts=(tickvaluesequence=(start=1 end=5 increment=1)
                  viewmin=0.5 viewmax=5.5))
       yaxisopts=( display=(line ticks tickvalues) type=linear GRIDDISPLAY=on
                   linearopts=(tickvaluesequence=(start=1 end=30 increment=1)
                  viewmin=1 viewmax=30));
        scatterplot y=idy x=idx / group= alterat name='mutation';
        discretelegend  'mutation';
     endlayout;
    endgraph;
  end;
run;

proc sgrender data= scatdata template=scatplot;
run;
```

LEGEND ORDER = INPUT DATASET SORTING

After the previous linear conversion, X/Y-axes are no more dependent of the sorting of the dataset.
As the sorting of the legend is also controlled by the input dataset, we can now sort the input dataset to get the expected order.

```
Proc template;
run;

proc sort data= scatdata;
  by alterat;
run;

proc sgrender data= scatdata template=scatplot;
run;
```

ATTRIBUTE MAP DATASET: CONTROL OF SYMBOLS AND COLORS

Why should we be controlling colors and symbols?

- Consistency: Keep the same symbol/color within a study/project for one category
- Context: Some colors are meaningful,(E.g. green: good effect; red: bad effect)
- Cosmetic: Allow the use of gradient palette,

One of the new features of SAS 9.4 is to associate a GTL template with an attribute map dataset. It allows us to develop this dataset externally in order to reuse it in other reports (E.g. Specify a single color for the same treatment group across an entire study).

Step 1 – Define an attribute map dataset:

An attribute map dataset contains a predefined list of variables (See the full set of variables allowed in appendix)

E.g. Alteration of gene (color + symbol)

Name: attrds_alterat

ID	VALUE	MARKERCOLOR	MARKERSYMBOL
alt	Amplification	CXE11520	trianglefilled
alt	Deletion	CX3A81E3	Squarefilled
alt	Mutation	CX0D9A81	circlefilled

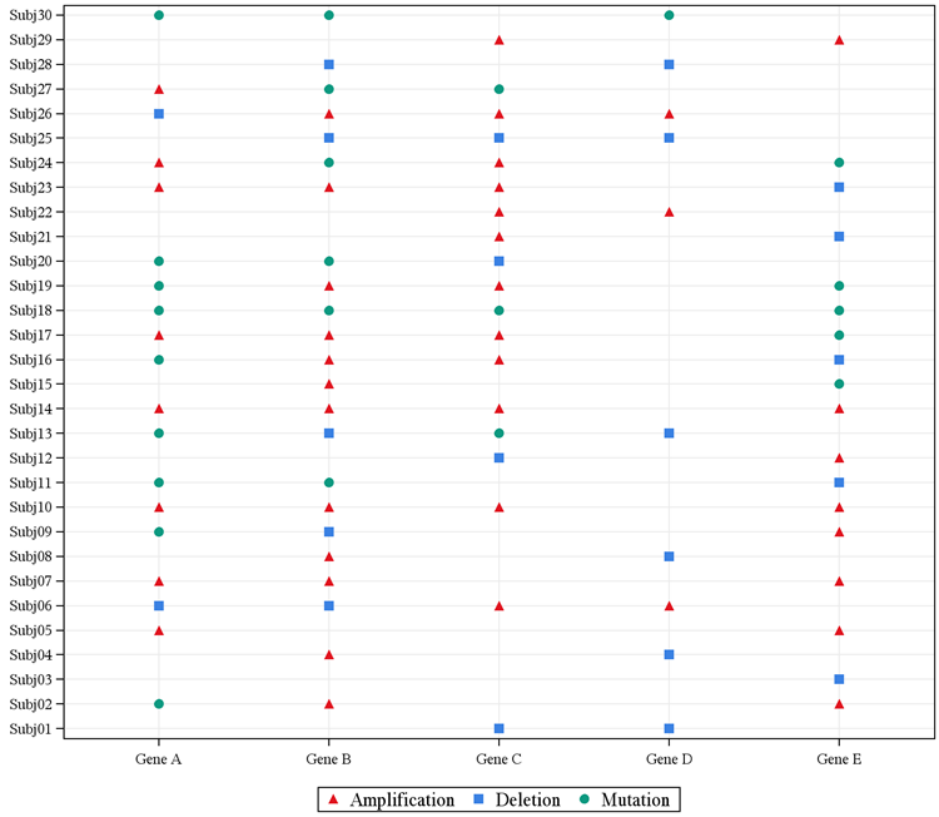
CXxxxxxx values are colors codes defined in hexadecimal. This format allows a larger panel of colors (useful for gradient color).
Amplification will be displayed as a red filled triangle, deletion as blue filled square and mutation as green filled circle

Step 2 – Link a variable to an attribute map dataset:
Add a discreteattrvar statement in the template:
Discreteattrvar var= variable in input dataset
attrmap=ID of associated attribute map
attrvar= Name of the new variable (replace original group variable)

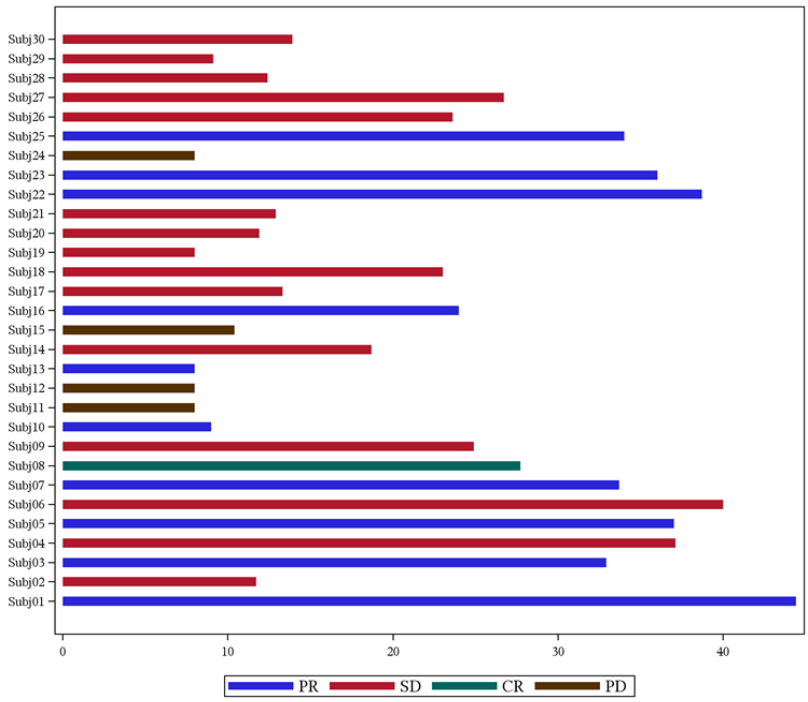
Step 3 – Link an attribute map dataset to a template:
Add the statement dattrmap=*attribute map dataset name* in the proc sgrender.

