SAS/STAT® 15.1 User's Guide

Statistical Graphics Using ODS

This document is an individual chapter from *SAS/STAT® 15.1 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2018. *SAS/STAT® 15.1 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/STAT® 15.1 User's Guide

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

November 2018

Chapter 21

Statistical Graphics Using ODS

Contents

Introduction

Effective graphics are indispensable in modern statistical analysis. They reveal patterns, differences, and uncertainty that are not readily apparent in tabular output. Graphics provoke questions that stimulate deeper investigation, and they add visual clarity and rich content to reports and presentations. Statistical graphs are produced by ODS Graphics, which is an extension of ODS (the Output Delivery System). ODS manages procedure output (including both tables and graphs) and lets you display it in a variety of destinations, such as HTML and RTF. With ODS Graphics, statistical procedures produce graphs as automatically as they produce tables, and graphs are integrated with tables in the ODS output. ODS Graphics is available in procedures in SAS/STAT, Base SAS, SAS/ETS, SAS/QC, and other SAS products (see the section "Procedures That Support ODS Graphics" on page 631). ODS Graphics is automatically provided with Base SAS software.

ODS Graphics might or might not be enabled by default, depending on your operating system, whether you are in the SAS windowing environment, your registry, your system options, and your configuration file settings. For more information about default settings and enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" on page 623.

You can enable ODS Graphics by specifying the following statement:

```
ods graphics on;
```

When ODS Graphics is enabled, procedures that support ODS Graphics create the appropriate graphs, either by default or when you specify procedure options to request specific graphs. These options are documented in the "Syntax" section of each procedure chapter, and the "Details" section of each chapter provides an "ODS Graphics" subsection that lists the available graphs. Once ODS Graphics is enabled, it stays enabled for the duration of your SAS session unless you disable it.

You can disable ODS Graphics by specifying the following statement:

```
ods graphics off;
```

You might consider disabling ODS Graphics if your goal is solely to produce computational results. Often, you can enable ODS Graphics and then leave it enabled. Throughout this chapter, ODS Graphics is enabled only once per section.

Chapter Reading Guide

This chapter provides a basic introduction to ODS Graphics along with more detailed information. The following list provides a guide to reading this chapter:

- If you want to see a few of the many graphs that statistical procedures produce by using ODS Graphics, see the section "Getting Started with ODS Statistical Graphics" on page 606.
- If you are using ODS Graphics for the first time, read the section "A Primer on ODS Statistical Graphics" on page 622, which provides the minimum information that you need to get started.
- If you need to create plots of raw data or your own customized plots of statistical results, see the section "Statistical Graphics Procedures" on page 715, which describes SAS procedures that use ODS Graphics.

- If you need information about specialized topics such as accessing your graphs, making changes to your graphs, and working with ODS styles, see the section "Syntax" on page 632 and the sections "Examples of ODS Statistical Graphics" on page 724, "Selecting and Viewing Graphs" on page 638, "Graphic Image Files" on page 644, "Graph Size and Resolution" on page 650, "ODS Graphics Editor" on page 651, "The Default Template Stores and the Template Search Path" on page 655, "ODS Styles" on page 657, and "Examples of ODS Statistical Graphics" on page 724.

If you are unfamiliar with ODS, see Chapter 20, "Using the Output Delivery System," for an introduction. For complete documentation about the Output Delivery System, see the *SAS Output Delivery System: User's Guide*. For an introduction to graph template modification, see Chapter 22, "ODS Graphics Template Modification." For more information about modifying the Kaplan and Meier (1958) plot in PROC LIFETEST, see Chapter 23, "Customizing the Kaplan-Meier Survival Plot." For an introduction to ODS Graphics, ODS styles, the Graph Template Language, the style template language, the statistical graphics procedures, and graph template modification, see Kuhfeld (2016). For complete documentation about ODS graph templates, see the *SAS Graph Template Language: User's Guide* and the *SAS Graph Template Language: Reference*. For complete documentation about the ODS Graphics Editor, see the *SAS ODS Graphics Editor: User's Guide*. Also see the *SAS ODS Graphics: Procedures Guide* for information about the statistical graphics procedures and Kuhfeld (2015) for advance examples.

Assumptions about ODS Defaults in This Chapter

Default settings such as destinations and whether or not ODS Graphics is enabled vary depending on your operating system, registry settings, configuration file settings, and system options and whether you are using the SAS windowing environment or batch mode. For this reason, this chapter makes no assumptions about these defaults. Instead, destinations are often explicitly closed without assuming which destination (usually LISTING or HTML) is open, destinations are explicitly opened when needed, and ODS Graphics is explicitly enabled and disabled as needed. In some examples, when all destinations are closed, the LISTING destination is opened at the end of the step so that some destination is available for subsequent output. If you know the defaults for your environment, you do not need to use many of the ODS statements that are used in this chapter.

Getting Started with ODS Statistical Graphics

This section provides examples that illustrate the most basic uses of ODS Graphics by showing a few of the many plots that are produced by statistical procedures.

Default Plots for Simple Linear Regression with PROC REG

This example is based on the section "Getting Started: REG Procedure" on page 8432 in Chapter 102, "The REG Procedure." The Class data set that this example uses is available in the Sashelp library. The following statements use PROC REG to fit a simple linear regression model in which Weight is the response variable and Height is the independent variable:

```
ods graphics on;

proc reg data=sashelp.class;
   model Weight = Height;
quit;
```

The ODS GRAPHICS ON statement requests ODS Graphics in addition to the usual tabular output. The statement ODS GRAPHICS OFF is not used here, but it can be specified to disable ODS Graphics.

The graphical output consists of a fit diagnostics panel, a residual plot, and a fit plot. These plots are integrated with the tabular output and are shown in Figure 21.1, Figure 21.2, and Figure 21.3, respectively.

The results are displayed in the HTMLBLUE style. ODS styles control the colors and general appearance of all graphs and tables, and the SAS System provides several styles that are recommended for use with statistical graphics. The default style that you see when you run SAS depends on the ODS destination, system options, and SAS registry settings. For more information about styles, see the section "ODS Styles" on page 624 and the section "ODS Styles" on page 657.

Figure 21.1 Fit Diagnostics Panel

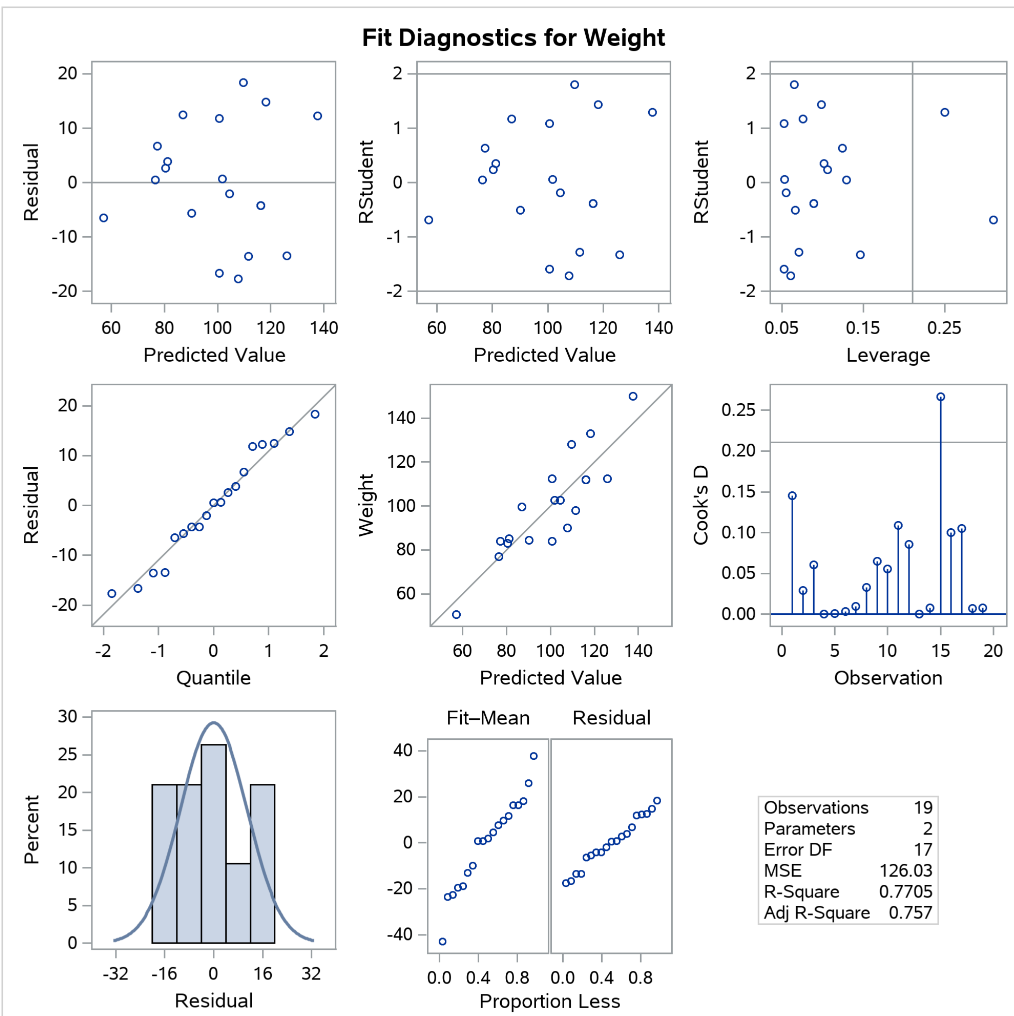


Figure 21.2 Residual Plot

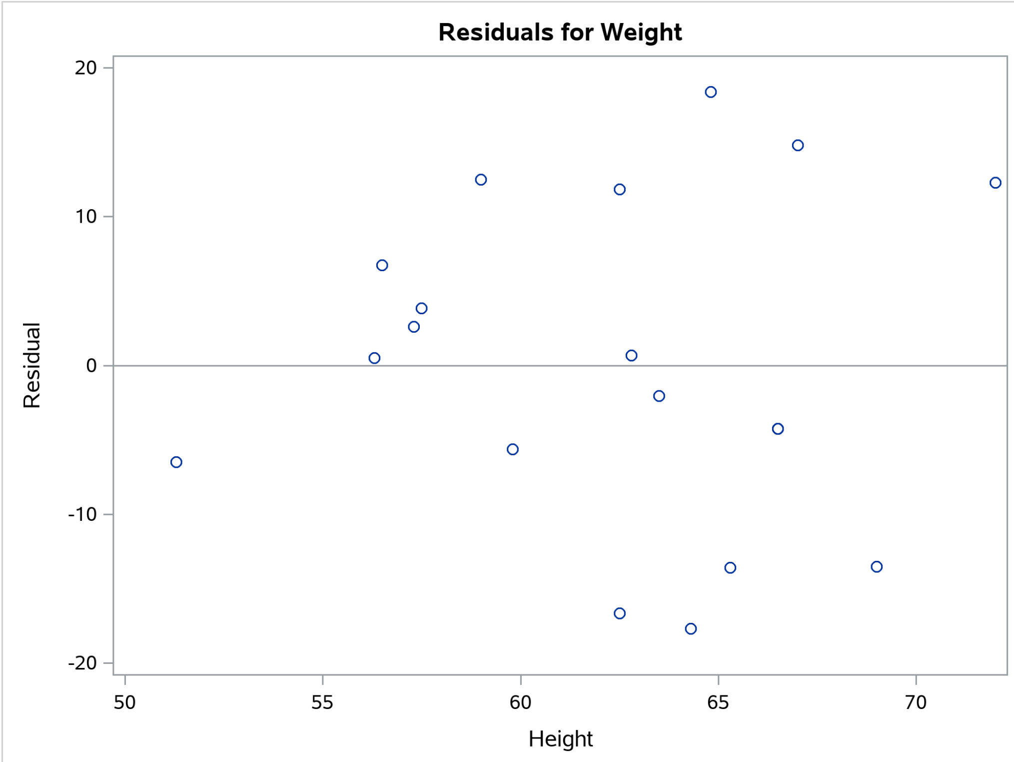
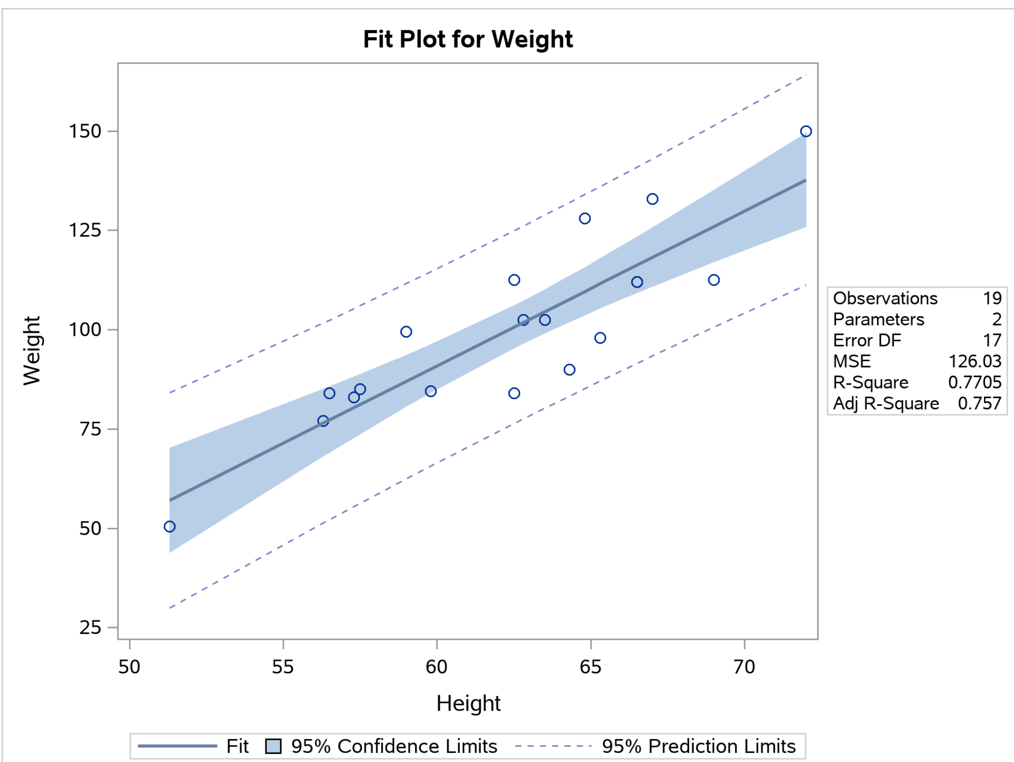


Figure 21.3 Fit Plot



Survival Estimate Plot with PROC LIFETEST

This example is taken from Example 74.2 in Chapter 74, "The LIFETEST Procedure." It shows how to construct a product-limit survival estimate plot. Both the ODS GRAPHICS statement and procedure options are used to request the plot. This examples uses the bone marrow transplant data set, which is available from the Sashelp library. The data set contains disease-free times for three risk categories.

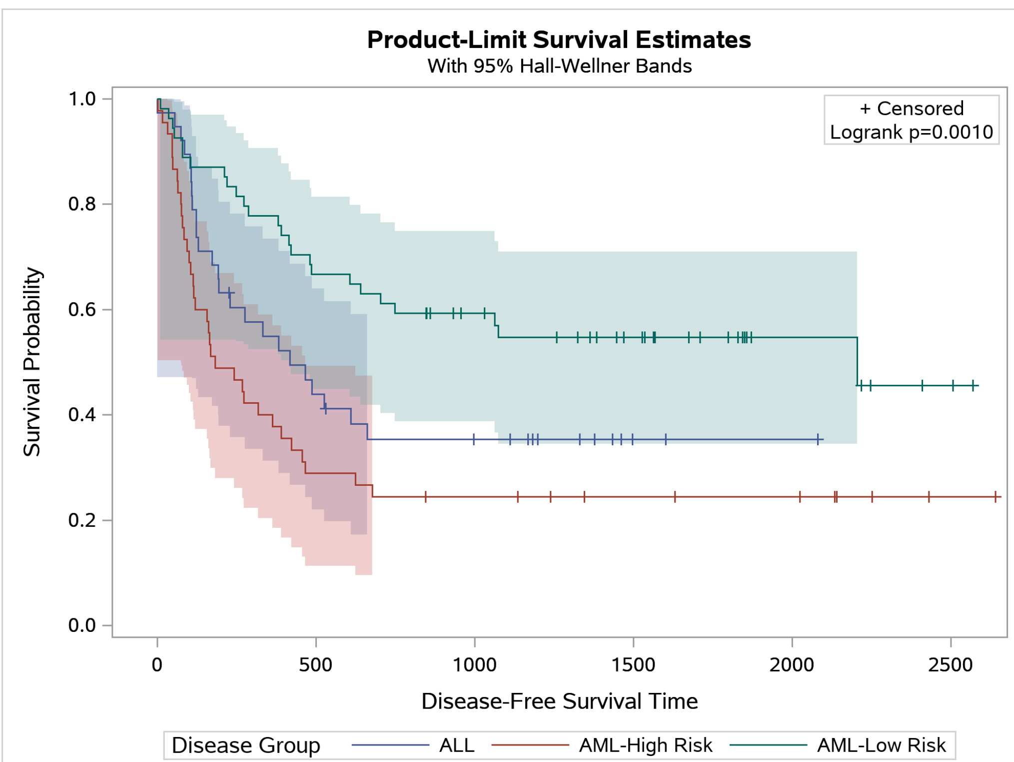
The following statements use PROC LIFETEST to compute the product-limit estimate of the survivor function for each risk category:

```
proc lifetest data=sashelp.BMT plots=survival(cb=hw test);
   time T * Status(0);
   strata Group / test=logrank;
run;
```

ODS Graphics was enabled in a previous step, and the PLOTS=SURVIVAL option requests a plot of the estimated survival curves. The CB=HW suboption requests Hall-Wellner confidence bands, and the TEST suboption displays the *p*-value for the log-rank test in a plot inset. For more information about modifying the Kaplan and Meier (1958) plot in PROC LIFETEST, see Chapter 23, "Customizing the Kaplan-Meier Survival Plot."

Figure 21.4 displays the plot. Patients in the AML-Low Risk group are disease-free longer than those in the ALL group, who in turn fare better than those in the AML-High Risk group.

Figure 21.4 Survival Plot



Contour and Surface Plots with PROC KDE

This example is taken from the section "Getting Started: KDE Procedure" on page 5322 in Chapter 70, "The KDE Procedure." Here, in addition to the ODS GRAPHICS statement, procedure options are used to request plots. The following statements simulate 1,000 observations from a bivariate normal density that has means (0,0), variances (10,10), and covariance 9:

```
data bivnormal;
   do i = 1 to 1000;
      z1 = rannor(104);
      z2 = rannor(104);
      z3 = rannor(104);
      x  = 3*z1+z2;
      y  = 3*z1+z3;
      output;
   end;
run;
```

The following statements request a bivariate kernel density estimate for the variables x and y:

```
proc kde data=bivnormal;
   bivar x y / plots=contour surface;
run;
```

The PLOTS= option requests a contour plot and a surface plot of the estimate (displayed in Figure 21.5 and Figure 21.6, respectively). For more information about the graphs available in PROC KDE, see the section "ODS Graphics" on page 5347 in Chapter 70, "The KDE Procedure."

Figure 21.5 Contour Plot of Estimated Density

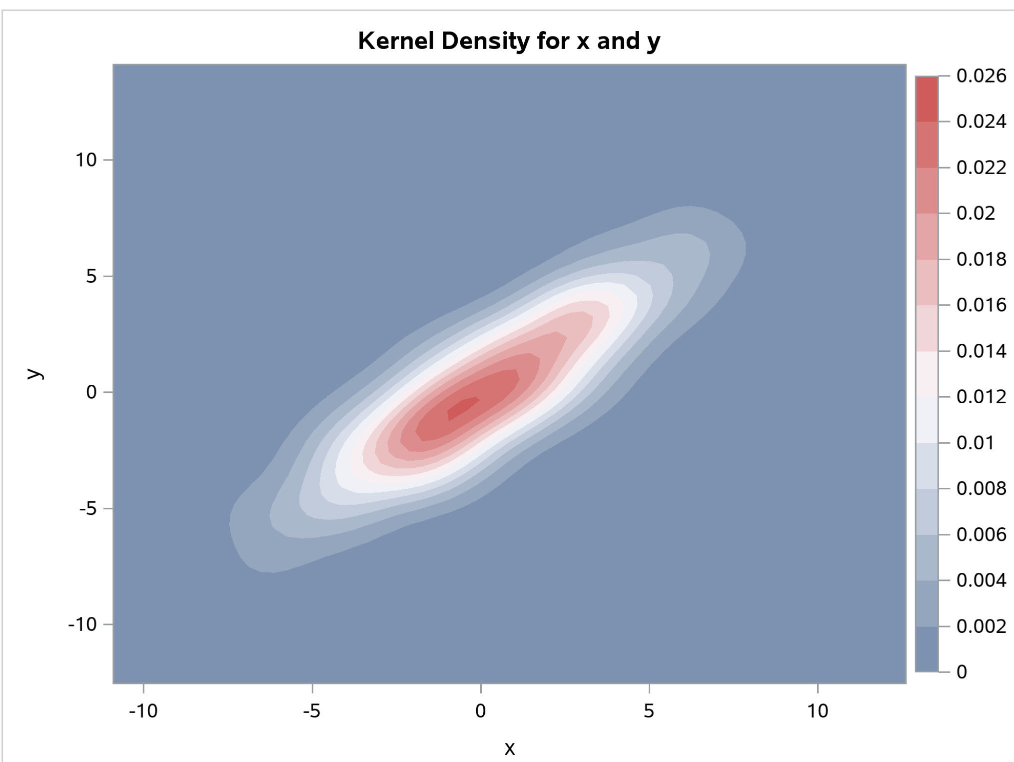
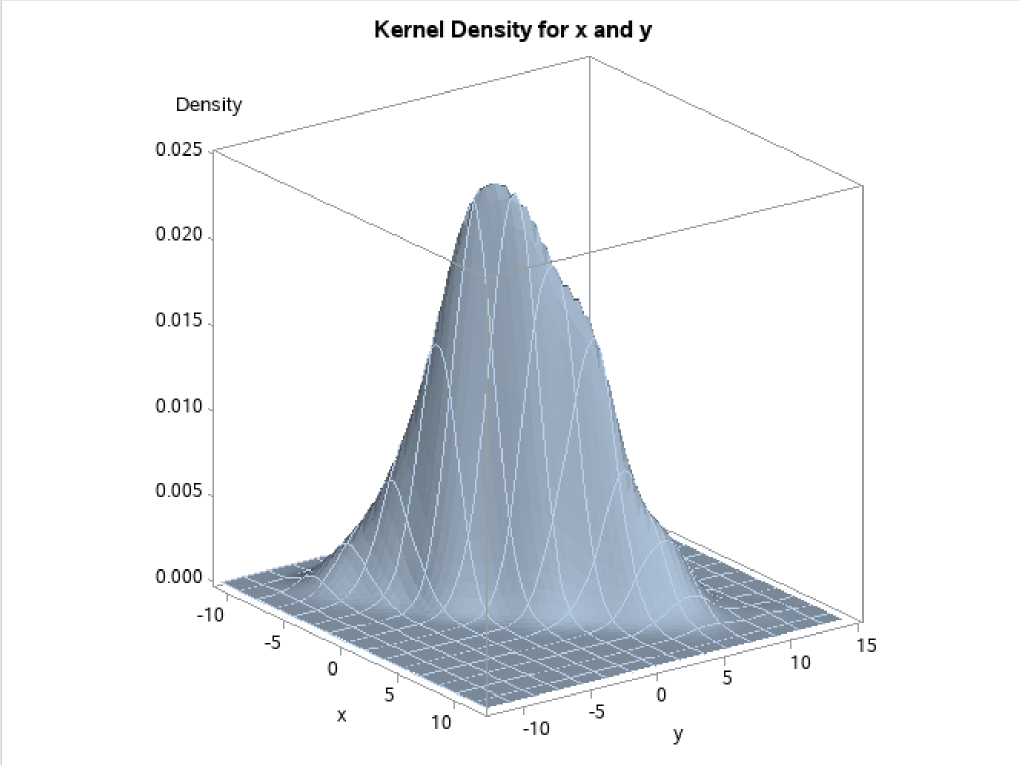


Figure 21.6 Surface Plot of Estimated Density



Contour Plots with PROC KRIGE2D

This example is taken from Example 71.2 in Chapter 71, "The KRIGE2D Procedure." The coal seam thickness data set is available from the Sashelp library. The following statements create a SAS data set that contains a copy of these data along with some artificially added missing data:

```
data thick;
   set sashelp.thick;
   if _n_ in (41, 42, 73) then thick = .;
run;
```

The following statements run PROC KRIGE2D:

```
proc krige2d data=thick outest=predictions
             plots=(observ(showmissing)
                    pred(fill=pred line=pred obs=linegrad)
                    pred(fill=se line=se obs=linegrad));
   coordinates xc=East yc=North;
   predict var=Thick r=60;
   model scale=7.2881 range=30.6239 form=gauss;
   grid x=0 to 100 by 2.5 y=0 to 100 by 2.5;
run;
```

The PLOTS=OBSERV(SHOWMISSING) option produces a scatter plot of the data along with the locations of any missing data. The PLOTS=PRED option produces maps of the kriging predictions and standard errors. Two instances of the PLOTS=PRED option are specified along with suboptions that customize the plots. The results are shown in Figure 21.7.

Figure 21.7 Spatial Distribution

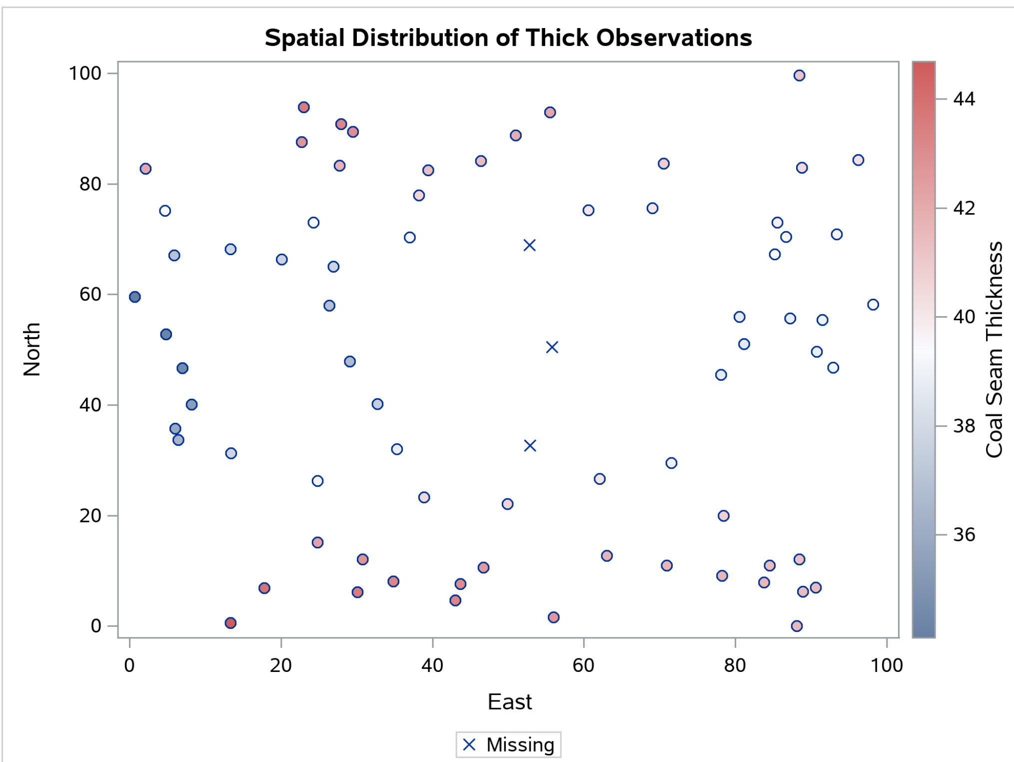
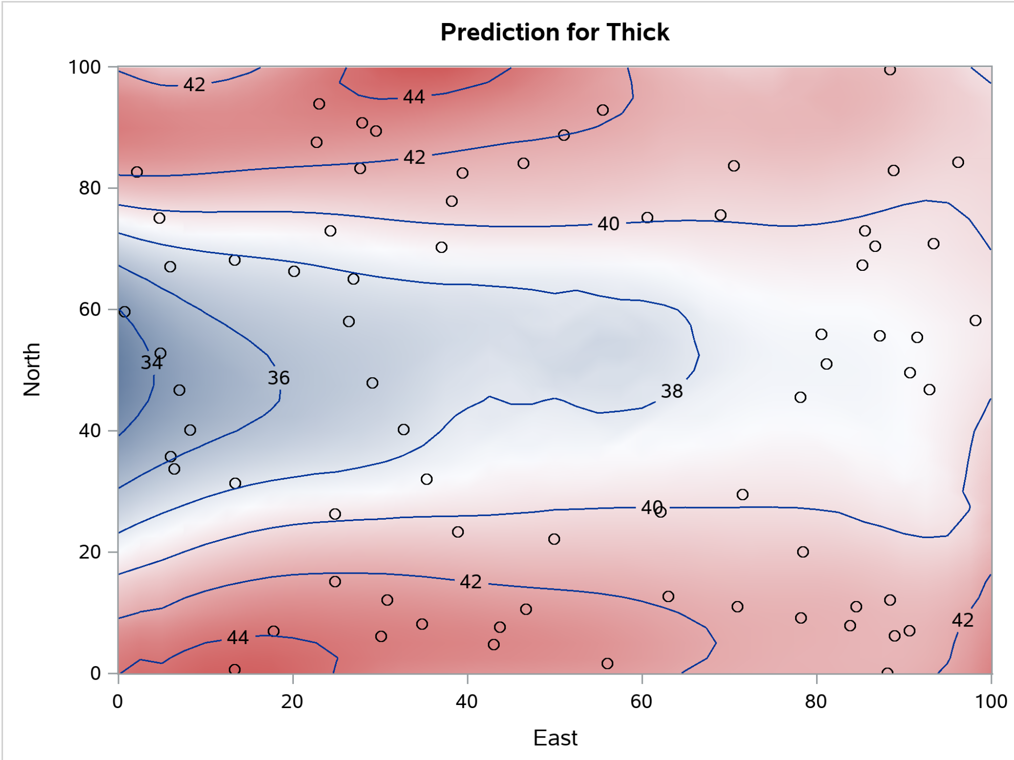
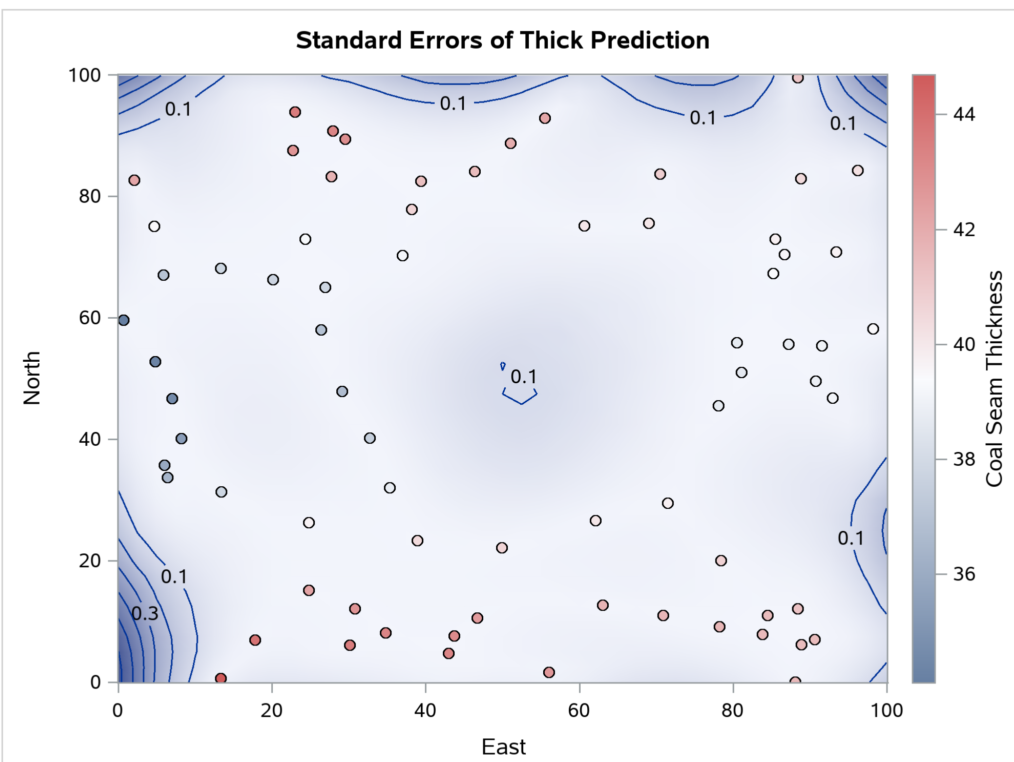


Figure 21.7 *continued*





Partial Least Squares Plots with PROC PLS

This example is taken from the section "Getting Started: PLS Procedure" on page 7583 in Chapter 92, "The PLS Procedure." The following statements create a SAS data set that contains measurements of biological activity in the Baltic Sea:

```
data Sample;
   input obsnam $ v1-v27 ls ha dt @@;
   datalines;
EM1    2766 2610 3306 3630 3600 3438 3213 3051 2907 2844 2796
       2787 2760 2754 2670 2520 2310 2100 1917 1755 1602 1467
       1353 1260 1167 1101 1017         3.0110  0.0000   0.00
EM2    1492 1419 1369 1158  958  887  905  929  920  887  800

   ... more lines ...

;
```

The following statements run PROC PLS:

```
proc pls data=sample cv=split cvtest(seed=104);
   model ls ha dt = v1-v27;
run;
```

By default, the procedure produces a plot for the cross validation analysis and a correlation loading plot (see Figure 21.8).

