PharmaSUG 2017 - Paper DV14

When one is not enough or multi-celled plots: comparison of different approaches

Vladlen Ivanushkin, inVentiv Health Clinical, Berlin, Germany

ABSTRACT

Data visualization is a very powerful way to present data and therefore has been used for a long time in clinical trials. However, sometimes it happens that the number of plots required for a project is huge, or maybe you just don't want to scroll pages in a document, or open many files to compare a couple of plots. In such situations, it's a common practice to put several plots on one page to spare some space, to have better comparison possibilities, or just for a better overview.

Some years ago a programmer would need to write tons of code to create some non-standard multi-celled graphics. However, the progress does not stand still and many tasks, which used to require much time and effort in the past, can now be replaced with just few lines of code. Having so many opportunities to create as sophisticated plots as one can only imagine which way should the programmer choose, and which one is the most efficient?

In this paper I would like to describe different approaches of creating multi-celled plots and specify pros and cons for each of them.

INTRODUCTION

Before SAS® 9.2 the opportunities for building multi-celled plots were quite limited however after the release of the SG-procedures and Graph Template Language (GTL) things have changed. Nowadays programmers have a broad variety of approaches at their disposal for building up any kind of plot and multi-celled plots in particular. In order to help programmers select the best approach for their needs we look in detail into the most common ways of creating multi-celled plots and compare them to identify their respective advantages and disadvantages.

SGREPLAY METHOD

Let us start with the old-fashioned GREPLAY procedure. Of course, it is possible to use the G-procedures, but let's try to combine GREPLAY with some SG-procedures.

Let's use the SASHELP.CARS data set and create scatter plots with regression lines for weight against length using car type as the classification variable and origin as the grouping variable.

This approach requires the following steps:

1. Create required plots.
2. Create a plot with legend, titles, footnotes, etc.
3. If the plots are created with the SG-procedures, then GSEG catalogue entries must be created for each plot.
4. Create template for replaying and run GREPLAY.

Let's take a closer look at each step.

The first step can easily be performed with, for example, a simple SGPLOT procedure call. Also, to display more features, let's add some inset information. For this type of figure the regression equations could be handy to show. This means that the regression parameters should be derived first and merged to the SASHELP.CARS data. These can be shown afterwards using the TEXT statement of the SGPLOT procedure (note that the TEXT statement for SGPLOT is only available starting from SAS 9.4). The data set may look like the data shown in the DATA 1 screenshot.

WORK.CARS_WITH_REG_EQ

	Type	Origin	Length Length (IN)	Weight Weight (LBS)	reg_eq
1	SUV	Asia	189	4451	Asia length=131.899+0.013*weight~Europe length=148.722+0.008*weight~USA length=128.636+0.014*weight
2	SUV	Asia	188	4387	Asia length=131.899+0.013*weight~Europe length=148.722+0.008*weight~USA length=128.636+0.014*weight
3	SUV	Asia	179	3258	Asia length=131.899+0.013*weight~Europe length=148.722+0.008*weight~USA length=128.636+0.014*weight

DATA 1

```
proc sgplot data=cars_with_reg_eq noautolegend;
    by type;
    text x=x1text y=y1text text=reg_eq / contributeoffsets=none outline
    position=bottomleft textattrs=(size=5) splitchar='~' splitpolicy=split;
    reg  x=weight y=length / group=origin ;
    xaxis  min=2000 max=7000 offsetmax=.08;
    yaxis  min=140  max=220  offsetmax=.1;
run;
```

If a common legend is required for all the plots across the page, then the NOAUTOLEGEND option can be used. As the result we get four separate plots for each value of the car type variable along with the regression equations as shown in Figure 1 - Figure 4.