```
proc template;
  define statgraph scatplot;
    begingraph /;
      discreteattrvar var=alterat attrmap="alt" attrvar=alteratmd;
      layout overlay;
         scatterplot y=idy x=idx / group= alteratmd name='mutation';
      endlayout;
    endgraph;
  end;
run;

proc sort data= scatdata;
  by alterat;
run;

proc sgrender data=scatdata template=scatplot dattrmap= attrds_alterat;
run;
```

FINAL ENHANCED SCATTERPLOT



SINGLE GRAPH EXAMPLE (2/2): SWIMMING PLOT



The swimming plot above displays the subjects' duration of exposure. The color of the line is showing efficacy response based on RECIST[(1)] criteria. As for the scatterplot, we start with a very basic version using the SAS defaults.

We want to present it with the following features:

- Sorted by treatment group, longest duration of exposure, best response and subject ID
- Gradient color for CR/PR/SD and red for PD
- Sort the legend in a specific order: CR – PR – SD – PD

PLOT STATEMENT: HIGHLOWPLOT

Highlowplot is a very flexible procedure. It draws vertical or horizontal lines/bars by defining a low and high values as well as a value on X (vertical) or Y (horizontal). As it allows to draw multiple lines/bars on the same X or Y values, we are then able to draw non-continuous lines where the color changes.

Syntax for horizontal line:
highlowplot y=*y-axis value* low=*start of line value* high=*end of line value* / group= *group value* type=line;

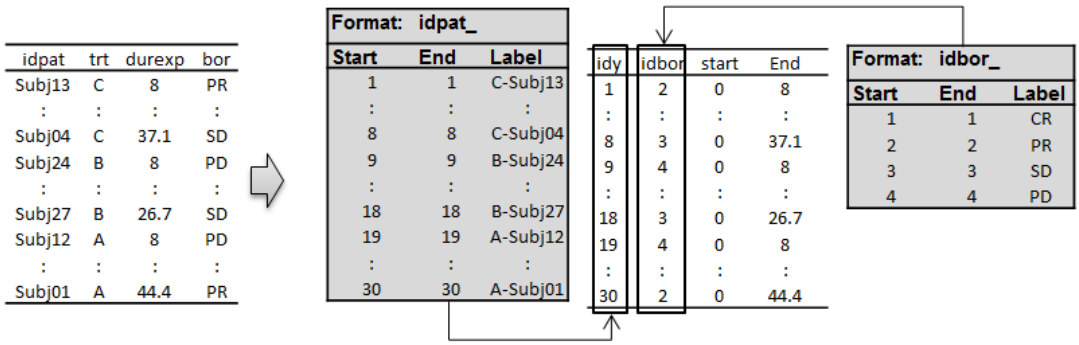
Note: To turn a line as a bar you can increase the thickness in the lineattrs option.

(1) Response Evaluation Criteria In Solid Tumors:
Complete response (CR) > Partial Response (PR) > Stable Disease (SD) > Progressive Disease (PD)

AXES IN LINEAR – LEGEND ORDER AS PER INPUT DATASET SORTING – ATTRIBUTE MAP

As we showed it previously, the character variables will be formatted to numeric in order to move the type of the axis from discrete to linear. In this graph we only format the Y-axis concerned as the X-axis is already numeric.

From original dataset to GTL input dataset:



The same logic is applied to idpat as in the previous example. We need to take into account the fact that both treatment group and subject ID need to be displayed. That is the reason why trt and idpat variables have been concatenated.

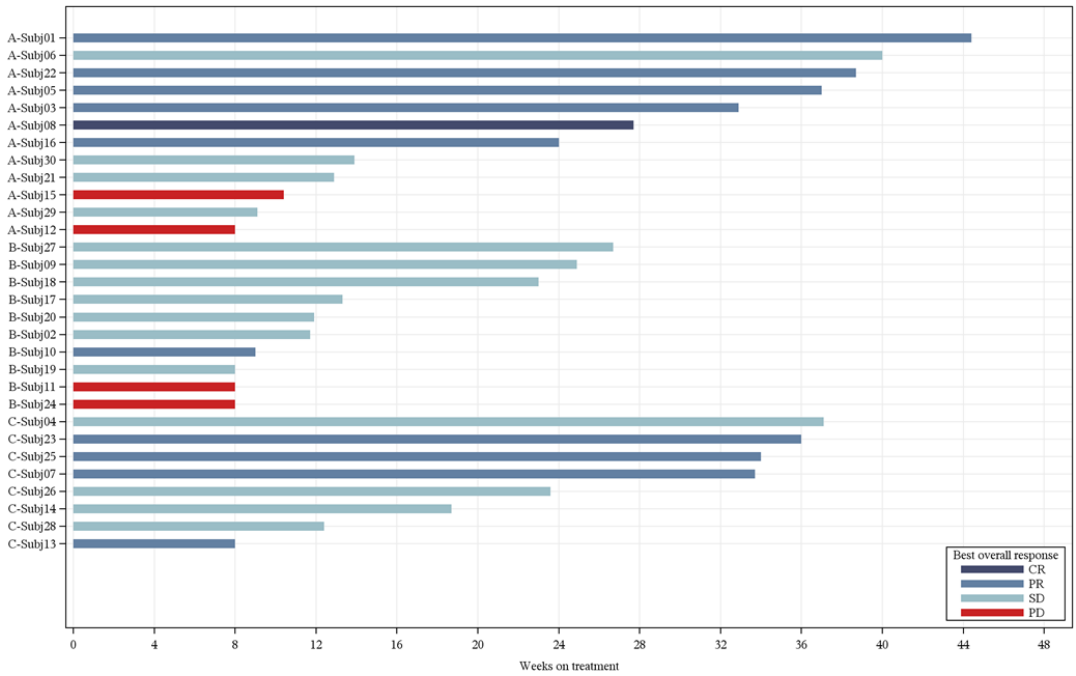
The variable bor is also formatted in numeric in order to have, in the legend, responses sorted by efficiency. The attribute map dataset contains attributes for responses and VALUE should contain decode values of idbor:

Name: attrds_idbor

ID	VALUE	LINECOLOR
bor	CR	CX434A6c
bor	PR	CX6380A2
bor	SD	CX9ABDC6
bor	PD	CXC92224

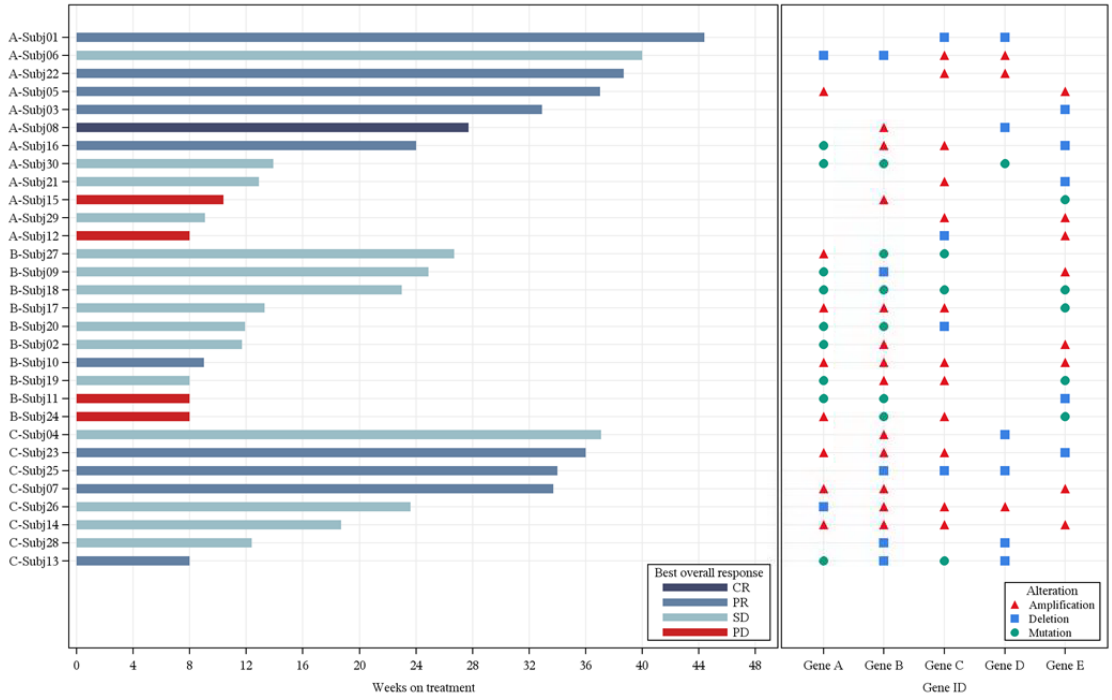
```
proc template;
  define statgraph swimingplot;
    begingraph /;
      discreteattrvar var=idbor attrmap="bor" attrvar=bormd ;
      layout overlay/
        xaxisopts=( display=(label line ticks tickvalues) type=linear griddisplay=on
                    linearopts=( tickvaluesequence=(start=0 end=48 increment=4)
                     viewmin=0 viewmax=48 label='weeks on treatment'))
        yaxisopts=( display=(line ticks tickvalues) type=linear griddisplay=on
                   linearopts=(tickvaluesequence=(start=1 end=30 increment=1)
                    viewmin=-2 viewmax=30));
