PhUSE 2017

Paper PP04

The graph template language of SAS®

Stephanie Fechtner, Grünenthal GmbH, Aachen, Germany

ABSTRACT

To create a graph in SAS®, a programmer has different options at hand. They may use procedures such as `PROC SGPLOT`, `PROC SGPANEL` or `PROC SGSCATTER` or they may opt for the more adaptable graph template language (GTL). Within SAS®, it is possible to create graphs with different layouts including various types of plots. Furthermore, special attributes like the legend, titles or the labels of the axes can be embedded into the graphs. However, to create more complicated graphs, it is worth considering the use of the combination of `PROC TEMPLATE` and `PROC SGRENDER`, belonging to the GTL. With this language, it is far more effective to create a graph just the way the programmer would like, however the syntax of this language is often quite complex. This paper describes in which situations it is worth using the GTL and how it can simplify the programming of a good figure in SAS®.

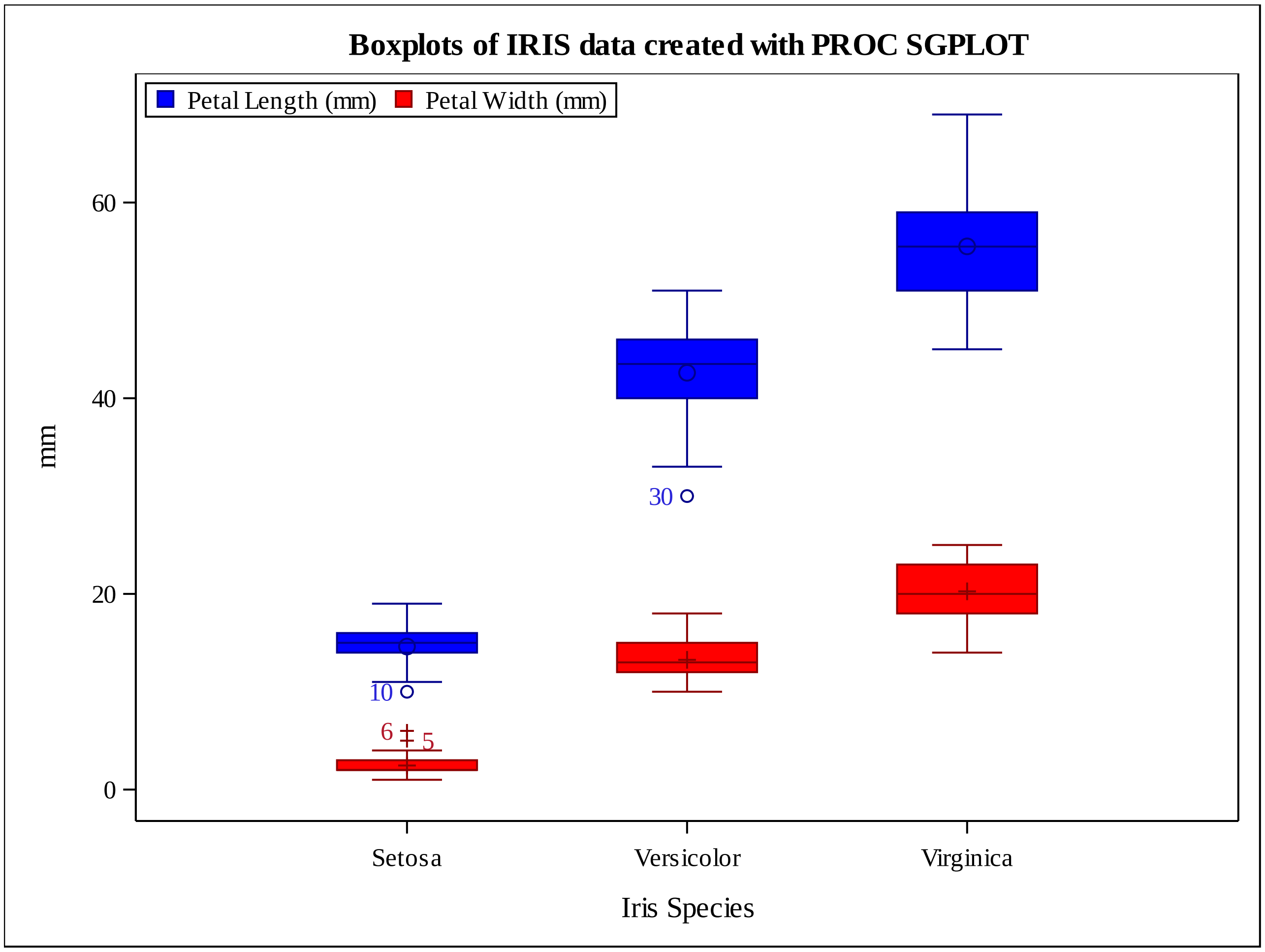
CREATING PLOTS – SAS® GRAPH PROCEDURES

With the aid of SAS® procedures, it is possible to create different kinds of graphs like boxplots, series plots, scatterplots or histograms. In this paper, the IRIS dataset of the library `SASHELP` will be used to illustrate how the layouts of graphs such as the title, the attributes of the axes and the appearance of the legend can be controlled and changed by the programmer. The three procedures `PROC SGPLOT`, `PROC SGPANEL` and `PROC SGSCATTER` will be explained. However, there are several other procedures which a programmer can use to create a graphic (see SAS® support).

PROC SGPLOT

With this procedure, the programmer can create one or more plots within one graph. If there are several plots inside `PROC SGPLOT`, they will be overlaid. The graph title and the label and attributes of the axes can be chosen by the programmer as well as the content and the position of the legend. Furthermore, layouts like the color, size and appearance of the plots can be changed depending on the kind of plot they would like to present.
The following example shows how boxplots of the IRIS dataset could be created by using `PROC SGPLOT`. Here, the distribution of the length and the width of petal were shown by using blue and red boxes, the variable *Species* was set as the category variable and `SPREAD` prevented an overlapping of possible identical values which were categorized as outliers:

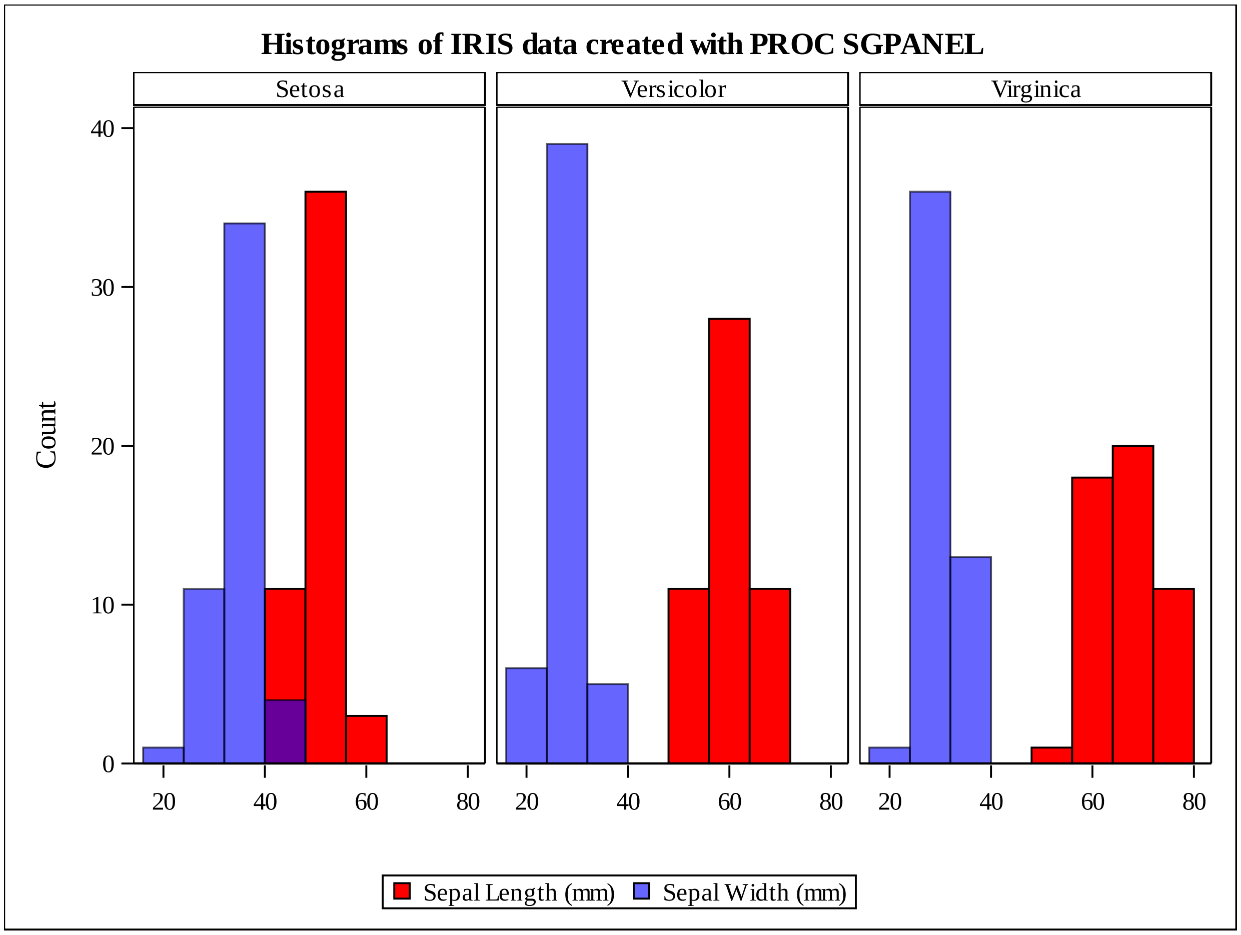
```
title 'Boxplots of IRIS data created with PROC SGPLOT';
proc sgplot data=sashelp.iris;
       vbox PetalLength/ category=Species datalabel BOXWIDTH=0.5 spread
                    fillattrs=(color=blue) WHISKERATTRS=(color=darkblue)
                    MEDIANATTRS=(color=darkblue) LINEATTRS=(color=darkblue)
                    OUTLIERATTRS=(color=darkblue) MEANATTRS=(color=darkblue);
       vbox PetalWidth/ category=Species datalabel BOXWIDTH=0.5 spread
                    fillattrs=(color=red) WHISKERATTRS=(color=darkred)
                    MEDIANATTRS=(color=darkred) LINEATTRS=(color=darkred)
                    OUTLIERATTRS=(color=darkred) MEANATTRS=(color=darkred);
       keylegend / location=inside position=topleft; /*Set the location and the
                                                   position of the legend*/
       xaxis label="Iris Species";    /*Set the label of the x-axis*/
       yaxis label="mm" min=0 max=70; /*Set the label and the range of the y-axis*/
run;
```



PROC SGPANEL

With this procedure, it is possible to create the same kinds of plots as with the procedure `PROC SGPLOT`. However, `PROC SGPANEL` enables the programmer to create a panel of graphs divided by one or more arbitrary class variable(s); that means the plots will be displayed in different cells. The number of variables which are used for the division can be chosen and the programmer can decide if the name(s) of the used class variable(s) should be displayed or not. Furthermore, the programmer can control the order of the plots as well as the space between the panels.
The following example shows how histograms of the length and the width of sepal belonging to the IRIS dataset can be created. Here, the vertical axis displayed the frequency count and the class variable *Species* was used to split the graph into different panels. Furthermore, some space was added between the figures and the name of the class variable was not displayed:

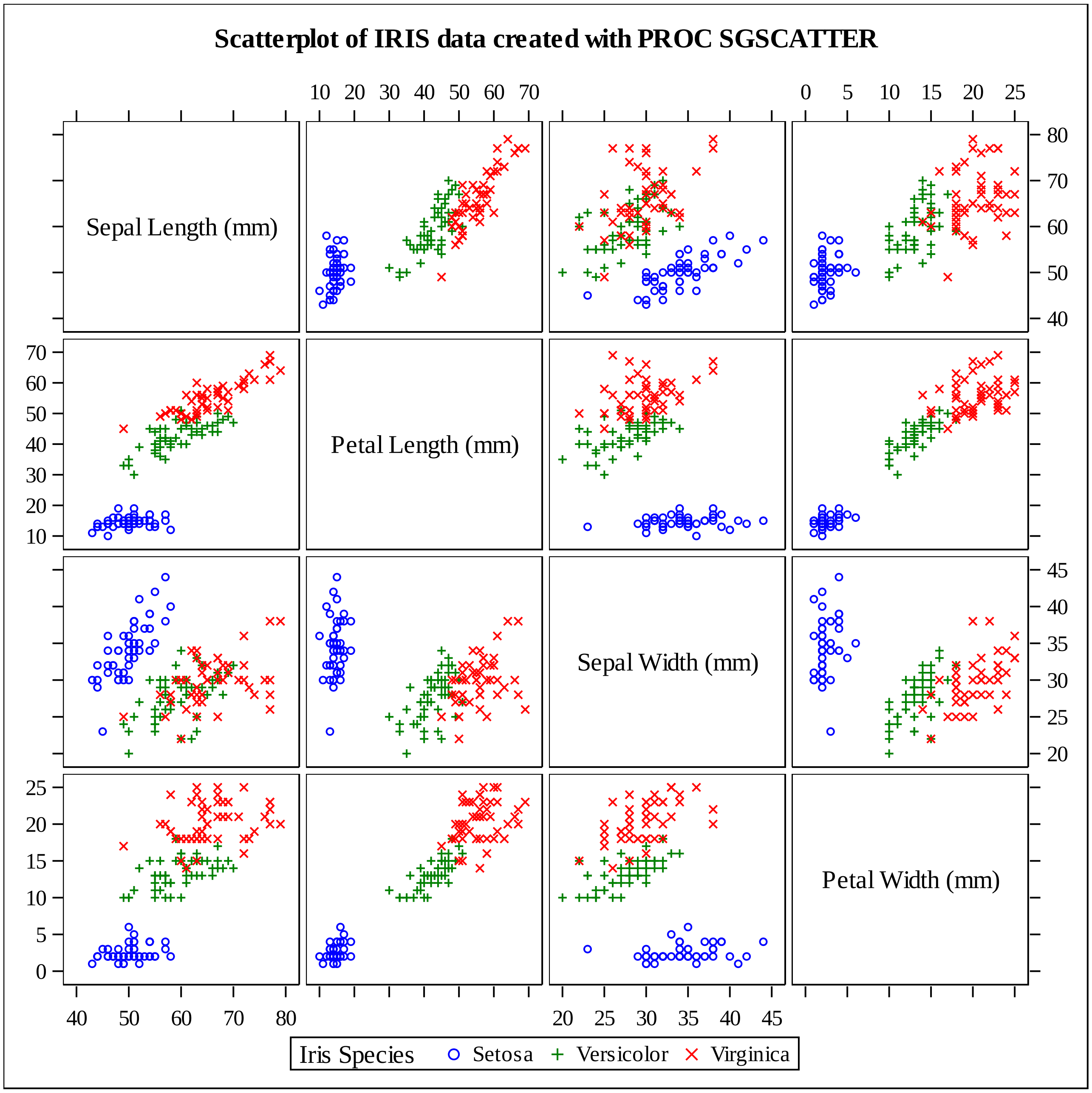
```
title 'Histograms of IRIS data created with PROC SGPANEL';
proc sgpanel data=sashelp.iris;
       panelby Species / layout=columnlattice columns=3 onepanel sort=data novarname
                           spacing=5;
       histogram SepalLength / transparency=0.0 scale=count fillattrs=(color=red);
       histogram SepalWidth / transparency=0.4 scale=count fillattrs=(color=blue);
       keylegend / position=bottom;
       colaxis label= " "; /*Set the label of the x-axis (here: "colaxis")*/
run;
```