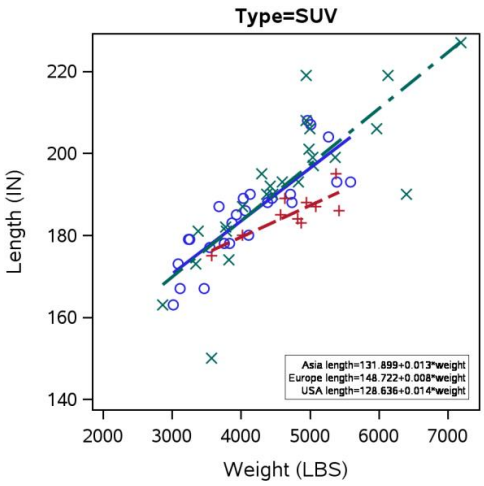

Figure 1. SUV

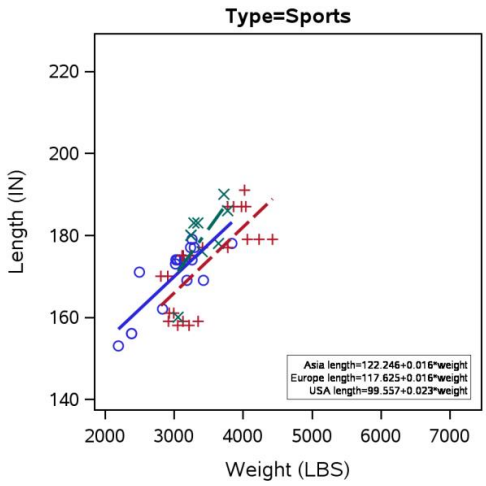

Figure 3. SPORTS

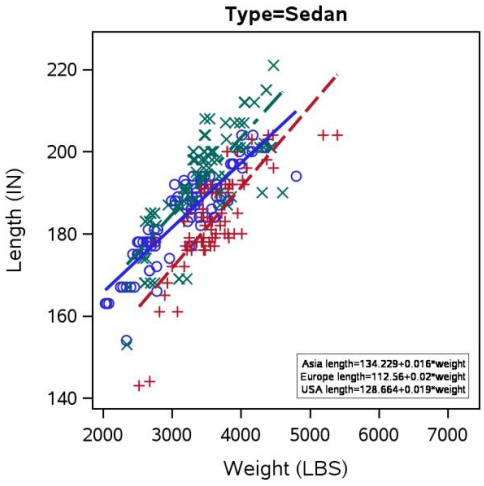

Figure 2. SEDAN

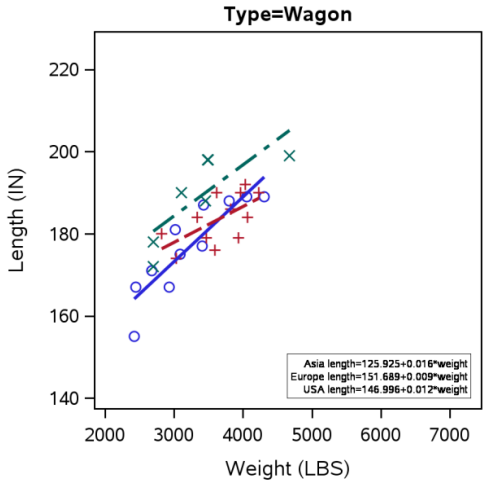

Figure 4. WAGON

Now an additional plot that will contain the legend and the title is needed. To create it, again the SGPLOT procedure can be used. If we specify the axes ranges which the actual data set does not contain, we will get an empty plot - Figure 5.

```
title1 "Figure: Scatter plot of
weight versus length of cars with
different origin and type using
GREPLAY";
proc sgplot data=
sashelp.cars(where=(type
^in ('Hybrid' 'Truck')))
noborder;
    series  x=weight y=length /
          group=origin markers;
    xaxis  min=-100 max=-10
    display=none;
    yaxis  min=-100 max=-10
    display=none;
run;
```

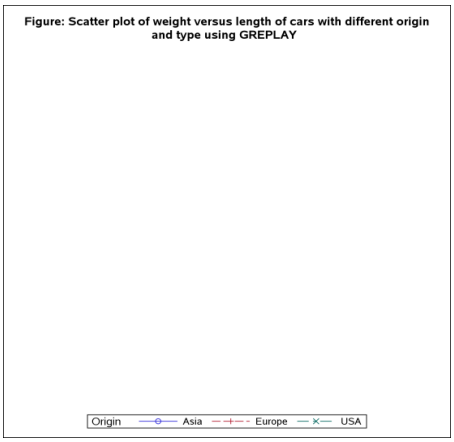

Figure 5. EMPTY PLOT

Using the SERIES statement here gives us a merged legend which represents both scatters and lines.

Before proceeding to the fourth step, it is important to keep in mind that the GREPLAY procedure can only replay the GSEG catalogue entries, however the SG-procedures do not create such entries. Taking this into account, the next step should be to create the catalogue entries for already existing PNG files. This is still possible with the GSLIDE procedure and the IBACK option:

```
goptions device=png300 nodisplay agestyle=fit iback="<path to .png file>";
proc gslide;
run;
quit;
```

And now, when everything is ready for the replaying, we can proceed to the final step

```
filename fig "<desired path where .emf file is to be stored>";
goptions reset=all device=emf gsfname=fig  hsize=6in vsize=6in;
proc greplay nofs igout=work.gseg tc=tempcat;
   tdef spec
   1/ llx=0       lly=0
      ulx=0       uly=100
      urx=100     ury=100
      lrx=100     lry=0
   2/ llx= 6.5    lly=48.5
      ulx= 6.5    uly=91.5
      urx=49.5    ury=91.5
      lrx=49.5    lry=48.5
   3/ llx=49.5    lly=48.5
      ulx=49.5    uly=91.5
      urx=92.5    ury=91.5
      lrx=92.5    lry=48.5
   4/ llx= 6.5    lly= 5.5
      ulx= 6.5    uly=48.5
      urx=49.5    ury=48.5
      lrx=49.5    lry= 5.5
   5/ llx=49.5    lly= 5.5
      ulx=49.5    uly=48.5
      urx=92.5    ury=48.5
      lrx=92.5    lry= 5.5;
```

```
    template spec;
    treplay 1:gslide4 2:gslide 3:gslide1 4:gslide2 5:gslide3;
run;
quit;
```

The result is shown on the Figure 6. Thus, it is possible to combine any GSEG catalogue entries using the GREPLAY procedure.

