Paper 3193-2015

Mapping out SG Procedures and Using PROC SGPLOT for Mapping

Frank Poppe, PW Consulting

ABSTRACT

This paper describes the main functions of the SAS® Statistical Graphics (SG) procedures (also referred to as SAS ODS Graphics procedures) and how they are related. It also offers a way to create geographic maps with regions colored through response variables.

SAS users who have experience with SAS/GRAPH® procedures will hesitate to change over to SG procedures. However, for virtually all tasks that you can perform with the well-known SAS/GRAPH procedures, the corresponding SG procedures are easily pointed out, frequently providing you with even more enhanced features. And, because the SG procedures continue to be enhanced with new features and the appearance of many elements is governed by the ODS styles, they are very well suited to providing a consistent style across all your output, text, and graphics.

The paper first describes the SGPLOT procedure, and then moves on to the more elaborate possibilities of the SGPANEL and SGSCATTER procedures. Both these procedures can create a matrix or panel of graphs. The different goals of these two procedures are explained: comparing a group of variables versus comparing the levels of two variables.

This paper also describes two utilities related to the SG procedures, the Graphics Editor and the Graphics Designer, which are delivered as SAS® Foundation applications.

Finally, this paper shows the few steps that are necessary to convert the data sets that contain your data and your map coordinates into data sets that enable you to use the power and features of PROC SGPLOT (instead of the GMAP procedure) to create your map in any projection system and any coordinate window.

INTRODUCTION

For a few years now, the SG procedures (PROC SGPLOT, PROC SGSCATTER, PROC SGPANEL, and so on) have been part of Base SAS® and thus available for everybody.
“SG” originated as “Statistical Graphics”, but nowadays the procedures are often referred to as SAS® ODS Graphics. With the syntax in a 1000+ page document, it is quite a challenge to start using them. Also, SAS® Enterprise Guide® currently has no graphics tasks that generate code for the SG procedures (except those in the statistical arena). For a long time SAS/GRAPH has been the vehicle for producing presentation-ready graphs of your data. In particular, the SAS users that have experience with the procedures from SAS/GRAPH will hesitate to change over. But the SG procedures continue to be enhanced with new features. And, because the appearance of many elements is governed by the ODS styles, they are very well suited to provide a consistent style across all your output, text and graphics.

PROC SGPLOT – PROC SGPANEL – PROC SGSCATTER

The ‘nucleus’ of the set of ODS Graphics procedures is the SGPLOT procedure. It supports a variety of statements to produce line plots, scatter plots, bars, histograms, plots showing statistical properties, together with statements to add all kinds of additional information. For those thinking in old terms: it combines the features of most SAS/Graph procedures. This encompasses the old main vehicles GPLOT and GCHART, but also the results of the later added procedures GAREABAR, GBARLINE and GRADAR. And starting with SAS 9.4 also the area colored maps maps (choropleth) of PROC GMAP can be produced.

PROC SGPANEL can be seen as a generalization of PROC SGPLOT by adding a PANELBY statement. The function of this statement is comparable to the well known BY statement. A *BY statement* creates output for each value of the variable or variables on the BY statement, starting a new page for each combination of values.

The *PANELBY statement* creates rows (or columns) of graphs, one graph for each combination of values of the variables on the PANELBY statement.
If you have exactly two variables on the PANELBY statement you can also choose for a 'lattice' layout. Instead of just enumerating all combinations of values and putting them next to each other, filling rows, the lattice layout let the values of one variable create the rows and the other the columns.
Almost all statements that can be used to create graphs in PROC SGPLOT can be used in PROC SGPANEL, with also most of the options for those statements.

PROC SGSCATTER can also be seen as a generalization of PROC SGPLOT, but then as a more focused one; and at the same time a more limited one. As the name already suggests only scatterplots can be produced, with a few statistical extras.
It always produces a panel of graphs, similar to PROC SGPANEL.
The difference is that the rows and the columns of the lattice now are defined by *variables* instead of by the combination of *values* of variables, as was the case with PROC SGPANEL.