PROC SGSCATTER

With this procedure, the programmer can create a paneled graph of scatterplots for multiple combinations of variables. Within the `MATRIX`-statement they define which variables should be considered for the creation of the graph. Here, it is also possible to specify an arbitrary class variable as group variable whereby the programmer can define the attributes of the different values of this variable, such as colors and symbols. To do this, a separate dataset which contains an ID variable and the information about the attributes must be embedded into the SAS® graph procedure by using the command `DATTRMAP`. `ATTRID` specifies the value of the ID variable included in this dataset. This approach can also be applied when `PROC SGPLOT` or `PROC SGPANEL` is used.
To create a matrix of scatterplots for the four variables belonging to the IRIS dataset with predefined colors and symbols of the markers, a programmer could use the following code:

```
data irisattrs;
      retain id "myid";
      length value $ 11 markercolor $ 6 markersymbol $ 6;
      input value $ markercolor $ markersymbol $;
      cards;
       Setosa blue circle
       Versicolor green plus
       Virginica red x
      ;
run;

title 'Scatterplot of IRIS data created with PROC SGSCATTER';
proc sgscatter data=sashelp.iris dattrmap=irisattrs;
    matrix sepallength petallength sepalwidth petalwidth/ group=species attrid=myid;
run;
```



In summary, a programmer can, with the aid of the SAS® procedures, create different types of plots with various layouts in a single cell or in several cells. However, there are some limitations when these procedures are used to generate a graph. For example, it can be very difficult or even impossible to display different graphs with various axes on one page. It could also be difficult to add some information next to the graph, like the number of patients at specific timepoints.
For more complex graphs, a programmer should consider the use of the graph template language (GTL) which can be described as an extension to the Output Delivery System (ODS). This language enables the programmer to create more sophisticated graphics and to get a graph just the way they would like in a more effective way (see also SAS® support). The GTL and the necessary steps will be presented in the following section.

THE GRAPH TEMPLATE LANGUAGE (GTL)

To create a graph with the aid of the GTL of SAS®, a programmer needs to implement two steps. First, they need to build the template by using the procedure `PROC TEMPLATE`. Here, the programmer can choose between different layouts to produce different kinds and combinations of graphs in a way they would like. In the second step, the graph is actually produced by using `PROC SGRENDER`. Within this step the programmer needs to choose the created template and to specify the dataset which contains the plot variables.
Furthermore, in SAS® 9.4 it is possible to include annotation objects into the graph. To do this, the programmer needs to create an annotation dataset first and the creation of the graph takes place afterwards. In previous SAS® versions it was just possible to embed annotation objects into the graphs if SAS® graph procedures were used; within the GTL this was not possible.
In the following subsections these three steps will be explained. Afterwards, an example will be given, how the boxplots, histograms and scatterplots of the IRIS dataset which were shown in the previous examples can be combined into one graph. Annotation objects will be embedded into the graph as well.

THE CREATION OF AN ANNOTATION DATASET (OPTIONAL)

The programmer has the opportunity to add some annotations like text, arrows or lines to the graph by building an annotation dataset first. The variables of this dataset have predefined names and contain information about the attributes and the position of the annotation objects, respectively. Each observation represents a command to draw or perform an action.
Some of these annotation variables are shown in the following passage. However, all available variables as well as detailed information about them can be obtained from the internet (SAS® support).