Note that the first slide used is the one with the title and the legend. It is also worth mentioning that GREPLAY does not work with ODS GRAPHICS.

Obviously, this approach takes a while to program and many factors must be considered. Also, GREPLAY is quite an outdated way to create multi-celled plots and it does not have all the possibilities that the GTL has. Nevertheless, it is a valid method which can be used under certain circumstances.

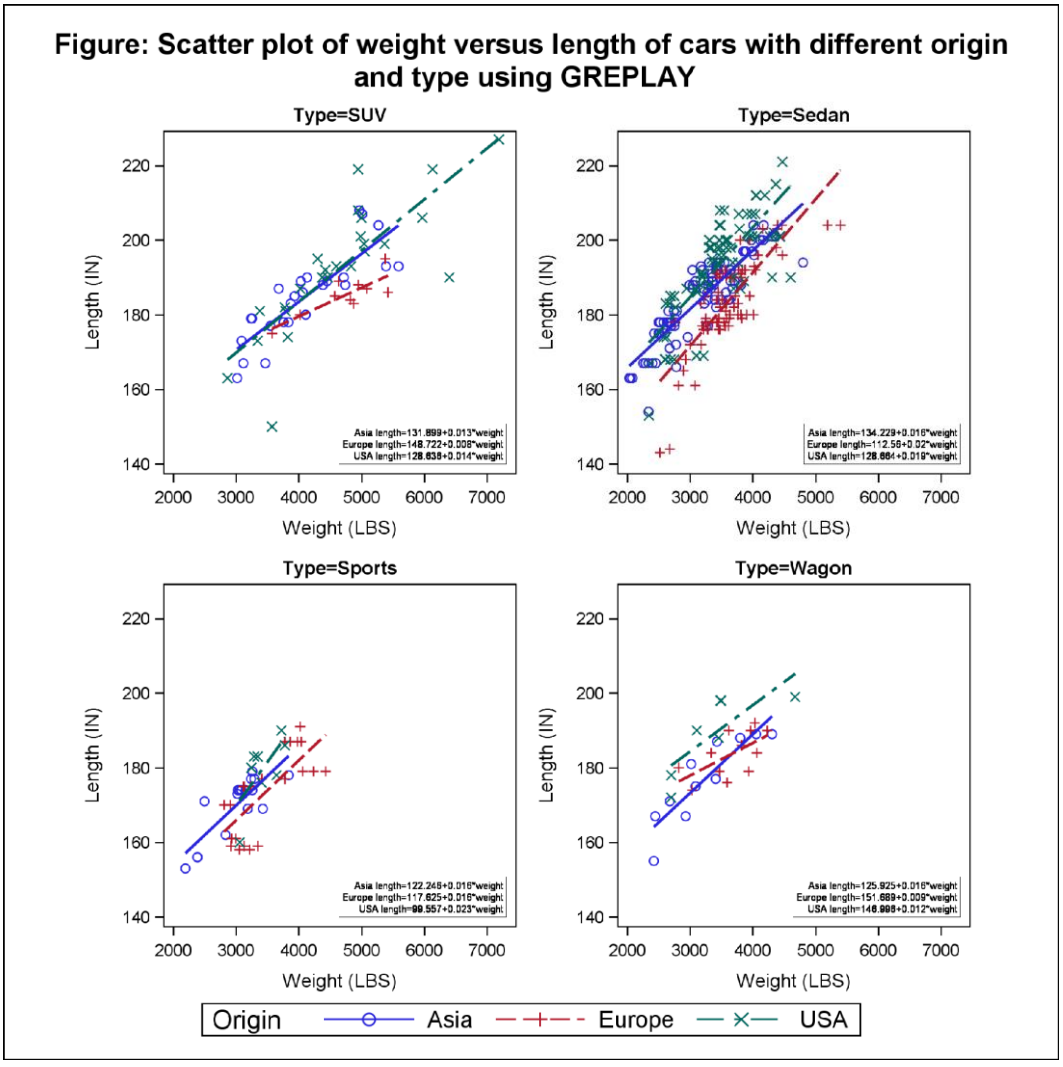

Figure 6. GREPLAY

SGPANEL METHOD

Another way, the most efficient and quick one, is to use the SGPANEL procedure, the very purpose of which is to produce multi-celled plots.

Indeed, applying the following very few lines of code we already get what we actually need - Figure 7.

```
title1 "Figure: Scatter plot of weight versus length of cars with different
origin and type using SGPANEL";
proc sgpanel data=cars_reg_for_plot;
    panelby type / novarname ;
    inset reg_eq_asia reg_eq_europe reg_eq_usa /
          border position=bottomright textattrs=(size=5);
    reg x=weight y=length / name="plot1" group=origin;
    scatter x=weight y=length / name="plot2" group=origin;
    keylegend "plot1" "plot2" / across=3 down=1;
run;
```

Figure: Scatter plot of weight versus length of cars with different origin and type using SGPANEL