There are three variants:

- Rows of graphs of all requested combinations of variables: the PLOT statement);
- A lattice of graphs defined by a list of variables defining the rows and another list defining the columns: the COMPARE statement;
- A matrix of all combinations of a list of variables (thus defining both columns and rows): the MATRIX statement.

PROC SGDESIGN – PROC SGRENDER

With PROC SGDESIGN you can *not*, as one might expect, design a graph. With this procedure you can create a graph defined by the *ODS Graphics Designer* application. This application can be used through Enterprise Guide, *but only if you have SAS installed locally,* or a SAS Foundation session. This application creates a SGD file, which is what SGDESIGN requires as in put.

PROC SGRENDER does what the name suggests: it renders a graph. The specifications for this graph come from a *graphics template*. PROC TEMPLATE creates such templates.
Alternatively one can use the specifications created by the SAS ODS Graphics Editor. These are SGE files. The latter application is also only available through a SAS Enterprise Guide with SAS installed locally or a SAS Foundation session.

A description of PROC TEMPLATE is beyond the scope of this paper, but the concept of PROC TEMPLATE is the following.
Templates are not unique to SGRENDER: templates are the basis for all output created by ODS, both tables and graphs. So also everything produced by PROC SGPLOT, SGPANEL and SGSCATTER is based on templates.
And PROC TEMPLATE not only creates the templates for tables and graphs, but also for styles, which define colors, sizes, fonts, spacing, etcetera for all elements of output.
Finally PROC TEMPLATE can also be used to define or adjust *tagsets*, which define how an output object, which is the result of combining data with a template and a style, should be translated to an external format like HTML (optionally with CSS), XML, LaTeX, etcetera.

THE RELATIONS BETWEEN THE PROCEDURES AND FILES THEY PRODUCE

The **ODS Graphics Designer** stores its results in a SGD file. **PROC SGDESIGN** can create ODS output from this file, either using the SAS data file used creating he design, or by using an alternative SAS data set. By default the alternative SAS data set should have exactly the same variables as in the data set that was used while designing the graph. But it is also possible to parameterize the design, making it possible to map variables with a different name (but with of course the same characteristics).
The SGD file is in effect a collection of files in zip-format. The archive contains a small example of the result, typically called icon.png, and a XML-file.

This XML-file contains a description of the parameters and the PROC TEMPLATE code to create a graphic template.

This code can by surfaced also by the 'view code' option in the application. As a consequence there is second route to use the result of a Graphics Designer session. One can use it as input to PROC SGDESIGN, but alternatively one can save the **PROC TEMPLATE** code, then generate the custom template and use that as input for **PROC SGRENDER**. One could use the designer as a first approximation to the desired result, and adjust the code to fine-tune it with PROC TEMPLATE.

The **ODS Graphcis Editor** takes an external PNG file as input, or a previously with the Graphics Editor created SGE file. The output is also either a SGE file or a PNG file.
Like a SGD file a SGE file is in fact a zip-archive. It contains the image in PNG format, plus XML describing the elements that are added to it.
PROC SGRENDER can create ODS output from the SGE file.

STATEMENTS

The number of statements that can be used to create different types of plots is large. The current documentation (for SAS 9.4) describes, for PROC SGPLOT only, the syntax of over 40 statements. The documentation lists them alphabetically.
In an attempt to create some structure I have organized the PROC SGPLOT statements in a number of categories (and, as mentioned before, PROC SGPANEL uses the same statements). Within these groups the required arguments are usually the same (or very much alike…). Let us therefore first pay some attention to the arguments, at the same time introducing some first examples of code.

ARGUMENTS

The statements mostly have a few required arguments, usually two, which in the syntax come before the slash. They are in the *argument=value* form, unless there is only one argument, or one *kind* of argument (with multiple values), then only the value can be specified.And usually there is a large number of optional arguments, or options, which appear *after* the slash.