Figure 21.8 Partial Least Squares

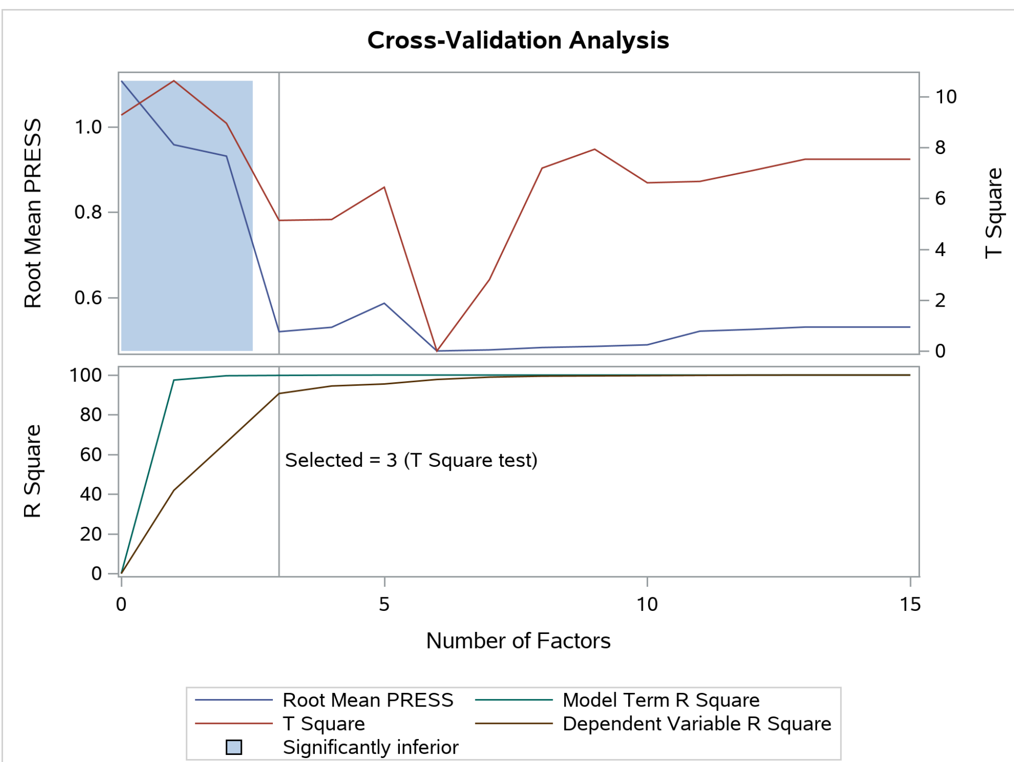
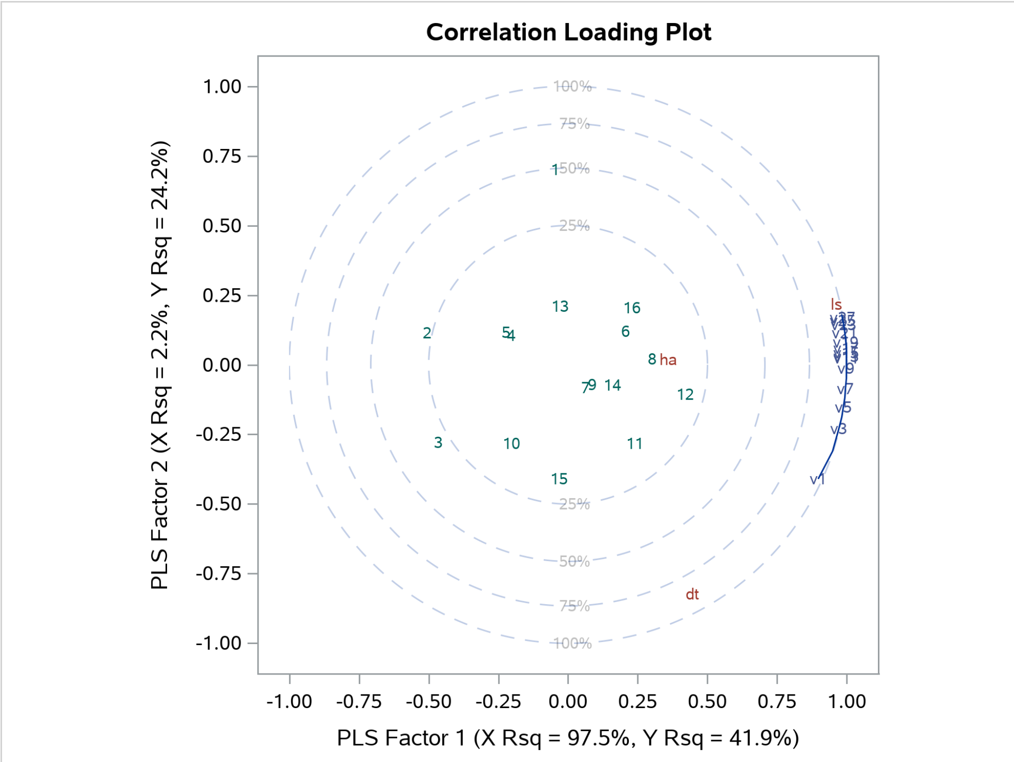


Figure 21.8 *continued*



Box-Cox Transformation Plot with PROC TRANSREG

This example is taken from Example 123.2 in Chapter 123, "The TRANSREG Procedure." The following statements create a SAS data set that contains failure times for yarn:

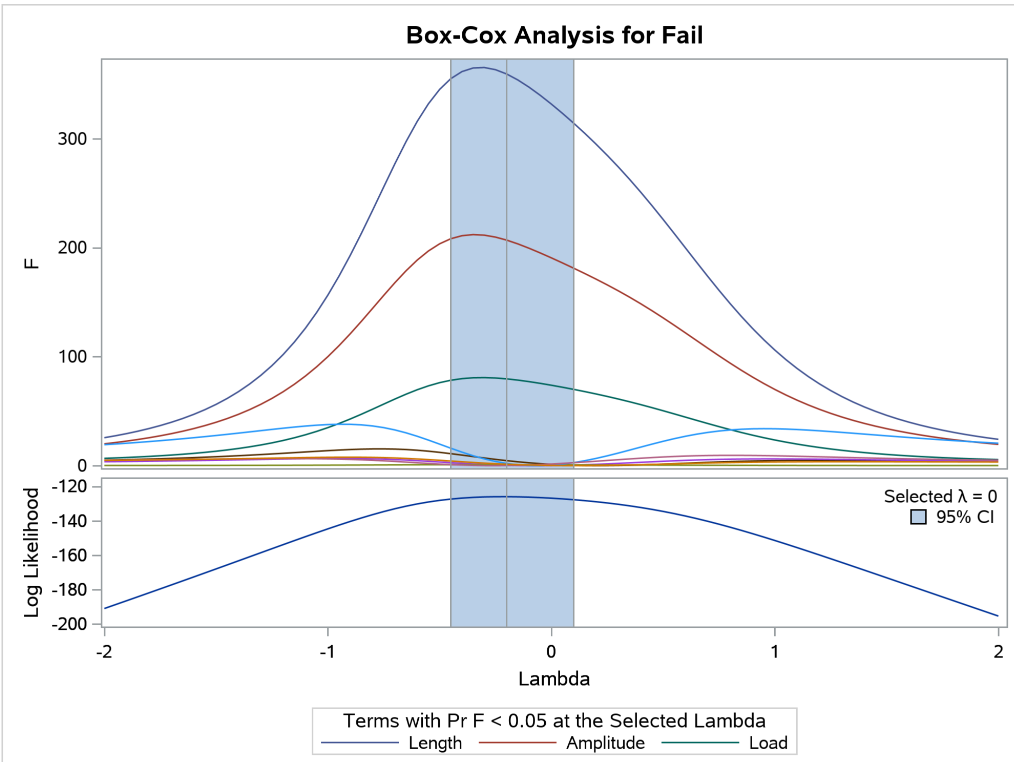
```
proc format;
   value a -1 =   8 0 =   9 1 =  10;
   value l -1 = 250 0 = 300 1 = 350;
   value o -1 =  40 0 =  45 1 =  50;
run;

data yarn;
   input Fail Amplitude Length Load @@;
   format amplitude a. length l. load o.;
   label fail = 'Time in Cycles until Failure';
   datalines;
 674 -1 -1 -1     370 -1 -1  0     292 -1 -1  1     338  0 -1 -1
 266  0 -1  0     210  0 -1  1     170  1 -1 -1     118  1 -1  0

   ... more lines ...

;
```

The following statements run PROC TRANSREG:

```
proc transreg data=yarn;
   model BoxCox(fail / convenient lambda=-2 to 2 by 0.05) =
         qpoint(length amplitude load);
run;
```

The log-likelihood plot in Figure 21.9 suggests a Box-Cox transformation where $\lambda = 0$.

Figure 21.9 Box-Cox “Significant Effects”



LS-Means Diffogram with PROC GLIMMIX

This example is taken from the section “Graphics for LS-Mean Comparisons” on page 3819 in Chapter 49, “The GLIMMIX Procedure.” The following statements create a SAS data set that contains measurements from an experiment that investigates how snapdragons grow in various soils:

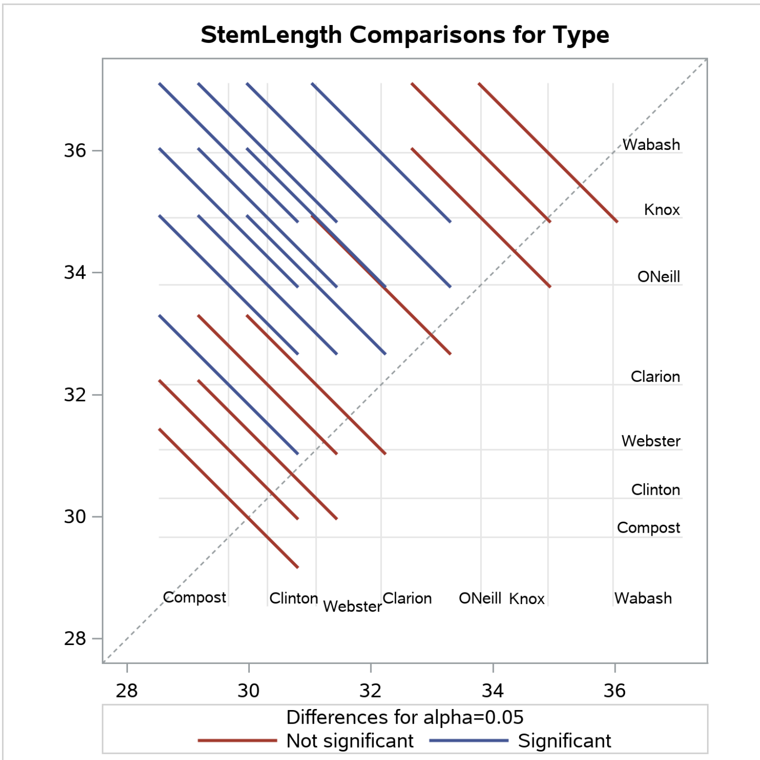
```
data plants;
   input Type $ @;
   do Block = 1 to 3;
      input StemLength @@;
      output;
   end;
   datalines;
Clarion    32.7 32.3 31.5   Clinton    32.1 29.7 29.1   Knox       35.7 35.9 33.1
ONeill     36.0 34.2 31.2   Compost    31.8 28.0 29.2   Wabash     38.2 37.8 31.9
Webster    32.5 31.1 29.7
;
```

The following statements run PROC GLIMMIX:

```
proc glimmix data=plants order=data plots=diffogram;
   class Block Type;
   model StemLength = Block Type;
   lsmeans Type;
run;
```

The PLOTS=DIFFOGRAM option produces a diffogram, shown in Figure 21.10, that displays all the pairwise least squares mean differences and indicates which are significant.

Figure 21.10 LS-Means Diffogram



Principal Component Analysis Plots with PROC PRINCOMP

This example is taken from Example 95.3 in Chapter 95, "The PRINCOMP Procedure." The following statements create a SAS data set that contains job performance ratings of police officers:

```
options validvarname=any;

data Jobratings;
   input ('Communication Skills'n
          'Problem Solving'n
          'Learning Ability'n
          'Judgment Under Pressure'n
          'Observational Skills'n
```

```
             'Willingness to Confront Problems'n
             'Interest in People'n
             'Interpersonal Sensitivity'n
             'Desire for Self-Improvement'n
             'Appearance'n
             'Dependability'n
             'Physical Ability'n
             'Integrity'n
             'Overall Rating'n) (1.);
   datalines;
26838853879867
74758876857667

   ... more lines ...

;
```

The following statements run PROC PRINCOMP:

```
proc princomp data=Jobratings(drop='Overall Rating'n) n=2
              plots=(Matrix PatternProfile);
run;
```

The plots are requested by the PLOTS=(MATRIX PATTERNPROFILE) option. The results, shown in Figure 21.11, contain the default scree and variance-explained plots, along with a scatter plot matrix of component scores and a pattern profile plot.

Figure 21.11 Principal Component Analysis

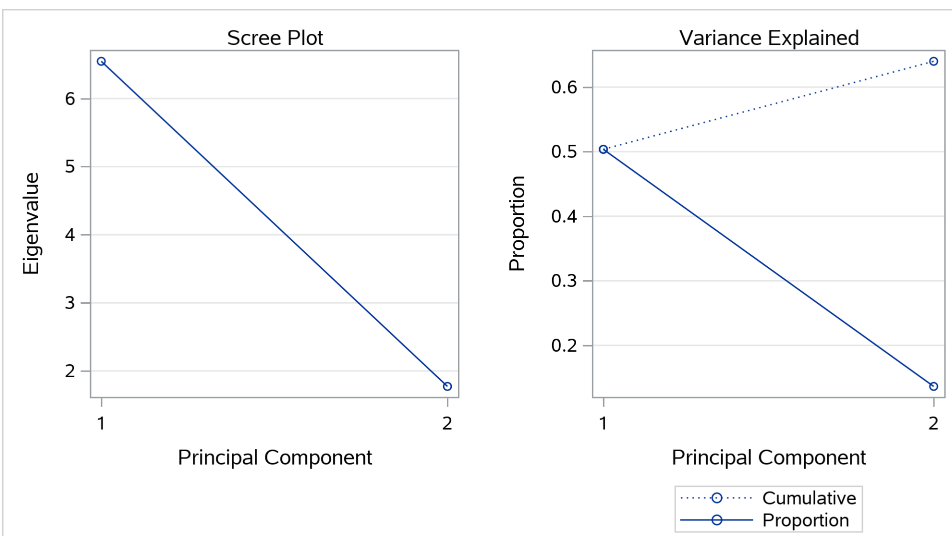


Figure 21.11 *continued*

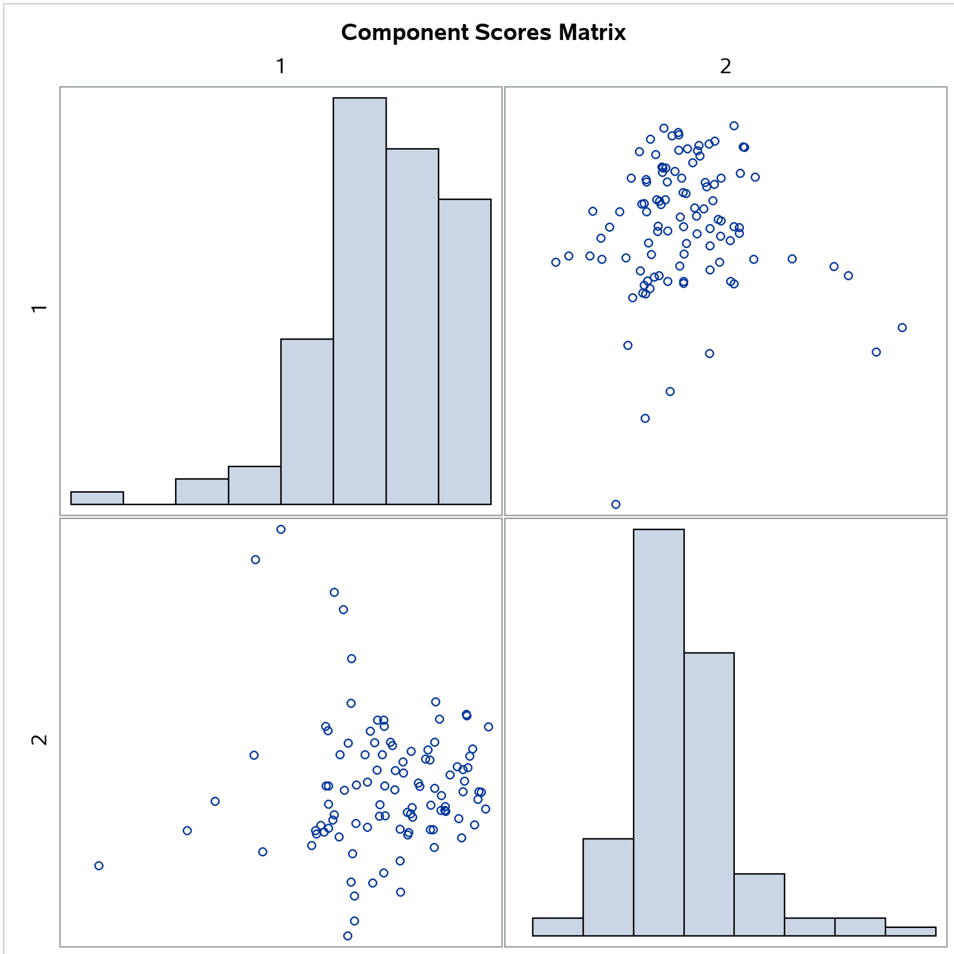
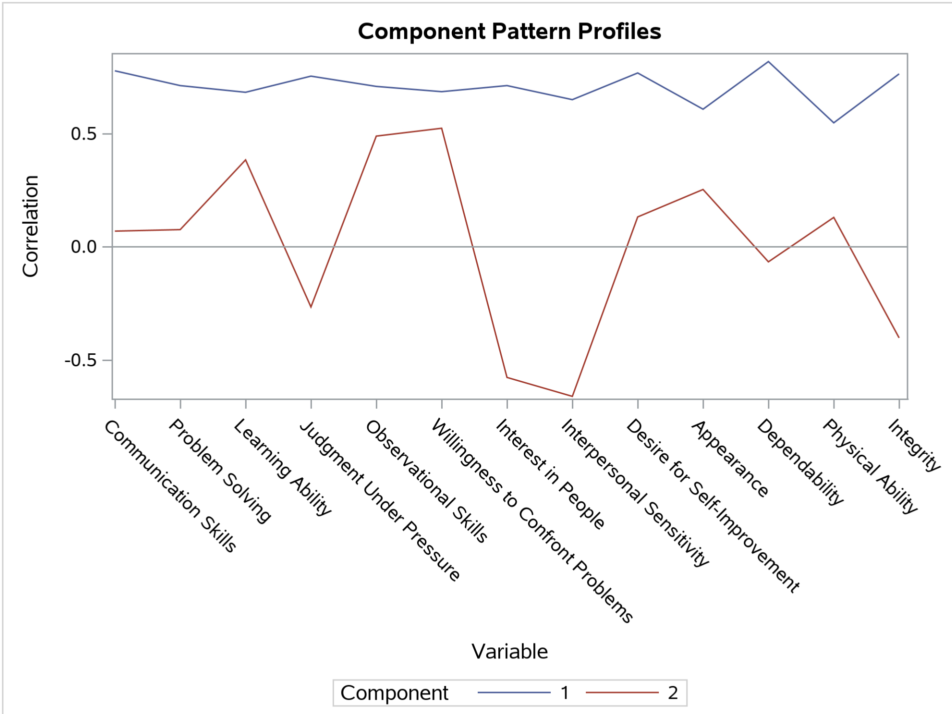


Figure 21.11 *continued*



Grouped Scatter Plot with PROC SGPLOT

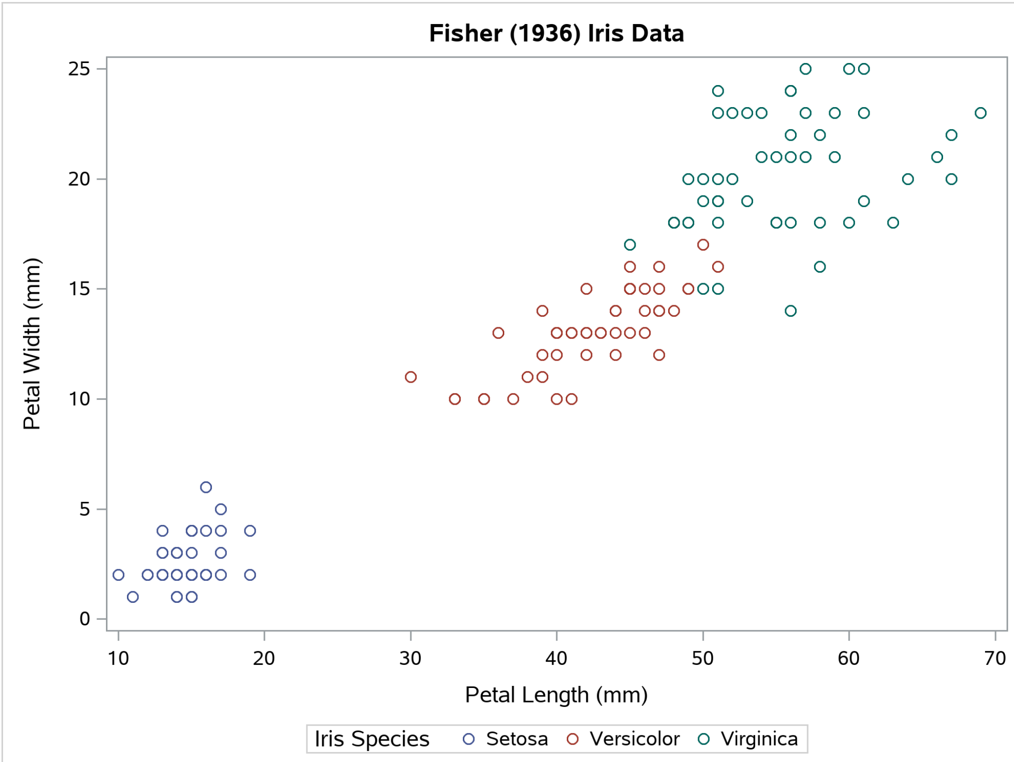
This example is taken from Example 39.1 in Chapter 39, “The DISCRIM Procedure.” It uses the Fisher iris data set, which is available from the Sashelp library.

The following statements run PROC SGPLOT to make a scatter plot, grouped by iris species:

```
proc sgplot data=sashelp.iris;
   title 'Fisher (1936) Iris Data';
   scatter x=petallength y=petalwidth / group=species;
run;
```

The results are shown in Figure 21.12.

Figure 21.12 Iris Data

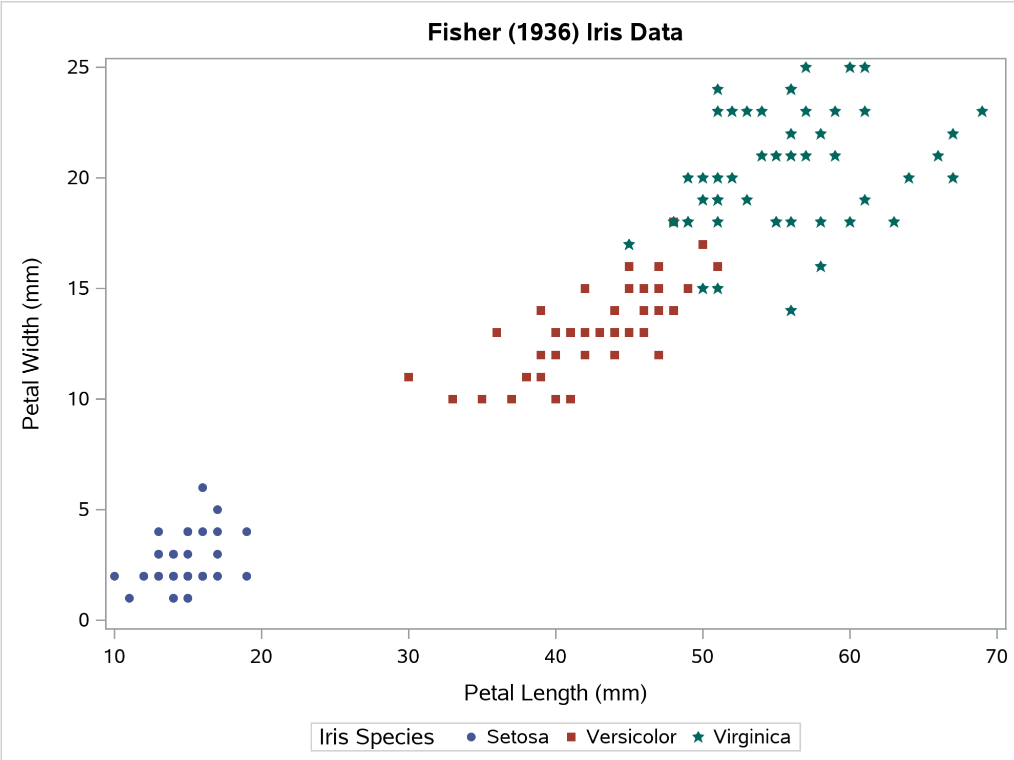


For more information about PROC SGPLOT (statistical graphics plot) and other SG procedures, see the section "Statistical Graphics Procedures" on page 715 and the *SAS ODS Graphics: Procedures Guide*. You do not need to enable ODS Graphics in order to use SG procedures (because making plots by using ODS Graphics is their sole function). However, you can use the ODS GRAPHICS statement to change options, as in the following example:

```
ods graphics on / attrpriority=none;

proc sgplot data=sashelp.iris;
   title 'Fisher (1936) Iris Data';
   styleattrs datasymbols=(circlefilled squarefilled starfilled);
   scatter x=petallength y=petalwidth / group=species markerattrs=(size=5px);
run;

ods graphics / reset;
```

The STYLEATTRS statement specifies the symbols for the three groups of observations. The results are shown in Figure 21.13. The groups in Figure 21.13 are distinguished by varying colors and symbols. In contrast, the groups in Figure 21.12 are distinguished only by colors. The ATTRPRIORITY=NONE option in the ODS GRAPHICS statement ensures that the second and third symbols are used. The HTMLBLUE style is an ATTRPRIORITY='Color' style, so by default, the symbol and line style attributes are not used to distinguish groups. The ATTRPRIORITY=NONE option in the ODS GRAPHICS statement overrides the ATTRPRIORITY='Color' option in the style, so that all three symbols are used. For more information, see the section "Attribute Priority and Overriding How Groups Are Distinguished" on page 660.

Figure 21.13 Iris Data Controlling the Symbols



A Primer on ODS Statistical Graphics

You can enable ODS Graphics by specifying the following statement:

```
ods graphics on;
```

ODS Graphics remains enabled for all procedure steps until you disable it by submitting the following statement:

```
ods graphics off;
```

When ODS Graphics is enabled, creating graphical output with procedures is as simple as creating tabular output. For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" on page 623. For more information about the most commonly used ODS GRAPHICS statement options, see the section "Syntax" on page 632.

You can control your output by using the following ODS components:

- ODS destination statements (such as ODS HTML or ODS RTF), which specify where you want your graphs to be displayed. For an example of HTML output, see Figure 21.24. For a list of the supported destinations, see the section "ODS Destination Statements" on page 635. For more information about the most commonly used ODS destination statement options, see the section "Syntax" on page 632 .
- ODS SELECT and ODS EXCLUDE statements, which select and exclude graphs from your output. See the section "Selecting and Excluding Graphs" on page 642 for an example of how to select graphs.
- ODS OUTPUT statements, which create SAS data sets from the data object that is used to make the plot. See the section "Specifying an ODS Destination for Graphics" on page 638 for an example.
- procedure options, which specify what graphs to create. For each procedure, these options are described in the "Syntax" section of the procedure chapter. Usually, you use the PLOTS= option to control all graphs. The available graphs are listed in the "ODS Graphics" section, which is found in the "Details" section of each procedure chapter. Many graphs are produced by default.
- ODS style templates, which control the general appearance and consistency of all graphs and tables. You can modify the styles that SAS provides and override style information in several ways. For more information about styles, see the sections "ODS Styles" on page 624 and "ODS Styles" on page 657.
- ODS graph templates, which modify the layout and details of each graph. For more information about graph templates, see the section "Graph Templates" on page 782 in Chapter 22, "ODS Graphics Template Modification."

  NOTE: SAS provides a default template for each graph, so you do not need to know anything about templates in order to create statistical graphics.

You can also access individual graphs, control the resolution and size of graphs, and modify your graphs (as explained in the sections beginning with "Selecting and Viewing Graphs" on page 638). Alternatively, you can use special statistical graphics procedures to create custom graphs directly (see the section "Statistical Graphics Procedures" on page 715).

Enabling and Disabling ODS Graphics

You can enable ODS Graphics by specifying the following statement:

```
ods graphics on;
```

ODS Graphics remains enabled for all procedure steps until you disable it by submitting the following statement:

```
ods graphics off;
```

ODS Graphics might or might not be enabled by default. This depends on a number of factors. ODS Graphics is usually enabled by default in the SAS windowing environment; ODS Graphics is usually disabled by default when you invoke SAS in other ways. However, you can change these defaults in a number of ways. You can enable or disable ODS Graphics by default in an *autoexec.sas* file, a configuration file such as *SASV9.CFG*, or the SAS registry. You can change the default in the SAS windowing environment by selecting **Tools ► Options ► Preferences** from the main SAS window. Then on the **Results** tab, select the **Use ODS Graphics** check box to enable ODS Graphics by default, or clear the check box to disable ODS Graphics by default. You can also change the default output destination (HTML or LISTING) on the **Results** tab. For more information about default ODS Graphics settings and default destinations, see the section "HTML Output in the SAS Windowing Environment" on page 532 in Chapter 20, "Using the Output Delivery System."

When ODS Graphics is enabled, procedures that support ODS Graphics create graphs, either by default or when you specify procedure options for requesting specific graphs. Often, you can leave ODS Graphics enabled for the duration of your SAS session. However, you might consider disabling ODS Graphics if your goal is solely to produce computational results, particularly for large data sets or with many BY groups.

ODS Styles

ODS styles control the overall appearance of graphs and tables. They specify colors, fonts, line styles, symbol markers, and other attributes of graph elements. There are two types of ODS styles:

- ATTRPRIORITY='Color' style, which distinguishes groups of observations by color changes and not by line style or symbol changes.[1] ATTRPRIORITY='Color' styles include HTMLBLUE, PEARL, PEARLJ, and SAPPHIRE. If you want to control the markers or lines that are displayed for groups of observations when using an ATTRPRIORITY='Color' style, be sure to first specify the ATTRPRIORITY=NONE option in the ODS GRAPHICS statement or switch to an ATTRPRIORITY='None' style. For more information, see the sections "Attribute Priority and Overriding How Groups Are Distinguished" on page 660.
- ATTRPRIORITY='None' style, which distinguishes groups of observations by simultaneous color, marker, and line changes. Most ODS styles are ATTRPRIORITY='None' styles. They are compromise styles in the sense that some graph elements are intentionally overdistinguished to facilitate black-and-white printing. For example, fit lines that correspond to different classification levels are distinguished by both colors and line patterns. You can use the ATTRPRIORITY='Color' styles (such as HTMLBLUE, PEARL, PEARLJ, and SAPPHIRE) when you want groups to be distinguished only by color.

Although you can use any ODS style, only a few styles are usually used with ODS Graphics. They are described in Table 21.1.

[1]More precisely, an ATTRPRIORITY='Color' style such as HTMLBLUE distinguishes the first 12 groups of observations only by color. Markers and lines change for groups 13–24 and then again for groups 25–36. Figure 21.55 shows how colors, markers, and line styles change in the HTMLBLUE style, and Figure 21.54 shows how these change in most other ODS styles.

Table 21.1 ODS Styles Most Often Used with ODS Graphics

Style	Recommended Destinations	Attribute Priority	Description
ANALYSIS	HTML	None	A color style, with sans serif fonts, whose dominant colors are yellow, green, and tan. See Figure 21.27.
DEFAULT	HTML	None	A color style, with bold sans serif fonts, whose dominant colors are gray, blue, and white. See Figure 21.23.
HTMLBLUE	HTML	Color ✓	A color style, with sans serif fonts, whose dominant colors are shades of blue. See Figure 21.24. Default for HTML destination and SAS/STAT documentation.
HTMLBLUECML	HTML	None ✓	An ATTRPRIORITY='None' version of HTMLBLUE. See Figure 21.25.
JOURNAL, JOURNAL1A	PDF, PS, RTF, PRINTER	None ✓	A black-and-white style with sans serif fonts and filled areas. See Figure 21.28, Figure 21.33, and Figure 21.34.
JOURNAL2, JOURNAL2A	PDF, PS, RTF, PRINTER	None ✓	A black-and-white style, similar to JOURNAL but with empty areas. Grouped bar charts use crosshatching to show groups. See Figure 21.35 and Figure 21.36.
JOURNAL3, JOURNAL3A	PDF, PS, RTF, PRINTER	None ✓	A black-and-white style, similar to JOURNAL2 but with a mix of filled areas and crosshatching in grouped bar charts. See Figure 21.37 and Figure 21.38.
LISTING	HTML, LISTING	None	A color style, similar to DEFAULT but with a white background. See Figure 21.29. Default for the LISTING destination.
PEARL	PDF, PS, RTF, PRINTER	Color ✓	A color style, with sans serif fonts and a white background, whose dominant colors are shades of blue. See Figure 21.31 and Figure 21.39. Default for PDF destination.
PEARLJ	PDF, PS, RTF, PRINTER	Color ✓	A color style, with sans serif fonts and a white background, whose dominant colors are shades of blue. See Figure 21.40. Default for PDF tables in SAS/STAT documentation.
RTF	RTF	None	A color style, with serif (Times Roman) fonts, whose dominant colors are blue, white, and black. See Figure 21.30 and Figure 21.41. Default for RTF destination.
SAPPHIRE	PDF, PS, RTF, PRINTER	Color ✓	A color style, with sans serif fonts, a white background, and a light blue table heading background, whose dominant colors are shades of blue. See Figure 21.32 and Figure 21.42.
STATISTICAL	HTML	None	A color style, with sans serif fonts, whose dominant colors are blue, gray, and white. See Figure 21.26.

✓indicates newer styles that are recommended for use with statistical graphics.
JOURNAL# styles differ from JOURNAL#A styles in that the former use italic fonts in table headings.

You specify an ODS style by using the STYLE= option in the ODS destination statement. For example, the following statement creates RTF output and specifies the JOURNAL style:

```
ods rtf style=Journal;
```

The following statement sets the style for the LISTING destination:

```
ods listing style=HTMLBlue;
```

The style that is specified by the STYLE= option in the ODS LISTING statement applies only to graphs. SAS monospace format is used for tables.

More generally, you can modify the colors, fonts, and other attributes of graph elements in an ODS style by editing the style template. For more information, see the section "ODS Styles" on page 657 and *SAS Output Delivery System: User's Guide*. You can also perform ODS style modifications by using the %MODSTYLE SAS autocall macro. For more information, see the section "ODS Style Template Modification Macro" on page 701.