         highlowplot y=idy low=start high=end /
                     group=bormd type=line lineattrs=(THICKNESS=8 pattern=1) name='swim';
        discreteLegend 'swim' /location=inside valign=bottom halign=right border=0
                               across=1 title='Best overall response';
      endlayout;
    endgraph;
  end;
run;
proc sort data= plotdata_3;
  by idbor;
run;
proc sgrender data=plotdata_3 template=swimingplot dattrmap=attrds_idbor;
run;
```

ENHANCED SWIMMING PLOT



COMBINED PLOTS

In the previous parts, we improved and enhanced the code in order to have more control on sorting and on symbols and colors. We will now see how to enrich a graph by combining it with other graphs.



The purpose of this graph is to present the patients' exposure as well as the alterations observed on the five gene of interest.

ONE INPUT DATASET, ONE ATTRIBUTE MAP DATASET AND ONE TEMPLATE

Step 1: Combine input datasets

Using GTL implies having a unique input dataset. To combine them a simple set of datasets is performed. However the key variable needed for the sorting of the common axis (here Y-axis) have to be retained.

	idpat	Trt	durexp	bor	Gene	alterat	start	end
Swimplot	**Subj01**	**A**	**44.4**	**PR**			0	44.4
	Subj02	**B**	**11.7**	**SD**			0	11.7
Scatterplot	**Subj01**	**A**	**44.4**	**PR**	Gene C	Deletion		
	Subj01	**A**	**44.4**	**PR**	Gene D	Deletion		
	Subj02	**B**	**11.7**	**SD**	Gene A	Mutation		

Step 2: Transform character variables to numeric variables

Below steps have to be done in the combined input dataset:

1) Bor to idbor: create a numeric variable for bor by sorting ascending level of response (1=CR, 2=PR, 3=SD, 4=PD)
2) Gene to idx: create a numeric variable such as for the previous scatterplot
3) idpat to idy: create a numeric variable such as for the previous Swimming plot

The sorting key 'by idbor alterat' will ensure the legend is correctly ordered.

Step 3: Combine attribute map datasets

A simple set of the attribute map datasets is done as their structures are the same.

ID	VALUE	MARKERCOLOR	MARKERSYMBOL	LINECOLOR
bor	CR			CX434A6c
bor	PR			CX6380A2
bor	SD			CX9ABDC6
bor	PD			CXC92224
alt	Amplification	CXE11520	trianglefilled	
alt	Deletion	CX3A81E3	Squarefilled	
alt	Mutation	CX0D9A81	circlefilled	

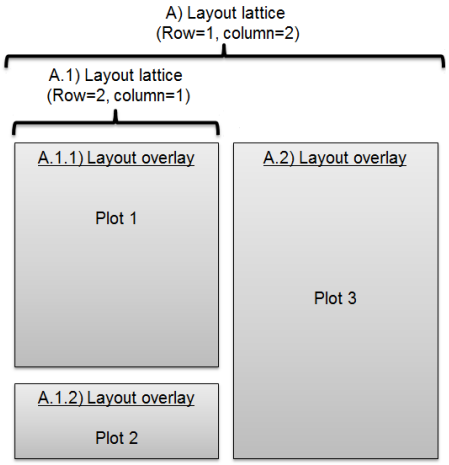
In the template a discreteattrvar statement has to be defined for every ID variables.

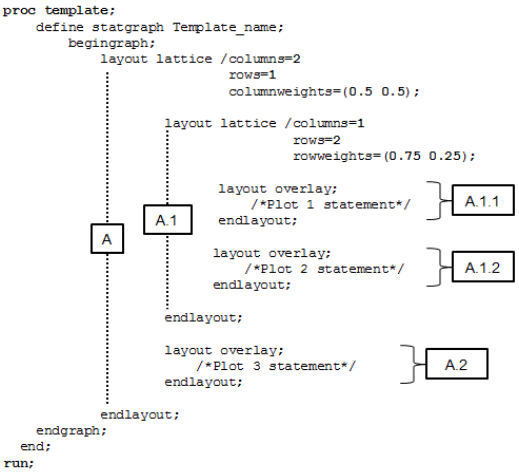
```
proc template;
  define statgraph combinedplot;
    begingraph /;
      discreteattrvar var=alterat attrmap="alt" attrvar=alteratmd;
      discreteattrvar var=bor attrmap="bor" attrvar=bormd;
    endgraph;
  end;
run;
```

Step 4: Combine graph templates

GTL allows the design of combined and nested plots by playing with the layout lattice/overlay:

- A layout overlay produces a single plot layout
- A layout lattice is defined by a number of column and rows
- A layout lattice can contain multiple nested layout lattice or overlay
- Dimension of each of the layout area is defined with <column/row>weights=(part1 part2 …), the sum of weights should be 1



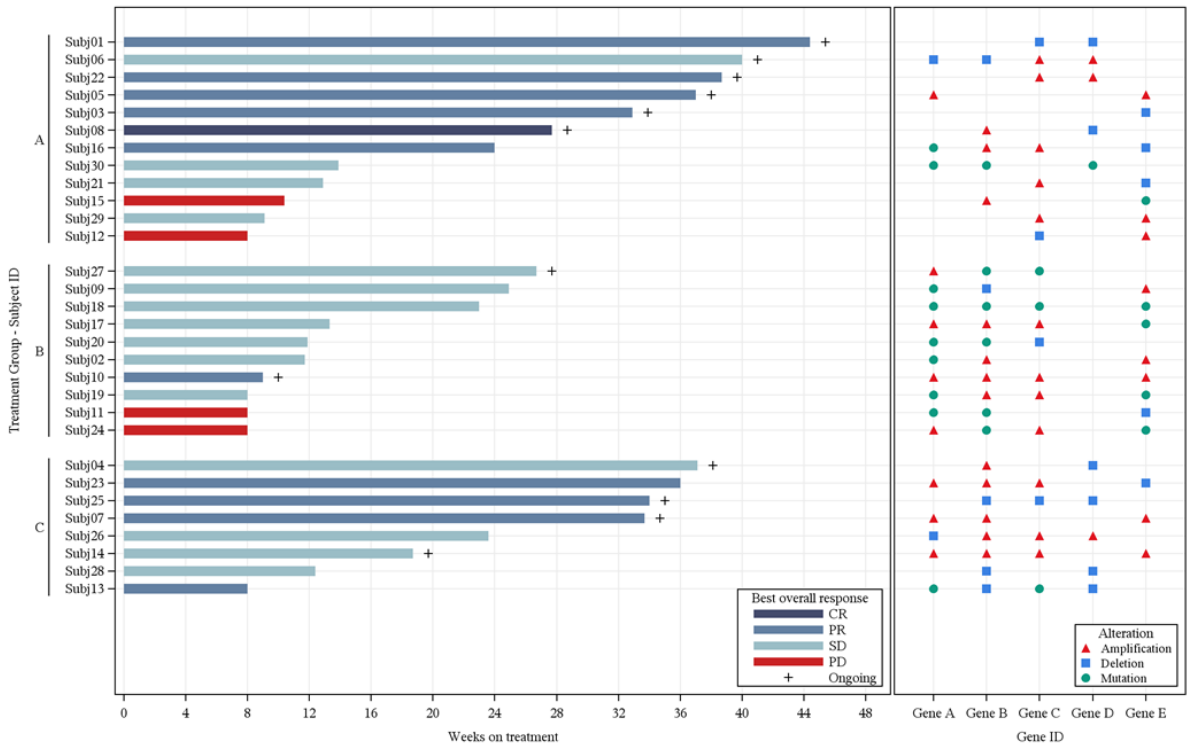


In our graph, we define a layout lattice with row=1 and column=2 to combine the swimming plot and the scatter plot:

```
proc template;
  define statgraph combinedplot;
    begingraph /;
      layout lattice / columns=2 rows=1 columnweights=(0.60 0.40) ;
        layout overlay
          /*Swimplot layout overlay*/ ;
        endlayout;
        layout overlay
          /*Scatterplot layout overlay*/ ;
        endlayout;
      endlayout;
    endgraph;
  end;
run;
```

CUSTOMIZATION

This part shows three examples of customization that can improve the overall layout and consequently the readability of the graphs. We will improve the overall layout (adding blank space between treatments, displaying treatment group with line, adding ongoing flag) as below:



Note: The SAS code to create this graph is available in the annexes section.

ADDING SPACES TO SEPARATE TREATMENT GROUP

One of the benefits to work with linear axes is that we can choose what and where we want to display.
During the formatting of axes, we can decide to have a gap of 2 instead of 1 between the last subject of one treatment group and the first subject of the following treatment group.

Format: idpat_		
Start	**End**	**Label**
1	1	C-Subj13
:	:	:
8	8	C-Subj04
9	**9**	
10	10	B-Subj24
:	:	:
19	19	B-Subj27
20	**20**	
21	21	A-Subj12
:	:	:
32	32	A-Subj01

idy	idbor	start	End
1	2	0	8
:	:	:	:
8	3	0	37.1
10	4	0	8
:	:	:	:
19	3	0	26.7
21	4	0	8
:	:	:	:
32	2	0	44.4

We also update the yaxisopts statement to not display the ticks 9 and 20.
We need to replace the tickvaluesequence to tickvaluelist:

```
yaxisopts=( display=(line ticks tickvalues) type=linear griddisplay=on
            linearopts=(tickvaluelist=(0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 21 22 23
                                       24 25 26 27 28 29 30 31 32) viewmin=-2 viewmax=30));
```

ADDING NEW PLOT STATEMENTS

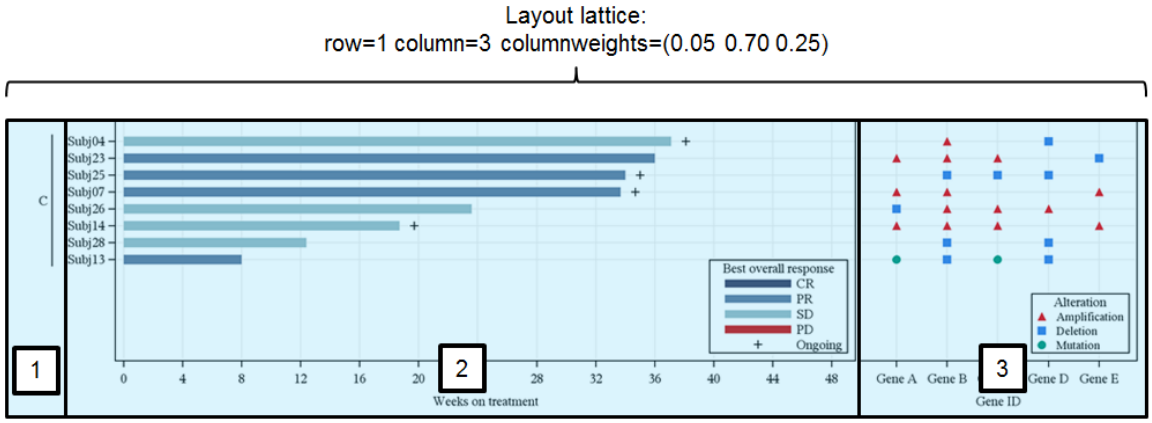
We can enrich a graph by adding new plot statement into one layout overlay.
E.g. we added the '+' symbol in the swimming plot to indicate that the subject is still on treatment at the time of the graph creation.