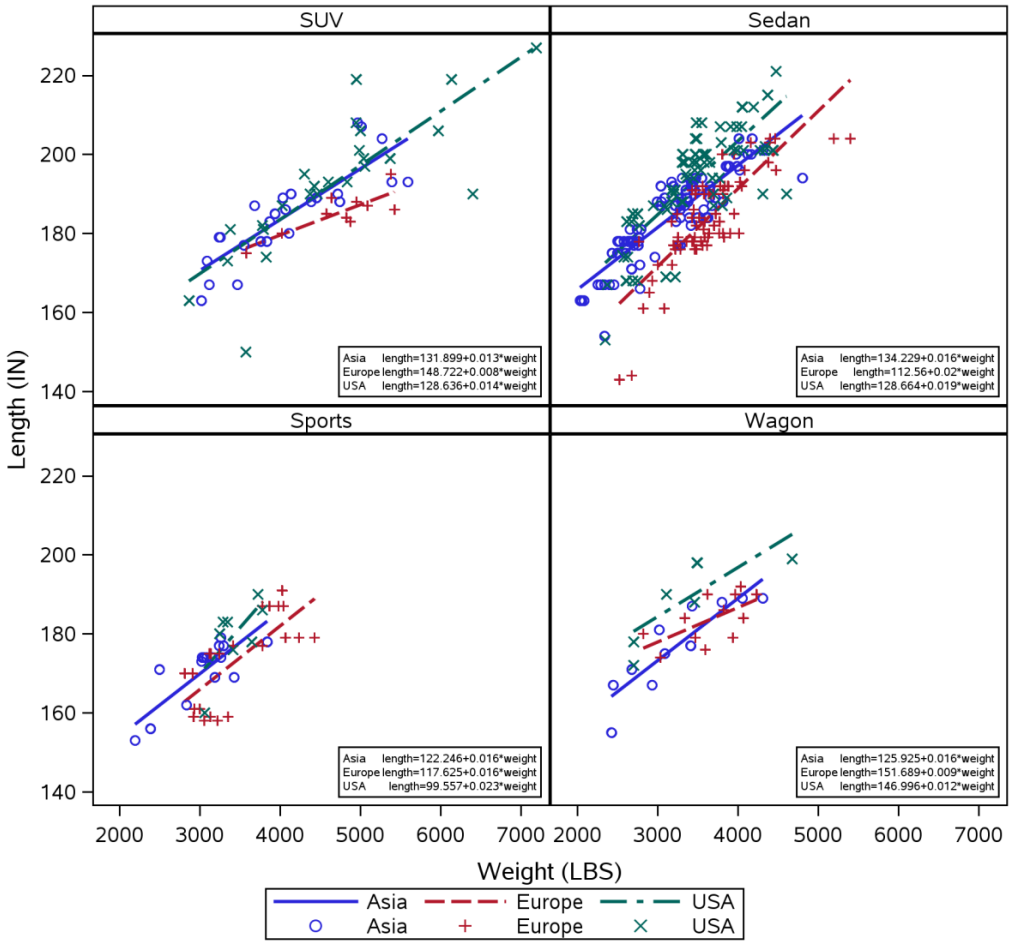

Figure 7. SGPANEL

Again, as in the previous example the inset information has been added using the INSET statement and DATA 2 (note that the INSET statement for SGPANEL is only available starting from SAS 9.4).

WORK.CARS_WITH_REG_EQ

	Type	Origin	Length Length (IN)	Weight Weight (LBS)	reg_eq_Asia Asia	reg_eq_Europe Europe	reg_eq_USA USA
1	SUV	Asia	189	4451	length=131.899+0.013*weight	length=148.722+0.008*weight	length=128.636+0.014*weight
2	SUV	Asia	188	4387	length=131.899+0.013*weight	length=148.722+0.008*weight	length=128.636+0.014*weight
3	SUV	Asia	179	3258	length=131.899+0.013*weight	length=148.722+0.008*weight	length=128.636+0.014*weight

DATA 2

As the REG statement does not draw marker symbols in the legend, the SCATTER statement and the KEYLEGEND have been used. However as there is no merge option and no MERGEDLEGEND statement, the marker symbols and lines are presented separately. Using the PANELBY statement, it is possible to influence the number of cells and their layout. By default, the PANEL layout is used which supports any number of classification variables. This is the best way to create multi-celled plots if the procedure's functionality is enough and only the same type of plot across one page is required.

GTL DATAPANEL LAYOUT

SGPANEL is quite a comprehensive and powerful procedure though it lacks some features. If this is the case, you can turn to GTL. Similar to the SGPANEL, which uses the PANEL layout, GTL also offers the DATAPANEL layout. Of course, this approach is more complex as it requires defining your own template using GTL, however at the same time it gives you more freedom and flexibility.

So let's try to reproduce the same plot but now using GTL and the DATAPANEL layout.

One of the things a programmer should keep in mind when using the DATAPANEL layout is that to construct cells within the layout, the nested PROTOTYPE layout should be used, and this has some restrictions - it supports neither nested layouts nor computed plots. Regression plot is a computed plot, hence it is not possible to use the REGRESSIONPLOT statement within the PROTOTYPE layout. Fortunately, it is still possible to draw regression lines having calculated the parameters of regression beforehand and afterwards applying the LINEPARM plot statement.