ODS Destinations

ODS can send your graphs and tables to a number of different destinations, including RTF (rich text format), HTML (hypertext markup language), LISTING (the SAS LISTING destination), DOCUMENT (the ODS document), and PDF (portable document format). You use an ODS statement to open a destination, as in the following examples:

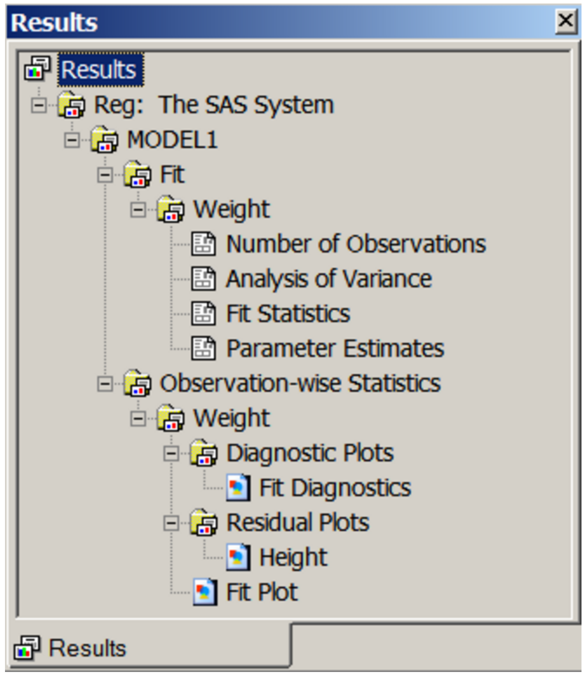
```
ods html body='b.htm';
ods rtf;
ods listing;
ods document name=MyDoc(write);
ods pdf file="contour.pdf";
```

You can close destinations individually or all at once, as in the following examples:

```
ods html close;
ods rtf close;
ods listing close;
ods document close;
ods pdf close;
ods _all_ close;
```

For most ODS destinations (such as HTML, RTF, and PDF), graphs and tables are integrated in the output, and you view your output by using an appropriate viewer, such as a web browser for HTML. However, the LISTING destination is different. If you are using the LISTING destination in the SAS windowing environment, you view your graphs individually by clicking the graph icons in the Results window, shown in Figure 21.14. This action invokes a host-dependent graph viewer (for example, Microsoft Photo Editor in Windows). The graphs that ODS Graphics produces are *not* displayed along with traditional graphs in the Graph window.

Figure 21.14 SAS Results Window



If you are using the SAS windowing environment and you prefer to view integrated output, you should use a destination such as HTML or RTF. In many cases, HTML is the default destination in the SAS windowing environment (see the section "HTML Output in the SAS Windowing Environment" on page 532 in Chapter 20, "Using the Output Delivery System"). You can change destinations in the SAS windowing environment by selecting **Tools ► Options ► Preferences** from the main SAS window and then clicking the **Results** tab.

You can prevent the Output window from appearing by using ODS statements to close the LISTING destination, as follows:

```
ods listing close;
ods html;
```

ODS creates a graph for every open destination. When you open a new destination, you should close all destinations that you do not need. Closing destinations makes your jobs run faster and use fewer resources, because fewer tables and graphs are produced.

Accessing Individual Graphs

If you are writing a paper or creating a presentation, you need to access your graphs individually. There are various ways to do this, depending on the ODS destination. Three particularly useful methods are as follows:

- If you are viewing RTF output, you can simply copy your graphs from the viewer and paste them into a Microsoft Word document or PowerPoint slide.
- If you are viewing HTML output, you can copy and paste your graphs from the viewer, or you can right-click the graph and save it to a file. For more information, see the section "Specifying the Size and Resolution of Graphs" on page 629.
- You can save your graphs in image files and then include them in a paper or presentation. For example, you can save your graphs as PNG files and include them in an HTML document or in a paper that you are writing in LaTeX.

You can specify the graphics image format and the file name in the ODS GRAPHICS statement. You can specify the graph resolution by using the IMAGE_DPI= option in an ODS destination statement. For example, the following statements, when submitted before a procedure step that produces multiple graphs, save the graphs in PostScript files named *myname.ps*, *myname1.ps*, and so on:

```
ods _all_ close;
ods latex;
ods graphics on / outputfmt=ps imagename='myname';
```

The following statements save the graphs in PNG files with a resolution of 300 dots per inch (DPI):

```
ods _all_ close;
ods html image_DPI=300;
ods graphics on;
```

For more information about the file types available with various destinations, how they are named, and how they are saved, see the section "Image File Types" on page 644. For more information about resolution, see the section "Specifying the Size and Resolution of Graphs" on page 629. For more information about scalable vector graphics, see the section "Scalable Vector Graphics" on page 645. If you are using the LISTING destination and the SAS windowing environment, you can copy from the viewer into a Microsoft Word document or PowerPoint slide.

Specifying the Size and Resolution of Graphs

Two factors to consider when you are creating graphs for a paper or presentation are the size of the graph and its resolution. You can specify the size of a graph in the ODS GRAPHICS statement. The following examples show typical ways to change the size of your graphs:

```
ods graphics on / width=6in;
ods graphics on / height=4in;
ods graphics on / width=4.5in height=3.5in;
```

You can change the resolution by specifying the IMAGE_DPI= option in any ODS destination statement, as in the following example:

```
ods html image_dpi=300;
```

The default resolution of graphs that you create by using the HTML and LISTING destinations is 96 DPI, whereas the default resolution in the RTF destination is 200 DPI. An increase in resolution often improves the quality of the graphs, but it also increases the size of the image file. For more information about graph size and resolution, see the section "Graph Size and Resolution" on page 650.

Modifying Your Graphs

Although ODS Graphics is designed to automate the creation of high-quality statistical graphics, you might occasionally need to modify your graphs. There are two ways to do this, depending on whether the changes that you want to make are data-dependent and immediate (for a specific graph you are preparing for a paper or presentation) or persistent (applied to a graph each time you run the procedure). You can make immediate, ad hoc changes by using the ODS Graphics Editor, which provides a point-and-click interface. You can make persistent changes by modifying the ODS graph template for a particular plot. (For an introduction to graph template modification, see Chapter 22, "ODS Graphics Template Modification.") A graph template is a program, written in the Graph Template Language (GTL), that specifies the layout and details of a graph.

NOTE: The SAS System provides a template for each graph that it creates, so you do not need to know anything about templates in order to create statistical graphics.

You can use the ODS Graphics Editor to customize titles and labels, annotate data points, add text, and change the properties of graph elements. After you modify your graph, you can save it as a PNG image file or an SGE file; the latter preserves the editing context. You can open SGE files by using the ODS Graphics Editor and resume editing.

You can invoke the ODS Graphics Editor in the SAS windowing environment, provided that you have enabled ODS Graphics to create editable graphs. The steps for doing this are described in the section "ODS Graphics Editor" on page 651. Also see *SAS ODS Graphics Editor: User's Guide*.

Figure 21.15 shows the ODS Graphics Editor window for a fit plot that is created by PROC REG. Figure 21.16 shows modifications that are made by using tools in the ODS Graphics Editor. The title has been changed, and the legend has been repositioned.

Figure 21.15 ODS Graphics Editor Invoked to Edit a Fit Plot

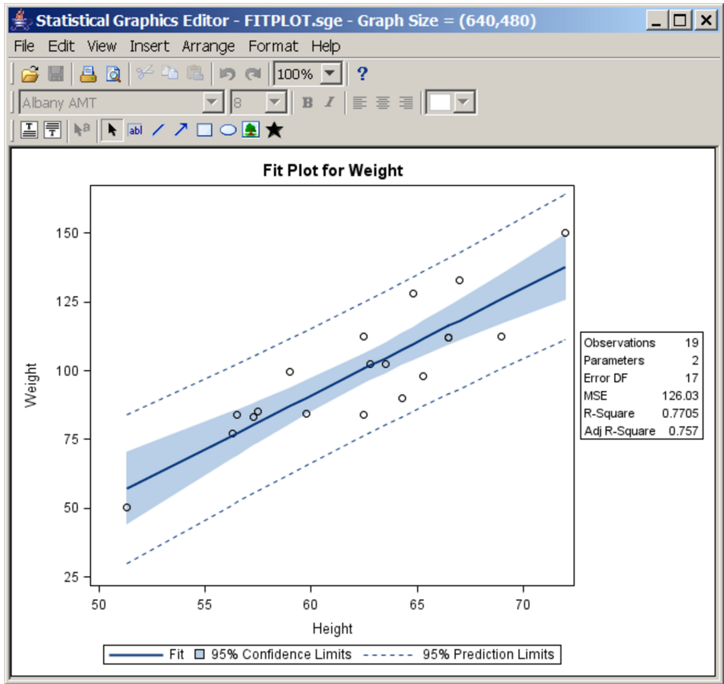
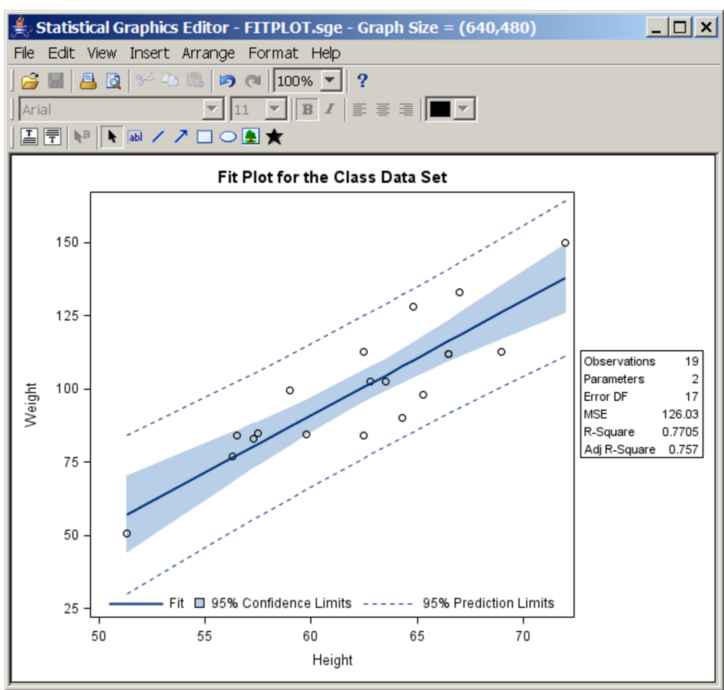


Figure 21.16 Point-and-Click Modifications Made Using the ODS Graphics Editor



Procedures That Support ODS Graphics

SAS procedures that support ODS Graphics include the following:

SAS/STAT		**Base SAS**	**SAS/ETS**
ACECLUS	MIXED	CORR	ARIMA
ADAPTIVEREG	MULTTEST	FREQ	AUTOREG
ANOVA	NLIN	UNIVARIATE	COPULA
BCHOICE	NPAR1WAY		COUNTREG
BGLIMM	ORTHOREG	**SAS/QC**	ENTROPY
BOXPLOT	PHREG	ANOM	ESM
CALIS	PLM	CAPABILITY	EXPAND
CAUSALTRT	PLS	CUSUM	HPCDM
CLUSTER	POWER	MACONTROL	HPQLIM
CORRESP	PRINCOMP	MVPDIAGNOSE	HPSEVERITY
FACTOR	PRINQUAL	MVPMONITOR	MODEL
FMM	PROBIT	MVPMODEL	PANEL
FREQ	PSMATCH	PARETO	PDLREG
GAM	QUANTLIFE	RAREEVENTS	QLIM
GAMPL	QUANTREG	RELIABILITY	SEVERITY
GEE	QUANTSELECT	SHEWHART	SIMILARITY
GENMOD	REG		SSM
GLIMMIX	RMSTREG	**Other**	SYSLIN
GLM	ROBUSTREG	HPF	TIMEDATA
GLMPOWER	RSREG	HPFENGINE	TIMEID
GLMSELECT	SEQDESIGN		TIMESERIES
HPFMM	SEQTEST	SAS Risk	TMODEL
HPSPLIT	SIM2D	Dimensions	UCM
ICLIFETEST	SPP		VARMAX
ICPHREG	STDRATE		X12
IRT	SURVEYFREQ		X13
KDE	SURVEYLOGISTIC		
KRIGE2D	SURVEYMEANS		
LIFEREG	SURVEYPHREG		
LIFETEST	SURVEYREG		
LOESS	TPSPLINE		
LOGISTIC	TRANSREG		
MCMC	TTEST		
MDS	VARCLUS		
MI	VARIOGRAM		

For information about the specific graphs that a particular procedure creates, see the PLOTS= option syntax and the “ODS Graphics” section in the corresponding procedure chapter. For the SAS/STAT procedures, the procedure names in the preceding table link to the “ODS Graphics” section.

Procedures That Support ODS Graphics and Traditional Graphics

A number of procedures that support ODS Graphics produced traditional graphics in previous releases of SAS. These include the UNIVARIATE procedure in Base SAS software; the LIFEREG, LIFETEST, and REG procedures in SAS/STAT software; and the ANOM, CAPABILITY, CUSUM, MACONTROL, PARETO, RELIABILITY, and SHEWHART procedures in SAS/QC software. All these procedures continue to produce traditional graphics, but in some cases they do so only when ODS Graphics is not enabled. For more information about the interaction between traditional graphics and ODS graphics in other procedures, see the documentation for those procedures.

Traditional graphs are saved in SAS graphics catalogs and are controlled by the GOPTIONS statement. In contrast, ODS Graphics produces graphs in standard image file formats (not graphics catalogs), and their appearance and layout are controlled by ODS styles and templates.

Syntax

The following sections document some of the most commonly used options in the ODS GRAPHICS statement (see the section "ODS GRAPHICS Statement" on page 632) and other ODS Graphics statements (see the section "ODS Destination Statements" on page 635). You can find the complete syntax in the *SAS Output Delivery System: User's Guide*. In addition, information about the PLOTS= option is provided in the section "PLOTS= Option" on page 636. All statistical procedures that produce ODS Graphics have a PLOTS= option that selects graphs and controls some aspects of the graphs.

ODS GRAPHICS Statement

ODS GRAPHICS < **OFF | ON** > < / *options* > **;**

The ODS GRAPHICS statement enables ODS to create graphs. You can enable ODS Graphics by using either of the following equivalent statements:

```
ods graphics on;
ods graphics;
```

You specify one of these statements before invoking your procedure, as illustrated in the examples beginning in the section "Default Plots for Simple Linear Regression with PROC REG" on page 606. Any procedure that supports ODS Graphics then produces graphs, either by default or when you specify procedure options to request particular graphs.

To disable ODS Graphics, specify the following statement:

```
ods graphics off;
```

ODS Graphics might or might not be enabled by default, depending on your operating system, whether you are in the SAS windowing environment, your registry, your system options, and your configuration file settings. For more information about default settings and enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" on page 623.

The following is a subset of the options, syntax, and capabilities available in the ODS GRAPHICS statement. For more information, see the *SAS Output Delivery System: User's Guide*.

ANTIALIAS=ON | OFF

controls the use of antialiasing to smooth the components of a graph. Without antialiasing, pixels are simply set or not set. With antialiasing, pixels at the edge of a line or other object are set to an intermediate color, which makes smoother and more professional-looking graphics. Text that is displayed in a graph is always antialiased. Antialiasing is very time-consuming for larger graphs, and its benefits decrease as the number of points increases, so it is disabled by default for plots that have many points. If the number of graph elements exceeds the ANTIALIASMAX= threshold (4,000 by default), then antialiasing is not used, even if you specify ANTIALIAS=ON. By default, ANTIALIAS=ON.

ANTIALIASMAX=*n*

specifies the maximum number of markers or lines to be antialiased before antialiasing is disabled. For example, if there are more than 4,000 markers and ANTIALIASMAX=4,000 (the default), then no markers are antialiased.

ATTRPRIORITY=COLOR | NONE

specifies the priority for cycling group attributes.

COLOR cycles through the lists of colors while holding the marker symbol and line pattern constant. When all the colors are exhausted, the marker symbol and line style attributes increment to the next element, and then the colors in the list are repeated.

NONE simultaneously cycles through colors, marker symbols, and line patterns.

You can specify ATTRPRIORITY=COLOR or ATTRPRIORITY=NONE in the ODS GRAPHICS statement to change the attribute priority for all graphs. You can specify ATTRPRIORITY=COLOR or ATTRPRIORITY=NONE in the BEGINGRAPH statement in the GTL to change the attribute priority for individual graphs. The default attribute priority is set by the ATTRPRIORITY='Color' or ATTRPRIORITY='None' option in the CLASS GRAPH statement in each style.

BORDER=ON | OFF

specifies whether to draw the graph with a border. By default, BORDER=ON.

BYLINE=FOOTNOTE | TITLE | NOBYLINE

specifies how the BY-group line is displayed in graphs when the analysis contains a BY statement. By default, a BY line is not displayed. Specify BYLINE=FOOTNOTE (recommended) to display the BY line as a left-justified graph footnote or BYLINE=TITLE (not recommended) to display the BY line as a centered graph title. Most graph templates control the placement of the BY line as follows:

```
if (_BYTITLE_)
   entrytitle _BYLINE_ / textattrs=GraphValueText;
else
   if (_BYFOOTNOTE_)
      entryfootnote halign=left _BYLINE_;
   endif;
endif;
```

You can modify the graph template if you want the BY line to be displayed in some other way. Because most graphs have titles and few graphs have footnotes, the BY line looks better when it is displayed as a footnote.

HEIGHT=*dimension*
specifies the height of the graph. By default, HEIGHT=480PX (480 pixels). You can also specify height in inches (for example, HEIGHT=5IN) or centimeters (for example, HEIGHT=12CM).

IMAGEMAP=ON | OFF
controls tooltip generation in the HTML destination. By default, IMAGEMAP=OFF, which means that no tooltips are generated. Tooltips are text boxes that appear in HTML output when you rest your mouse pointer over a part of the plot (see Example 21.1).

IMAGENAME=< *base-file-name* >
specifies the base image file name. The default is the name of the output object. You can determine the name of the output object by using the ODS TRACE statement (see the section "Determining Graph Names and Labels" on page 640). The base image file name should not include an extension. ODS automatically adds the increment value and the appropriate extension (which is specific to the output destination). See the section "Specifying Base File Names" on page 646 for an example.

LABELMAX=*n*
specifies the maximum number of labeled areas before labeling is disabled. For example, if LABELMAX=50, and there are more than 50 points that have labels, then no points are labeled. By default, LABELMAX=200.

MAXLEGENDAREA=*n*
specifies the maximum percentage of the overall graph area that a legend can occupy. By default, MAXLEGENDAREA=20. Larger legends are dropped from the display.

OUTPUTFMT=< *image-file-type* | **STATIC** >
specifies the image format for graphs. The OUTPUTFMT= option was previously named the IMAGEFMT= option. By default, OUTPUTFMT=STATIC, and ODS dynamically uses the best-quality static image format for the active output destination. The available image formats include BMP (Microsoft Windows device-independent bitmap), DIB (Microsoft Windows device-independent bitmap), EMF (Microsoft NT enhanced metafile), EPSI (Adobe encapsulated PostScript interchange), GIF (graphic interchange format), JFIF (JPEG file interchange format), JPEG (Joint Photographic Experts Group format), PBM (portable bitmap), PCD (Photo CD), PCL (Printer Command Language), PDF (portable document format), PICT (QuickDraw picture format), PNG (Portable Network Graphics), PS (PostScript image file format), SVG (Scalable Vector Graphics format), TIFF (Tagged Image File Format), WMF (Microsoft Windows Metafile format), XBM (X Bitmap), and XPM (X-Windows Pixelmap). If the specified image format is not valid for the active output destination, the device is automatically remapped to the default image format.

RESET< =*option* >
resets one or more ODS GRAPHICS options to their default settings. RESET and RESET=ALL are equivalent. If you want to reset more than one option, but not all the options, then you must specify RESET= separately for each option that you reset (for example, `ods graphics on / reset=antialias reset=index;`). The RESET= *options* include the following:

ALL	resets all resettable options to their defaults.
ANTIALIAS	resets the ANTIALIAS= option to its default.
ANTIALIASMAX	resets the ANTIALIASMAX= option to its default.
BORDER	resets the BORDER= option to its default.
INDEX	resets the index counter that is appended to static image files.

HEIGHT	resets the HEIGHT= option to its default.
IMAGEMAP	resets the IMAGEMAP= option to its default.
LABELMAX	resets the LABELMAX= option to its default.
SCALE	resets the SCALE= option to its default.
TIPMAX	resets the TIPMAX= option to its default.
WIDTH	resets the WIDTH= option to its default.

SCALE=ON | OFF
specifies whether the fonts and symbol markers are scaled proportionally to the size of the graph. By default, SCALE=ON. For examples, see Figure 21.74 and Figure 21.75.

SCALEMARKERS=ON | OFF
specifies whether markers are scaled in nested layouts. By default, SCALEMARKERS=ON.

TIPMAX=*n*
specifies the maximum number of distinct tooltips that are permitted before tooltips are disabled. Tooltips are text boxes that appear when you rest your mouse pointer over part of the plot. For example, if TIPMAX=400, and there are more than 400 points in a scatter plot, then no tooltips appear. By default, TIPMAX=500.

WIDTH=*dimension*
specifies the width of the graph. By default, WIDTH=640PX (640 pixels). You can also specify width in inches (for example, WIDTH=5IN) or centimeters (for example, WIDTH=12CM).

ODS Destination Statements

ODS has a number of statements that control the destination of ODS output. The ODS destination statements that are most commonly used in ODS Graphics are ODS DOCUMENT, ODS HTML, ODS LISTING, ODS PCL, ODS PDF, ODS PS, and ODS RTF. Specifying a statement opens a destination, unless the CLOSE option is specified. Each of the following statements opens an ODS destination:

```
ods html;
ods rtf;
ods html image_dpi=300;
ods listing style=HTMLBlue;
```

Each of the following statements closes an ODS destination:

```
ods html close;
ods rtf close;
ods listing close;
```

The following statement closes all open destinations:

```
ods _all_ close;
```

The following two options are commonly used in ODS destination statements to control aspects of ODS Graphics:

IMAGE_DPI=*dpi*
specifies the image resolution of graphical output, measured in number of dots per inch (DPI). The default varies depending on the destination. For example, the default is 96 for HTML and 200 for RTF.

STYLE=*style-name*

specifies the output style. Commonly used styles include HTMLBLUE, HTMLBLUECML, PEARL, PEARLJ, SAPPHIRE, DEFAULT, LISTING, STATISTICAL, JOURNAL, JOURNAL2, JOURNAL3, RTF, and ANALYSIS.

Other options provide ways to control the files that you create. For example, the following statement opens the HTML destination:

```
ods html body='b.html' contents='c.html' frame='a.html';
```

This statement also writes the body of the output to the file *b.html*, the table of contents to the file *c.html*, and an overall frame that contains both the contents and the output to the file *a.html*. Alternatively, you can specify FILE= instead of BODY=.

If you are using a destination for which individual graphs are created (for example, LISTING or HTML), you can use the GPATH= option to specify the directory where your graphics files are saved, as in the following example:

```
ods html gpath="C:\figures";
```

For more information about individual image files and options specified in the ODS destination statements, see the sections “Image File Types” on page 644, “Saving Graphic Image Files” on page 647, “LISTING Destination” on page 648, and “HTML Destination” on page 648. For complete information about the ODS destination statements, see the *SAS Output Delivery System: User’s Guide*.

PLOTS= Option

Each statistical procedure that supports ODS Graphics has a PLOTS= option that you use to select graphs and specify some options. The syntax of the PLOTS= option is as follows:

PLOTS < **(***global-plot-options***)** > < = *plot-request*< **(***options***)** > > **;**

PLOTS < **(***global-plot-options***)** > < =**(***plot-request*< **(***options***)** > < ... *plot-request*< **(***options***)** > >**)** > **;**

The PLOTS= option has a common overall syntax for all statistical procedures, but the specific *global-plot-options*, *plot-requests*, and *plot-options* vary across procedures. This section discusses only a few of the options available in the PLOTS= option. For more information about the PLOTS= option, see the “Syntax” section for each procedure that produces ODS Graphics. There are only a limited number of details that you can control by using the PLOTS= option. Most graphical details are controlled either by graph templates (see the section “Graph Templates” on page 782 in Chapter 22, “ODS Graphics Template Modification”) or by styles (see the section “ODS Styles” on page 657).

The PLOTS= option is usually specified in the PROC statement. However, for some procedures, certain analyses and hence certain plots can appear only if an additional statement is specified. These procedures might have a PLOTS= option in that other statement. For example, the PHREG procedure has a PLOTS= option in the BAYES statement, which is used to perform a Bayesian analysis. For more information, see the “Syntax” section of each procedure chapter. The following examples illustrate the syntax of the PLOTS= option:

```
plots=all
plots=none
plots=residuals
plots=residuals(smooth)
```

```
plots=(trace autocorr)
plots(unpack)
plots(unpack)=diagnostics
plots=diagnostics(unpack)
plots(only)=freqplot
plots=(scree(unpack) loadings(plotref) preloadings(flip))
plots(unpack maxparmlabel=0 stepaxis=number)=coefficients
plots(sigonly)=(rawprob adjusted(unpack))
```

Also see the "Getting Started" sections "Survival Estimate Plot with PROC LIFETEST" on page 609, "Contour and Surface Plots with PROC KDE" on page 610, "Contour Plots with PROC KRIGE2D" on page 611, "LS-Means Diffogram with PROC GLIMMIX" on page 616, and "Principal Component Analysis Plots with PROC PRINCOMP" on page 617 for examples of using the PLOTS= option.

The simplest PLOTS= specifications are of the form PLOTS=*plot-request* or PLOTS=(*plot-requests*). When there is more than one *plot-request*, the *plot-request* list must appear in parentheses. Each *plot-request* either requests a plot (for example, RESIDUALS) or provides a place to specify plot-specific options (for example, DIAGNOSTICS(UNPACK)). Some simple and typical *plot-requests* are explained next:

- PLOTS=ALL requests all plots that are relevant to the analysis. This does not mean that the procedure produces all plots that it can produce. Plots that are produced for one set of options might not appear when you specify PLOTS=ALL and a different set of options. In some cases, certain plots are not produced unless certain options or statements outside the PLOTS= option are specified.

- PLOTS=NONE disables ODS Graphics for just that step. You can use this option instead of specifying ODS GRAPHICS OFF before a procedure step and ODS GRAPHICS ON after the step when you want to suppress graphics for only that step.

- PLOTS=RESIDUALS requests a plot of residuals in a modeling procedure such as PROC REG.

- PLOTS=RESIDUALS(SMOOTH) requests the residuals plot along with a smooth fit function.

- PLOTS=(TRACE AUTOCORR) requests trace and autocorrelation plots in procedures that have Bayesian analysis options.

Global-plot-options appear in parentheses after the option name and before the equal sign. These options affect many or all of the plots. The UNPACK option is a commonly used *global-plot-option*. It specifies that plots that are usually produced containing multiple plots per panel (or "packed") should be unpacked so they appear in multiple panels that each contain one plot. The specification PLOTS(UNPACK)=(*plot-requests*) unpacks all paneled plots. The UNPACK option is also used as an option in a *plot-request* when you want to unpack only certain panels. For example, the option PLOTS=(DIAGNOSTICS(UNPACK) PARTIAL PREDICTIONS) unpacks only the diagnostics panel. In some cases, unpacked plots contain additional information that is not found in the smaller packed versions. The UNPACK option is not available for all *plot-requests*; it is available only for plots that have multiple panels by default.

Another commonly used *global-plot-option* is the ONLY option. Many procedures produce default plots, and additional plots can be requested in the PLOTS= option. Specifying PLOTS=(*plot-requests*) while omitting the default plots does not prevent the default plots from being produced. You use the ONLY option when you want to see only the plots that are specifically cited in the *plot-request* list. Procedures that produce no default plots usually do not provide an ONLY option. You can use ODS SELECT and ODS EXCLUDE (see the section "Selecting and Excluding Graphs" on page 642) to select and exclude graphs, but in some situations the ONLY option is more convenient. It is usually more efficient to select plots by using the

PLOTS(ONLY)= option, because the procedure does not do extra work to generate a plot that is excluded by the PLOTS(ONLY)= option. In contrast, ODS SELECT and ODS EXCLUDE have their effect after the procedure does the work to generate the plot.

Selecting and Viewing Graphs

This section describes techniques for selecting and viewing your graphs. Topics include the following:

- specifying an ODS destination for graphics
- viewing your graphs in the SAS windowing environment
- referring to graphs by name when using ODS
- selecting and excluding graphs from your output

Specifying an ODS Destination for Graphics

If you do not specify an ODS destination, then either the LISTING or HTML destination is used by default. Here is an example of how you can explicitly specify the HTML destination:

```
ods graphics on;
ods html;

proc reg data=sashelp.class;
   model Weight = Height;
quit;

ods html close;
```

This ODS HTML statement creates an HTML file that has a default name. For information about specifying a file name, see the section "Specifying a File for ODS Output" on page 639. Other destinations are specified in a similar way. For example, you can specify an RTF destination by using the following statements:

```
ods graphics on;
ods rtf;

. . .

ods rtf close;
```

The destinations that ODS supports for graphics are as follows:

Destination	Destination Family
DOCUMENT	
HTML	MARKUP
LATEX*	MARKUP
LISTING	
PCL	PRINTER
PDF	PRINTER
PS	PRINTER
RTF	

* The LATEX destination is experimental.

You can close all open destinations if you are interested only in displaying your output in a nondefault destination. For example, if you want to see your output only in the RTF destination, you can specify the following statements:

```
ods graphics on;
ods _all_ close;
ods rtf;

. . .

ods rtf close;
ods listing;
```

Closing unneeded destinations makes your jobs run faster and creates fewer files. More generally, it makes your jobs consume fewer resources, because a graph is created for every open destination. The last statement opens the LISTING destination after you are finished using the RTF destination.

You can also use the ODS OUTPUT destination to create an output data set from the data object that is used to make a plot. Here is an example:

```
ods graphics on;

proc reg data=sashelp.class;
   ods output fitplot=myfitplot;
   model Weight = Height;
quit;
```

Specifying a File for ODS Output

You can specify a file name for your output by using the FILE= option in the ODS destination statement, as in the following example:

```
ods html file="test.htm";
```

The output is written to the file *test.htm*, which is saved in the SAS current folder. At start-up, the SAS current folder is the same directory in which you started your SAS session. If you are using the SAS windowing environment, then the current folder is displayed in the status line at the bottom of the main SAS window. If you do not specify a file name for your output, then the SAS System provides a default file name, which depends on the ODS destination. This file is saved in the SAS current folder. You can always check the SAS log to verify the name of the file in which your output is saved. For example, suppose you specify the following statement:

```
ods html;
```

Then the following message is displayed in the SAS log:

```
NOTE: Writing HTML Body file: sashtml.htm
```

The default file names for each destination are specified in the SAS Registry. For example, Figure 21.76 shows that the default file name in the SAS Registry for the RTF destination is *sasrtf.rtf*. For more information, see the *SAS Companion* for your operating system.

Viewing Your Graphs in the SAS Windowing Environment

The mechanism for viewing graphs that are created by ODS can vary depending on your operating system, which viewers are installed on your computer, and the ODS destination that you select. If you do not specify an ODS destination, then the default destination is either HTML or LISTING.

If you are using the SAS windowing environment and the HTML destination, then the results are displayed by default in the SAS Results Viewer unless you are using an external browser. To use an external viewer, select **Tools ► Options ► Preferences** from the main SAS window. Then click the **Results** and **Web** tabs to make your selection.

If you are using the PS destination, you must use a PostScript viewer, such as GSview. For information about the windowing environment in a different operating system, see the *SAS Companion* for that operating system.

If you do not want to view the results as they are being generated, then select **Tools ► Options ► Preferences** from the main SAS window. Then on the **Results** tab, clear the **View results as they are generated** check box.

If you are using the SAS windowing environment and the LISTING destination, go to the Results window and find the icon for the corresponding graph. You can double-click the graph icon to display the graph in the default viewer that is configured on your computer for the corresponding image file type (see Figure 21.14).

Determining Graph Names and Labels

Procedures assign a name to each graph that they create using ODS Graphics. This name enables you to refer to ODS graphs in the same way that you refer to ODS tables (see the section "The ODS Statement" on page 539 in Chapter 20, "Using the Output Delivery System"). You can determine the names of graphs in several ways:

- You can look up graph names in the "ODS Graphics" section of chapters for procedures that use ODS Graphics. For example, see the section "ODS Graphics" on page 8557 in Chapter 102, "The REG Procedure."
- You can use the Results window to view the names of ODS graphs that are created in your SAS session. For more information, see the section "The SAS Results Window" on page 543 in Chapter 20, "Using the Output Delivery System."

- You can use the ODS TRACE ON statement to list the names of graphs that are created during your SAS session. This statement adds identifying information in the SAS log (or optionally in the SAS LISTING) for each graph that is produced. For more information, see the section "The ODS Statement" on page 539 in Chapter 20, "Using the Output Delivery System."

The graph name is not the same as the name of the image file that contains the graph (see the section "Naming Graphic Image Files" on page 645).

This example revisits the analysis that is described in the section "Contour and Surface Plots with PROC KDE" on page 610. To determine which output objects are created by ODS, you specify the ODS TRACE ON statement before the procedure statements as follows:

```
ods graphics on;
ods trace on;

proc kde data=bivnormal;
   bivar x y / plots=contour surface;
run;

ods trace off;
```

The trace record from the SAS log is as follows:

```
Output Added:
-------------
Name:       Inputs
Template:   Stat.KDE.Inputs
Path:       KDE.Bivar1.x_y.Inputs
-------------

Output Added:
-------------
Name:       Controls
Template:   Stat.KDE.Controls
Path:       KDE.Bivar1.x_y.Controls
-------------

Output Added:
-------------
Name:       ContourPlot
Label:      Contour Plot
Template:   Stat.KDE.Graphics.Contour
Path:       KDE.Bivar1.x_y.ContourPlot
-------------

Output Added:
-------------
Name:       SurfacePlot
Label:      Density Surface
Template:   Stat.KDE.Graphics.Surface
Path:       KDE.Bivar1.x_y.SurfacePlot
-------------
```

By default, PROC KDE creates table objects named Inputs and Controls, and it creates graph objects named ContourPlot and SurfacePlot. In addition to the name, the trace record provides the label, template, and path for each output object. Graph templates are distinguished from table templates by a naming convention that uses the procedure name in the second level and **Graphics** in the third level. For example, the fully qualified template name for the surface plot that PROC KDE creates is **Stat.KDE.Graphics.SurfacePlot**.