```
      layout overlay/
        highlowplot y=idy low=start high=end /
                    group=bormd type=line lineattrs=(THICKNESS=8 pattern=1) name='swim';
       scatterplot y=idy x=ongoing / markerattrs=(symbol=PLUS) name='ongo' legendlabel='Ongoing';
       discreteLegend 'swim' 'ongo' /location=inside valign=bottom halign=right border=1
                                 across=1 title='Best overall response';
      endlayout;
```

Note: Only one legend is defined as it contains the legend of both plot statement 'swim' and 'ongo'

DISPLAYING OF TREATMENT GROUP

To add the grouping, a new layout overlay will be inserted in the previous layout lattice:



Step 1: Create a dataset from the previous input dataset (with formatted character variables)

trt	Ref_x	Ref_y1	Ref_y2	Idx_trt	Idy
A	1	32.4	20.6	0	27
B	1	19.4	9.6	0	15
C	1	8.4	0.6	0	5

Ref_y1 and Ref_y2 are derived from the formatted variable idy.
E.g. Treatment C: The text marker 'C' is added at coordinates: (x=0,y=5) and the line at (x=1, low=0.6 high=8.4).

Step 2: Set this dataset to the previous input dataset

Step 3: Add a new layout overlay to the previous layout lattice:

```
layout lattice / columns=3 rows=1 columnweights=(0.05 0.70 0.25) ;
  layout overlay /border=FALSE walldisplay=none
               xaxisopts=(display=NONE type=linear
                          linearopts=(tickvaluelist=( 0 2 ) viewmin=0 viewmax=2))
               yaxisopts=(display= none type=linear);
    highlowplot low=ref_y2 high=ref_y1 X=ref_x;
    scatterplot y=idy x=idx_trt /markercharacter=trt;
  endlayout;

  /*layout overlay 2*/
  /*layout overlay 3*/
endlayout;
```

Note: To insert text into a graph, we used a scatterplot statement combined with the markercharacter option. The textplot statement can replace it if you have SAS9.4 release 2 or later.

CONCLUSION

The first approach when creating a graph is to understand the SAS default options and how to set them better. Moving axis type from discrete to linear allows a full control of the display by assigning X/Y coordinates to each marker/line. This is managed independently from the sorting of the input dataset.
The attribute map dataset functionality enables users to define standard display characteristics (such as symbols, colors, patterns, etc.).
With GTL, displaying multiple distinct plots into one graph is feasible with the combination of layout lattice and overlay
Further customization is also becoming feasible with only a few plot statements such as scatterplot or higlowplot.
Considering all these methods will give you the keys to raise your graphs to the next level.

REFERENCES

Support SAS 9.4 Graph template Language:
http://support.sas.com/documentation/cdl/en/grstatgraph/67882/HTML/default/viewer.htm#titlepage.htm

Hexadecimal color:
http://www.color-hex.com/

ACKNOWLEDGMENTS

With thanks to Sebastien Jolivet, Nassim Sleiman and Annabelle Foulon.

CONTACT INFORMATION

Author Name: Mathias Lebreton
Company: Novartis Basel
Email: mathias.lebreton@novartis.com

ANNEXES

Macro to formatting of axes