So the first step here would be to derive parameters for the regression lines. Also if we want to include some additional inset information, like the regression equations, we will need to create the corresponding variables. If we use the match-merge principle to add the information to the data set then the appropriate DATASCHEME option must be used, i.e. DATASCHEME=MATCHED (the option DATASCHEME is only available for the DATAPANEL layout starting from SAS 9.4).

WORK.CARS_WITH_REG_EQ

	Origin	Type	Length Length (IN)	Weight Weight (LBS)	Intercept Intercept	slope Weight (LBS)	reg_eq_Asia Asia	reg_eq_Europe Europe	reg_eq_USA USA
1	Asia	SUV	189	4451	131.89895791	0.0128871778	length=131.899+0.013*weight	length=148.722+0.008*weight	length=128.636+0.014*weight
2	Asia	SUV	188	4387	131.89895791	0.0128871778	length=131.899+0.013*weight	length=148.722+0.008*weight	length=128.636+0.014*weight
3	Asia	SUV	179	3258	131.89895791	0.0128871778	length=131.899+0.013*weight	length=148.722+0.008*weight	length=128.636+0.014*weight

DATA 3

Now we can proceed to the template itself.

```
proc template;
    define statgraph regr_scatter;
    dynamic x y classvar group;
    begingraph;
        entrytitle "Figure: Scatter plot of weight versus length of cars
                with different origin and type using GTL DATAPANEL layout";
        layout datapanel classvars=(classvar) /
          columns=2 rows=2 headerLabelDisplay=value inset=(inset1 inset2
          inset3) insetopts=(datascheme=matched border=true valign=bottom
          halign=right textattrs=(size=5));
            layout prototype;
                lineparm x=0 y=intercept slope=slope /
                      clip=true group=group name="plot1";
                scatterplot x=x y=y /primary=true group=group name="plot2";
            endlayout;
            sidebar / align=bottom;
                mergedlegend  "plot1" "plot2";
/*discretelegend "plot1" "plot2" / merge=yes; SAS 9.2 or earlier release*/
            endsidebar;
        endlayout;
```

```
      endgraph;
      end;
run;
```

As we need a four-celled layout, we can set the number of rows and columns to two. Also, it is possible to use the MERGEDLEGEND (starting from SAS 9.3) statement to consolidate the legend entries.

To reproduce the defined template, the SGRENDER procedure is used:

```
proc sgrender template=regr_scatter data=cars_with_reg_eq;
   dynamic classvar="type" x="weight" y="length" group="origin"
           inset1="reg_eq_asia" inset2="reg_eq_europe" inset3="reg_eq_usa";
run;
```

Figure: Scatter plot of weight versus length of cars with different origin and type using GTL DATAPANEL layout