An example will clarify this.
The SERIES statement creates a line plot, connecting a series of points, in the order they are read from the source. The minimal syntax is

```
SERIES X=var Y=var ;
```

If one wants to add a reference line there is the REFLINE statement. The only argument is the value, or the values, for which one wants a reference line. The valyue can be specified in two ways: through a variable and through a constant (or list of constants). But the values always can have only one meaning, so there is no keyword. So the minimal syntax is:

```
REFLINE var ;
```

(*var* referring to a variable) or:

```
REFLINE value1 <… value2> ;
```

(*value1*, etc, refererring to one or more constant values).

One might wonder how the system knows whether these are values on the horizontal or on the vertical axis. This is specified by the AXIS option, which has a default of Y (creating horizontal lines).

When these two statements are translated to example, using SASHELP.CLASS without any additional work, the following code creates Figure 1.

```
proc SGPLOT data=sashelp.class ;
series x=height y=weight ;
refline 80 100 ;
run ;
```

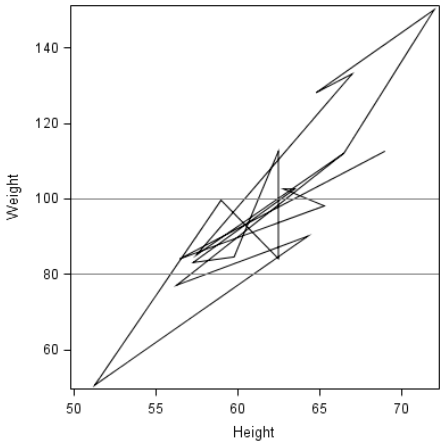


Figure 1. Series statement only

Note that the lines criss-cross because the observations are not ordered in any way.
Using one of the variables on the REFLINE statement, and ordering them this time on the variable for the x-axis, one will get Figure 2.

```
proc sort data = sashelp.class out=class ;
by height ;
run ;
proc sgplot data = class ;
series x = height y = weight ;
refline weight / lineattrs=(pattern=35 color=gray) ;
run ;
```

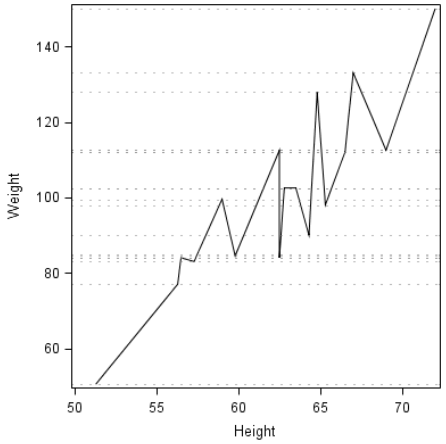


Figure 2. Series statement with reference line for axis variable

Now we have a more ordered series, and a reference line is created for each value of weight. To distinguish them from the lines between the data points the reference lines are specified to be dotted gray.

One could use another variable to create the values for the reference lines. For the example in Figure 3

we have created a new variable showing the nearest multiple of 10 for each value. The LINEATTRS option is used as well.

```
data class ;
  set class ;
  refWeight = round ( weight , 10 ) ;
run ;
proc sgplot data = class ;
series x = height y = weight ;
refline refWeight / lineattrs = ( pattern = 1 color = red thickness = 5 )
transparency = .75 ;
run ;
```

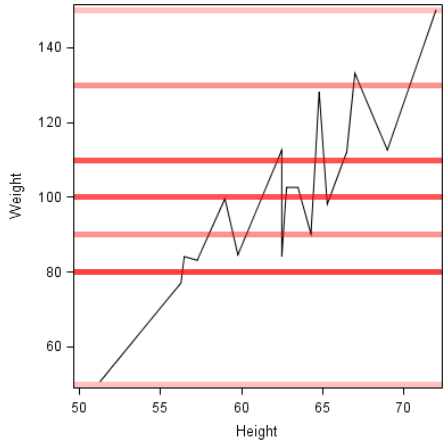


Figure 3. Reference lines from a new variable

The reference lines are made extra thick and transparent. The different shades of red are defined by the number of times a reference line is drawn for that value (they are superimposed on each other).