You can specify the LISTING option in the ODS TRACE ON statement to write the trace record to the LISTING destination as follows:

```
ods trace on / listing;
```

Each table and graph has a path (or name path), which was previously shown in the trace output. The path consists of the plot name, preceded by the names of one or more output groups. Each table and graph also has a label path, which can be seen by adding the LABEL option to the ODS TRACE ON statement, after a slash, as follows:

```
ods trace on / label;

proc kde data=bivnormal;
   bivar x y / plots=contour surface;
run;

ods trace off;
```

A portion of the trace output is shown next:

```
Path:       KDE.Bivar1.x_y.Inputs
Label Path: 'The KDE Procedure'.'Bivariate Analysis'.'x and y'.'KDE.Bivar1.x_y'

Path:       KDE.Bivar1.x_y.Controls
Label Path: 'The KDE Procedure'.'Bivariate Analysis'.'x and y'.'KDE.Bivar1.x_y'

Path:       KDE.Bivar1.x_y.ContourPlot
Label Path: 'The KDE Procedure'.'Bivariate Analysis'.'x and y'.'Contour Plot'

Path:       KDE.Bivar1.x_y.SurfacePlot
Label Path: 'The KDE Procedure'.'Bivariate Analysis'.'x and y'.'Density Surface'
```

The label path contains the information that you see in the HTML table of contents. Names are fixed (they do not vary and they are not data- or context-dependent). In contrast, labels often reflect data- or context-dependent information.

Selecting and Excluding Graphs

You can use the ODS SELECT and ODS EXCLUDE statements along with graph and table names to specify which ODS outputs to display. For more information about how to use these statements, see the section "The ODS Statement" on page 539 in Chapter 20, "Using the Output Delivery System."

This section shows several examples of selecting and excluding graphs by using the data set and trace output that are created in the section "Determining Graph Names and Labels" on page 640. The following statements use the ODS SELECT statement to select only two graphs, ContourPlot and SurfacePlot, for display in the output:

```
proc kde data=bivnormal;
   ods select ContourPlot SurfacePlot;
   bivar x y / plots=contour surface;
run;
```

Equivalently, the following statements use the ODS EXCLUDE statement to exclude the two tables:

```
proc kde data=bivnormal;
   ods exclude Inputs Controls;
   bivar x y / plots=contour surface;
run;
```

You can select or exclude graphs by using either the name or the label. Labels must be specified in quotes. In the context of this example, the following two statements are equivalent:

```
ods select contourplot;
ods select 'Contour Plot';
```

You can also specify multiple levels of the path, as in the following examples:

```
ods select x_y.contourplot;
ods select 'x and y'.'Contour Plot';
ods select 'x and y'.contourplot;
ods select x_y.'Contour Plot';
```

You can mix name and label paths, as in the last two statements. All four of the preceding statements select the same plot. Furthermore, selection based directly on the names and labels is case-insensitive. The following statements all select the same plot:

```
ods select x_y.contourplot;
ods select 'x and y'.'Contour Plot';
ods select X_Y.CONTOURPLOT;
ods select 'X AND Y'.'CONTOUR PLOT';
```

It is sometimes useful to specify a WHERE clause in an ODS SELECT or ODS EXCLUDE statement. This enables you to specify expressions based on either the name path or the label path. You can base your selection on two automatic variables, _path_ and _label_. The following two statements select every object whose path contains the string 'plot' and every object whose label path contains the string 'plot', respectively, ignoring the case in the name and label:

```
ods select where = (lowcase(_path_)  ? 'plot');
ods select where = (lowcase(_label_) ? 'plot');
```

The question mark operator means that the second expression (the string 'plot') is contained in the first expression (the lowercase version of the name or label). For example, all the following names match 'plot' in the WHERE clause: plot, SurfacePlot, SURFACEPLOT, FitPlot, pLoTtInG, Splotch, and so on. Because WHERE clause selection is based on SAS string comparisons, selection is case-sensitive. The LOWCASE function is used to ensure a match even when the case of the specified string does not match the case of the actual name or label.

WHERE clauses are particularly useful when you want to select all the objects in a group. A group is a level of the name path or label path hierarchy before the last level. In the following step, all the objects whose name path contains 'DiagnosticPlots' are selected:

```
proc reg data=sashelp.class plots(unpack);
   ods select where = (_path_ ? 'DiagnosticPlots');
   model Weight = Height;
quit;
```

These are the plots that come from unpacking the PROC REG diagnostics panel of plots. All are in a group named 'DiagnosticPlots'.

Graphic Image Files

Accessing your graphs as individual image files is useful when you want to include them in various types of documents. The default image file type depends on the ODS destination, but you can specify other supported image file types. You can also specify the names for your graphic image files and the directory in which you want to save them. This section describes the image file types that ODS Graphics supports, and it explains how to name and save graphic image files.

Image File Types

If you are using the LISTING or HTML destination, your graphs are individually produced in a specific image file type, such as PNG. If you are using a destination in the PRINTER family or the RTF destination, the graphs are contained in the ODS output file and cannot be accessed as individual image files. However, you can open an RTF output file in Microsoft Word and then copy and paste the graphs into another document, such as a Microsoft PowerPoint presentation. This is illustrated in Example 21.2.

Table 21.2 shows the various ODS destinations that ODS Graphics supports, the viewer that is appropriate for displaying graphs in each destination, and the image file types that each destination supports.

Table 21.2 Destinations and Image File Types Supported by ODS Graphics

Destination	Destination Family	Recommended Viewer	Image File Types
DOCUMENT		Not applicable	Not applicable
HTML	MARKUP	Web browser	PNG (default), GIF, JPEG
LATEX	MARKUP	PostScript or PDF viewer after the $\LaTeX$ file is compiled	PostScript (default), EPSI, GIF, JPEG, PDF, PNG
LISTING		Default viewer in your system for the specified file type	PNG (default), BMP, DIB, EMF, EPSI, GIF, JFIF, JPEG, PBM, PS, TIFF, WMF
PCL	PRINTER	Not applicable	Contained in PRN file
PDF	PRINTER	PDF viewer, such as Adobe Reader	Contained in PDF file
PS	PRINTER	PostScript viewer, such as GSview	Contained in PostScript file
RTF		Word processor, such as Microsoft Word	Contained in RTF file

For destinations such as PDF and RTF, you can control the types of the images that the file contains even though individual files are not made for each image. The default image file type is PNG, and other image types are available. For more information, see the *SAS Output Delivery System: User's Guide*.

Scalable Vector Graphics

Scalable vector graphics output is supported in ODS Graphics. The output type support depends on the ODS destination that you use. You can specify the OUTPUTFMT= option in the ODS GRAPHICS statement to specify the output type for any destination. For destinations that generate vector graphics by default, you can get image output by specifying OUTPUTFMT=STATIC.

Vector graphics are not supported for all graph types. When vector graphics are requested but not supported, the graph automatically changes to image output. Vector graphics are not supported for the following graph types:

- three-dimensional graph
- contour plots that have smooth gradient fills
- graphs that have continuous legends
- graphs that have data skins (such as bars charts that appear to be shiny or have depth)
- graphs that have rotated annotation images
- graphs that have transparency (EMF and PS only)

The LISTING destination can generate all the supported forms of vector-based output: PDF, PS, EMF, SVG, and PCL. Each graph is generated in a separate file that can be included in a larger report. The default output format is a PNG image.

Like the LISTING destination, the ODS PRINTER destination can generate all the supported vector output types. The output format depends on the type of printer that you select. If you select the PDF, SVG, or PCL5C printer, vector-based output is automatically produced. However, if you select the PS or EMF printer, you need to set the OUTPUTFMT= option in the ODS GRAPHICS statement to PS or EMF, respectively, to create vector-based output. By default, the output from this destination is contained in one file instead of individual files for each graph.

The PDF destination produces PDF vector output by default, except for the exceptions noted. You can specify OUTPUTFMT=STATIC in the ODS GRAPHICS statement to produce an embedded image in the PDF file.

The experimental LATEX destination produces PS vector output by default, except for the exceptions noted. You can specify OUTPUTFMT=STATIC in the ODS GRAPHICS statement to produce an embedded image in the PostScript file.

The RTF destination produces PNG image output by default. It also supports vector-based EMF output for this destination. You can specify OUTPUTFMT=EMF in the ODS GRAPHICS statement to select this output type. If one of the noted exceptions occurs, the output type for that graph changes to a PNG image type.

The HTML destination produces PNG image output by default. It also supports vector-based SVG output for this destination. You can specify OUTPUTFMT=SVG in the ODS GRAPHICS statement to select this output type. If one of the noted exceptions occurs, the output type for that graph changes to a PNG image type.

In most cases, the vector graphics file is much smaller than a comparable static image file. However, in some cases, the vector graphics file is larger than the image version. This is likely for scatter plots of data sets that have a large number of observations.

Naming Graphic Image Files

The following discussion applies to the destinations where ODS graphs are created as individual image files (for example, HTML and LISTING). The names of graphic image files are determined by a base file name, an index counter, and an extension. By default, the base file name is the ODS graph name (see the section “Determining Graph Names and Labels” on page 640). There is an index counter for each base file name. The extension indicates the image file type. The first time a graph object that has a particular base file name is created, the file name consists only of the base file name and the extension. If a graph that has the same base file name is created multiple times, then an index counter is appended to the base file name to avoid overwriting previously created images.

To illustrate, consider the following statements:

```
proc kde data=bivnormal;
   ods select ContourPlot SurfacePlot;
   bivar x y / plots=contour surface;
run;
```

If you run this step at the beginning of a SAS session, the two graphic image files that are created are *ContourPlot.png* and *SurfacePlot.png*. If you immediately rerun these statements, then ODS creates the same graphs in different image files named *ContourPlot1.png* and *SurfacePlot1.png*. The next time, the image files are named *ContourPlot2.png* and *SurfacePlot2.png*. The index starts at 0, and 1 is added each time the same name is used. However, if the index is at 0, then 0 is not included in the file name.

Resetting the Index Counter

You can specify RESET=INDEX in the ODS GRAPHICS statement to reset the index counter. This is useful when you need to have predictable names. It is particularly useful when you are running a SAS program multiple times in the same session. The following statement resets the index:

```
ods graphics on / reset=index;
```

The index counter is reinitialized at the beginning of your SAS session or if you specify RESET=INDEX in the ODS GRAPHICS statement. Graphic image files that have the same name are overwritten.

Specifying Base File Names

You can specify a base file name for all your graphic image files by using the IMAGENAME= option in the ODS GRAPHICS statement as follows:

```
ods graphics on / imagename="MyName";
```

You can also specify RESET=INDEX as follows:

```
ods graphics on / reset=index imagename="MyName";
```

The IMAGENAME= option overrides the default base file name. In the preceding statement, the graphic image files are named *MyName*, *MyName1*, *MyName2*, and so on.

Specifying Image File Types

You can specify the image file type for the LISTING, HTML, or LATEX destination by specifying OUTPUTFMT= in the ODS GRAPHICS statement as follows:

```
ods graphics on / outputfmts=gif;
```

For more information, see the section “ODS GRAPHICS Statement” on page 632.

Naming Graphic Image Files with Multiple Destinations

Because the index counter depends only on the base file name, if you specify multiple ODS destinations for your output, then the index counter is increased independently of the destination. For example, the following statements create image files named *ContourPlot.png* and *SurfacePlot.png* that correspond to the LISTING destination and *ContourPlot1.png* and *SurfacePlot1.png* that correspond to the HTML destination:

```
ods listing;
ods html;
ods graphics on / reset;

proc kde data=bivnormal;
   ods select ContourPlot SurfacePlot;
   bivar x y / plots=contour surface;
run;

ods _all_ close;
ods listing;
```

When you specify one of the destinations in the PRINTER family or the RTF destination, your ODS graphs are embedded in the document, so the index counter is not affected. For example, the following statements create the image files *ContourPlot.png* and *SurfacePlot.png* for the LISTING destinations but no image files for the RTF destination:

```
ods listing;
ods rtf;
ods graphics on / reset;

proc kde data=bivnormal;
   ods select ContourPlot SurfacePlot;
   bivar x y / plots=contour surface;
run;

ods _all_ close;
```

Saving Graphic Image Files

Knowing where your graphic image files are saved and how they are named is particularly important if you are running in batch mode, if you have disabled the SAS Results window (see the section "Viewing Your Graphs in the SAS Windowing Environment" on page 640), or if you plan to access the files for inclusion in a paper or presentation. The following discussion assumes that you are running SAS on the Windows operating system. If you are running on a different operating system, see the *SAS Companion* for your operating system.

In the SAS windowing environment, the current folder is displayed in the status line at the bottom of the main SAS window. When the **Use WORK folder** check box is cleared on the **Results** tab (which you access by selecting **Tools ► Options ► Preferences** from the main SAS window), graphic image files are saved in the current folder and available after your SAS session ends. They can accumulate over time and take up a great deal of space. When **Use WORK folder** is selected, graphic image files are stored in the Work folder and are not available after your SAS session ends.

If you are running your SAS programs in batch mode, the graphs are saved by default in the same directory where you started your SAS session. For example, suppose the SAS current folder is *C:\myfiles*. If ODS

Graphics is enabled, then your graphic image files are saved in the directory *C:\myfiles*. Traditional graphics are always saved in a catalog in your Work directory.

If you are using the LISTING, HTML, and LATEX destinations, you can specify a directory for saving your graphic image files. If you are using a destination in the PRINTER family or the RTF destination, you can specify a directory only for your output file. The remainder of this discussion provides details about each destination type.

LISTING Destination

If you are using the LISTING destination, the individual graphs are created as PNG files by default. You can use the GPATH= option in the ODS LISTING statement to specify the directory where your graphics files are saved. For example, if you want to save your graphic image files in the *C:\figures* directory, then you can specify the following:

```
ods listing gpath="C:\figures";
```

It is important to note that the GPATH= option applies only to ODS Graphics. It does not affect the behavior of graphs that are created by traditional SAS/GRAPH procedures.

HTML Destination

If you are using the HTML destination, the individual graphs are created as PNG files by default. You can use the PATH= and GPATH= options in the ODS HTML statement to specify the directory where your HTML and graphics files, respectively, are saved. These options also give you more control over your graphs. For example, if you want to save your HTML file named *test.htm* in the *C:\myfiles* directory, but you want to save your graphic image files in *C:\myfiles\png*, then you can specify the following:

```
ods html path  = "C:\myfiles"
         gpath = "C:\myfiles\png"
         file  = "test.htm";
```

When you specify the URL= suboption along with the GPATH= option, SAS creates relative paths for the links and references to the graphic image files in the HTML file. This is useful for building output files that are easily moved from one location to another. For example, the following statements create a relative path to the *png* directory in all the links and references that the file *test.htm* contains:

```
ods html path  = "C:\myfiles"
         gpath = "C:\myfiles\png" (url="png/")
         file  = "test.htm";
```

If you do not specify the URL= suboption, SAS creates absolute paths that are hard-coded in the HTML file. These can cause broken links if you move the files. For more information, see the ODS HTML statement in the *SAS Output Delivery System: User's Guide*.

LATEX Destination (Experimental)

LaTeX is a document preparation system for high-quality typesetting of mathematical and scientific material. The ODS LATEX statement produces output in the form of a LaTeX source file that is ready to compile in LaTeX. When you request ODS Graphics for a LATEX destination,[2] ODS creates the requested graphs as PostScript files by default, and the LaTeX source file includes references to these graphic image files. You can compile the LaTeX file, or you can ignore this file and simply access the individual PostScript files to include

[2]The LATEX destination is experimental.

your graphs in a different LaTeX document, such as a paper that you are writing. You can specify the PATH= and GPATH= options in the ODS LATEX statement, as explained previously for the ODS HTML statement; see Example 21.3 for an illustration. The ODS LATEX statement is an alias for the ODS MARKUP statement with the TAGSET=LATEX option. For more information, see the *SAS Output Delivery System: User's Guide*.

The default image file type for the LATEX destination is PostScript. When you use LaTeX to compile your document, the graphics format for included images is PostScript. However, if you prefer to use pdfLaTeX, you can specify a different format, such as JPEG, PDF, or PNG, any of which can be directly included in your pdfLaTeX document. To specify one of these formats, you use the OUTPUTFMT= option in the ODS GRAPHICS statement. For more information, see the LaTeX documentation for the `graphicx` package.

Creating Graphs in Multiple Destinations

This section illustrates how to send your output to more than one destination by using a single execution of your SAS statements. For example, to create LISTING, HTML, and RTF output, you can specify the ODS LISTING, ODS HTML, and ODS RTF statements before your procedure statements. The ODS _ALL_ CLOSE statement closes all open destinations before and after the other statements are run.

```
ods _all_ close;
ods listing;
ods html;
ods rtf;

. . .

ods _all_ close;
```

You can also specify multiple instances of the same destination. For example, using the data in the section "Contour and Surface Plots with PROC KDE" on page 610, the following statements save the contour plot to the file *contour.pdf* and the surface plot to the file *surface.pdf*:

```
ods _all_ close;
ods pdf file="contour.pdf";
ods pdf select ContourPlot;
ods pdf(id=srf) file="surface.pdf";
ods pdf(id=srf) select SurfacePlot;
ods graphics on;

proc kde data=bivnormal;
   ods select ContourPlot SurfacePlot;
   bivar x y / plots=contour surface;
run;

ods _all_ close;
```

The ID= option assigns the name **srf** to the second instance of the PDF destination. Without the ID= option, the second ODS PDF statement closes the destination that was opened by the first ODS PDF statement, and it opens a new instance of the PDF destination. In that case, the file *contour.pdf* is not created. For more information, see the ODS PDF statement in the *SAS Output Delivery System: User's Guide*.

Graph Size and Resolution

ODS provides options for specifying the size and resolution of graphs. You can specify the size of a graph in the ODS GRAPHICS statement and the resolution in an ODS destination statement. There are two other ways to change the size of a graph, but they are rarely needed. The three methods are as follows:

- Usually, you specify the WIDTH= or HEIGHT= option (or both) in the ODS GRAPHICS statement to change the size of a graph.
- You can modify the size of a particular graph by specifying the dimensions in the DESIGNHEIGHT= and DESIGNWIDTH= options in the BEGINGRAPH statement in the template. Some templates contain the specification DESIGNWIDTH=DEFAULTDESIGNHEIGHT, which sets the width of the graph to the default height, or DESIGNHEIGHT=DEFAULTDESIGNWIDTH, which sets the height of the graph to the default width.
- You can modify the size of all your ODS graphs by specifying the dimensions in the OUTPUTHEIGHT= and OUTPUTWIDTH= options in the style template.

The following examples show typical ways to change the size of your graphs:

```
ods graphics on / width=6in;
ods graphics on / height=4in;
ods graphics on / width=4.5in height=3.5in;
```

The dimensions of the graph can be specified in pixels (such as 200PX), inches (such as 3IN), or centimeters (such as 8CM). The default dimensions of ODS Graphics are 640 pixels wide and 480 pixels high, and these values determine the default aspect ratio. The actual size of the graph in inches depends on your printer or display device. For example, if the resolution of your printer is 100 dots per inch (DPI) and you want a graph that is 4 inches wide, you should set the width to 400 pixels.

If you specify only one dimension, the other dimension is determined by the default aspect ratio—that is, height = $0.75 \times$ width. For best results, you should create your graphs by using the exact size that is used to display the graphs in your paper or presentation. In other words, avoid generating graphs at one size and then expanding or shrinking them for inclusion in your document.

By default, fonts and symbol markers are automatically scaled along with the size of the graph. You can suppress this scaling by specifying the SCALE= option, as in the following example:

```
ods graphics on / scale=off;
```

The default resolution of graphs that are created in the HTML and LISTING destinations is 96 DPI, whereas the default for the RTF destination is 200 DPI. The 200 DPI value is recommended if you are copying and pasting graphs into a Microsoft PowerPoint presentation or a Microsoft Word document. Graphs in SAS/STAT documentation are usually generated at 300 DPI for display in PDF and 96 DPI for display in HTML.

You can change the resolution by using the IMAGE_DPI= option in any ODS destination statement, as in the following example:

```
ods html image_dpi=300;
```

An increase in resolution often improves the quality of the graphs, but it also greatly increases the size of the image file. Going from 96 DPI to 300 DPI increases the size of the image file by approximately a factor of $(300/96)^2 = 9.77$. Even when you are using a higher resolution for most of your graphs, you should consider using a lower resolution for some, such as contour plots, that create large files even at a lower resolution.

If you increase the resolution, you might need to compensate by reducing the size of the graph, as in the following example:

```
ods graphics on / width=4.5in  height=3.5in;
```

Increasing resolution also increases the amount of memory that is needed for your program to run. You can increase the amount of memory available to ODS Graphics by specifying an option when you invoke SAS, as in the following example:

```
-jreoptions '(-Xmx256m)'
```

You can modify the default amount of memory available to ODS Graphics by changing JREOPTIONS in your SAS configuration file to the settings –Xmx*nnn*m –Xms*nnn*m, where *nnn* is the amount of memory in megabytes. An example is –Xmx256m –Xms256m. In either case, the exact syntax varies depending on your operating system, and the amount of memory that you can allocate varies from system to system. For more information, see the *SAS Companion* for your operating system.

ODS Graphics Editor

The ODS Graphics Editor is a point-and-click interface that you can use to modify a specific graph created by ODS Graphics. For example, if you need to enhance a graph for a paper or presentation, you can use the ODS Graphics Editor to customize the title, modify the axis labels, annotate particular data points, and change graph element properties such as fonts, colors, and line styles.

This section explains how to enable ODS Graphics to create editable graphs and how to invoke the ODS Graphics Editor. You can use the ODS Graphics Editor in the SAS windowing environment, provided that the LISTING destination is open and you have first enabled ODS Graphics to create editable graphs. **NOTE:** The LISTING destination is usually open by default. There are three steps that you must take to edit a graph:

1 First enable the creation of editable graphs in one of three ways:
   - Use an ODS statement to temporarily enable this feature.
   - Use a SAS command to temporarily enable this feature.
   - Use the SAS Registry Editor to permanently enable this feature.

   Creating editable graphs requires additional resources, so you might not want to permanently enable this feature.

2 Submit your SAS code and create editable graphs.

3 Invoke the ODS Graphics Editor and edit the plot.

Step 2 involves submitting SAS code in the usual way; no special instructions are needed for creating graphs that can be edited. Steps 1 and 3 are explained in more detail in the following sections.

Enabling the Creation of Editable Graphs

Temporarily Enable Creation of Editable Graphs by Using an ODS Statement

You can enable the creation of editable graphs within a SAS session by submitting one of the following statements:

```
ods listing sge=on;
ods html sge=on;
```

You can disable the creation of editable graphs by submitting one of the following statements:

```
ods listing sge=off;
ods html sge=off;
```

Permanently Enable Creation of Editable Graphs across SAS Sessions

You can create a default setting that enables or disables the creation of editable graphs across SAS sessions by using the "ODS Graphics Editor" setting in the SAS Registry. You can change this setting in the SAS windowing environment as follows:

1 Open the Registry Editor by entering **regedit** on the command line.

2 Select **SAS_REGISTRY** ► **ODS** ► **GUI** ► **RESULTS**.

3 Click **ODS Graphics Editor** to open the Edit String Value window. In the **Value Data** field, type **On** to enable the creation of editable graphs or **Off** to disable it.

4 Click **OK**.

Editing a Graph with the ODS Graphics Editor

The ODS Graphics Editor is illustrated using the following example:

```
data growth;
   length Country $ 20;
   input country &$ GDP LFG EQP NEQ GAP;
   datalines;
Argentina              0.0089 0.0118 0.0214 0.2286 0.6079
Austria                0.0332 0.0014 0.0991 0.1349 0.5809

   ... more lines ...

Zambia                -0.0110 0.0275 0.0702 0.2012 0.8695
Zimbabwe               0.0110 0.0309 0.0843 0.1257 0.8875
;

ods graphics on;
ods html sge=on;

proc robustreg data=growth plots=(ddplot histogram);
   model GDP  = LFG GAP EQP NEQ / diagnostics leverage;
   output out=robout r=resid sr=stdres;
run;

ods _all_ close;
ods html;
```

The DATA and PROC ROBUSTREG steps are submitted to the SAS System, in this case from the SAS windowing environment, as shown in Figure 21.17. Two versions of the graph are created: one in an uneditable PNG file (for example, *DDPlot.png*) and one in an editable SGE file (for example, *DDPlot.sge*). Both are saved in the SAS current folder. You can edit the graph in one of three ways:

- In the Results window, double-click the second graph icon for the graph that you want to edit (see Figure 21.17). The second graph icon corresponds to the SGE file, and the first graph icon corresponds to the PNG file. Clicking the first graph icon invokes a host-dependent graph viewer (for example, Microsoft Photo Editor in Windows), not the ODS Graphics Editor. **NOTE:** The ODS Graphics Editor window might be hidden behind other windows in the SAS windowing environment.

- You can edit the graph by selecting it in the SAS Explorer window. You must first navigate to the SAS current folder and to the SGE files.

- You can open the graph from outside the SAS System. For example, if you are running SAS on the Windows operating system, you can click on the graph's SGE file to open it by using the ODS Graphics Editor.

Figure 21.17 Results Window with Icons for Editable Plots

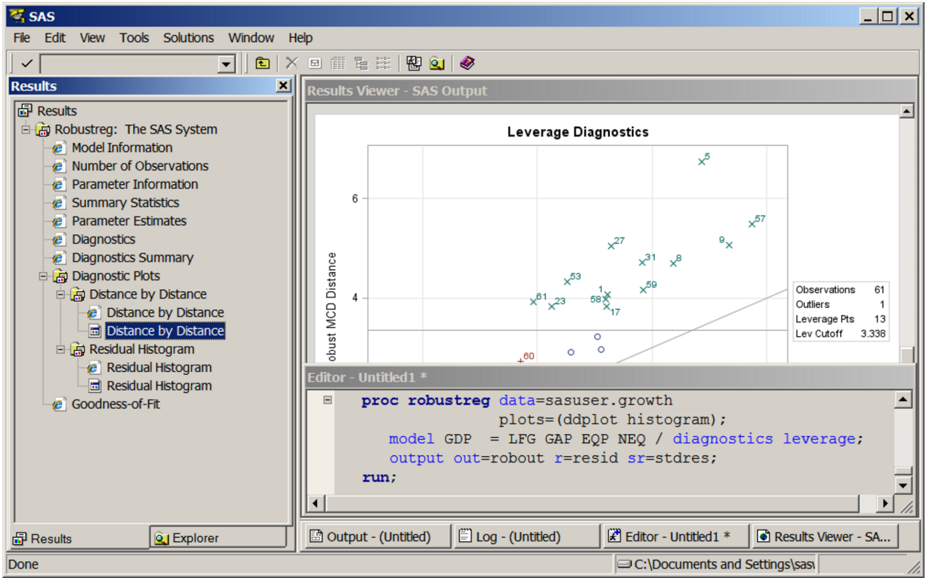


Figure 21.18 shows the ODS Graphics Editor window for the editable diagnostic plot that PROC ROBUSTREG creates. In Figure 21.19, various tools in the ODS Graphics Editor are used to modify the title and annotate a particular point. You can save the edited plot as a PNG file or an SGE file by selecting **File ► Save As**. After saving the plot, you can edit it again through the SAS Explorer window or by selecting **File ► Open** from the ODS Graphics Editor window. Alternatively, you can reopen the saved plot for editing without first invoking the SAS System. For example, if you are running SAS on the Windows operating system, you can click on the plot to open it by using the ODS Graphics Editor.

The ODS Graphics Editor does not permit you to make structural changes to a graph (such as moving the positions of data points). The ODS Graphics Editor provides you with a point-and-click way to make one-time changes to a specific graph, whereas the template language (see the section "Graph Templates" on page 782 in Chapter 22, "ODS Graphics Template Modification") provides you with a programmatic way to make template changes that persist every time you run the procedure. For complete information about the tools available in the ODS Graphics Editor, see *SAS ODS Graphics Editor: User's Guide*.

Figure 21.18 Diagnostic Plot before Editing

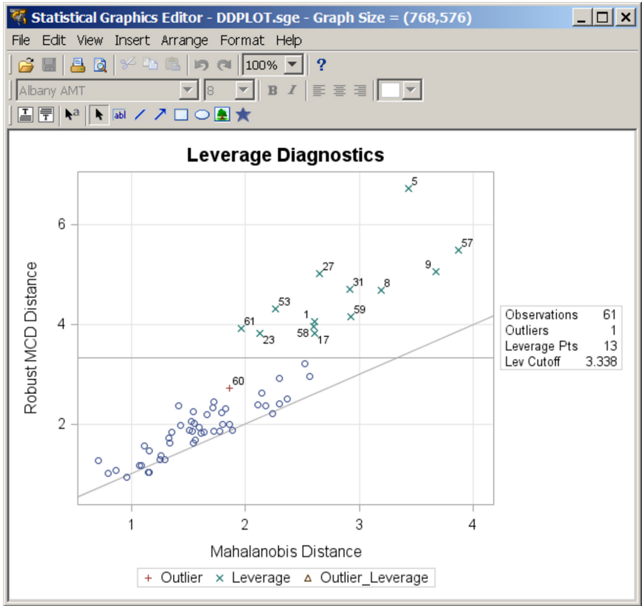
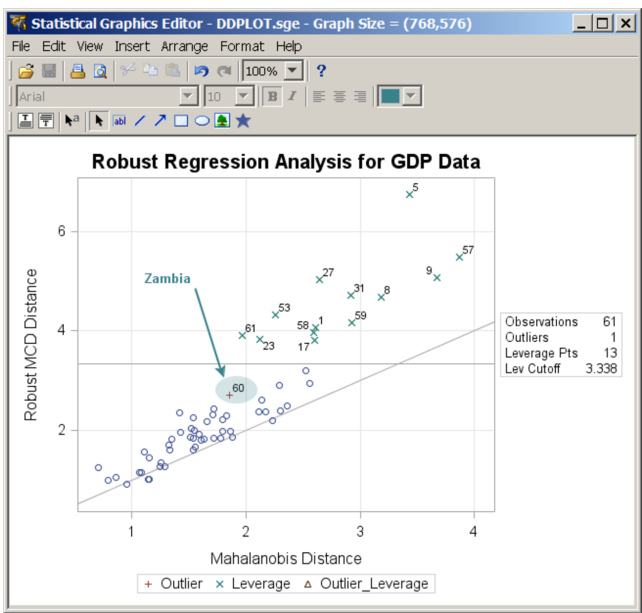


Figure 21.19 Diagnostic Plot after Editing



The Default Template Stores and the Template Search Path

Compiled templates are stored in a template store, which is a type of item store. (An item store is a special type of SAS file.) You can see the list of template stores by submitting the following statement:

```
ods path show;
```

The results are as follows:

```
Current ODS PATH list is:

1. SASUSER.TEMPLAT(UPDATE)
2. SASHELP.TMPLMST(READ)
```

These results show that the default template search path consists of Sasuser.Templat followed by Sashelp.Tmplmst. You can add template stores that you create, or you can change the order in which the template stores are searched. For more information, see the sections "Saving Customized Templates" on page 791, "Using Customized Templates" on page 791, and "Reverting to the Default Templates" on page 792 in Chapter 22, "ODS Graphics Template Modification," and the sections "The ODS PATH Statement" on page 544 and "Controlling Output Appearance with Templates" on page 546 in Chapter 20, "Using the Output Delivery System."

This section discusses the default template stores that you use when you have not modified the template search path by using the ODS PATH statement. By default, templates that you write are stored in Sasuser.Templat. If you store a modified template in Sasuser.Templat, ODS finds and uses your modified template. Otherwise, ODS finds the templates that it provides in Sashelp.Tmplmst. You can see a list of all the templates that you have modified as follows:

```
proc template;
   list / store=sasuser.templat;
run;
```

You can delete any template that you modified (so that ODS finds the default SAS template) by specifying it in a DELETE statement, as in the following statement:

```
proc template;
   delete Stat.REG.Graphics.ResidualPlot / store=sasuser.templat;
run;
```

Specifying STORE=SASUSER.TEMPLAT is not required. However, if you have administrator privileges on your computer, this option helps you ensure that you do not accidentally delete templates from Sashelp.Tmplmst. Unless you have administrator privileges, ODS never deletes a template in Sashelp.Tmplmst, so you can safely run the preceding step without the STORE= option, even if the template that you specify does not exist in Sasuser.Templat. You can run the following step to delete the entire Sasuser.Templat store of customized templates so that ODS uses only the templates that SAS supplies:

```
ods path sashelp.tmplmst(read);
proc datasets library=sasuser nolist;
   delete templat(memtype=itemstor);
run;
ods path sasuser.templat(update) sashelp.tmplmst(read);
```

It is good practice to delete templates that you have customized when you are done using them, so that they are not unexpectedly used later. For more information, see the section "Reverting to the Default Templates" on page 792 in Chapter 22, "ODS Graphics Template Modification."

ODS Styles

ODS styles control the overall appearance of your output. Usually, the only thing that you need to do with styles is specify them in an ODS destination statement, as in the following example:

```
ods html body='b.html' style=HTMLBlue;
```

However, you can also modify existing styles and even write your own styles. You can also specify style elements in custom templates that you write, or you can modify which style elements are used in templates that SAS supplies. This section provides an overview of ODS styles and style elements, which are the components of a style. It also describes how to customize a style template and how to specify a default style for your output. Only the most commonly used styles, style elements, and style changes are discussed here. For complete information about ODS styles, see the *SAS Output Delivery System: User's Guide*.

An Overview of ODS Styles

An ODS style template provides formatting information for specific visual aspects of your SAS output (see the section "ODS Style Elements and Attributes" on page 665). The appearance of tables and graphs is coordinated within a particular style. For tables, this information includes a list of fonts and a list of colors. Each font definition specifies a family, size, weight, and style. Colors are associated with common areas of output, including titles, footnotes, BY groups, table headings, and table cells. For graphs, ODS styles also control the appearance of graph elements, including lines, markers, fonts, and colors. ODS styles also include elements specific to statistical graphics, such as the style of fitted lines, confidence bands, and prediction limits. For more information about ODS styles, see Kuhfeld (2016) and the *SAS Output Delivery System: User's Guide*.

You can specify a style by using the STYLE= option in an ODS destination statement such as HTML, PDF, RTF, or PRINTER. You can also specify a style in the LISTING destination; however, this style affects graphs but not tables. Output that is produced by using different styles has the same content but a different appearance. For example, the following statement requests output that is produced by using the JOURNAL style:

```
ods rtf style=Journal;
```

You can use any ODS style or any style that you define yourself. The following statements list the names of all the styles and then display five of them:

```
proc template;
   list styles;
   source Styles.Default;
   source Styles.Statistical;
   source Styles.Journal;
   source Styles.RTF;
   source Styles.HTMLBlue;
run;
```

The results of this step (not shown) include a list of more than 50 styles in the SAS listing and five style templates in the SAS log. Style templates are often hundreds of lines long. For more information about style templates, see the section "Style Templates and Colors" on page 667.

Each ODS destination has its own default style, as shown in Table 21.1 and mentioned throughout this section. Most graphs in SAS/STAT documentation use the HTMLBLUE style. However, throughout this chapter, you can see examples of other styles. For more information about styles, see the *SAS Output Delivery System: User's Guide*.

The rest of this section describes a few ODS styles. There are many more ODS styles than are listed here. Most styles are not designed for statistical work. The following styles are used most often for statistical work.