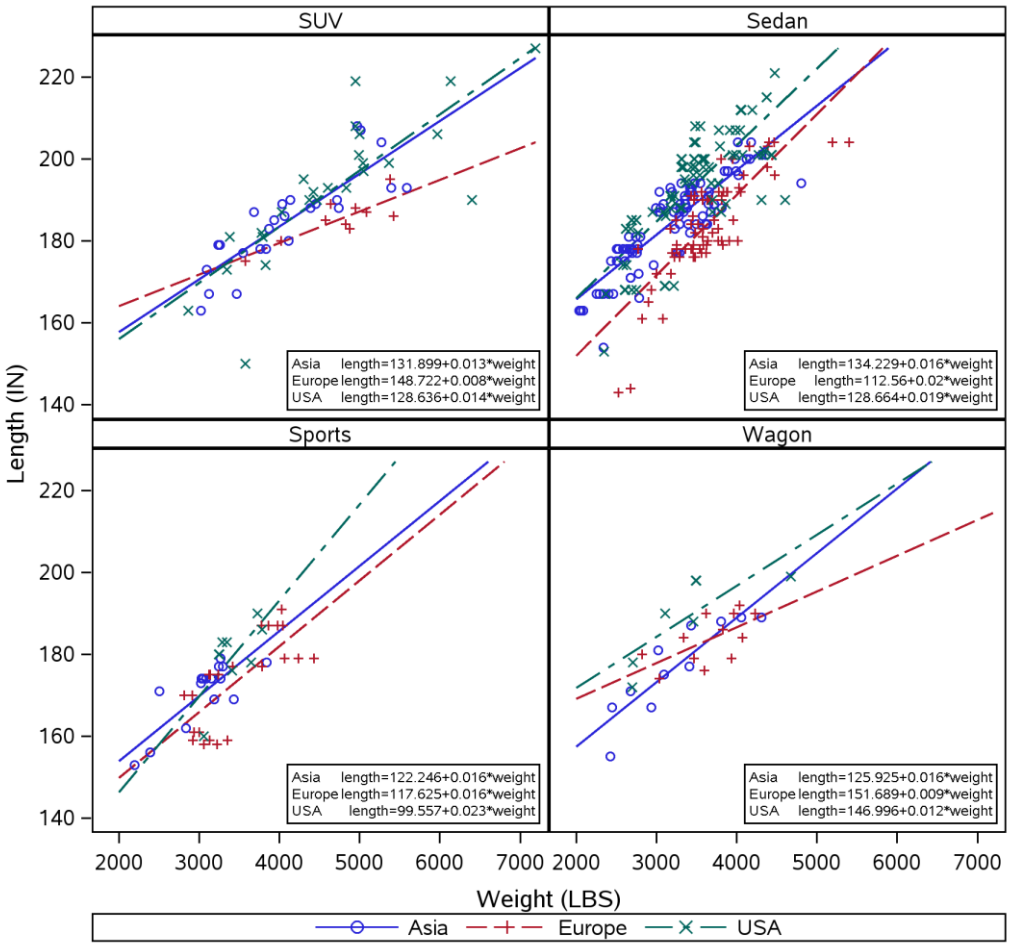

Figure 8. DATAPANEL LAYOUT

And we get a similar result in Figure 8 with some modifications though. Compared to the result of the SGPANEL procedure, here we have the merged legend and, also, the DATAPANEL layout does not restrict the regression lines by the actual data for each particular group.

GTL LATTICE LAYOUT

In case different types of plots are required, the LATTICE layout may come in handy. However, in this case each cell must be defined separately as the layout does not support any classification variables. Let's reproduce the same plot using the LATTICE layout. Some modifications are required for the input

data as we need a separate pair of variables for each cell, i.e. the data should be transformed to a vertical structure.

WORK.CARS_TRANS

	Origin	length_SUV SUV	weight_SUV SUV	length_Sedan Sedan	weight_Sedan Sedan	length_Sports Sports	weight_Sports Sports	length_Wagon Wagon	weight_Wagon Wagon
1	Asia	.	.	197	3880	.	.	.	.
2	Asia	.	.	197	3893	.	.	.	.
3	Asia	.	.	.	.	169	3428	.	.
4	Asia	.	.	.	.	169	3188	.	.
5	Asia	189	4035	.	.	.	.	.	.

DATA 4

And the template itself:

```
proc template;
    define statgraph regr_scatter;
    mvar reg_eq1_asia   reg_eq2_asia   reg_eq3_asia   reg_eq4_asia
         reg_eq1_europe reg_eq2_europe reg_eq3_europe reg_eq4_europe
         reg_eq1_usa    reg_eq2_usa    reg_eq3_usa    reg_eq4_usa;
    dynamic x1 y1 x2 y2 x3 y3 x4 y4 group;
    begingraph;
        entrytitle "Figure: Scatter plot of weight versus length of cars
                    with different origin and type using GTL LATTICE layout";
        layout lattice / rows=2 columns=2
                         rowdatarange=unionall columndatarange=unionall;
            cell;
                cellheader;
                    layout gridded / border=true;
                        entry eval(collabel(y1));
                    endlayout;
                endcellheader;
                layout overlay;
                     regressionplot x=x1 y=y1 / group=group name="plot1";
                     scatterplot    x=x1 y=y1 / group=group name="plot2";
                     layout gridded / autoalign=(bottomright) border=true;
                         entry reg_eq1_asia   / textattrs=(size=5);
                         entry reg_eq1_europe / textattrs=(size=5);
                         entry reg_eq1_usa    / textattrs=(size=5);
                     endlayout;
                endlayout;
            endcell;
/*The same code is repeated three more times to define all the four cells*/
                ...
            sidebar / align=left;
              entry "Length (IN)" / rotate=90;
            endsidebar;
            sidebar / align=bottom;
               entry "Weight (LBS)";
            endsidebar;
            sidebar / align=bottom;
               mergedlegend "plot1" "plot2";
            endsidebar;
            rowaxes;
               rowaxis / display=(ticks tickvalues);
               rowaxis / display=(ticks tickvalues);
             endrowaxes;
             columnaxes;
               columnaxis / display=(ticks tickvalues) offsetmin=0.06;
```

```
                columnaxis / display=(ticks tickvalues) offsetmin=0.06;
              endcolumnaxes;
          endlayout;
      endgraph;
      end;
run;
```

The template contains four cells. As the cell headers are required, each OVERLAY layout is nested in the CELL block. To obtain shared axes the UNIONALL value for the ROWDATARANGE and the COLUMNDATARANGE options in the LATTICE layout is used along with the corresponding ROWAXES and COLUMNAXES statements. Also, to obtain axis labels the SIDEBAR statements are used within the LATTICE layout. Another thing to consider is that the macro variables for the inset information based on the ENTRY statement must be created beforehand. After running the SGRENDER procedure with all the required DYNAMIC statements we get results that look very similar to the previous ones - Figure 9.

```
proc sgrender template=regr_scatter data=cars_trans;
   dynamic x1="weight_suv"     y1="length_suv"
           x2="weight_sedan"   y2="length_sedan"
           x3="weight_sports"  y3="length_sports"
           x4="weight_wagon"   y4="length_wagon" group="origin";
run;
```

Figure: Scatter plot of weight versus length of cars with different origin and type using GTL LATTICE layout