CATEGORIES OF STATEMENTS

The following categories of statement can be distinguished:

- **Basic plot statements**, typically with two arguments: x=*var* and y=*var*. These statements all show the data points in the data set.
- **Enhancing plot statements**, most of them also requiring the arguments x=*var* and y=*var*. Most need additional specifications, variables or constants. These statements mostly are used to annotate plots created by basic plot statements. *(I would have called them annotating statements if 'annotate' didn't already refer to another part of SAS graphics statements.)*
- Statements for **fit and confidence plots**, again having x=*var* and y=*var* as the required arguments, but producing the result of a statistical computation rather then present the values itself. Also these statements can be used together with basic plot statements and with the enhancing plot statements; in that case the fit or confidence information is usually the primary information.
- Statements for **distribution plots**, requiring only an *analysis* variable (and, because there is only one argument, without a keyword).

- Statements for **categorization plots**, requiring only a *category* variable (again without a keyword).
- **Miscellaneous plot statements** that are difficult to place in a category: POLYGON, INSET, XAXISTABLE and YAXISTABLE
- **Specification statements** that determine several aspects of the graph.
- **Utility statements** that only help in relation to other statements.

The difference between the basic and the enhancing plot statements is rather arbitrary. Most enhancing plot statements are seldomly used on itself, but rather to emphasize of enhance the graph created by a basic plot statement. But sometimes a special type of plot can be created by using only enhancing plot statements.
The statements from the first three groups can be mixed into one graph. The statements for distribution plots and categorization plots can only be mixed within that group.

DESCRIPTION OF THE STATEMENTS

BASIC PLOT STATEMENTS

The following table describes the essentials of the basic plot statements. When not noted otherwise the arguments are X=*var* and Y=*var*. Table 1 includes a small example of the statements.

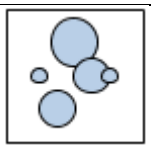
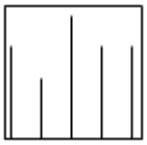
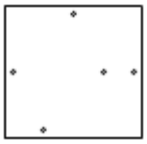
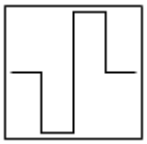
Statement	Example	About the arguments and options
Bubble		Additional required argument: `SIZE=`*`var`*. Typical options: COLORRESPONSE and related options to specify color of bubble.
Needle		Typical options: MARKERCHAR and related options to specify how the datapoints are marked.
Scatter		Typical options: MARKERCHAR and related options to specify how the datapoints are marked.
Series		Typical options: MARKERCHAR and related options to specify how the datapoints are marked. LINEATTRS to specify line appearance.
Step		As with the SERIES statement.
Vector		Origin of vector can be specified with the XORIGIN and YORIGIN options.

Table 1. Basic plot statements

ENHANCING PLOT STATEMENTS

In Table 2 the main characteristics of the 'enhancing plot statements' are summarized.

Statement	Arguments	Description
Band	X=*var* ***or*** Y=*var* UPPER=*var*|*val* LOWER=*var*|*val*	Creates a horizontal ***or*** vertical shaded band between *upper* and *lower*.
Block	X=*catvar* BLOCK=*block-var*	Creates vertical blocks marking areas with the same value for the block-var. Typical option: POSITION=TOP|CENTER|BOTTOM
Dropline	X=*var*|*val* Y=*var*|*val*	Draws a line from a data point to one or both axes. Typical option: DROPTO=BOTH|X|Y
Fringe	*numvar*	Marks the datapoints on the X-axis. Typical option: HEIGHT=*dim*<*unit*>
HighLow	X=*var* ***or*** Y=*var* HIGH=*numvar* LOW=*numvar*	Creates a horizontal ***or*** vertical line from *low* to *high*. Typical options: OPEN=*numvar* CLOSE=*numvar*
LINEPARM	X=*var*|*val* Y=*var*|*val* SLOPE=*var*|*val* (note:all numeric)	Draws a straight line. LINEATTR= option to specify appearance.
TEXT	X=*var* Y=*var* TEXT=*var*	Places *text* at the datapoint.