- The HTMLBLUE style is a modern color style that is recommended for use in web pages or color print media. (See Figure 21.24 for an example.) The HTMLBLUE style is an ATTRPRIORITY='Color' style. The HTMLBLUE style inherits most of its attributes from the STATISTICAL style, which inherits some of its attributes from the DEFAULT style. The HTMLBLUE style has a brighter appearance than its parents, and it has color coordination between the tables and graphs. The dominant color is blue.

  The HTMLBLUE style is the default style for the HTML destination. It is also the default style in SAS/STAT documentation for tables displayed in the HTML format and graphs displayed in both the PDF and HTML formats.[3]

  Output that you create by using the HTMLBLUE style might not print well on black-and-white devices (particularly when you create graphs that have groups of observations). If you need an alternative to the HTMLBLUE style that varies colors, lines, and markers, use the HTMLBLUECML style or most other styles. If you need an alternative to the HTMLBLUE style that is designed for printer destinations such as PRINTER, PDF, PS, and RTF, see the PEARL, PEARLJ, and SAPPHIRE styles.

- The HTMLBLUECML style is a modern color style that is recommended for use in web pages or color print media. (See Figure 21.25 for an example.) It inherits most of its attributes from the HTMLBLUE style. The dominant color is blue. The HTMLBLUECML style is an ATTRPRIORITY='None' style. In graphs, groups of observations are distinguished by simultaneous color, line style, and symbol changes. If you need an alternative to the HTMLBLUECML style that is all color, use the HTMLBLUE style instead. Output that you create by using the HTMLBLUECML style might not print well on black-and-white devices.

- The PEARL style is a modern color style that is recommended for use in documents that are created by using printer destinations such as PRINTER, PDF, PS, and RTF. Graphs are displayed in color, and tables are displayed in black and white. (See Figure 21.31 for an example.) The PEARL style shares most of its attributes with the SAPPHIRE style. Both styles inherit most of their attributes from the HTMLBLUE style; hence the dominant color is blue. The differences are outside the graphs (in the tables and in the page background). The HTMLBLUE style has a very light blue background, and the PEARL style has a white background. The PEARL style also has a white background for row and column table headings, whereas the SAPPHIRE style has a light blue background. The PEARL and SAPPHIRE styles use fonts that are appropriate for printer destinations. The PEARL style is an ATTRPRIORITY='Color' style. In graphs, groups of observations are distinguished by color. Output that you create by using the PEARL style might not print well on black-and-white devices (particularly when you create graphs that have groups of observations).

- The PEARLJ style is a modern style that is recommended for use in documents that are created by using printer destinations such as PRINTER, PDF, PS, and RTF. Graphs are displayed in color, and tables are displayed in black and white. The PEARLJ style inherits all of its graphical attributes and most of its other attributes from the PEARL style. The two styles display tables differently. The PEARLJ style uses smaller fonts and less cell padding, and it displays horizontal but not vertical

[3] The PEARLJ style is the default style for the PDF tables displayed in SAS/STAT documentation.

lines. The PEARLJ style is the default style for tables that are displayed in the PDF version of SAS/STAT documentation. The PEARLJ style is an ATTRPRIORITY='Color' style. In graphs, groups of observations are distinguished by color. Output that you create by using the PEARL style might not print well on black-and-white devices (particularly when you create graphs that have groups of observations).

- The SAPPHIRE style is a modern color style that is recommended for use in documents that are created by using printer destinations such as PRINTER, PDF, PS, and RTF. (See Figure 21.32 for an example.) The SAPPHIRE style shares most of its attributes with the PEARL style. Both styles inherit most of their attributes from the HTMLBLUE style; hence the dominant color is blue. However, unlike the HTMLBLUE style (which has a very light blue background), the SAPPHIRE style has a white background. The SAPPHIRE style has a light blue background for row and column table headings, whereas the PEARL style has a white background. The SAPPHIRE and PEARL styles use fonts that are appropriate for printer destinations. The SAPPHIRE style is an ATTRPRIORITY='Color' style. In graphs, groups of observations are distinguished by color. Output that you create by using the SAPPHIRE style might not print well on black-and-white devices (particularly when you create graphs that have groups of observations).

- The JOURNAL family of styles (JOURNAL, JOURNAL2, and JOURNAL3) consists of black-and-white or grayscale styles that are recommended for graphs that appear in journals and other black-and-white publications. (See Figure 21.28 for an example of the JOURNAL style, see Figure 21.9 for an example of the JOURNAL2 style, and see Example 21.3 for a comparison of the three styles.) The JOURNAL1A, JOURNAL2A, and JOURNAL3A styles inherit most of their attributes from the JOURNAL, JOURNAL2, and JOURNAL3 styles, respectively, but use fewer italic fonts. For color alternatives to the JOURNAL family of styles, see the PEARL, PEARLJ, and SAPPHIRE styles.

- The DEFAULT style is a legacy color style. (See Figure 21.23 for an example.) Most other styles inherit some of their elements from this style. The DEFAULT style was the default style for the HTML destination in earlier SAS releases. The dominant color is gray. The DEFAULT style is an ATTRPRIORITY='None' style. In graphs, groups of observations are distinguished by simultaneous color, line style, and symbol changes. Output that you create by using the DEFAULT style might not print well on black-and-white devices.

- The STATISTICAL style is a legacy color style. (See Figure 21.26 for an example.) The STATISTICAL style inherits elements from the DEFAULT style, and it is similar in some ways (other than color) to the ANALYSIS style. The dominant colors are blue and gray. The STATISTICAL style is an ATTRPRIORITY='None' style. In graphs, groups of observations are distinguished by simultaneous color, line style, and symbol changes. Output that you create by using the STATISTICAL style might not print well on black-and-white devices.

- The ANALYSIS style is a legacy color style. (See Figure 21.27 for an example.) The ANALYSIS style inherits elements from the DEFAULT style, and it is similar in some ways (other than color) to the STATISTICAL style. The dominant colors are green and yellow. The ANALYSIS style is an ATTRPRIORITY='None' style. In graphs, groups of observations are distinguished by simultaneous color, line style, and symbol changes. Output that you create by using the ANALYSIS style might not print well on black-and-white devices.

- The RTF style is a legacy color style that is designed for graphs that will be inserted into a Microsoft Word document or PowerPoint slide. (See Figure 21.30 for an example of the RTF style, which is the default style for the RTF destination.) The RTF style inherits elements from the DEFAULT style. The dominant color is gray. The ANALYSIS style is an ATTRPRIORITY='None' style. In graphs, groups of observations are distinguished by simultaneous color, line style, and symbol changes. Output that you create by using the RTF style might not print well on black-and-white devices. For alternatives to the RTF style that are brighter and less gray, see the PEARL, PEARLJ, and SAPPHIRE styles.

- The LISTING style is a legacy color style that is similar to the DEFAULT style, but the LISTING style has a lighter background. (See Figure 21.29 for an example.) It is the default style for the LISTING destination. The LISTING style inherits elements from the DEFAULT style. The dominant colors are black and white. The LISTING style is an ATTRPRIORITY='None' style. In graphs, groups of observations are distinguished by simultaneous color, line style, and symbol changes. Output that you create by using the LISTING style might not print well on black-and-white devices.

Attribute Priority and Overriding How Groups Are Distinguished

ODS styles have an ATTRPRIORITY= option with the specification ATTRPRIORITY='Color' or ATTRPRIORITY='None'. The ODS GRAPHICS statement has an ATTRPRIORITY= option with the specification ATTRPRIORITY=COLOR or ATTRPRIORITY=NONE. The default attribute priority for the style option is 'None'. The default attribute priority for the ODS GRAPHICS statement option is the attribute priority from the style. The ODS styles that distinguish groups by color alone are ATTRPRIORITY='Color' styles. The ODS styles that distinguish groups by varying colors, markers, and line patterns are ATTRPRIORITY='None' styles. The ATTRPRIORITY='Color' styles include HTMLBLUE, PEARL, PEARLJ, and SAPPHIRE. Most other styles are ATTRPRIORITY='None' styles. (For more information about styles and attribute priority, see the section "ODS Styles" on page 624.) You can override the ATTRPRIORITY= option in all styles by specifying one of the following statements:

```
ods graphics on / attrpriority=color;
ods graphics on / attrpriority=none;
```

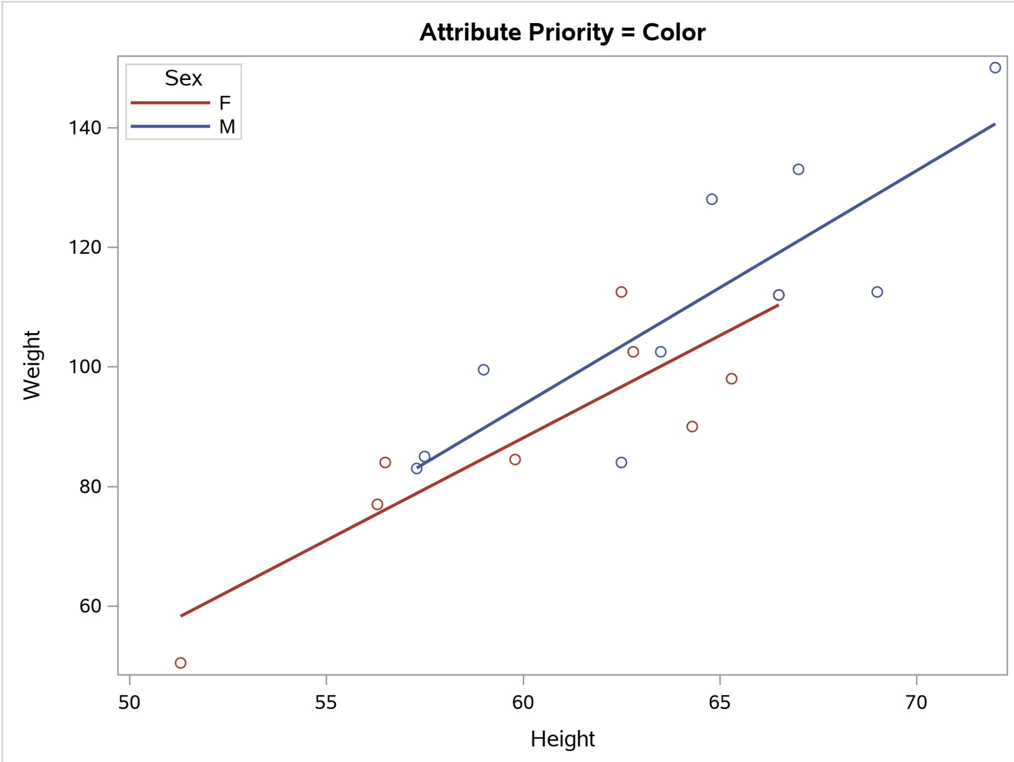
You can reset the default ODS Graphics options as follows:

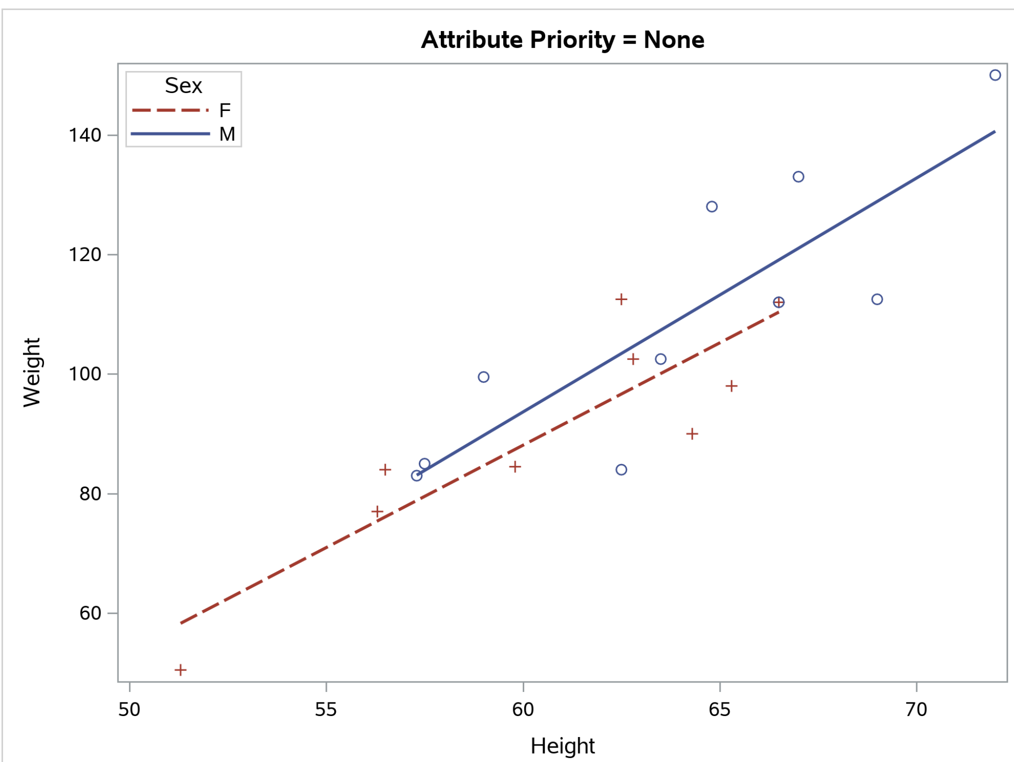
```
ods graphics on / reset;
```

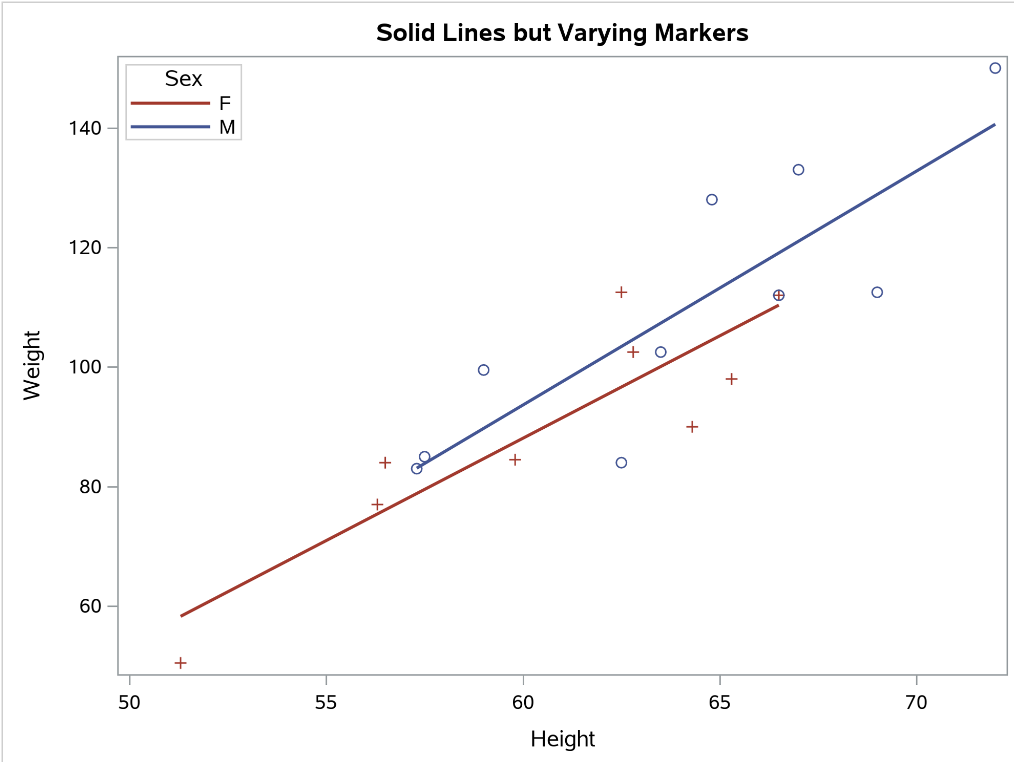
The following steps illustrate three ways to make graphs and create the graphs in Figure 21.20:

```
ods graphics on / attrpriority=color;
proc sgplot data=sashelp.class;
   title 'Attribute Priority = Color';
   reg y=weight x=height / group=sex;
   keylegend / location=inside position=topleft across=1;
run;

ods graphics on / attrpriority=none;
proc sgplot data=sashelp.class;
   title 'Attribute Priority = None';
   reg y=weight x=height / group=sex;
   keylegend / location=inside position=topleft across=1;
run;

proc sgplot data=sashelp.class;
   title 'Solid Lines but Varying Markers';
   styleattrs datalinepatterns=(solid);
   reg y=weight x=height / group=sex;
   keylegend / location=inside position=topleft across=1;
run;
ods graphics on / reset;
```

Figure 21.20 Attribute Priority







The settings that are shown in the last graph, which has solid lines but varying markers, often work well when there are multiple groups. You might not need to vary the line style, but varying the markers makes it easier to distinguish the points. For example, it is easy to see in the last two graphs that one point corresponds to both a male and a female point.

You can modify the attribute priority in any style by specifying the ATTRPRIORITY= option, as in the following examples:

```
proc template;
   define style styles.Default;
      parent = styles.default;
      style Graph from Graph / attrpriority = "Color";
   end;
   define style styles.HTMLBlue;
      parent = styles.HTMLBlue;
      style Graph from Graph / attrpriority = "None";
   end;
run;
```

You can delete the modified style templates as follows:

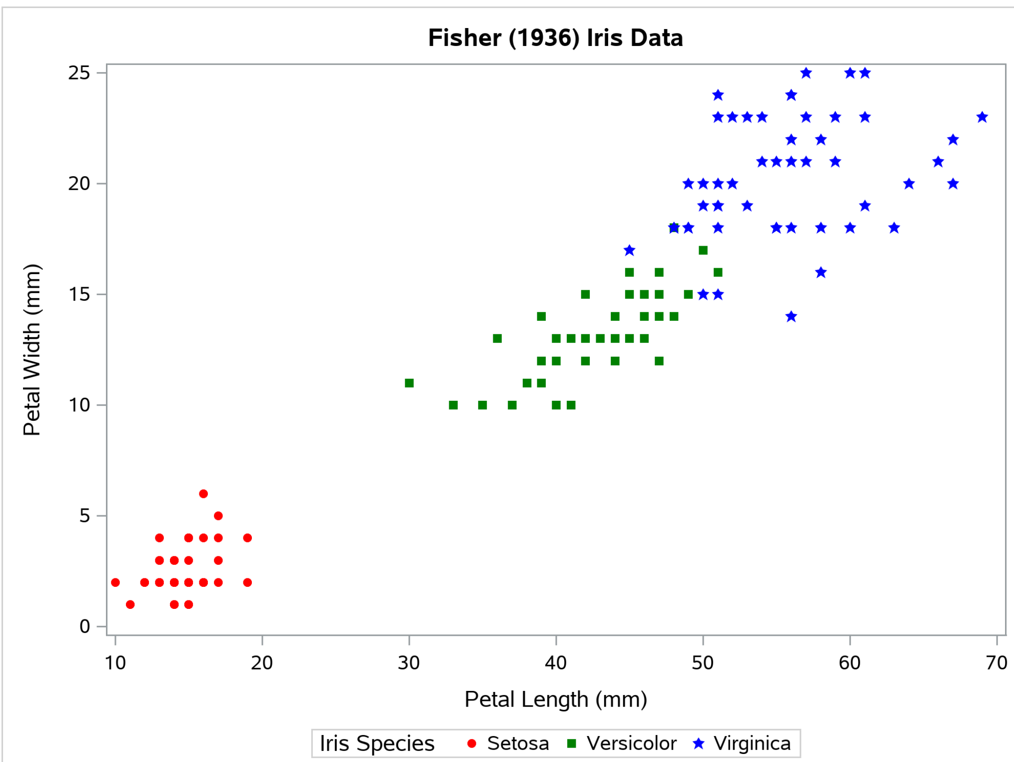
```
proc template;
   delete styles.Default  / store=sasuser.templat;
   delete styles.HTMLBlue / store=sasuser.templat;
run;
```

In the statistical graphics (SG) procedures, you can override the style attributes and modify how you distinguish groups by using the STYLEATTRS statement:

```
ods graphics on / attrpriority=none;

proc sgplot data=sashelp.iris;
   title 'Fisher (1936) Iris Data';
   styleattrs datasymbols=(circlefilled squarefilled starfilled)
              datacontrastcolors=(red green blue);
   scatter x=petallength y=petalwidth / group=species markerattrs=(size=5px);
run;

ods graphics / reset;
```

The results are displayed in Figure 21.21.

Figure 21.21 Iris Data



Note: Specifying more than one value in a STYLEATTRS statement option list works only when the attribute priority is 'None'. The STYLEATTRS statement overrides the default style elements, but it does not force the style elements `GraphData2`, `GraphData3`, and so on to be used.

The STYLEATTRS statement options are as follows:

DATACOLORS=(*color-list*)
: specifies the fill colors. You can specify common color names or values of the form CX*rrggbb*, where the last six characters specify RGB (red, green, blue) values on the hexadecimal scale of 00 to FF (0 to 255, base 10).

DATACONTRASTCOLORS=(*color-list*)
: specifies the contrast colors, which are used for lines and markers. You can specify common color names or values of the form CX*rrggbb*, where the last six characters specify RGB (red, green, blue) values on the hexadecimal scale of 00 to FF (0 to 255, base 10).

DATALINEPATTERNS=(*line-pattern-list*)
: specifies the list of line patterns. Some of the available line patterns are **Solid**, **MediumDash**, **MediumDashShortDash**, **LongDash**, **DashDashDot**, **LongDashShortDash**, **DashDotDot**, **Dash**, **ShortDashDot**, **MediumDashDotDot**, and **ShortDash**.

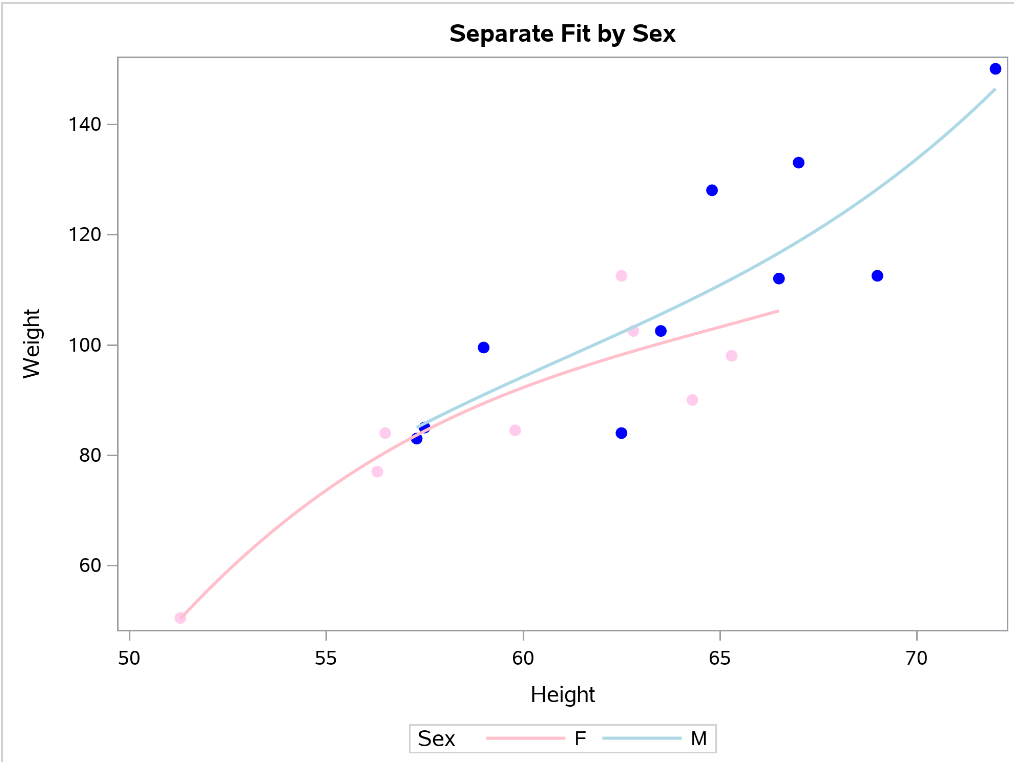
DATASYMBOLS=(*marker-symbol-list*)
: specifies the list of marker symbols. The available markers are **ArrowDown**, **Asterisk**, **Circle**, **CircleFilled**, **Diamond**, **DiamondFilled**, **GreaterThan**, **Hash**, **HomeDown**, **HomeDownFilled**, **Ibeam**, **LessThan**, **Plus**, **Square**, **SquareFilled**, **Star**, **StarFilled**, **Tack**, **Tilde**, **Triangle**, **TriangleDown**, **TriangleDownFilled**, **TriangleFilled**, **TriangleLeft**, **TriangleLeftFilled**, **TriangleRight**, **TriangleRightFilled**, **Union**, **X**, **Y**, and **Z**.

In the SG procedures, you can override the style attributes and modify how groups are distinguished by using attribute maps:

```
data myattrmap;
   retain ID 'Attr1' MarkerSymbol 'CircleFilled';
   input Value $ LineColor $ 3-11 MarkerColor $ 13-20;
   datalines;
F pink      cxFFCCEE
M lightblue blue
;

proc sgplot data=sashelp.class dattrmap=myattrmap;
   title 'Separate Fit by Sex';
   reg y=weight x=height / group=sex degree=3 attrid=Attr1;
run;
```

The results are displayed in Figure 21.22.

Figure 21.22 Color Changes with an Attribute Map



You can set the following variables and values in the PROC SGPLOT DATTRMAP=*SAS-data-set* option:

FillColor	color or style attribute
FillStyle	style element
ID	text string that contains the attribute map ID
LineColor	color or style attribute
LinePattern	line pattern or style attribute
LineStyle	style element
MarkerColor	color or style attribute
MarkerStyle	style element
MarkerSymbol	symbol name or style attribute
Value	text string that contains the group

ODS Style Elements and Attributes

An ODS style template is composed of a set of *style elements*. A style element is a collection of *style attributes* that applies to a particular feature or aspect of the output. A value is specified for each attribute in a style template. For example, `GraphFit` is the style element that is used for fit lines, and its attributes include `LineThickness`, `LineStyle`, `MarkerSize`, `MarkerSymbol`, `ContrastColor`, and `Color`.

In general, style templates control the overall appearance of ODS tables and graphs. For tables, style templates specify features such as background color, table borders, and color scheme, and they specify the fonts, sizes, and color for the text and values in a table and its headings. For graphs, style templates specify the following features:

- background color
- graph dimensions (height and width)
- borders
- line styles for axes and grid lines
- fonts, sizes, and colors for titles, footnotes, axis labels, axis values, and data labels (see the section "Modifying Graph Fonts in Styles" on page 709 for an illustration)
- marker symbols, colors, and sizes for data points and outliers
- line styles for needles
- line and curve styles for fitted models and predicted values (see the section "Modifying Other Graph Elements in Styles" on page 712 for an illustration)
- line and curve styles for confidence and prediction limits
- fill colors for histogram bars, confidence bands, and confidence ellipses
- colors for box plot features
- colors for surfaces
- color ramps for contour plots

The SAS System supplies a graph template for each graph that is created by statistical procedures. A graph template is a program that specifies the layout and details of a graph. For more information about templates, see the section "Graph Templates" on page 782 in Chapter 22, "ODS Graphics Template Modification." Some template options are specified by using a style reference of the form **`style-element`** or occasionally **`style-element:attribute`**. For example, the symbol, color, and size of markers for basic scatter plots are specified in a template SCATTERPLOT statement as follows:

```
scatterplot x=x y=y / markerattrs=GraphDataDefault;
```

The preceding statement specifies that the appearance for markers is controlled by the **`GraphDataDefault`** element. Consistent use of this element guarantees a common appearance of markers across all scatter plots, based on the style template that you are using.

In general, ODS Graphics features are determined by style element attributes unless they are overridden by a statement or option in the graph template. For example, suppose that a classification variable is specified in the GROUP= option in a SCATTERPLOT template statement as follows:

```
scatterplot x=X y=Y / group=GroupVar;
```

Then the colors for markers that correspond to the classification levels are assigned by using the style element attributes **`GraphData1:ContrastColor`** through **`GraphData12:ContrastColor`**.

Style templates are created and modified by using PROC TEMPLATE. For more information, see the *SAS Output Delivery System: User's Guide*. You need to understand the relationships between style elements and graph features if you want to create your own style template or modify a style template. These relationships are explained in the following sections.

Style Templates and Colors

The default style templates that the SAS System provides are stored in the *Styles* directory of Sashelp.Tmplmst. You can display, edit, and save style templates by using the same that are methods available for modifying graph and table templates, as explained in the section "The Default Template Stores and the Template Search Path" on page 655 and the series of sections beginning with the section "Displaying Templates" on page 788 in Chapter 22, "ODS Graphics Template Modification." In particular, you can display a style template by using one of the following methods:

- Open the Templates window by entering the command **odst** on the command line of the SAS windowing environment, expand the Sashelp.Tmplmst node under **Templates**, and then select **Styles** to display the contents of this folder.
- Use the SOURCE statement in PROC TEMPLATE.

For example, the following statements display the DEFAULT style template in the SAS log:

```
proc template;
   source Styles.Default;
run;
```

Some of the results are as follows:

```
define style Styles.Default;
   . . .
   class GraphColors
      "Abstract colors used in graph styles" /
      . . .
      'gconramp3cend' = cxFF0000
      'gconramp3cneutral' = cxFF00FF
      'gconramp3cstart' = cx0000FF
      . . .
      'gdata12' = cxDDD17E
      'gdata11' = cxB7AEF1
      'gdata10' = cx87C873
      'gdata9' = cxCF974B
      'gdata8' = cxCD7BA1
      'gdata6' = cxBABC5C
      'gdata7' = cx94BDE1
      'gdata4' = cxA9865B
      'gdata5' = cxB689CD
      'gdata3' = cx66A5A0
      'gdata2' = cxDE7E6F
      'gdata1' = cx7C95CA;
   . . .
```

The first part of this list shows that the shading for certain filled plots, such as some contour plots, goes from blue (`'gconramp3cstart' = cx0000FF`) to magenta (`'gconramp3cneutral' = cxFF00FF`) to red (`'gconramp3cend' = cxFF0000`). All colors are specified in values of the form CX*rrggbb*, where the last six characters specify RGB (red, green, blue) values on the hexadecimal scale of 00 to FF (0 to 255, base 10). The second part of the list (`'gdata1' = cx7C95CA`) shows that the dominant component of the

GraphData1 color is blue because the blue component of the color (CA, which corresponds to 202, base 10) is greater than both the green component (95, which corresponds to 149, base 10) and the red component (7C, which corresponds to 124, base 10).

You can change any part of the style and then submit the style back to the SAS System, after first submitting a PROC TEMPLATE statement. For more information about modifying, using, and restoring templates, see the sections "Saving Customized Templates" on page 791, "Using Customized Templates" on page 791, and "Reverting to the Default Templates" on page 792 in Chapter 22, "ODS Graphics Template Modification." The principles that are discussed in those sections apply to all templates—table, style, and graph.

Some Common ODS Style Elements

This section explains some common ODS style elements and produces most of the graphs that are displayed in the section "ODS Style Comparisons" on page 673.

The DEFAULT style is the parent for the styles that are used for statistical graphics work. You can see all the elements of the DEFAULT style by running the following step:

```
proc template;
   source styles.default;
run;
```

The source listing of the definition of the DEFAULT style is hundreds of lines long. If you run PROC TEMPLATE along with the SOURCE statement for most other styles, you see **parent = styles.default** (or in the case of the HTMLBLUE style, you see **parent = styles.statistical**, which inherits attributes from the DEFAULT style), and you do not see all the elements in the style unless you also add a slash and the EXPAND option to the SOURCE statement.

Only a few of the style elements are referenced in the templates that the SAS System provides for statistical procedures. The most commonly used style elements, along with the defaults for the noncolor attributes of the DEFAULT style, are shown next (**Color** applies to filled areas, and **ContrastColor** applies to markers and lines):

Graph	graph size, outer border appearance, and background color
	Padding = 0
	BackgroundColor
GraphConfidence	primary fit confidence interval
	LineThickness = 1px
	LineStyle = 1
	MarkerSize = 7px
	MarkerSymbol = "triangle"
	ContrastColor
	Color
GraphData1	attributes related to first grouped data items
	MarkerSymbol = "circle"
	LineStyle = 1
	ContrastColor
	Color

GraphData2	attributes related to second grouped data items **MarkerSymbol = "plus"** **LineStyle = 4** **ContrastColor** **Color**
GraphData3	attributes related to third grouped data items **MarkerSymbol = "X"** **LineStyle = 8** **ContrastColor** **Color**
GraphData4	attributes related to fourth grouped data items **MarkerSymbol = "triangle"** **LineStyle = 5** **ContrastColor** **Color**
GraphData*n*	attributes related to *n*th grouped data items **MarkerSymbol** **LineStyle** **ContrastColor** **Color**
GraphDataDefault	attributes related to ungrouped data items **EndColor** **NeutralColor** **StartColor** **MarkerSize = 7px** **MarkerSymbol = "circle"** **LineThickness = 1px** **LineStyle = 1** **ContrastColor** **Color**
GraphFit	primary fit line or a normal density curve **LineThickness = 2px** **LineStyle = 1** **MarkerSize = 7px** **MarkerSymbol = "circle"** **ContrastColor** **Color**
GraphFit2	secondary fit line or a kernel density curve **LineThickness = 2px** **LineStyle = 4** **MarkerSize = 7px** **MarkerSymbol = "X"** **ContrastColor** **Color**

GraphGridLines	horizontal and vertical grid lines drawn at major tick marks **Displayopts = "auto"** **LineThickness = 1px** **LineStyle = 1** **ContrastColor** **Color**
GraphOutlier	outlier data for the graph **LineThickness = 2px** **LineStyle = 42** **MarkerSize = 7px** **MarkerSymbol = "circle"** **ContrastColor** **Color**
GraphPredictionLimits	fills for prediction limits **LineThickness = 1px** **LineStyle = 2** **MarkerSize = 7px** **MarkerSymbol = "chain"** **ContrastColor** **Color**
GraphReference	horizontal and vertical reference lines and drop lines **LineThickness = 1px** **LineStyle = 1** **ContrastColor**
GraphDataText	text font and color for point and line labels **Font = GraphFonts('GraphDataFont')** (where **'GraphDataFont' =** **("<sans-serif>, <MTsans-serif>",7pt))** **Color**
GraphValueText	text font and color for axis tick values and legend values **Font = GraphFonts('GraphValueFont')** (where **'GraphValueFont' =** **("<sans-serif>, <MTsans-serif>",9pt)**) **Color**
GraphLabelText	text font and color for axis labels and legend title **Font = GraphFonts('GraphLabelFont')** (where **'GraphLabelFont' =** **("<sans-serif>, <MTsans-serif>",10pt,bold)**) **Color**
GraphFootnoteText	text font and color for footnotes **Font = GraphFonts('GraphFootnoteFont')** (where **'GraphFootnoteFont' =** **("<sans-serif>, <MTsans-serif>",10pt)**) **Color**

GraphTitleText	text font and color for titles **Font = GraphFonts('GraphTitleFont')** (where **'GraphTitleFont' = ("<sans-serif>, <MTsans-serif>",11pt,bold)**) **Color**
GraphWalls	vertical walls bounded by axes **LineThickness = 1px** **LineStyle = 1** **FrameBorder = on** **ContrastColor** **BackgroundColor** **Color**

You refer to these elements in graph templates **style-element** or **style-element:attribute** (for example, **GraphDataDefault:ContrastColor**). The default values are not shown for the color attributes because they are usually defined indirectly. For example, **Graph:BackgroundColor** (the color that fills the box outside the graph) is defined elsewhere in the style as **colors('docbg')**. The style also defines **'docbg' = color_list('bgA')** and **'bgA' = cxE0E0E0**. This shows that the background is a shade of gray that is much closer to white (CXFFFFFF) than to black (CX000000). You can see the background color in Figure 21.43. This shade of gray might seem darker (closer to CX000000) than you might expect based on only the RGB values. Your perception of a color change is not a linear function of the change in RGB values.