```
%macro fmtaxis(_inds=,        /*input dataset*/
               _var=,         /*input character variable*/
               _byvar=,       /*sorting on axis display*/
               _outds=,       /*output dataset*/
               _idvar=,       /*new numeric variable with format _fmtid*/
               _fmtid=,       /*name of the format associated to _idvar*/
               _blank=0,      /*_blank=1: add space between _blankvar*/
               _blankvar=,    /*variable of _byvar where space should be applied*/
               _viewmin=0.5,  /*space before 0 on axis*/
               _viewmax=0.5,  /*space after the last value on axis*/
               _type='N'      /*type of id variable*/);
       %if &_blank=1 %then %do;
               proc sort data= &_inds Out=_temp(where=(missing(&_var)=0) ) nodupkey;
                       by &_byvar;
               run;
               data _temp(keep= &_var);
                       set _temp end=eod;
                       output;
                       by &_byvar;
                       if last.&_blankvar and not eod then do;
                       &_var=' ';
                       output;
                       end;
               run;
       %end;
       %else %do;
               proc sort data= &_inds Out=_temp(keep= &_var where=(missing(&_var)=0) ) nodupkey;
                       by &_byvar;
               run;
       %end;
       data _temp1;
               set _temp;
               &_idvar=_N_;
       run;
       proc sort data= _temp1; by &_var; run;
       proc sort data= &_inds Out=_temp2; by &_var; run;
       data &_outds;
               merge _temp2 _temp1;
               by &_var;
       run;
       proc sort data= &_outds; by &_byvar; run;
       proc sql;
               create table fmt as
               select distinct &_idvar. as start, &_idvar. as end, &_var. as label,
                               &_type. as type, "&_fmtid." as fmtname
               from _temp1;
       quit;
       proc format cntlin=fmt;
       quit;
       data &_outds;
               set &_outds;
               format &_idvar &_fmtid..  ;
       run;
       %global tck_&_idvar;
       proc sql noprint;
               select distinct put(&_idvar,8.) into: tick1 separated by ' '
               from &_outds where &_var ne '' ;
               select "tickvaluelist=("||"&tick1"||") viewmin= "||
               compress(put(min(&_idvar)-&_viewmin,best.))||"viewmax="||
               compress(put(max(&_idvar)+&_viewmax,best.)) into: tck_&_idvar from &_outds;
       quit ;
       run;
%mend fmtaxis;
```

Data derivation

```
/*Input dataset*/
data plotdata_1;
        set swimdata scatterdata;
        if bor ='CR' then bor_ord=1;
        else if bor ='PR' then  bor_ord=2;
        else if bor ='SD' then  bor_ord=3;
        else if bor ='PD' then  bor_ord=4;
run;
/*formatting of axes*/
%fmtaxis(      _inds=plotdata_1,
               _var=bor,
               _byvar= bor_ord,
               _outds=plotdata_2,
               _idvar=idbor,
               _fmtid=bor_,
               _type='N');

%fmtaxis(      _inds=plotdata_2,
               _var=idpat,
               _byvar= descending trt durexp descending idbor descending idpat,
               _outds=plotdata_3,
               _idvar=idy,
               _fmtid=idpat_,
               _blank=1,
               _viewmin=4,
               _viewmax=0);

%fmtaxis(      _inds=plotdata_3,
               _var=gene,
               _byvar= gene,
               _outds=plotdata_4,
               _idvar=idx_gene,
               _fmtid=gene_,
               _type='N');
/*data derivation for custom treatment display*/
proc sql ;
        create table freq as
        select trt,count(distinct idy) as counta from plotdata_4 group by  trt;
        create table refline1 as
        select a.*,b.counta from plotdata_4 as a
        left join
        (select b.counta,b.trt from freq as b)  on a.trt=b.trt;
quit;
run;
proc sort data=refline1; by trt idy; run;
data refline2 (keep=trt  ref_y1 ref_y2 ref_x idy counta trt trttxt idx_trt);
        set refline1(where=(trt ne ''));
        by trt idy ;
        if last.trt ;
        ref_y1= idy+0.4; ref_y2=idy-counta+0.6;ref_x=1;
        idx_trt=0;
        trttxt=strip(trt);
        idy=idy-(counta-1)/2 ;
run;
data plotdata_5;
        set plotdata_4 refline2;
run;
/*Macro variable for tick values display option*/
proc sql noprint;
        select "tickvaluesequence=(start=0 end="||compress(put(ceil(max(end)/4)*4,best.))||
        " increment=4) viewmin=0 viewmax="||
        compress(put( 1 + ceil(max(end)/4)*4,best. ) )
        into: xtick_hp from plotdata_5;
quit ;
run;
```

Attribute map dataset definition:

```
data attrds_idbor;
        infile datalines delimiter=',' missover;
        length ID $8 VALUE $40 linecolor $12;
        input ID VALUE linecolor;
        datalines;
bor,CR,CX434a6c
bor,PR,CX6380a2
bor,SD,CX9abdc6
bor,PD,CXC92224
;
run;

data attrds_alterat;
        infile datalines delimiter=',' missover;
        length ID $8 VALUE $40 markercolor $12  markersymbol $20;
        input ID VALUE markercolor markersymbol;
        datalines;
alt,Amplification,CXe11520,trianglefilled
alt,Deletion,CX3a81e3,squarefilled
alt,Mutation,CX0d9a81,circlefilled
;
run;