➢ `ID`:	should be a unique character value which specifies the ID value of the annotation objects. This variable is required for the drawing of the annotation.
➢ `FUNCTION`:	specifies which kind of action should be taken (`DRAW`, `ARROW`, `MOVE`, `TEXT`, `LABEL`, `PIE`…).
➢ `X1/X2`:	specifies the numeric horizontal coordinates (start and endpoint).
➢ `Y1/Y2`:	specifies the numeric vertical coordinates (start and endpoint).
➢ `COLOR`:	specifies the color of the annotation object.
➢ `WIDTH`:	specifies the width of the annotation object.
➢ `WIDTHUNIT`:	specifies the unit of the measurement of `WIDTH.`
➢ `TEXTWEIGHT`:	specifies if the annotation character is written in `BOLD` or `NORMAL.`
➢ `TEXTSIZE`:	specifies the font size of the annotation character. The default unit of measurement is pixels.

THE CREATION OF A TEMPLATE $\longrightarrow$ PROC TEMPLATE

When a programmer creates a template for a graph, there are required and optional steps they need to implement. Some of these steps are shown in the following passage. Detailed information about these and all other steps can also be obtained from SAS® support.

- `DEFINE STATGRAPH`: Choose the name of your template
- `DYNAMIC`: Define one or more dynamic variables (optional)
- `BEGINGRAPH`: Start to specify your graph:
  - `ENTRYTITLE`: Define the title of your graph
  - `DISCRETEATTRMAP`: Define different attributes of your graph (optional)
  - `LAYOUT`: Choose the layout of your graph *(1)*
  - Define the attributes of your `ROWAXES` and `COLUMNAXES`
  - Define plot statements within the layout statement *(2)*
  - Add a `DISCRETELEGEND` or a `MERGEDLEGEND` to your plot(s)

  - If necessary, define further nested layout statements
  - New in SAS® 9.4: If desired, add an `ANNOTATE` statement to include annotation objects into the graph
- `ENDGRAPH`: End the specification of the graph
- `END`: End the creation of the template

(1) A programmer can decide between different (nested) layouts of their graph. Information about all available layouts can be obtained from SAS® support. The three different layouts that were used in the subsequent example are:

- `LAYOUT OVERLAY`: displays a 2D plot in a single cell
- `LAYOUT GRIDDED`: displays several independent plots in a multi-cell layout
- `LAYOUT LATTICE`: displays several plots in a multi-cell layout across different `ROWS` and `COLUMNS` with chosen `ROWWEIGHTS` and `COLUMNWEIGHTS`

(2) Some of the available plot statements are:

- `SCATTERPLOT`: generates a scatterplot of the input data
- `SCATTERPLOTMATRIX`: generates a matrix of all pairwise scatterplots, typically placed within a `LAYOUT GRIDDED` block
- `HISTOGRAM`: generates a univariate histogram computed from input data
- `BOXPLOT`: generates boxplots that are computed from input data
- `SERIESPLOT`: generates a series of line segments that connects the observations of input data