You can use the following program to see the color and other attributes for a number of style elements:

```
proc format;
   value vf 5 = 'GraphValueText';
run;

data x1;
   array y[20] y0 - y19;
   do x = 1 to 20; y[x] = x - 0.5; end;
   do x = 0 to 10 by 5; output; end;
   label y18 = 'GraphLabelText' x = 'GraphLabelText';
   format x y18 vf.;
run;

%macro d;
   %do i = 1 %to 12;
      reg y=y%eval(19-&i) x=x / lineattrs=GraphData&i markerattrs=GraphData&i
                                curvelabel="  GraphData&i" curvelabelpos=max;
   %end;
%mend;

%macro l(i, l);
   reg y=y&i x=x / lineattrs=&l markerattrs=&l curvelabel="  &l"
                   curvelabelpos=max;
%mend;
```

```
ods html style=default; * You can instead specify other destinations
                          such as LISTING, PDF, or RTF;
proc sgplot noautolegend data=x1;
   title 'GraphTitleText';
   %d
   %l(19, GraphDataDefault)
   %l( 6, GraphFit)
   %l( 5, GraphFit2)
   %l( 4, GraphPredictionLimits)
   %l( 3, GraphConfidence)
   %l( 2, GraphGridLines)
   %l( 1, GraphOutlier)
   %l( 0, GraphReference)
   xaxis values=(0 5 10);
run;
ods html close;
```

The results in Figure 21.43 display the attributes for a number of the elements of the DEFAULT style.

When there is a group or classification variable, the colors, markers, and lines that distinguish the groups are derived from the **GraphData***n* elements that are defined in the style. In the DEFAULT style, these are the elements **GraphData1** through **GraphData12**. There can be any number of groups, even though only 12 **GraphData***n* style elements are defined in the DEFAULT style. The following steps create a data set that contains 40 groups, display one line per group, and produce Figure 21.54:

```
data x2;
   do y = 40 to 1 by -1;
      group = 'Group' || put(41 - y, 2. -L);
      do x = 0 to 10 by 5;
         if x = 10 then do; z = 11; l = group; end;
         else           do; z = .;  l = ' ';   end;
         output;
      end;
   end;
run;

proc sgplot data=x2;
   title 'Colors, Markers, Lines Patterns for Groups';
   series  y=y x=x / group=group markers;
   scatter y=y x=z / group=group markerchar=l;
run;
```

The colors, markers, and line patterns in Figure 21.54 repeat in cycles. The **GraphData1** – **GraphData8** lines in Figure 21.43 exactly match the **Group1** – **Group8** lines in Figure 21.54. After that, there are differences due to the cyclic construction of the grouped style. This is explained next.

The DEFAULT style defines a marker symbol only in **GraphData1** through **GraphData7**. The seven markers are circle, plus sign, X, triangle, square, asterisk, and diamond. With the explicit style reference in Figure 21.43, the actual symbol, when no symbol is specified, is the circle. This is what you see for **GraphData8** through **GraphData12**. With the group variable in Figure 21.54, the symbols repeat in cycles. Hence, **Group1**, **Group8**, **Group15**, and so on, are all circles. Similarly, **Group2**, **Group9**, **Group16**, and so on, are all plus signs. The DEFAULT style defines 11 different line styles for **GraphData1** through **GraphData11**. You specify line styles by specifying an integer. The default lines styles are 1, 4, 8, 5, 14, 26, 15, 20, 41, 42, and 2. Hence, **Group1**, **Group12**, **Group23**, and so on, all have the same line style, which is a solid line (line style 1). Similarly, **Group2**, **Group13**, **Group24**, and so on, all have line style 4. There are 12 different colors, so **Group1**, **Group13**, **Group25**, and so on, all have the same colors. Overall, there are $12 \times 11 \times 7 = 924$ color, line, and marker combinations that appear before any combination repeats. You can use the %MODSTYLE SAS autocall macro (see the section "ODS Style Template Modification Macro" on page 701) to conveniently change these style attributes.

The HTMLBLUE style is an all-color style for the first 12 groups of observations. Most analyses have fewer than 12 groups. Markers and lines change for groups 13–24 and then again for groups 25–36. Figure 21.55 shows how colors, markers, and line styles change in the HTMLBLUE style. Figure 21.54 and Figure 21.55 through Figure 21.64 show how these elements change in other styles.

ODS Style Comparisons

In this section, some of the most commonly used styles are compared in a series of figures, most of which were generated in the preceding section. Figure 21.23 through Figure 21.32 show tables and graphs in the HTML destination for each of eight styles, for the following analysis:

```
proc reg data=sashelp.class;
   model Weight = Height;
quit;
```

The PEARL, PEARLJ, RTF, SAPPHIRE, and six JOURNAL styles are compared by running the following steps for each of the ten styles and capturing output in the PDF destination:

```
options nonumber nodate;
ods proctitle off;

ods pdf body="fPearlJ.pdf" style=PearlJ startpage=never;
title "PearlJ";
proc means data=sashelp.class maxdec=2;
run;

proc sgplot data=sashelp.class;
   vbar age / group=sex;
run;
ods pdf close;
```

The results of these steps are displayed in Figure 21.33 through Figure 21.42.

Figure 21.43 through Figure 21.53 show some of the most common style elements. Figure 21.54 through Figure 21.61 show how groups of observations are displayed in the graph.

The style comparisons are as follows:

- The ANALYSIS style is displayed in Figure 21.27, Figure 21.47, and Figure 21.58.
- The DEFAULT style is displayed in Figure 21.23, Figure 21.43, and Figure 21.54.
- The HTMLBLUE style is displayed in Figure 21.24, Figure 21.44, and Figure 21.55.
- The HTMLBLUECML style is displayed in Figure 21.25, Figure 21.45, and Figure 21.56.
- The JOURNAL style is displayed in Figure 21.28, Figure 21.33, Figure 21.48, and Figure 21.59.
- The JOURNAL1A style is displayed in Figure 21.34.
- The JOURNAL2 style is displayed in Figure 21.35.
- The JOURNAL2A style is displayed in Figure 21.36.
- The JOURNAL3 style is displayed in Figure 21.37.
- The JOURNAL3A style is displayed in Figure 21.38.
- The LISTING style is displayed in Figure 21.29, Figure 21.49, and Figure 21.60.
- The PEARL style is displayed in Figure 21.31, Figure 21.39, Figure 21.51, and Figure 21.62.
- The PEARLJ style is displayed in Figure 21.40, Figure 21.52, and Figure 21.63.
- The RTF style is displayed in Figure 21.30, Figure 21.41, Figure 21.50, and Figure 21.61.
- The SAPPHIRE style is displayed in Figure 21.32, Figure 21.42, Figure 21.53, and Figure 21.64.
- The STATISTICAL style is displayed in Figure 21.26, Figure 21.46, and Figure 21.57.

Figure 21.23 Statistical Output with the DEFAULT Style and HTML Destination

Parameter Estimates					
Variable	**DF**	**Parameter Estimate**	**Standard Error**	**t Value**	**Pr > |t|**
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

The REG Procedure
Model: MODEL1
Dependent Variable: Weight

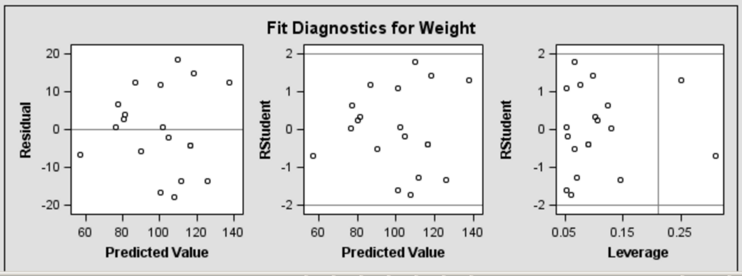


Figure 21.24 Statistical Output with the HTMLBLUE Style and HTML Destination

Parameter Estimates					
Variable	**DF**	**Parameter Estimate**	**Standard Error**	**t Value**	**Pr > |t|**
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

The REG Procedure
Model: MODEL1
Dependent Variable: Weight

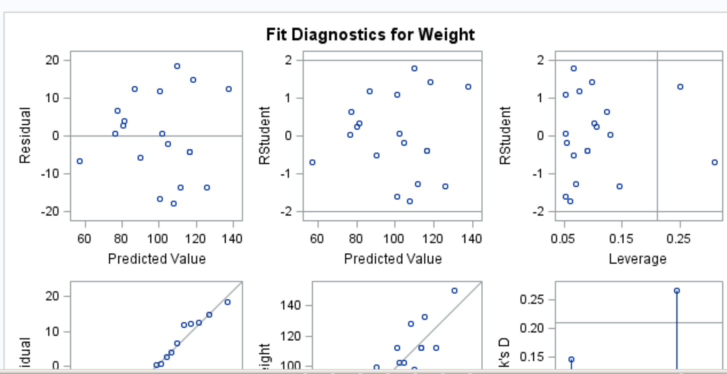
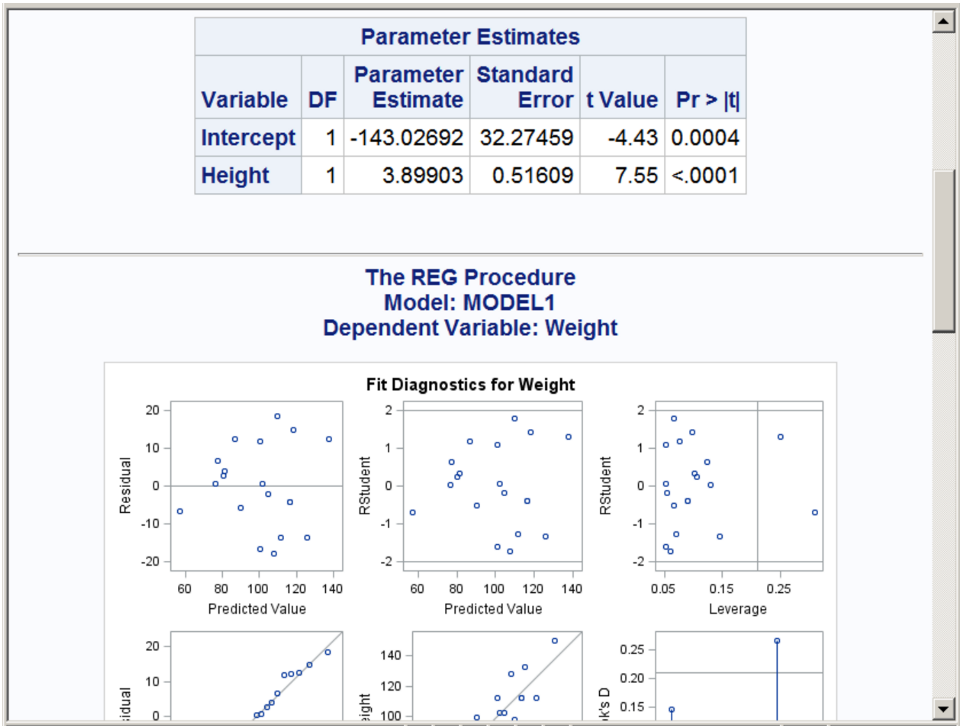


Figure 21.25 Statistical Output with the HTMLBLUECML Style and HTML Destination

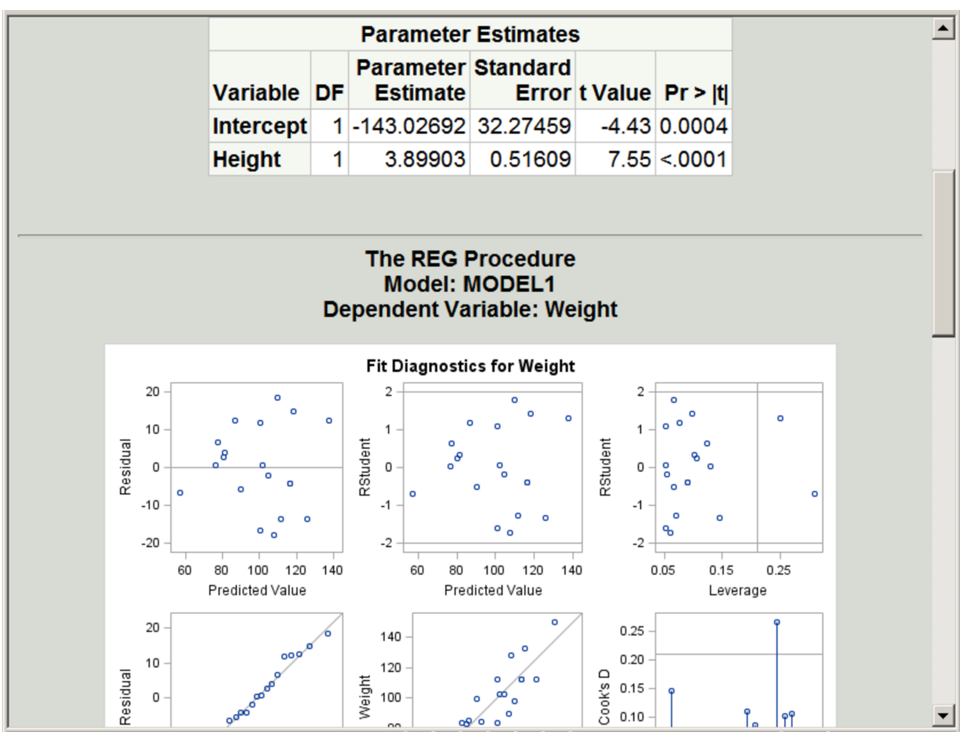


Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

The REG Procedure
Model: MODEL1
Dependent Variable: Weight

Figure 21.26 Statistical Output with the STATISTICAL Style and HTML Destination

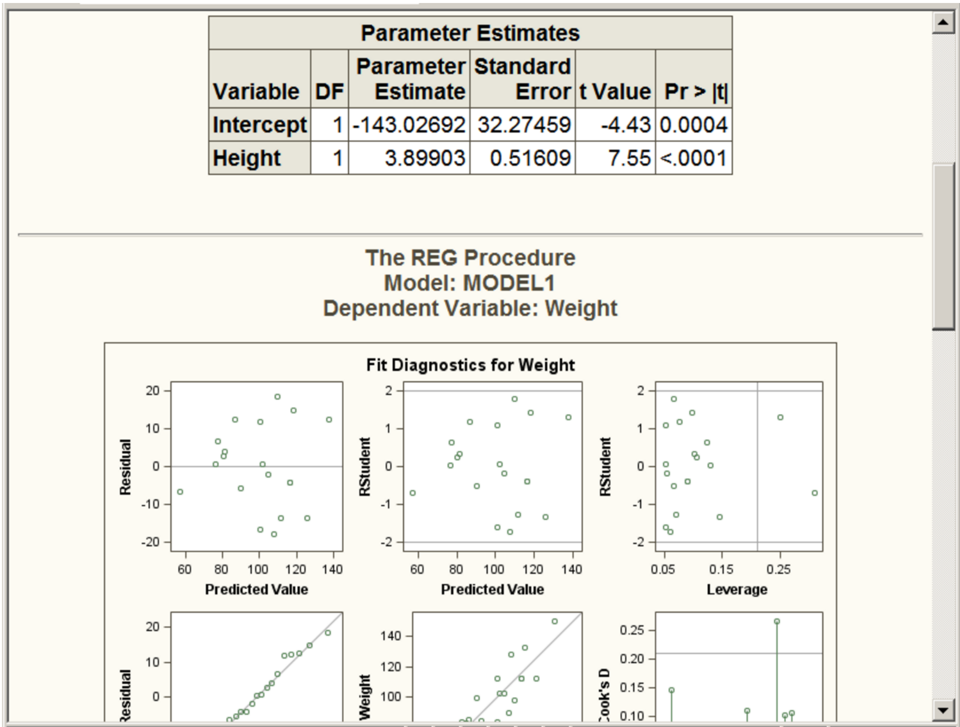


Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

The REG Procedure
Model: MODEL1
Dependent Variable: Weight

Figure 21.27 Statistical Output with the ANALYSIS Style and HTML Destination

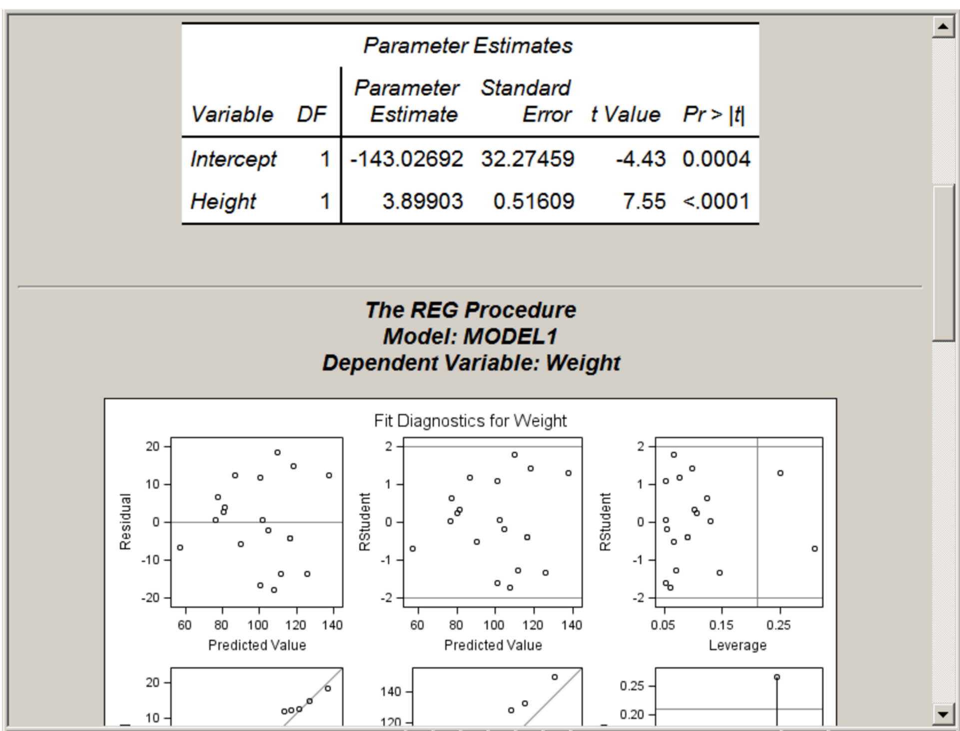


Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

The REG Procedure
Model: MODEL1
Dependent Variable: Weight

Figure 21.28 Statistical Output with the JOURNAL Style and HTML Destination

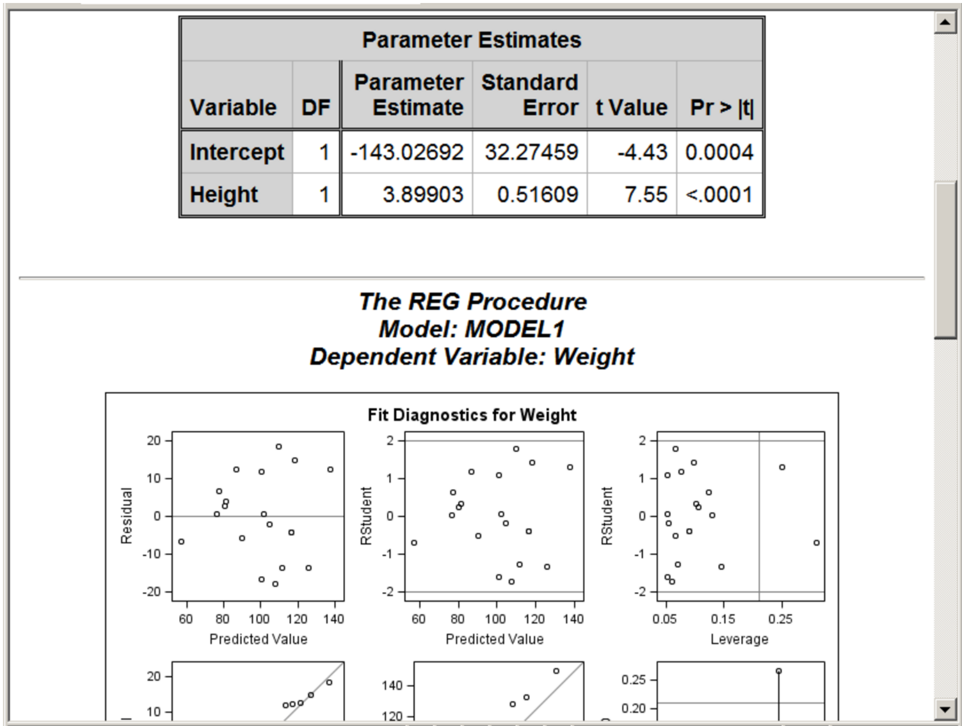


Parameter Estimates

Variable	*DF*	*Parameter Estimate*	*Standard Error*	*t Value*	*Pr > |t|*
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

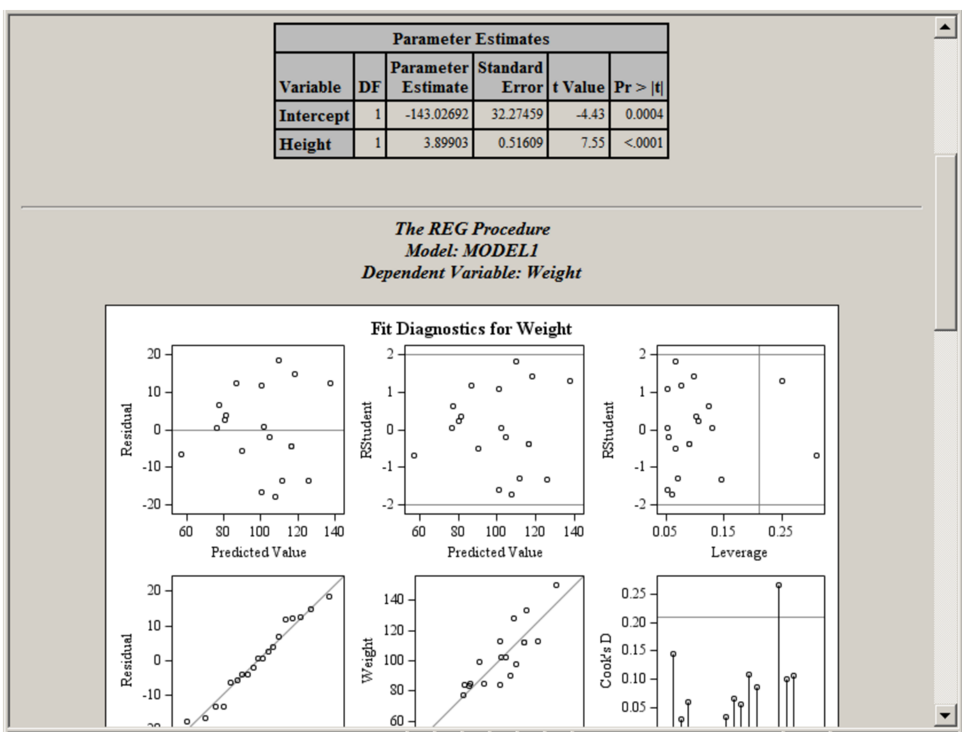
The REG Procedure
Model: MODEL1
Dependent Variable: Weight

Figure 21.29 Statistical Output with the LISTING Style and HTML Destination



Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

Figure 21.30 Statistical Output with the RTF Style and HTML Destination



Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

Figure 21.31 Statistical Output with the PEARL Style and HTML Destination

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

The REG Procedure
Model: MODEL1
Dependent Variable: Weight

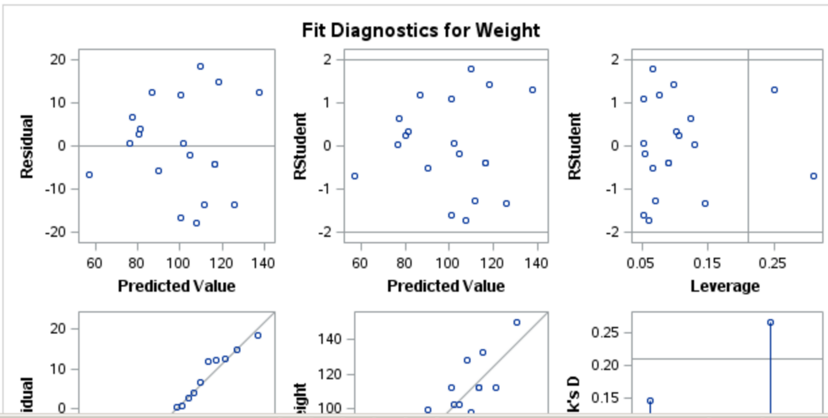


Figure 21.32 Statistical Output with the SAPPHIRE Style and HTML Destination

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-143.02692	32.27459	-4.43	0.0004
Height	1	3.89903	0.51609	7.55	<.0001

The REG Procedure
Model: MODEL1
Dependent Variable: Weight

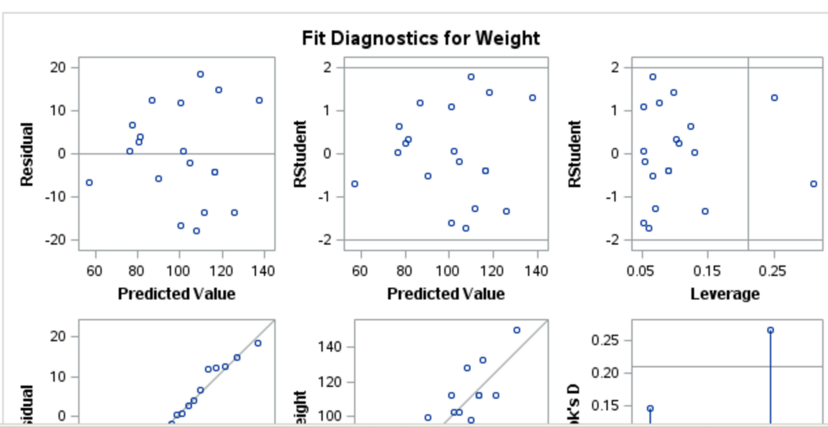


Figure 21.33 JOURNAL Style and PDF Destination

Journal

Variable	*N*	*Mean*	*Std Dev*	*Minimum*	*Maximum*
Age	19	13.32	1.49	11.00	16.00
Height	19	62.34	5.13	51.30	72.00
Weight	19	100.03	22.77	50.50	150.00

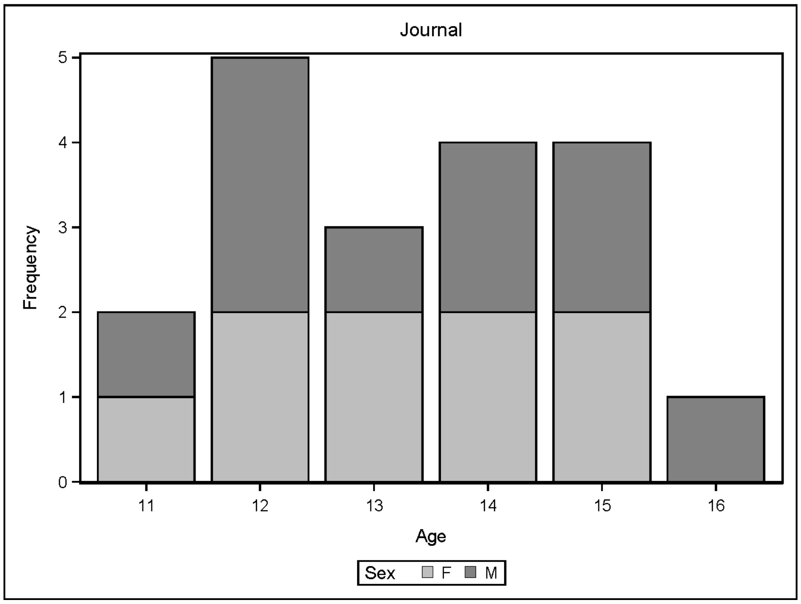


Figure 21.34 JOURNAL1A Style and PDF Destination

Journal1a

Variable	N	Mean	Std Dev	Minimum	Maximum
Age	19	13.32	1.49	11.00	16.00
Height	19	62.34	5.13	51.30	72.00
Weight	19	100.03	22.77	50.50	150.00

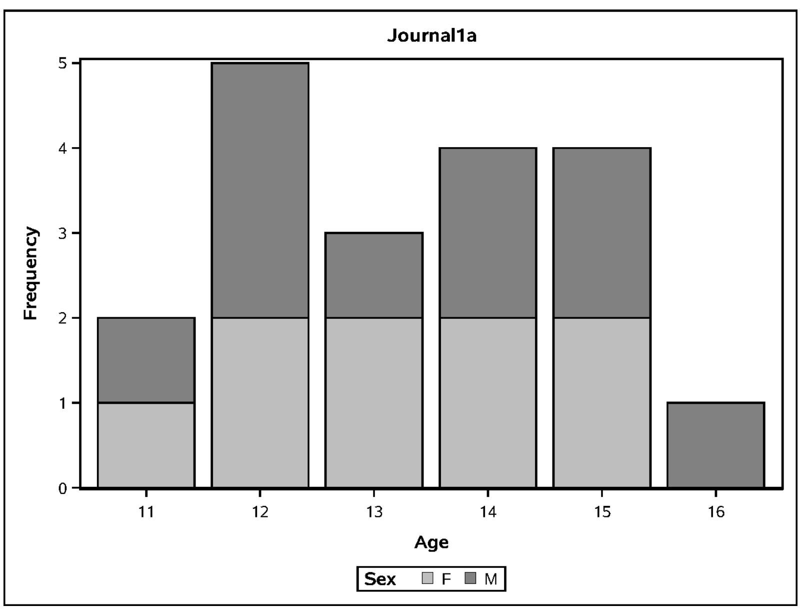


Figure 21.35 JOURNAL2 Style and PDF Destination

Journal2

Variable	*N*	*Mean*	*Std Dev*	*Minimum*	*Maximum*
Age	19	13.32	1.49	11.00	16.00
Height	19	62.34	5.13	51.30	72.00
Weight	19	100.03	22.77	50.50	150.00

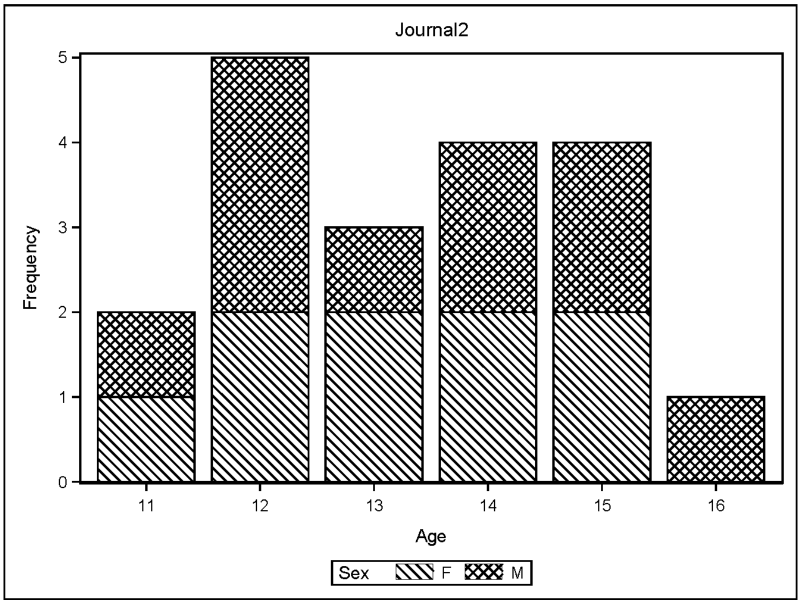


Figure 21.36 JOURNAL2A Style and PDF Destination

Journal2a

Variable	N	Mean	Std Dev	Minimum	Maximum
Age	19	13.32	1.49	11.00	16.00
Height	19	62.34	5.13	51.30	72.00
Weight	19	100.03	22.77	50.50	150.00

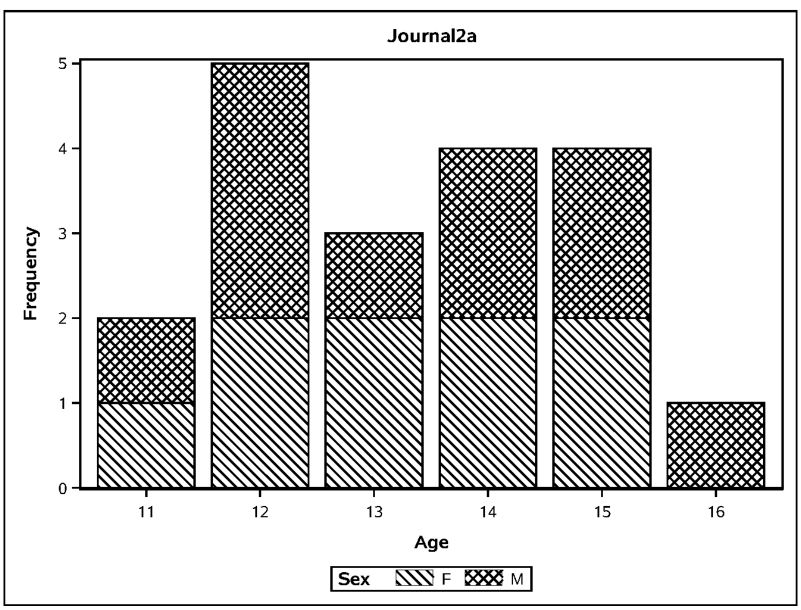


Figure 21.37 JOURNAL3 Style and PDF Destination

Journal3

Variable	*N*	*Mean*	*Std Dev*	*Minimum*	*Maximum*
Age	19	13.32	1.49	11.00	16.00
Height	19	62.34	5.13	51.30	72.00
Weight	19	100.03	22.77	50.50	150.00