Table 2. Enhancing plot statements

DISTRIBUTION PLOTS

The statements for distribution plots require only an *analysis* variable (and, because there is only one argument, without a keyword). Most graphs however become only meaningful when category or group variables are added through the different options.

The DENSITY and HISTOGRAM statements show the distribution of a response (or analysis) variable (the single argument) as a continuous curve or with discrete bars.

The HBOX and VBOX statements show the distribution by creating box-and-whisker plots. The length of the box is determined by the first and third quartile, the length of the whiskers show how much values extend outside that range. Usually the CATEGORY option is used to create box-and-whiskers plots side by side for each value. As with other statements, the GROUP option can specify a variable by which to group the plots.

CATEGORIZATION PLOTS

Also the statements for the categorization plots require only a *category* variable (thus again without a keyword).

The two main statements in this group are the HBAR and VBAR statements. The single argument specifies the category variable that define the different bars. The height of the bars is by default the frequency. The RESPONSE option can be used to specify a variable to use instead, with the STAT option to define the statistic to use.

If the data is already summarized with respect to the response variable one can use the HBARPARM, resp. VBARPARM statements. Consequently these statements have two required arguments: CATEGORY=*catvar* and RESPONSE=*numvar*.

There are three statements that can add additional information to the categorization plots. HLINE and VLINE add line plots to the bars, while DOT adds single data points. These statements require the same single argument specifying the category variables, which define in which bar the dot or data point for the line lies. Options to specify which variable should define where in that bar the dot or point should be placed are almost essential to produce a meaningful result.

MISCELLANEOUS PLOT STATEMENTS

There are four plot statements that are difficult to place in one of the category above: POLYGON, INSET, XAXISTABLE and YAXISTABLE.

The POLYGON statement creates polygons for observations sharing the same value for the variable in the ID argument. This is a third required argument, next to the X= and Y= arguments. The style of polygons can be specified in a number of ways: using a response variable with a model to define how values translate into colors, specifying them directly, or through an attribute map.

The INSET statement places boxes with text inside the graph. The ODS options to format text are available for the text.

The XAXISTABLE and YAXISTABLE statements produce a row or column of values for the specified variables, one for each value on the axis.

SPECIFICATION STATEMENTS

Specification statement are used to describe different aspects of the layout of the graph.

The **STYLEATTRS** statement provide a mechanism to change aspects of the ODS style for the duration of the PROC SGPLOT, PROC SGSCATTER or PROC SGPANEL.

Important in many graphs are the axis statements. There are four axis statements: an **XAXIS** and **XAXIS2** statements that describe the bottom and top axes respectively, and an **YAXIS** and **YAXIS2** statement for the left and right axes.
The options determine a.o. the value range (or ranges), the formatting of the values at the tick marks, the appearance of the axis, etc.

The **GRADLEGEND** and **KEYLEGEND** statements specify where legends should be places and how they should look.
A GRADLEGEND statement can be used when the graph contains one or more response variables for which a colormapping is defined.
A KEYLEGEND statement is appropriate when a group variable has been used to mark data elements that belong to a group with a distinct color.

UTILITY STATEMENTS

The remaining two statements have been named ‘utility statements’ because there sole function is to assist the function of another statement. The SYMBOLCHAR and SYMBOLIMAGE link a symbol (that can be specified using Unicode syntax) or an image to an identifier that can be used as a marker.

CREATING A MAP

In all the examples one can find on the SAS website of output produced by ODS Graphics procedure there is only one showing a geographic map. Yet examples of SAS/Graph code creating a map, i.e. showing geographic regions colored according to some measure, can be found easily.

Nevertheless it is, with the introduction of the POLYGON statement in SAS 9.4 relatively easy to create a map using PROC SGPLOT (or PROC SGPANEL). This statement draws a polygon, defined by a series of points grouped together by an Id-variable, with options to specify how the values of another variable make out the color of the polygon.
And isn't a colored geographic region nothing else but a polygon filled with some color defined by a variable?