data attrds;
        set attrds_idbor attrds_alterat;
run;
```

List of variables in attribute map dataset

Column Name	Type	Description
FillColor	String	Specifies the color for the filled areas for this classification value. When FillStyleElement is also specified, this column overrides the specified style element's color attribute.
FillStyleElement	String	Specifies the name of a style element such as GraphData3 that is to provide the fill attributes.
FillTransparency	Numeric	Specifies the transparency of the fill color for this classification value. The value ranges from 0 (opaque) to 1 (completely transparent).
LineColor	String	Specifies the line color for this classification value. When LineStyleElement is also specified, this column overrides the specified style element's contrast color attribute.
LinePattern	String or Numeric	Specifies the line pattern for this classification value. You can specify the pattern name or number. See Available Line Patterns.
LineStyleElement	String	Specifies the name of a style element such as GraphData3 that is to provide the line attributes.
LineThickness	Numeric	Specifies the line thickness, in pixels, for this classification value. The values must be integers.
MarkerColor	String	Specifies the color of the marker symbol for this classification value. When MarkerStyleElement is also specified, this column overrides the specified style element's contrast color attribute.
MarkerSize	Numeric	Specifies the size of the marker, in pixels, for this classification value. The values must be integers.
MarkerSyleElement	String	Specifies the name of a style element such as GraphData3 that is to provide the marker attributes.
MarkerSymbol	String	Specifies the marker symbol to use for this classification value. See SYMBOL= in Marker Options. When MarkerStyleElement is also specified, this column overrides the specified style element's marker symbol attribute.
MarkerTransparency	Numeric	Specifies the transparency of the marker for this classification value. The value ranges from 0 (opaque) to 1 (completely transparent).
TextColor	String	Specifies the text color for this classification value.
TextFont	String	Specifies the text font by name for this classification value.
TextSize	Numeric	Specifies the text font size in points for this classification value. The values must be integers.
TextStyle	String	Specifies the text style for this classification value. Valid values are NORMAL and ITALIC.
TextStyleElement	String	Specifies the name of a style element such as GraphLabelText that is to provide the text attributes.
TextWeight	String	Specifies the text weight for this classification value. Valid values are NORMAL and BOLD.

Template procedure

```
proc template;
        define statgraph combinedplot;
        begingraph /;

        discreteattrvar var=idbor attrmap="bor" attrvar=bormd ;
        discreteattrvar var=alterat attrvar=alteratmd attrmap="alt";
        layout lattice / columns=3 rows=1 columnweights=(0.05 0.70 0.25) ;

                /*Layout overlay 1 - Custom display of treatment*/
                layout overlay /border=false walldisplay=none
                        xaxisopts=(display=none type=linear tickvalueattrs=(size=3)
                                  linearopts=(tickvaluelist=( 0 1 ) viewmin=0 viewmax=1))
                        yaxisopts=(display= (label) tickvalueattrs=(size=8) type=linear
                                   labelattrs=(size=8) label='Treatment Group - Subject ID'
                                   linearopts=(&tck_idy.));
                        scatterplot y=idy x=idx_trt
                                /markercharacter=trttxt markercharacterattrs=(size=8);
                        hightlowplot low=ref_y2 high=ref_y1 X=ref_x
                                /lineattrs=(pattern=solid thickness=1 );
                endlayout;

                /*Layout overlay 2 - Swimming plot*/
                layout overlay/
                        xaxisopts=(display=( label line ticks tickvalues ) griddisplay=on
                                   tickvalueattrs=(size=8) type=linear labelattrs=(size=8)
                                   label='Weeks on treatment' linearopts=(&xtick_hp.))

                        yaxisopts=(display= ( line tickvalues ticks) griddisplay=on
                                   tickvalueattrs =(size=8) type=linear linearopts=(&tck_idy.));
                        hightlowplot y=idy low=start high=end
                                /group=bormd type=line lineattrs=(thickness=8 pattern=1)
                                includemissinggroup=false name='swim';
                        scatterplot y=idy x=ongoing
                                /markerattrs=(symbol=plus) name='ongo' legendlabel='Ongoing';
                        discreteLegend 'swim' 'ongo'
                                /location=inside backgroundcolor=white valign=bottom opaque=true
                                halign=right border=1 across=1 tittleattrs=(Size=8)
                                valueattrs=(Size=8) title='  Best overall response  ';
                endlayout;

                /*Layout overlay 3 - Scatterplot*/
                layout overlay /
                        xaxisopts=(display=( line ticks label tickvalues ) griddisplay=on
                                   tickvalueattrs=(size=8) labelattrs=(size=8) label='Gene ID'
                                   type=linear linearopts=(&tck_idx_gene))
                        yaxisopts=(display=(line ) griddisplay=on tickvalueattrs=(size=8)
                                   type=linear linearopts=(&tck_idy.));
                        scatterplot y=idy x=idx_gene
                                /group=alteratmd markerattrs=(size=10) name='scat'
                                includemissinggroup=false;

                        discreteLegend 'scat'
                                /autoitemsize=true backgroundcolor =white location=inside
                                opaque=true valign=bottom halign=right across=1 border=1
                                tittleattrs=(Size=8) valueattrs=(Size=7)  title='Alteration';
                endlayout;
        endlayout;

        endgraph;
        end;
run;
```

Sorting by legend variables

```
proc sort data= plotdata_5; by idbor alterat; run;
```

Sgrender procedure

```
proc sgrender data=plotdata_5 template= combinedplot dattrmap=attrds;
run;
```