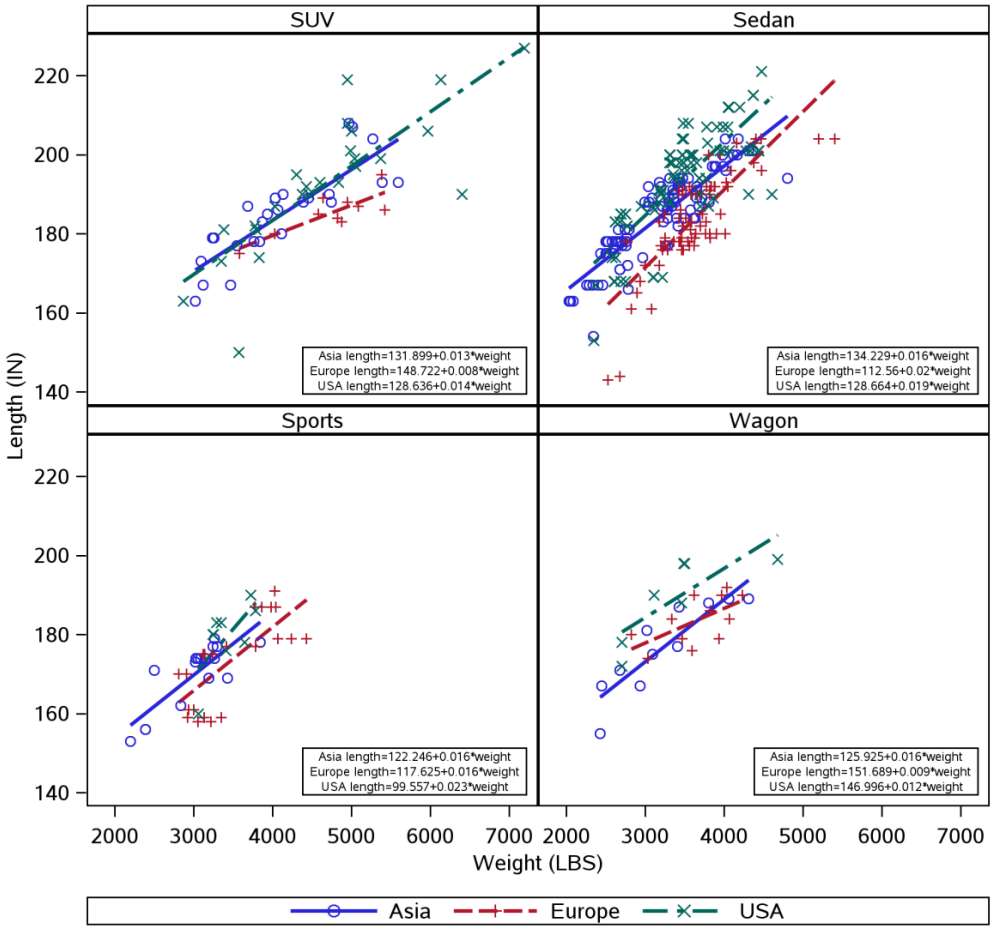

Figure 9. LATTICE LAYOUT

SGSCATTER METHOD

The last but not least way is to use the SGSCATTER procedure. While all the previous approaches work for almost any type of plot, this one is only applicable for scatter plots. This is not the most flexible and preferable way though it can be used when a programmer needs to create paneled scatter plots based on multiple combinations of variables. So the DATA 4 data set can be used for this example. To include the inset information the DATA 5 annotated data set is used.

WORK.ANNO

	Type	Origin	function	label	x1	y1	linethickness	width	height
1	SUV	Asia	TEXT	Asia length=131.899+0.013*weight	48	60	.	40	.
2	SUV	Europe	TEXT	Europe length=148.722+0.008*weight	48	58.9	.	40	.
3	SUV	USA	TEXT	USA length=128.636+0.014*weight	48	57.8	.	40	.
4	SUV	USA	RECTANGLE		48.5	59	1	20	4

DATA 5

```
proc sgscatter data=car_trans sganno=anno;
  plot length_suv*weight_suv length_sedan*weight_sedan
       length_sports*weight_sports  length_wagon*weight_wagon /
       reg group=origin uniscale=all legend=(across=3);
run;
```

Figure: Scatter plot of weight versus length of cars with different origin and type using SGSCATTER