So what would be necessary to use the map datasets delivered for use with PROC GMAP in a Polygon statement in PROC SGPLOT?
Not very much, it appeared.
And would it then be possible to use all the additional options in ODS Graphics to enrich the appearance of the graph? Easily, it appeared.
In the following paragraphs of this paper I will outline the steps needed.

PREPARATION

Datasets containing the coordinates for creating maps in SAS/Graph always conform to the following specifications.

- *They have one or more variables specifying the region.* More then one variable can be used e.g. to specify counties within all states, with numeric codes for the counties that can occur in several states. A single county is then uniquely specified by the combination of code for the state and the county.
- *There are two variables for the coordinates.* Usually these are X and Y variables. Sometimes these are the results of straightforward measurements in the terrain, but more often they are the result of the geographic *projection* of latitude and longitude values. In the GfK-data sets delivered with SAS/Graph these unprojected coordinates are always available as the variables as LAT and LONG.
- *A SEGMENT variable is often present as well.* This variable is required when there are regions that consist of not connected polygons (e.g., a region that includes islands). The SEGMENT value then defines the separate polygons within the value (or combination of values) that define the region.

The conversion from latitude-longitude values to X and Y values can be done with the SAS/Graph procedure GPROJECT. Often the conversion will already have been done, so we do not always have to address that, but sometimes it is convenient to do so, and PROC GPROJECT has some side-effects that can be used to our advantage, so it will get some attention also.

An important difference between the way PROC GMAP interprets data and the way the Polygon statement of PROC SGPLOT does is how a polygon is defined. From the requirements above it follows that for PROC GMAP a polygon is defined by a unique SEGMENT value within the combination of identification values.
For the POLYGON statement to work the polygons must have a unique value across all regions.

The following simple DATA STEP accomplishes this. The example uses the data set for The Netherlands in the mapsgfk libref.

```
data corrected (
  keep = name id newseg  x y lat long
  rename = ( newseg = segment)
  )
;
  set mapsgfk.netherlands ;
  by id segment ;
  retain newseg 0 ;
  if first.segment then do ;
```

```
    newseg+1 ;
  end ;
run ;
```

In this case there is only one variable identifying the regions (ID). We create a new variable, temporarily called NEWSEG but in the resulting dataset renamed (again) to SEGMENT. (This way the new data set still can be used with PROC GMAP, where the variable *must* be called SEGMENT. This new variable is initialized to zero and incremented each time the old SEGMENT changes value within ID.

We can then already, without further tricks, use this data set with PROC SGPLOT.

```
proc sgplot data = corrected
   aspect = 1 noautolegend noborder
;
xaxis
    grid  values = ( -.025 to .025 by .01 )
    display=none
;
yaxis
  grid  values =  ( -.025 to .025 by .01 )
  display=none
;
polygon
  x = x y = y
  id = segment
/
  group = id
  nooutline fill
  dataskin = matte
;
run ;
```

The result is shown in Figure 4:

Figure 4. Provinces of The Netherlands

One of the drawbacks of this map is that the country seems to be island: there is emptiness on all sides. This of course has been the case with PROC GMAP output as well. If one would add the borders for the countries that border on The Netherlands on the east and on the south, one would get the whole of these countries in the picture – and since those countries include Germany only a small area of the graph would be used for The Netherlands. There has never been an option in PROC GMAP to include only those parts of the boundaries of neighboring countries that fit in the window that has been used for The Netherlands in the figure above.

With PROC SGPLOT one could simply use the VALUES= option on the XAXIS and YAXIS statements to limit the area that is actually shown. The data would still be in the data set, and would have to be interpreted by the procedure to determine whether it falls inside or outside the defined range.