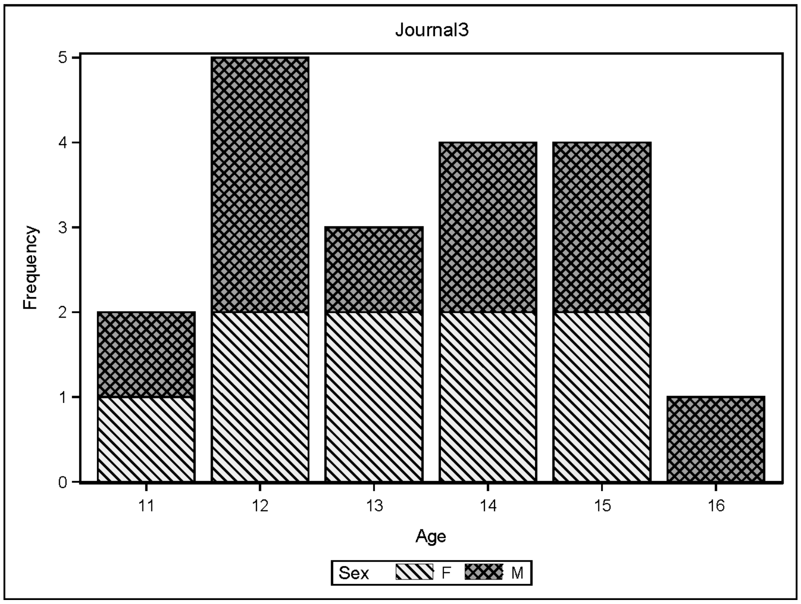


Figure 21.38 JOURNAL3A Style and PDF Destination

Journal3a

Variable	N	Mean	Std Dev	Minimum	Maximum
Age	19	13.32	1.49	11.00	16.00
Height	19	62.34	5.13	51.30	72.00
Weight	19	100.03	22.77	50.50	150.00

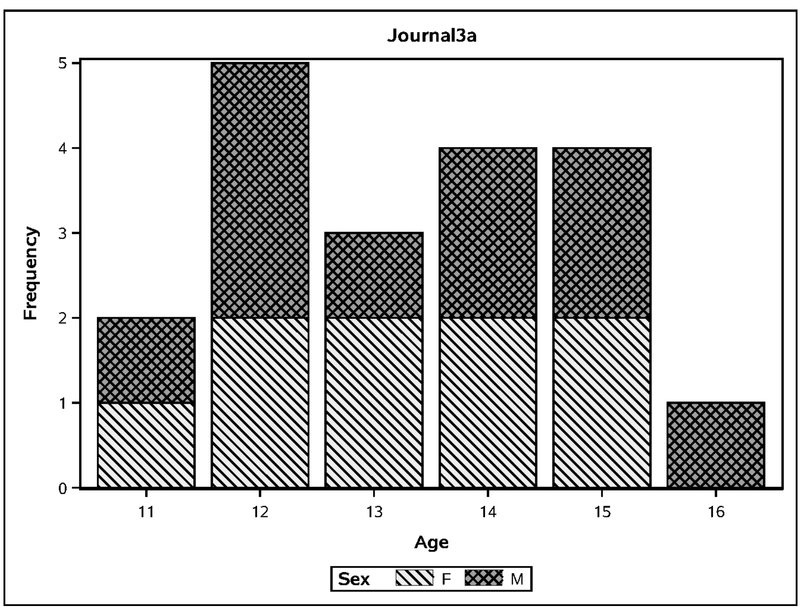


Figure 21.39 PEARL Style and PDF Destination

Pearl

Variable	N	Mean	Std Dev	Minimum	Maximum
Age	19	13.32	1.49	11.00	16.00
Height	19	62.34	5.13	51.30	72.00
Weight	19	100.03	22.77	50.50	150.00

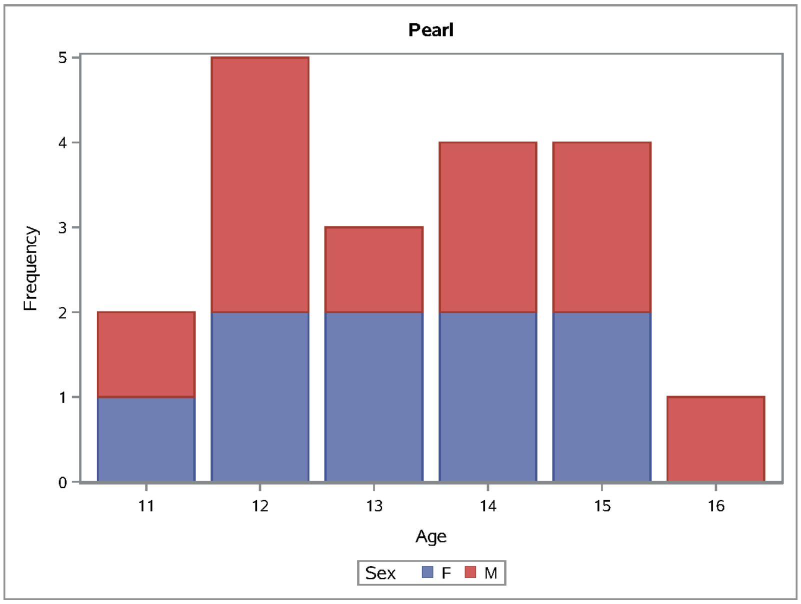


Figure 21.40 PEARLJ Style and PDF Destination

PearlJ

Variable	N	Mean	Std Dev	Minimum	Maximum
Age	19	13.32	1.49	11.00	16.00
Height	19	62.34	5.13	51.30	72.00
Weight	19	100.03	22.77	50.50	150.00

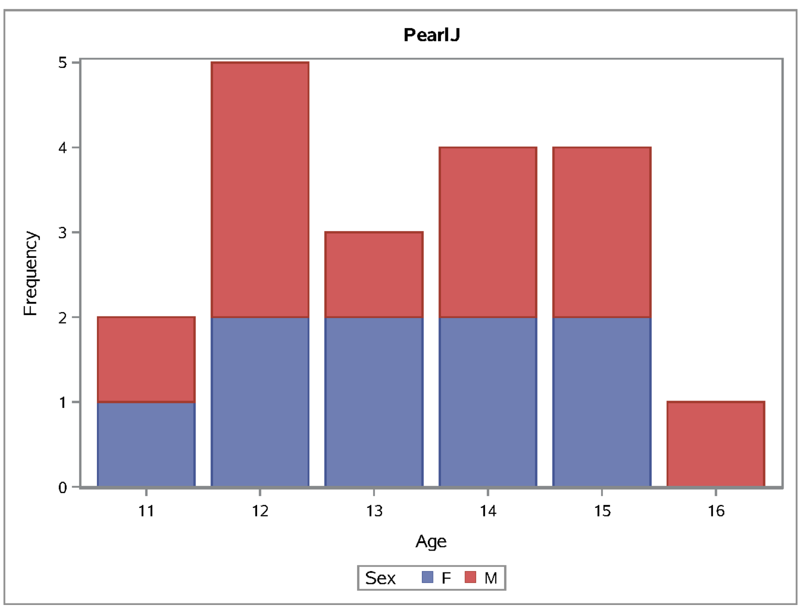


Figure 21.41 RTF Style and PDF Destination

RTF

Variable	N	Mean	Std Dev	Minimum	Maximum
Age	19	13.32	1.49	11.00	16.00
Height	19	62.34	5.13	51.30	72.00
Weight	19	100.03	22.77	50.50	150.00

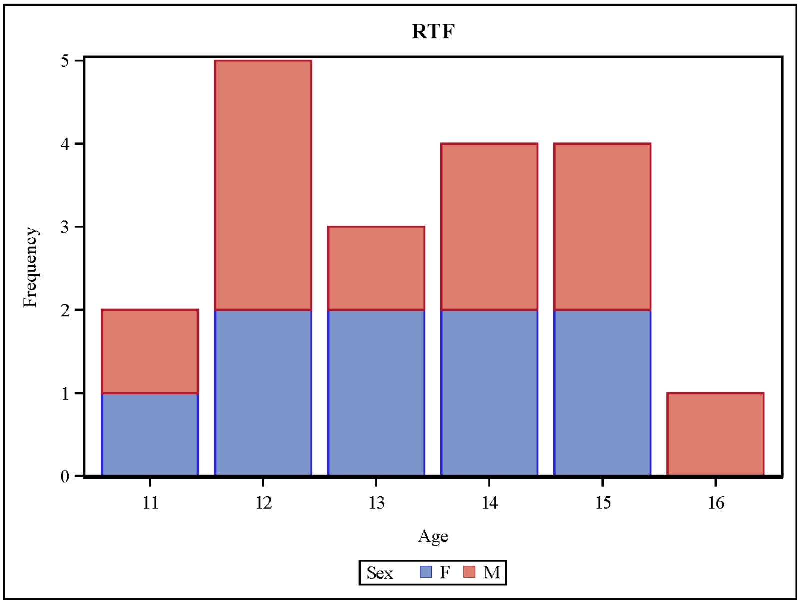


Figure 21.42 SAPPHIRE Style and PDF Destination

Sapphire

Variable	N	Mean	Std Dev	Minimum	Maximum
Age	19	13.32	1.49	11.00	16.00
Height	19	62.34	5.13	51.30	72.00
Weight	19	100.03	22.77	50.50	150.00