THE GENERATION OF THE FIGURE ⟶ PROC SGRENDER

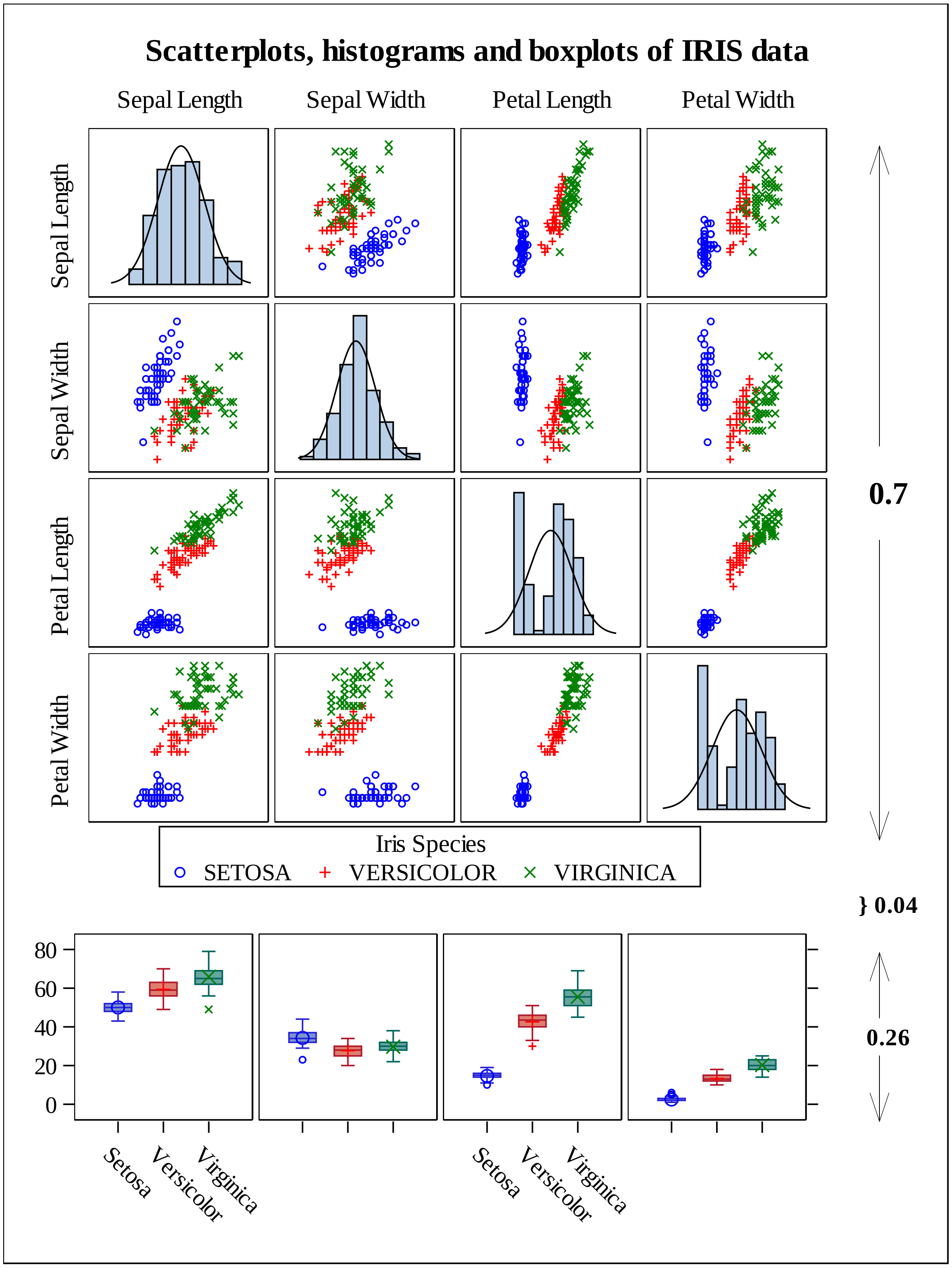
- Choose the created template and the dataset that contains the plot variables
- New in SAS® 9.4: Specify `SGANNO`, to embed an annotation dataset into the graph
- Define `DYNAMIC` variables, if necessary