There is however another problem to solve before one can add the boundaries from neighboring countries.
The map data sets in the GfK-libref all contain projected and unprojected coordinates. 'Unprojected' means they are in latitude and longitude form, these variables are called LAT and LONG. The names for the projected variables are X and Y. The X and Y variables can be used in a rectangular X-Y coordinate system, and are a projected approximation of the sphere the earth really is. But there are different methods to project latitude and longitude coordinates, and for each method the optimal parameters differ per area. So the X-Y coordinates for the boundary between The Netherlands and Germany in the projection preferred for The Netherlands will not exactly be the same as those for the same points projected according to the projection that is customary in Germany.

That is why we first have to append the coordinates for the Dutch provinces and the coordinates for the neighboring countries into one data set. Then they can be projected with PROC GPROJECT in one step, all according to the Dutch method and parameters. The full code for these steps can be found in the appendix.

The current version of PROC GPROJECT has the option to specify method and parameters in a TO parameter, using the internationally adopted EPSG numbering scheme. So you only have to know the EPSG code for your preferred projection, and you don't have to know the details of the projection method and the 5 or more parameters (with a required precision of many digits). This very useful option was added at some point in time without any publicity. In earlier version one had to specify the method and all the parameters explicitly.
(The procedure also has a FROM parameter, so it possible to go from any projection to any other projection, and back.)

The GPROJECT procedure also has the possibility to clip the output, by specifying minimum and maximum values for the input coordinates. In this case these are the latitude and longitude values. After converting to X-Y coordinates the clipped area would show as bended lines, like the meridians on a world map. This is usually what is preferred. In the example the clipping is done through the VALUES option on the AXIS statement.

An alternative would be to call PROC GPROJECT a second time, this time *without* options that ask for a projection, thus only invoking the clipping. Also this is a useful, yet not advertised, function of the procedure.

Where to clip

While covering the subject 'how to clip' we haven't yet addressed the question *where* to clip. We want to show the details of a particular area (in this case: The Netherlands), with some surroundings. If we would take just the minimum and maximum X and Y values for that area (the so called *bounding box*) we would get very little surrounding detail. So the a margin is created around that bounding box, at the same time making the new bounding box conform to the aspect ratio of the picture we want. This is accomplished by the macro `%AdjustFrame`, included in the code in the appendix.

There is one final step before turning the data over to PROC SGPLOT: specifying the way the different regions are colored.

If there is some measure to display one could leave that to PROC SGPLOT by specifying the variable in question, together with a color model that determines how to translate the measurement values into a color. In this example however we have limited this to two colors that are specified through an attribute map. The data set containing the attribute map specifies only two colors: ‘salmon’ for the provinces in The Netherlands, and ‘silver’ for the countries around The Netherlands.
When a response variable is being used one could also use this attribute map to enforce a coloring scheme that cannot be accomplished by the standard options of the POLYGON statement.

Fout! Verwijzingsbron niet gevonden. shows the result of the whole exercise.

CONCLUSION

The ODS Graphics procedures provide a very powerful tool to create your graphics and apply a consistent style to them. There are several ways to get your results, interactively or in batch, which also can be combined to profit from the advantages of both. This is both a strength and a potential weakness: much is possible, but which method do you choose and where and how do you begin?

Although not advertised as such, ODS Graphics also provides the tools to produce the choropleth graphs from PROC GMAP, bringing the ODS graphic styles to the maps.

For specific tasks like geographic projection the SAS/Graph procedures remain essential utilities.

ACKNOWLEDGMENTS

This is not the first attempt to provide SAS users, and particularly those with SAS/Graph experience, with an introduction to ODS Graphics. Personally I have found helpful suggestions and examples on the web site of Robert Allison (http://robslink.com), and useful overviews in the paper by Susan J. Slaughter and Lora D. Delwiche: *"Using PROC SGPLOT for Quick, High-Quality Graphs"* (http://www.wuss.org/proceedings14/47_Final_Paper_PDF.pdf).

CONTACT INFORMATION

Comments, suggestions, questions and tips are very welcome! You can reach me at:

Frank Poppe
PW Consulting, the Netherlands
+31 6 2264 0854
Frank.Poppe *at* PWconsulting *dot* NL
www.pwconsulting.nl