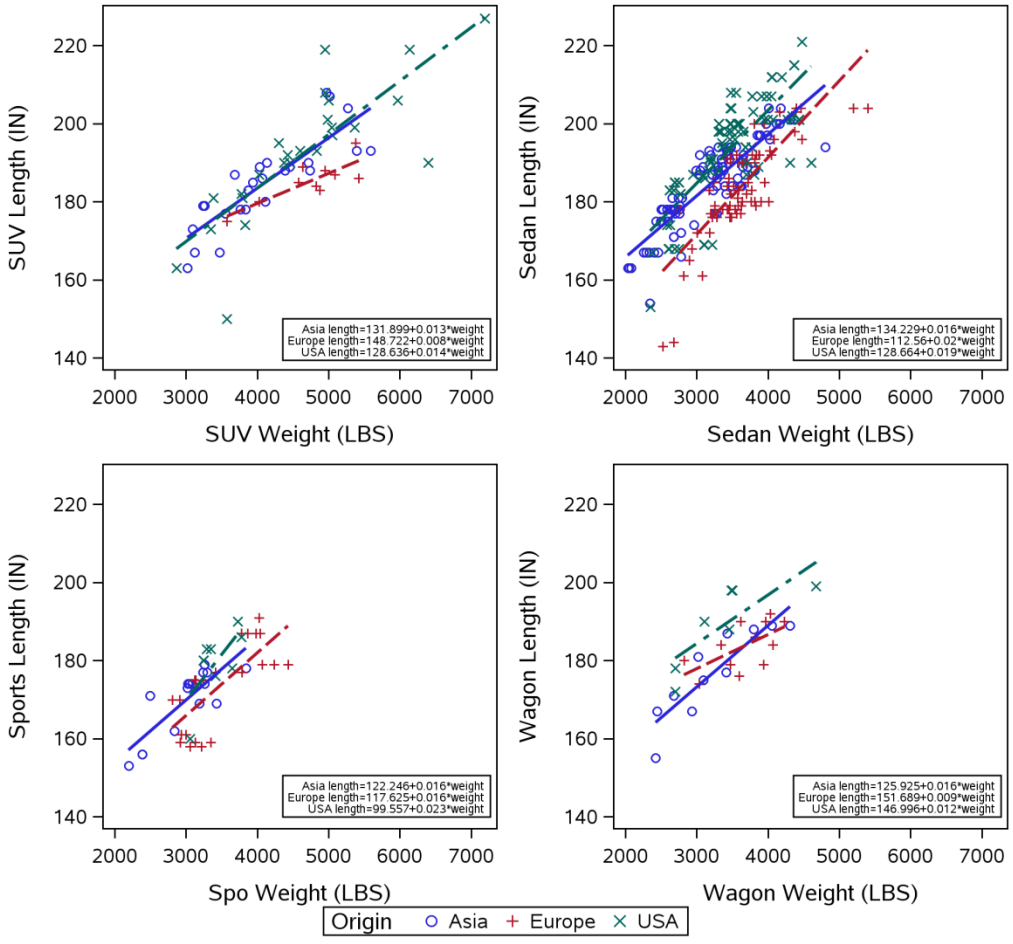

Figure 10. SGSCATTER

As we can see from Figure 10 - the PLOT statement doesn't support shared axes and the legend does not contain line patterns. Also there are no header labels for the plots, the different axis labels being displayed instead. Of course, the SGSCATTER has two more statements to draw the plots – i.e. MATRIX and COMPARE. However in this case they are not suitable. Summing up – the SGSCATTER has quite a limited flexibility and is only applicable for a narrow scope however it can still be helpful.

CONCLUSION

Obviously, each approach has its own advantages and disadvantages, and which one to use depends on the situation and the kind of plot. See Table 1. for an overview of the approaches described above. The SCGSCATTER procedure is not included as it is only applicable for scatter plots.

	GREPLAY	SGPANEL	GTL DATAPANEL	GTL LATTICE
Classification variable	Depends on what procedure is used to build single plots	Four ways of using class variable are available based on the PANELBY statement: PANEL – any number of class variables LATTICE – requires two variables and arranges cells in such a way that the values of the first variable are columns and the values of the second are rows COLUMNLATTICE/ROWLATTICE – allow one class variable which is used to arrange cells either in columns or in rows	Supports N class variables; builds a cell for each combination of class variables	Not supported
Different kinds of plots across one page		Not supported as based on the class variable	Not supported as based on the class variable	Any kinds of plot can be combined in different cells within one page
Computed plots		All basic computed plots are supported	Not supported, however it is possible to derive parameters for some plots and use the GTL plot statements for parametric plots, i.e. BARCHARTPAR, BOXPLOTPARM, LINEPARM etc.	All GTL plot statements can be used
Easy to use	Requires many steps and a lot of effort;	The easiest and the	Requires building	Requires building a template and

	only catalogue entries can be replayed and this involves even more work if the SG-procedures are used for single plots	most efficient way	a template	defining each cell separately
Flexibility	Quite flexible but already outdated. Also may be difficult to implement some features which can be easily programmed using GTL and the SG-procedures (like common axes)	Not as flexible as building your own template using GTL. Also there are some restrictions on using different plots together, i.e. for example, BOXPLOT cannot be combined with any other plot	Quite flexible, however has some restrictions like, for example, it is not possible to create different plots in different cells	The most flexible way as each cell is being built separately and all the GTL features are available; however may not be the best way for using with class variables
ODS GRAPHICS is supported	No	Yes	Yes	Yes
Easy to add inset information (“number of observations”, “regression parameters” etc.)	Depends on what procedure is used to build single plots	The SGANNO option is supported starting from SAS 9.3 The INSET statement is supported starting from SAS 9.4	The INSET statement is supported Starting from SAS 9.4 the SGANNO option is supported in the SGRENDER procedure	The INSET and ENTRY statements are supported Starting from SAS 9.4 the SGANNO option is supported in the SGRENDER procedure

Table 1. COMPARISON OF THE APPROACHES

CONTACT INFORMATION

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

Vladlen Ivanushkin
inVentiv Health Clinical
vladlen.ivanushkin@inventivhealth.com