EXAMPLE: IRIS DATA

```
/*Creation of an annotation dataset to show the influence of ROWWEIGHTS*/
data anno;
   length label $30 id $7;
   /*Create the first block including arrows and a number*/
      id='Weights'; function='arrow'; x1=140; y1=360; x2=140; y2=520; color='black';
   output;
      id='Weights'; function='text'; x1=145; y1=335; color='black'; label='0.7';
      width=1000; widthunit='pixel'; textweight='bold'; textsize=12;
   output;
      id='Weights'; function='arrow'; x1=140; y1=310; x2=140; y2=150; color='black';
   output;

   /*Create the second block including a curly bracket and a number*/
      id='Weights'; function='text'; x1=145; y1=115; color='black'; label='}0.04';
      width=1000; widthunit='pixel'; textweight='bold'; textsize=10;
   output;

   /*Create the third block including arrows and a number*/
      id='Weights'; function='arrow'; x1=140; y1=55; x2=140; y2=90; color='black';
   output;
      id='Weights'; function='text'; x1=145; y1=45; color='black'; label='0.26';
      width=1000; widthunit='pixel'; textweight='bold'; textsize=10;
   output;
      id='Weights'; function='arrow'; x1=140; y1=35; x2=140; y2=0; color='black';
   output;
run;
```

```
proc template;

    define statgraph iris_tem;    /*iris_tem: name of the template*/
       dynamic TITLE;             /*TITLE: dynamic variable*/
    begingraph;

              entrytitle TITLE;
              discreteattrmap name="attitudes"/ ignorecase=true;
                     value "Setosa" / markerattrs=(color=blue symbol=circle);
                     value "Versicolor" / markerattrs=(color=red symbol=plus);
                     value "Virginica" / markerattrs=(color=green symbol=x);
              enddiscreteattrmap;
              discreteattrvar attrvar=Spec_gr var=Species attrmap="attitudes";

/*LAYOUT LATTICE with 3 rows and 2 columns with specified ROWWEIGHTS and
COLUMNWEIGHTS*/
layout lattice/ rows=3 columns=2 rowweights=(.7 .04 .26) columnweights=(.9 .1);

       /*LAYOUT GRIDDED as a nested layout statement*/
       layout gridded;
              ScatterPlotMatrix SepalLength SepalWidth PetalLength PetalWidth /
                     NAME="matrix" Group=Spec_gr diagonal=(histogram normal)
                     datalabelattrs=(size=4);
              DiscreteLegend "attitudes" / type=marker title="Iris Species" across=3
                     location=outside;
       endlayout;

       /*Create three empty cells to gain some space under the ScatterPlotMatrix and on
       the right side of the graphic (for the annotation objects)*/
        cell; entry " "; endcell;
        cell; entry " "; endcell;
        cell; entry " "; endcell;

       /*LAYOUT LATTICE with 1 row and 4 columns (and some space between the columns)
       as a nested layout statement*/
       layout lattice/ rows=1 columns=4 columndatarange=union rowdatarange=union
                        columngutter=2;

            /*Define the attributes of the columnaxes and rowaxes*/
            columnaxes;
              columnaxis / display=(ticks tickvalues);
              columnaxis / display=(ticks);
              columnaxis / display=(ticks tickvalues);
              columnaxis / display=(ticks);
            endcolumnaxes;
            rowaxes;
              rowaxis / display=(ticks tickvalues) displaysecondary=(ticks);
            endrowaxes;

              /*Create 4 boxplot graphics, each with Species as group variable*/
              layout overlay; boxplot x=Species y=SepalLength/group=Spec_gr; endlayout;
              layout overlay; boxplot x=Species y=SepalWidth/group=Spec_gr; endlayout;
              layout overlay; boxplot x=Species y=PetalLength/group=Spec_gr; endlayout;
              /*Include the ID variable of the annotation dataset somewhere*/
              layout overlay;
                  boxplot x=Species y=PetalWidth/group=Spec_gr; annotate /id="Weights";
              endlayout;
       endlayout;

       /*Create an empty cell to gain some space on the right side of the graphic*/
       cell; entry " "; endcell;

endlayout;

endgraph;
end;
run;
```

```
/*Choose the template, the input dataset and the annotation dataset*/
proc sgrender data=sashelp.iris template=iris_tem sganno=anno;
    dynamic title="Scatterplots, histograms and boxplots of IRIS data";
run;
```



CONCLUSION

- SAS® procedures can be used for programming simple graphs
- GTL has a complex syntax
- GTL is more adaptable than SAS® graph procedures
- GTL allows the user to produce different graphs on one page
- Templates can be used to produce the same output from different datasets

REFERENCES

SAS® support (https://support.sas.com/en/support-home.html)

CONTACT INFORMATION

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

Author‘s Name	Stephanie Fechtner
Company	Grünenthal GmbH
Address	Zieglerstr.6
City / Postcode	Aachen, 52078, Germany
Work Phone:	+49 241 569-2848
Email:	Stephanie.fechtner@grunenthal.com
Web:	www.grunenthal.com