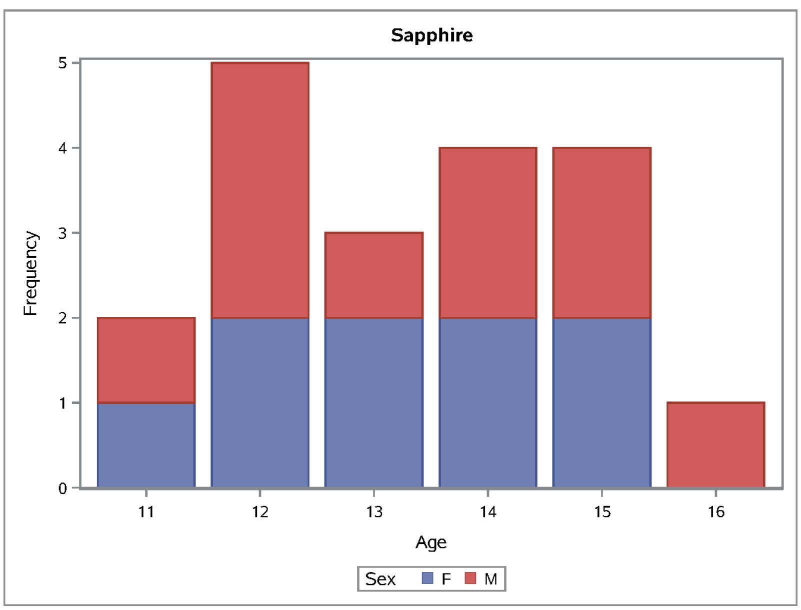


Figure 21.43 Attributes of Style Elements in the DEFAULT Style

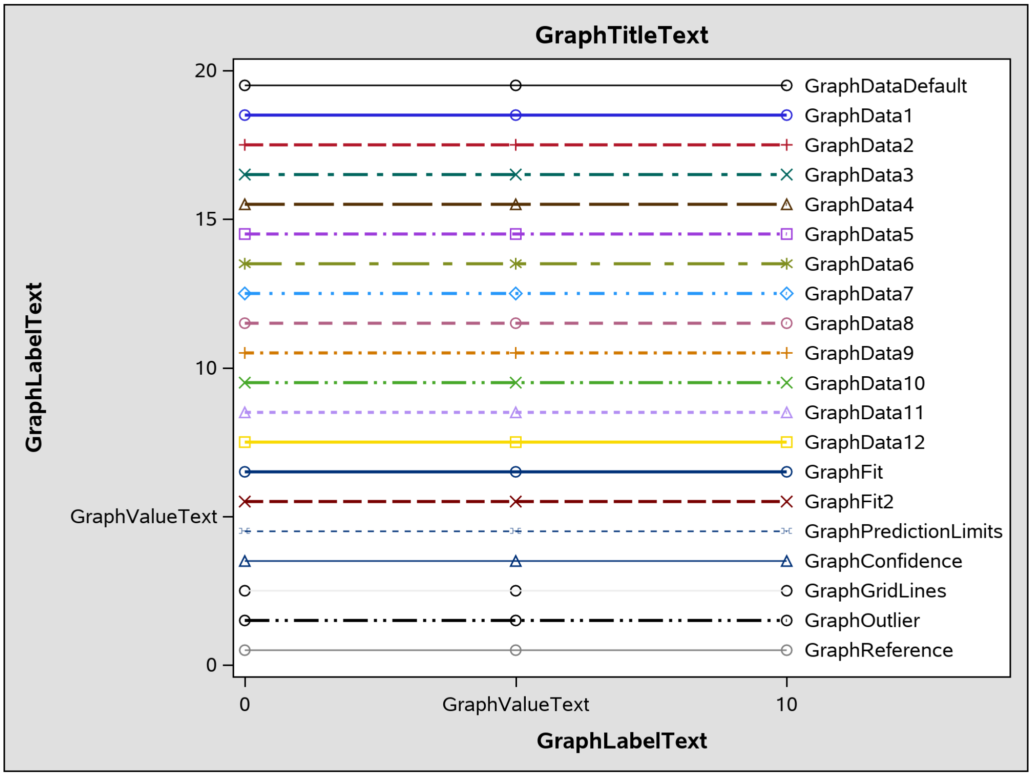


Figure 21.44 Attributes of Style Elements in the HTMLBLUE Style

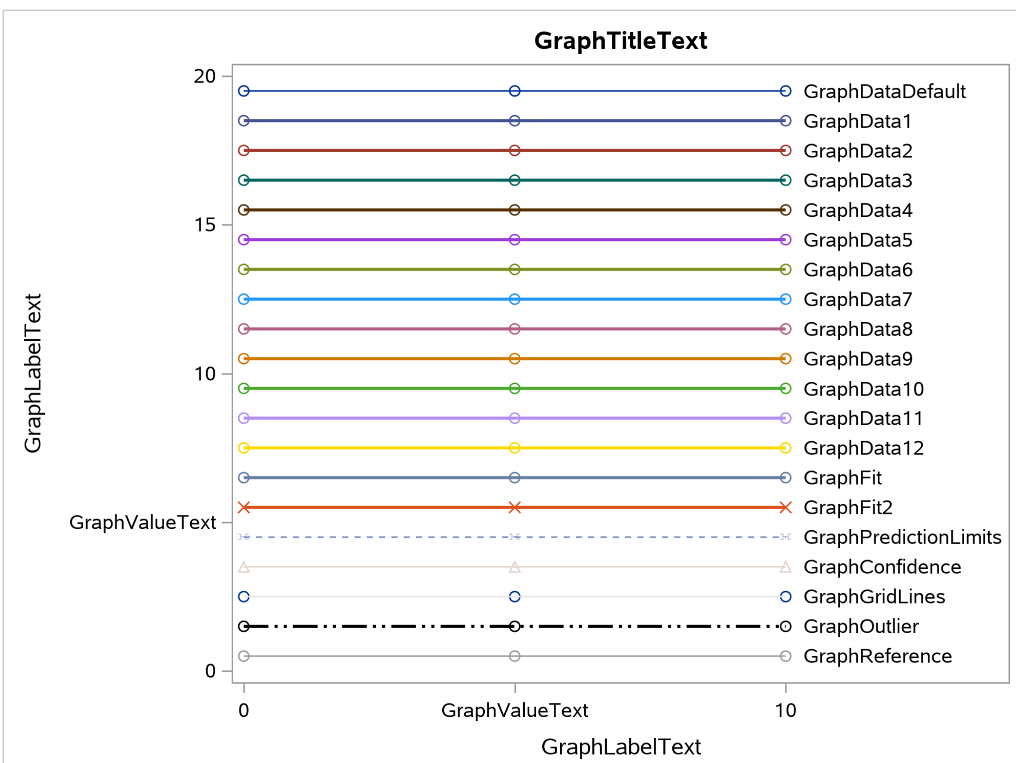


Figure 21.45 Attributes of Style Elements in the HTMLBLUECML Style

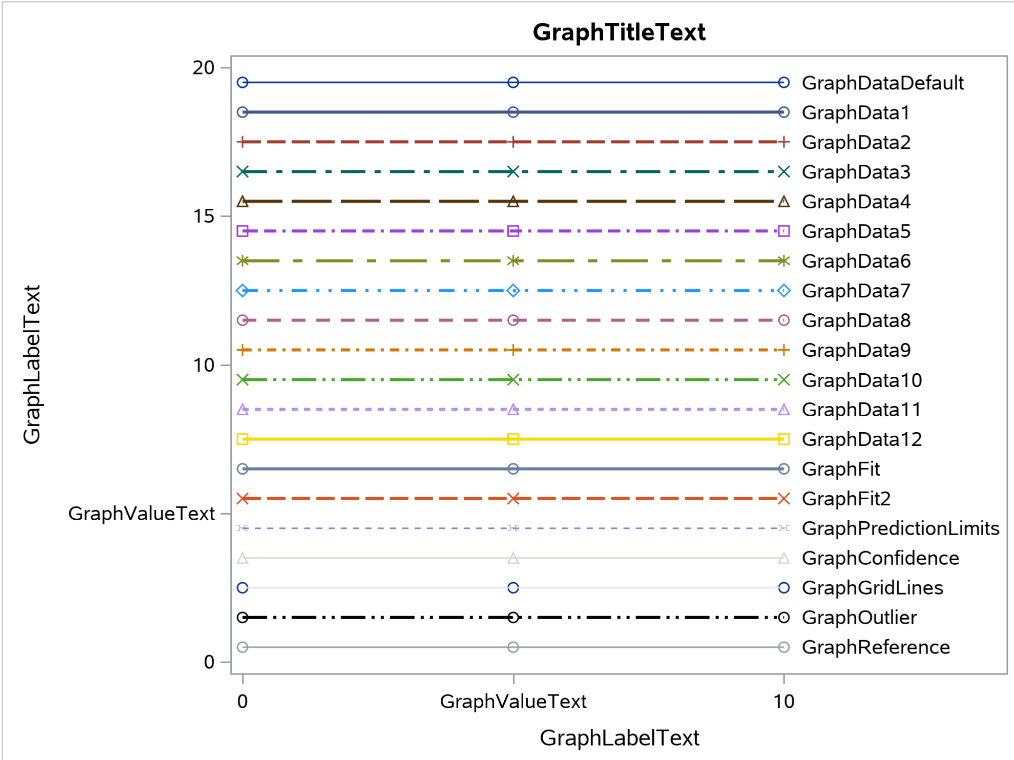


Figure 21.46 Attributes of Style Elements in the STATISTICAL Style

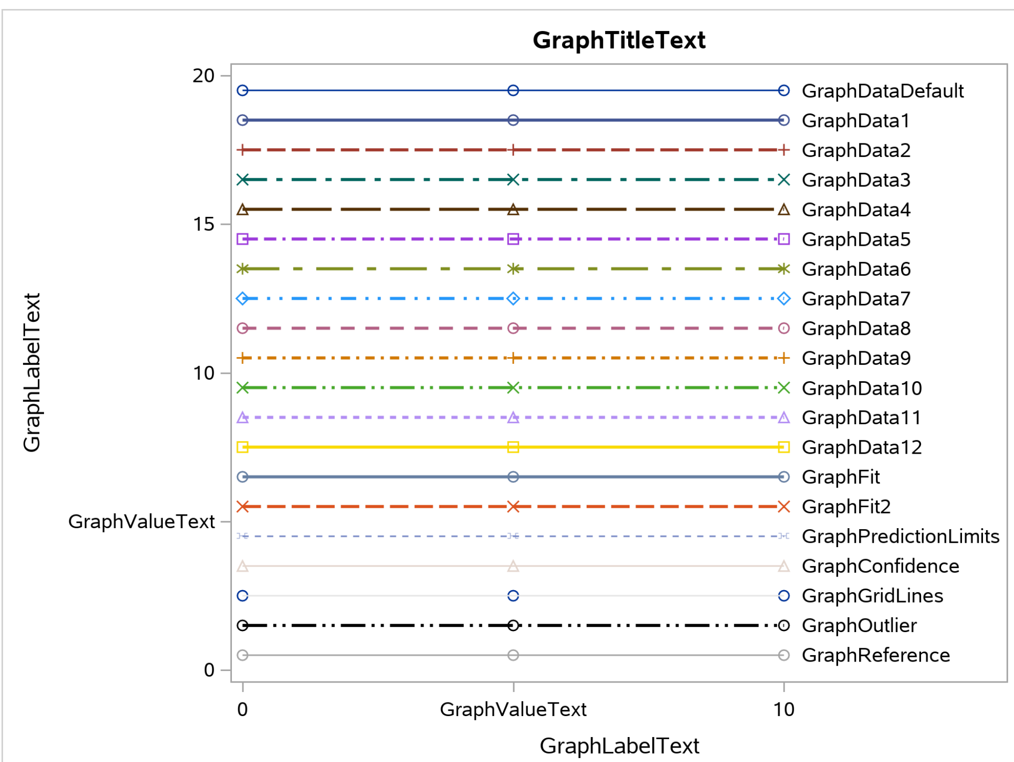


Figure 21.47 Attributes of Style Elements in the ANALYSIS Style

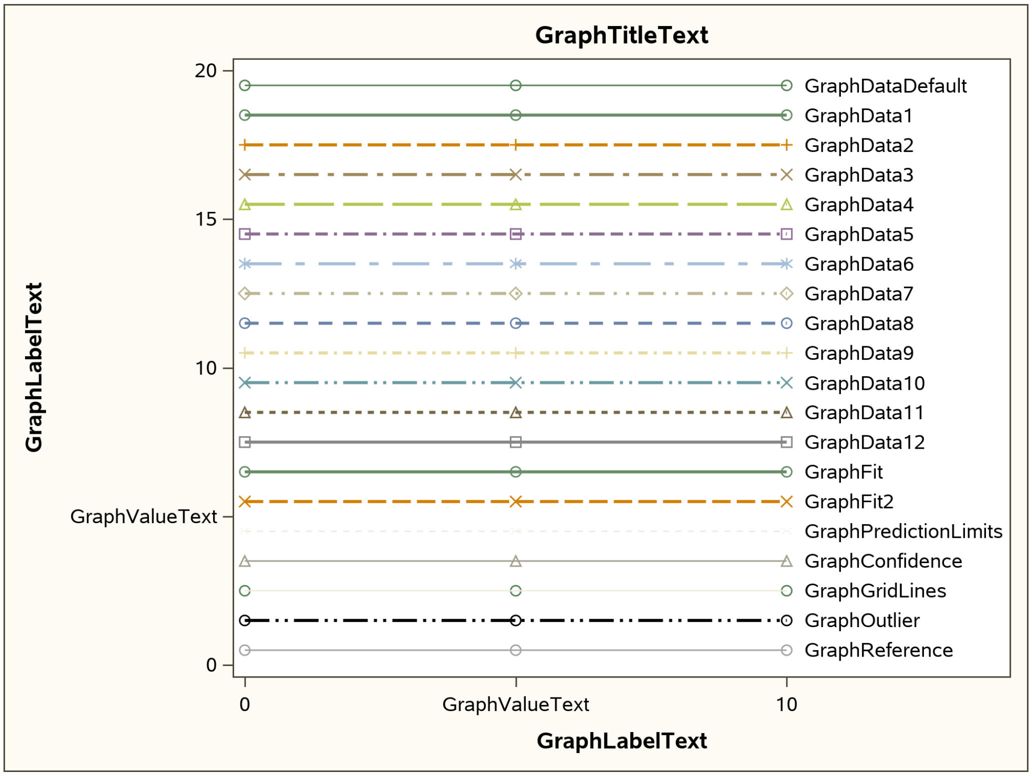


Figure 21.48 Attributes of Style Elements in the JOURNAL Style

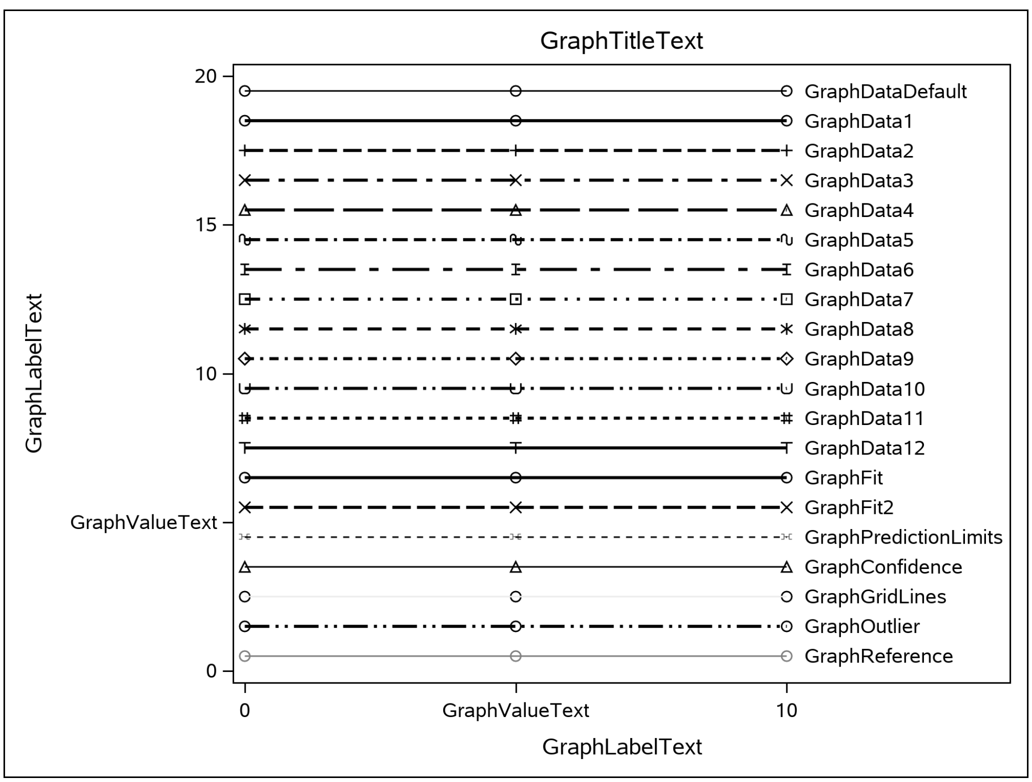


Figure 21.49 Attributes of Style Elements in the LISTING Style

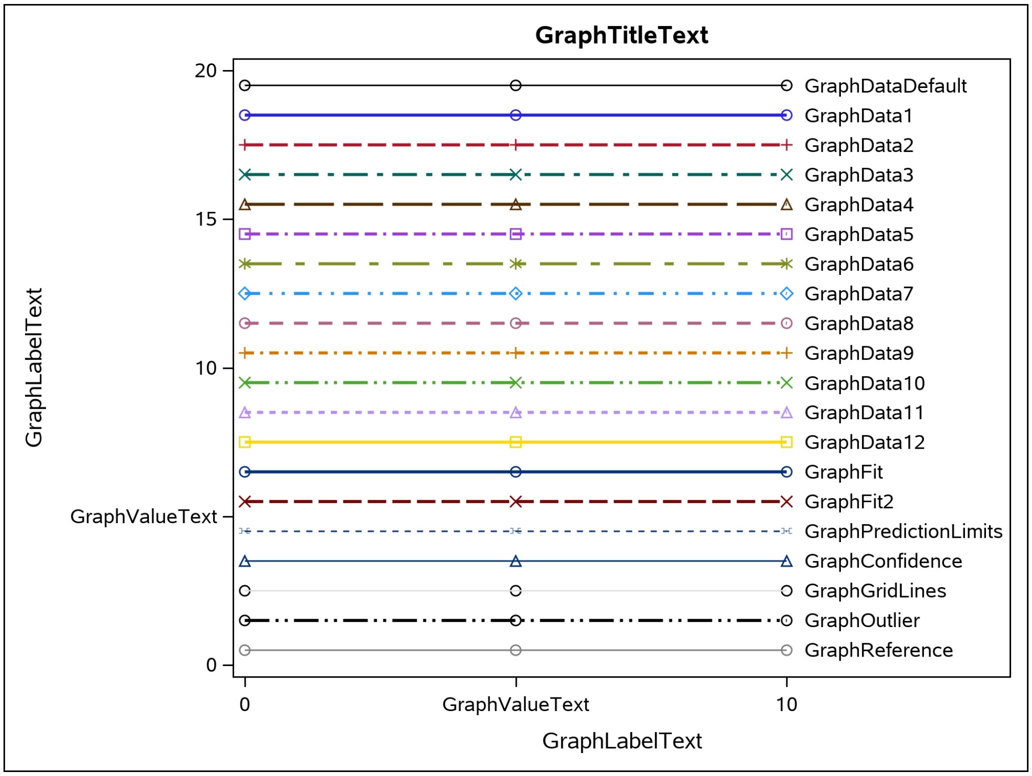


Figure 21.50 Attributes of Style Elements in the RTF Style

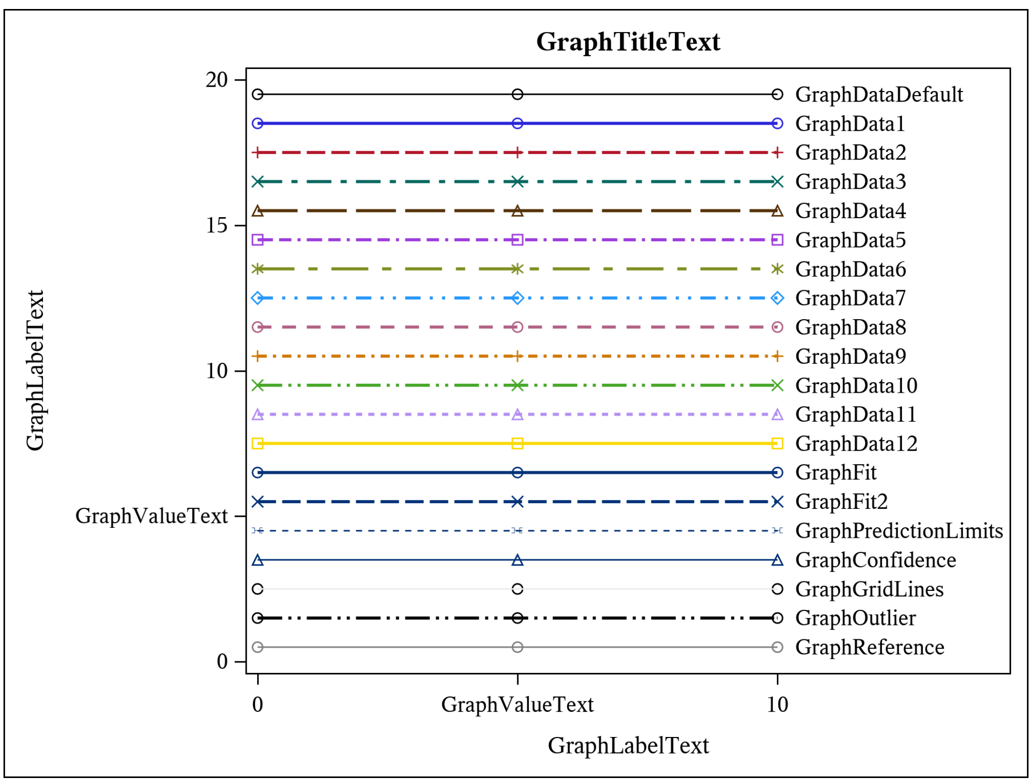


Figure 21.51 Attributes of Style Elements in the PEARL Style

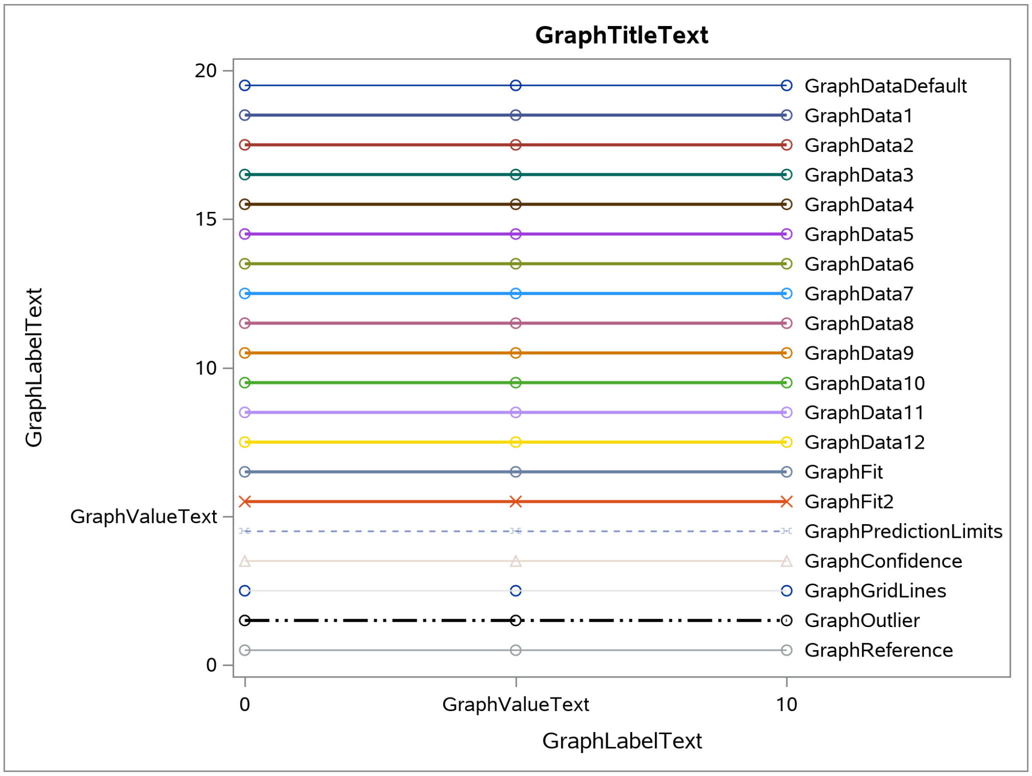


Figure 21.52 Attributes of Style Elements in the PEARLJ Style

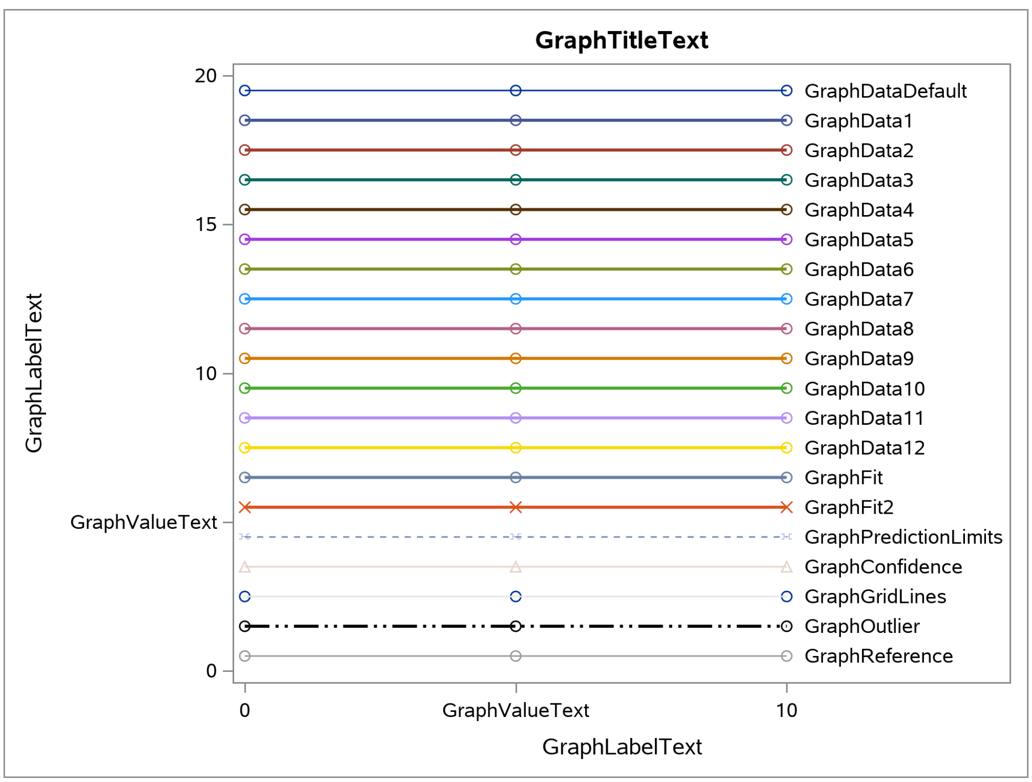


Figure 21.53 Attributes of Style Elements in the SAPPHIRE Style

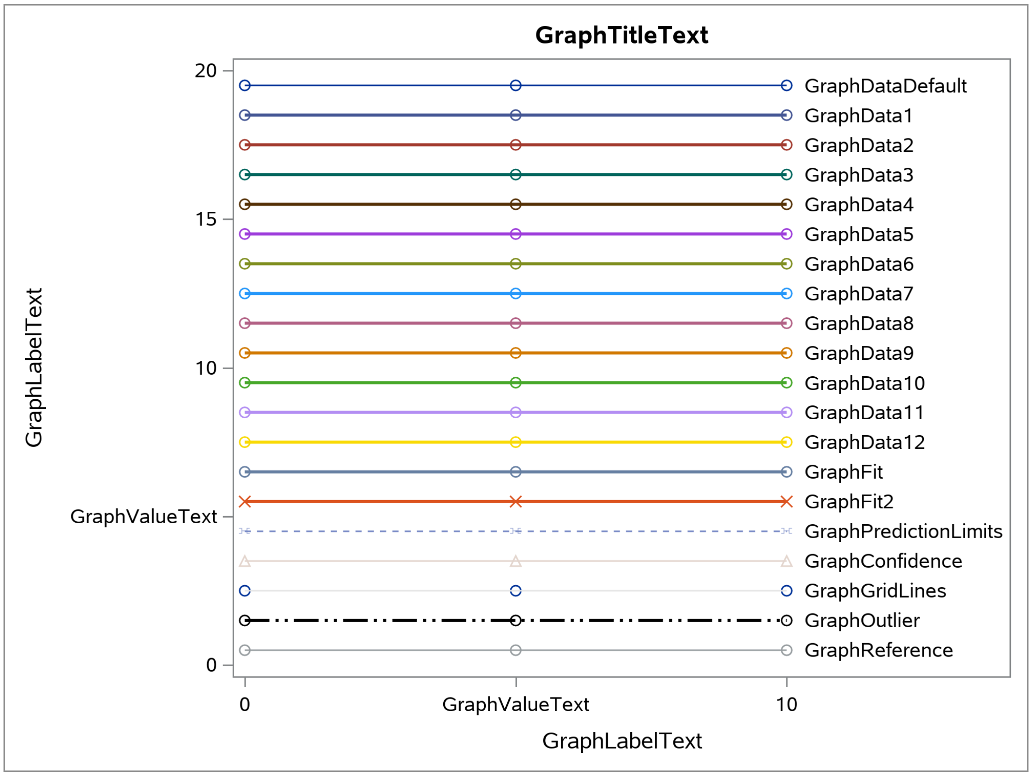


Figure 21.54 Markers, Lines, and Colors with Groups in the DEFAULT Style

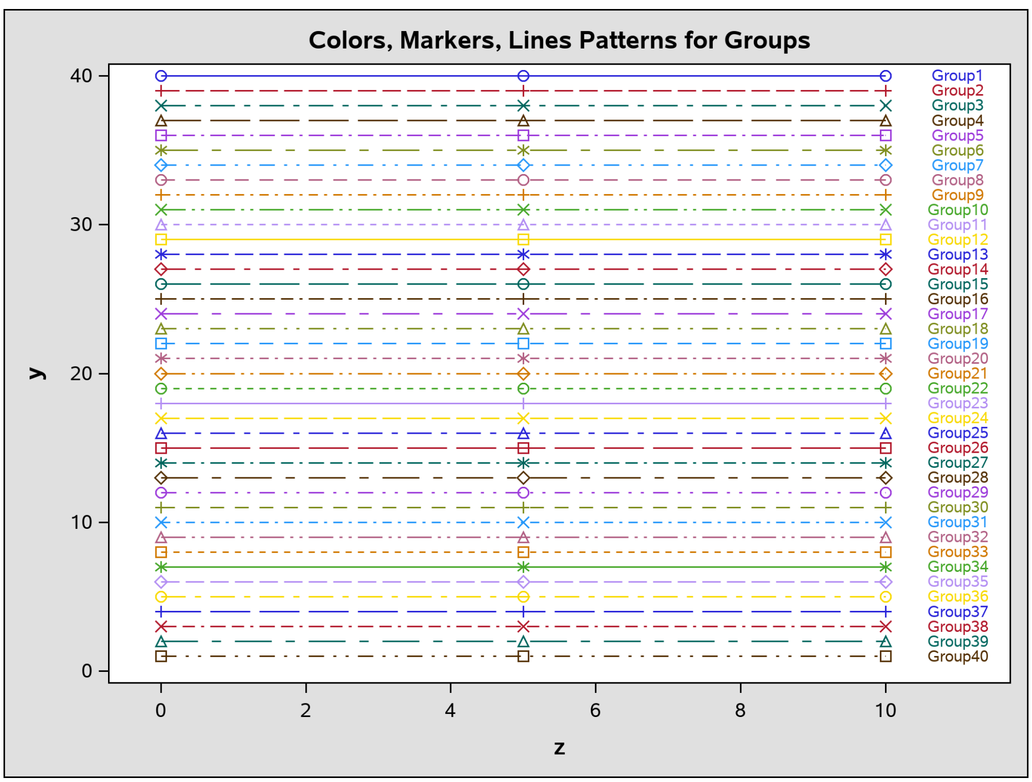


Figure 21.55 Markers, Lines, and Colors with Groups in the HTMLBLUE Style

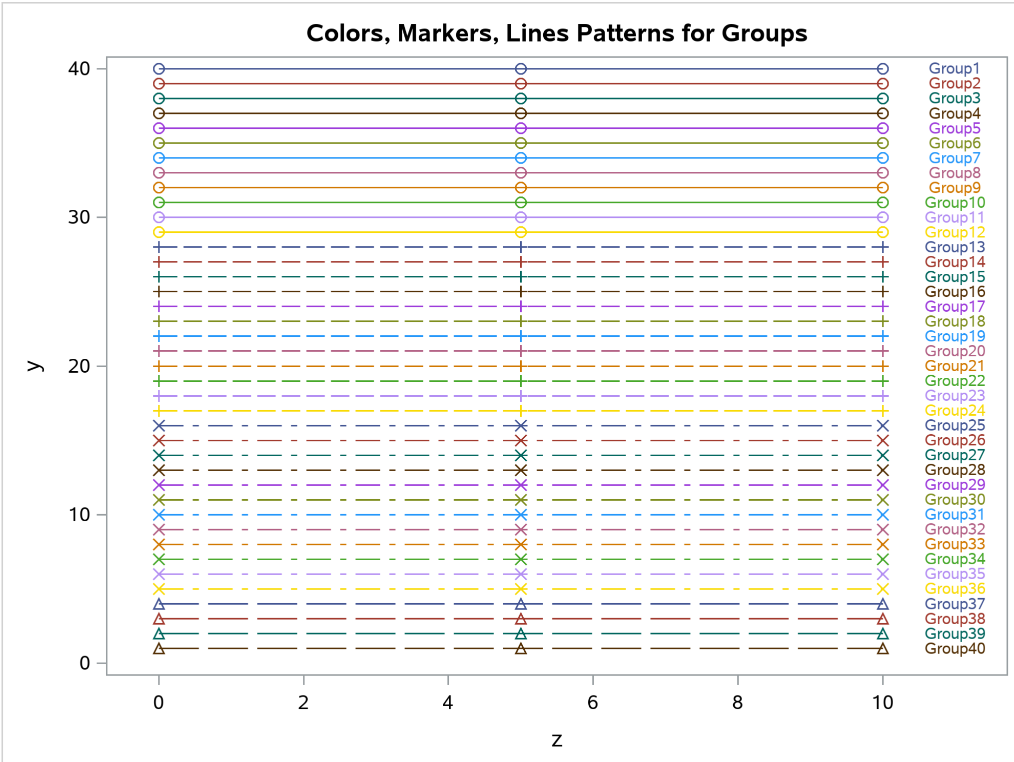


Figure 21.56 Markers, Lines, and Colors with Groups in the HTMLBLUECML Style

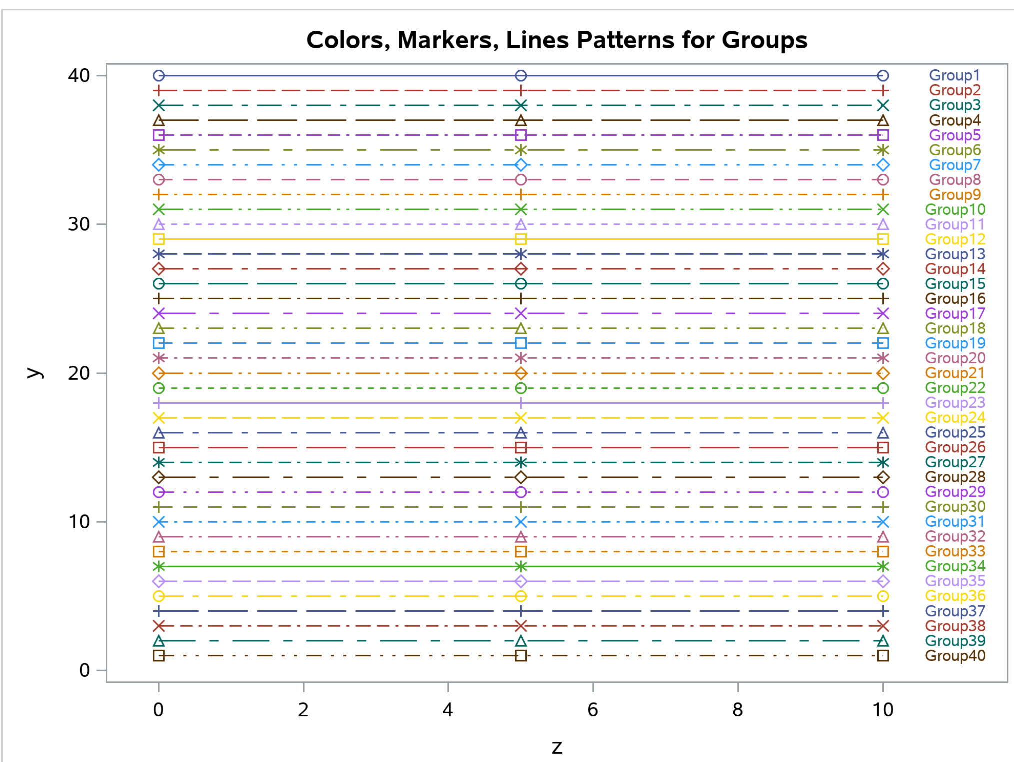


Figure 21.57 Markers, Lines, and Colors with Groups in the STATISTICAL Style

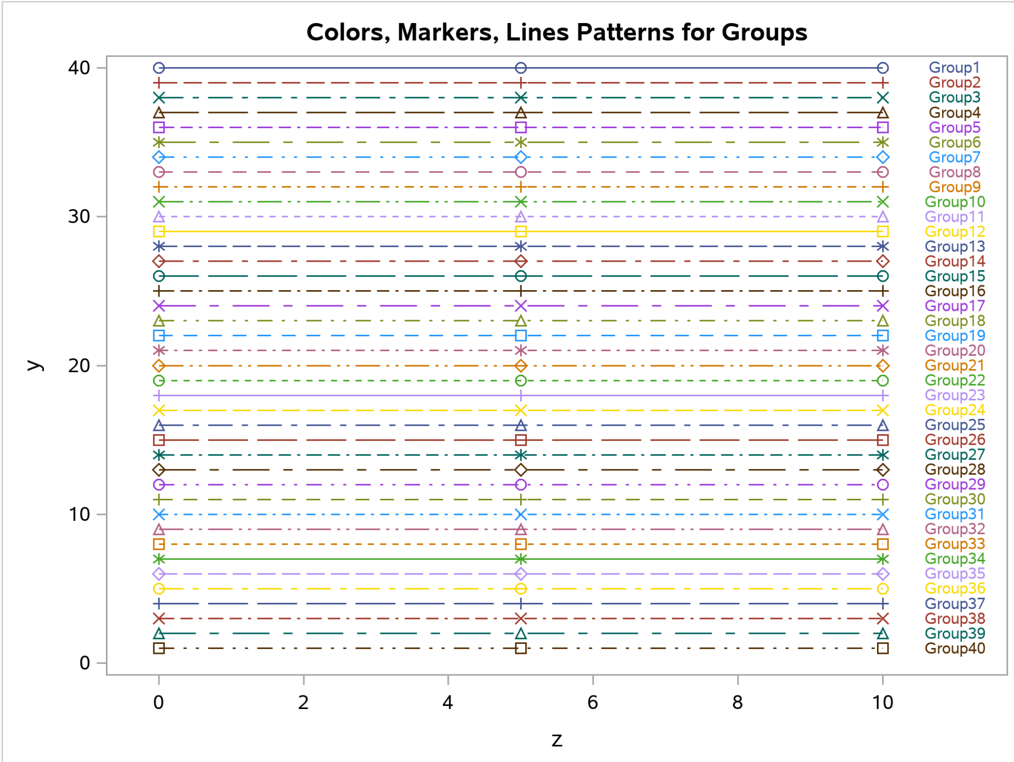


Figure 21.58 Markers, Lines, and Colors with Groups in the ANALYSIS Style

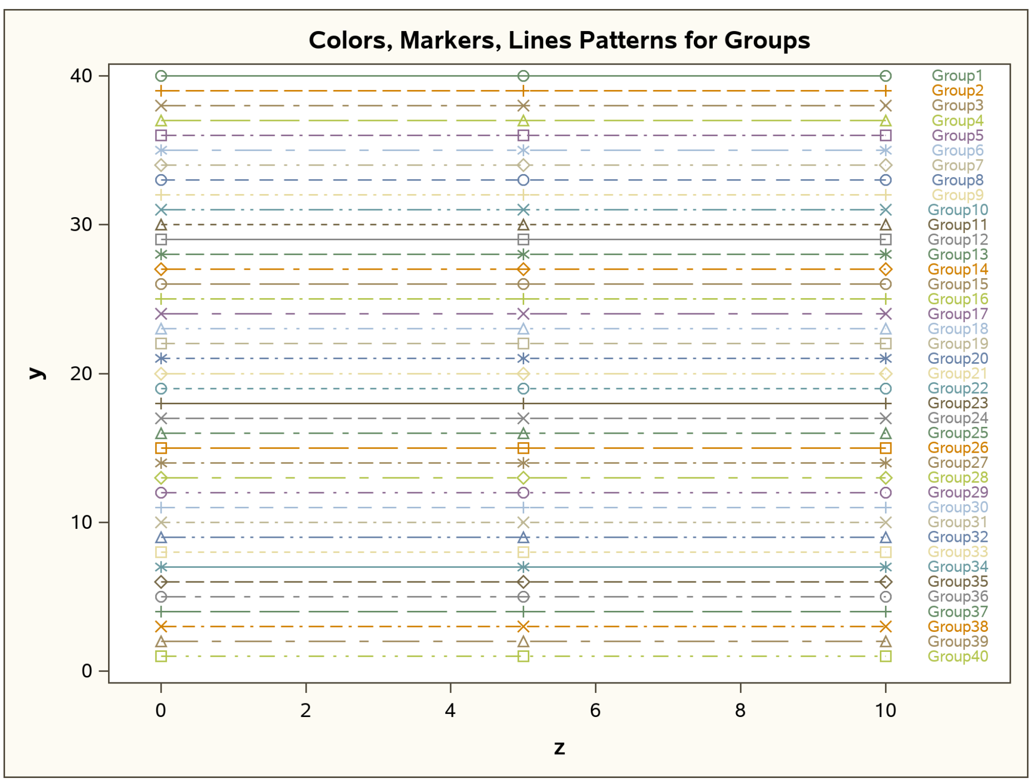


Figure 21.59 Markers, Lines, and Colors with Groups in the JOURNAL Style

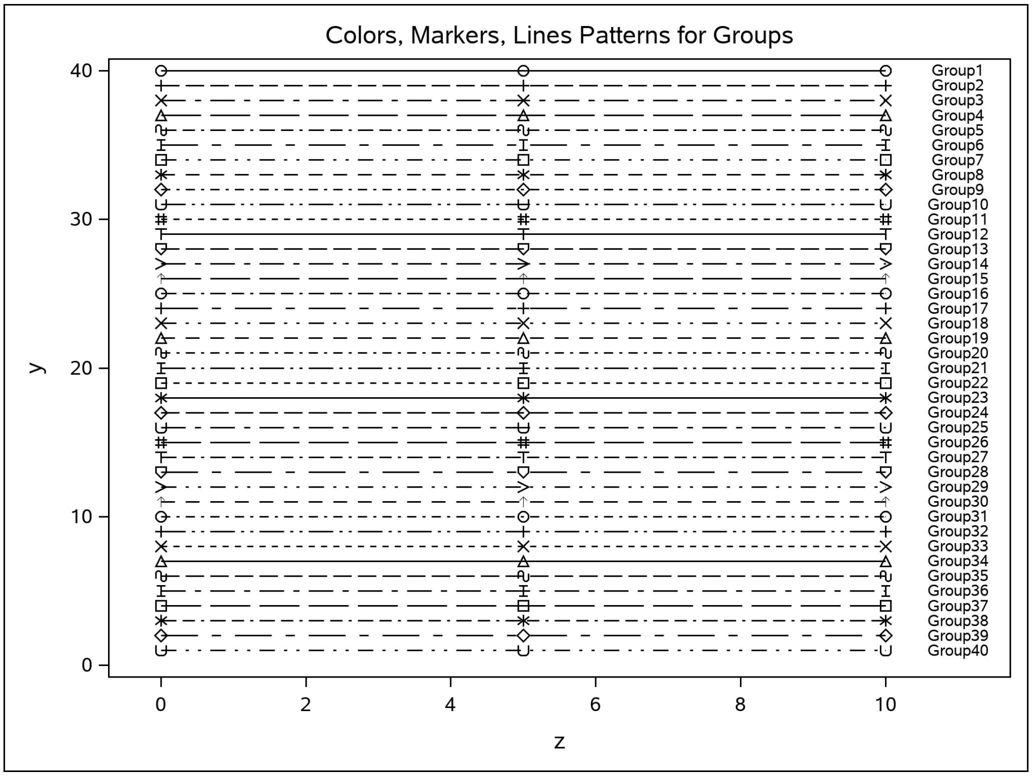


Figure 21.60 Markers, Lines, and Colors with Groups in the LISTING Style

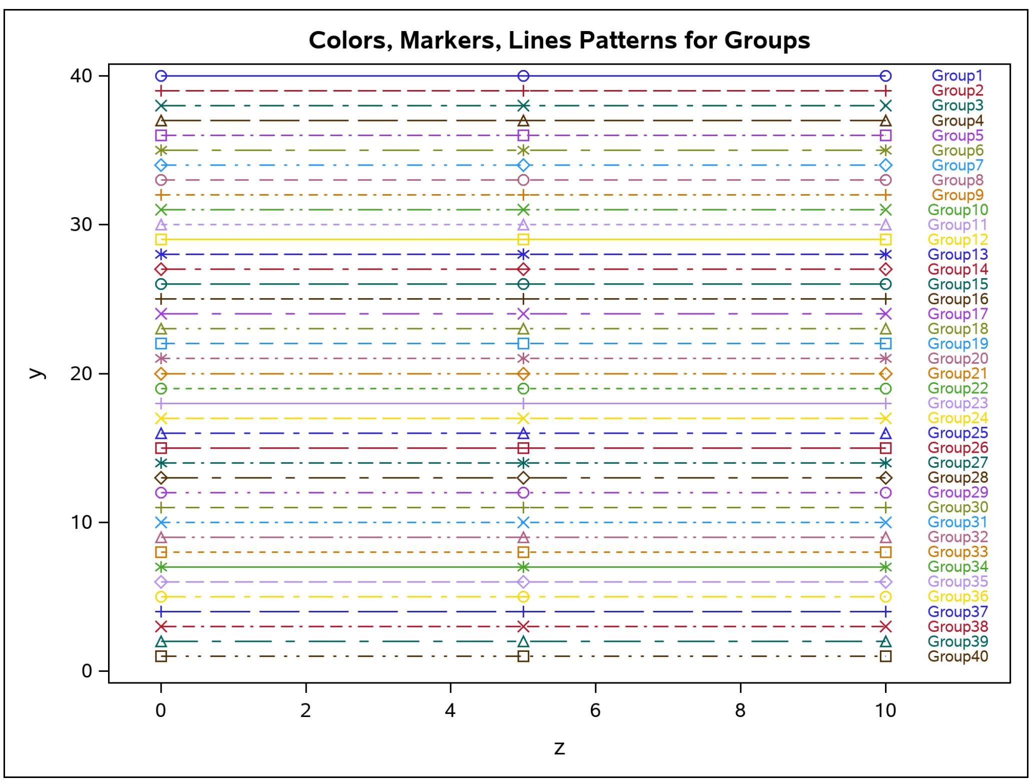


Figure 21.61 Markers, Lines, and Colors with Groups in the RTF Style

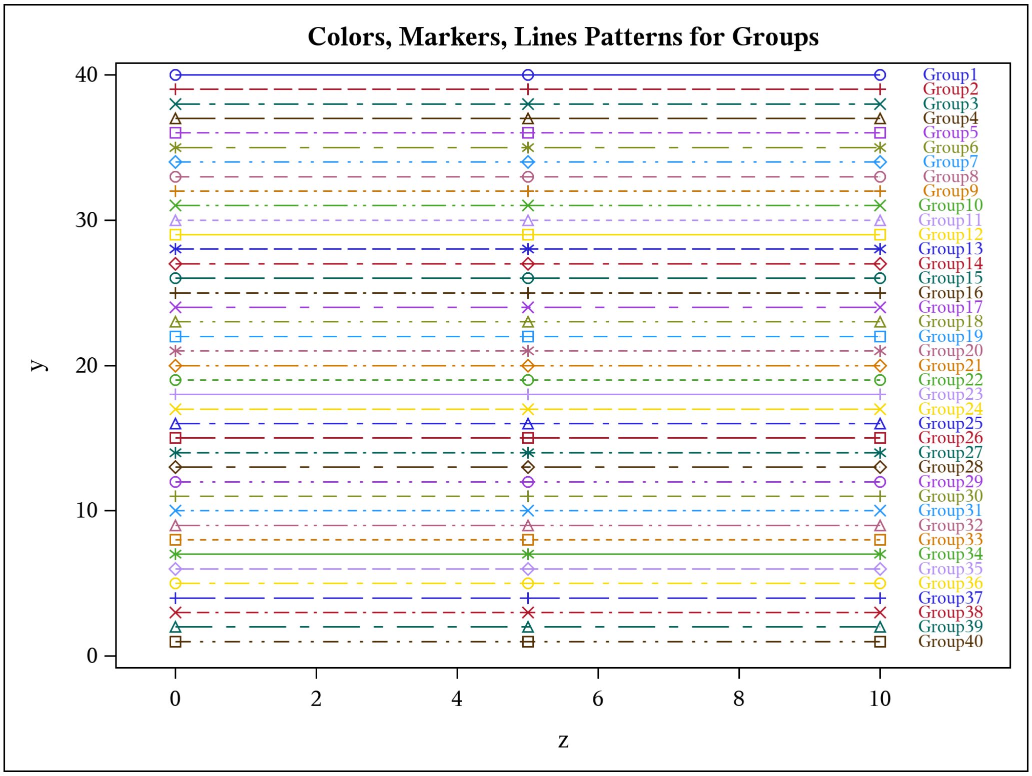


Figure 21.62 Markers, Lines, and Colors with Groups in the PEARL Style

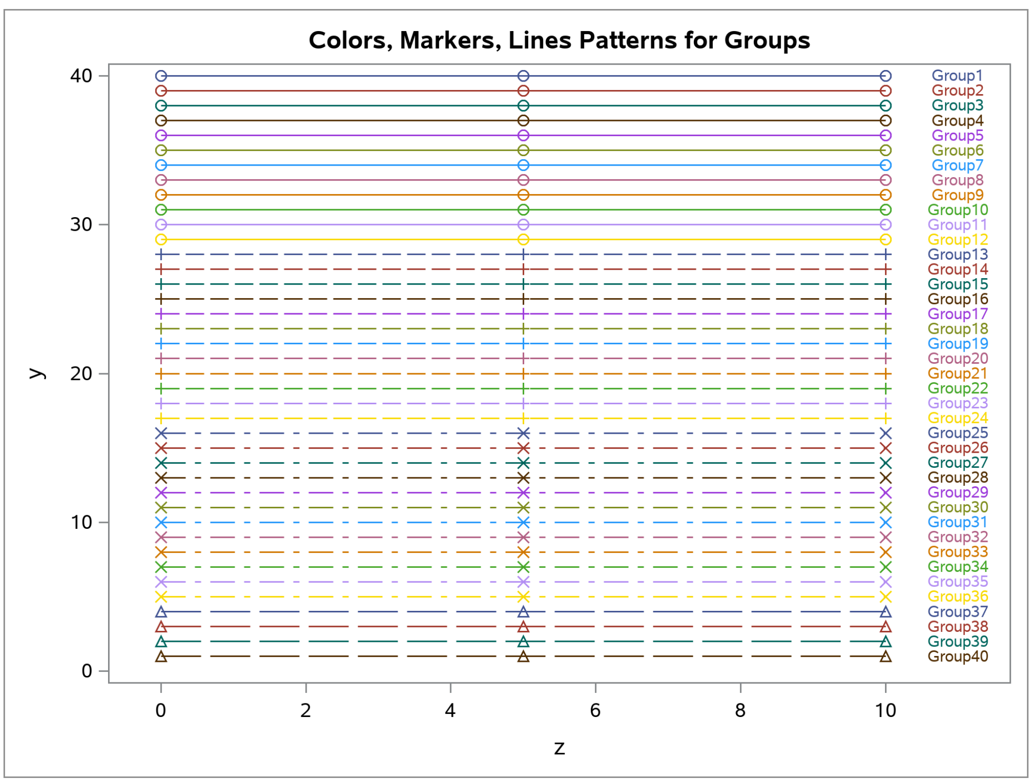


Figure 21.63 Markers, Lines, and Colors with Groups in the PEARLJ Style

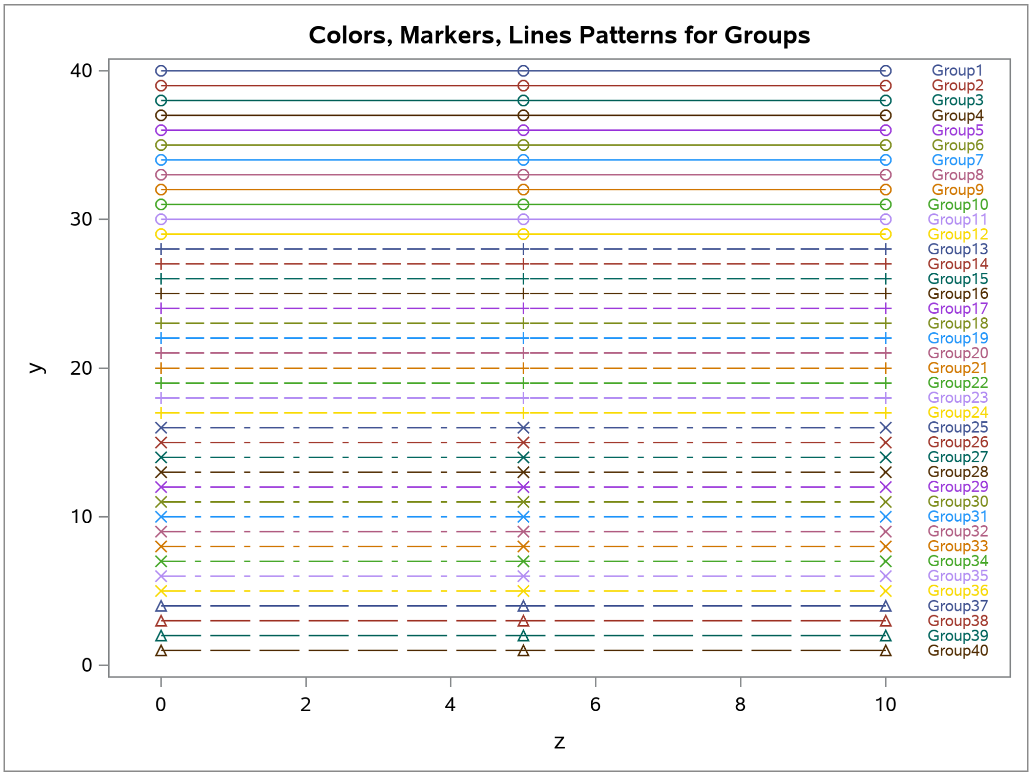
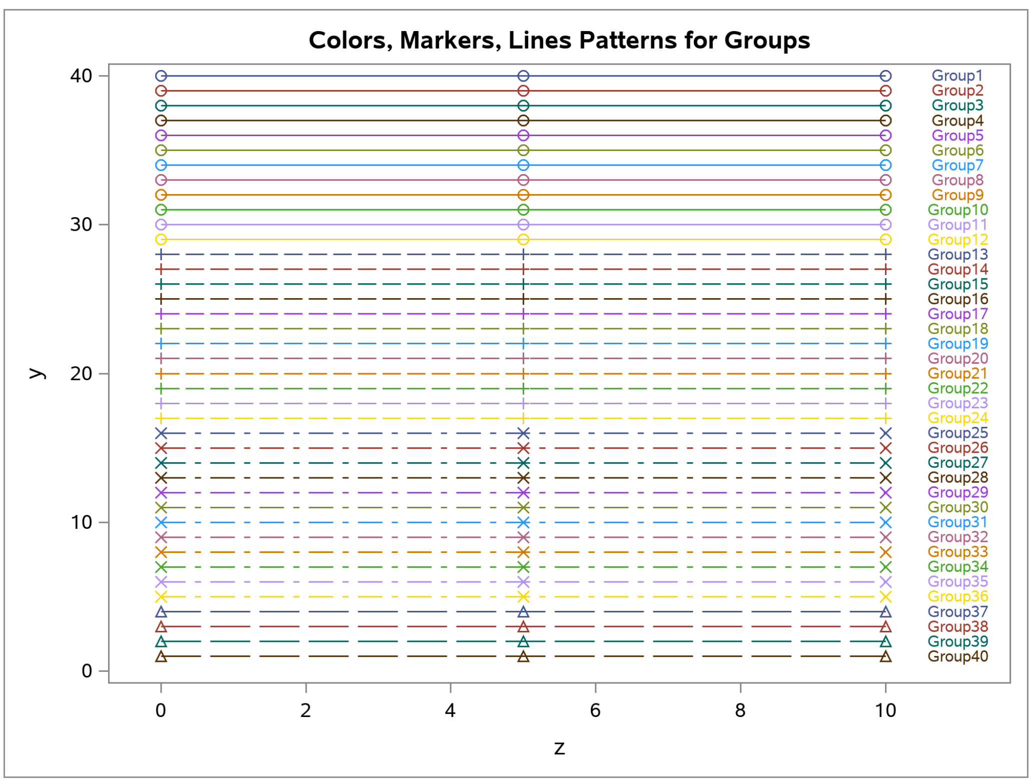


Figure 21.64 Markers, Lines, and Colors with Groups in the SAPPHIRE Style



Modifying the HTMLBLUE Style

The HTMLBLUE style is an all-color style for the first 12 groups of observations. After each set of 12 groups, the line style and marker change for the next 12 groups. (See Figure 21.55.) The HTMLBLUECML style is a color style in which groups of observations are distinguished by simultaneous color, line style, and symbol changes. (See Figure 21.56.) For some graphs, you might want more differentiation than you get in an all-color style like HTMLBLUE but without the overkill differentiation of the HTMLBLUECML style and other styles. This section defines four new styles for this purpose:

HTMLBLUEL – line styles and colors vary together with fixed markers for each set of 11 groups.
HTMLBLUEM – markers and colors vary together with fixed line styles for each set of 11 groups.
HTMLBLUEFL – line styles and colors vary together with fixed filled markers for each set of 11 groups.
HTMLBLUEFM – filled markers and colors vary together with fixed line styles for each set of 5 groups.

The following statements show part of the style template for each of these styles:

```
define style Styles.HTMLBlueL;    parent = styles.htmlbluecml;
   style GraphFit2   from GraphFit2   /                         linestyle = 1;
   style GraphData1  from GraphData1  / markersymbol = "circle" linestyle = 1;
   style GraphData2  from GraphData2  / markersymbol = "circle" linestyle = 4;
   style GraphData3  from GraphData3  / markersymbol = "circle" linestyle = 8;
   style GraphData4  from GraphData4  / markersymbol = "circle" linestyle = 5;
   style GraphData5  from GraphData5  / markersymbol = "circle" linestyle = 14;
   style GraphData6  from GraphData6  / markersymbol = "circle" linestyle = 26;
   style GraphData7  from GraphData7  / markersymbol = "circle" linestyle = 15;
   style GraphData8  from GraphData8  / markersymbol = "circle" linestyle = 20;
   style GraphData9  from GraphData9  / markersymbol = "circle" linestyle = 41;
   style GraphData10 from GraphData10 / markersymbol = "circle" linestyle = 42;
   style GraphData11 from GraphData11 / markersymbol = "circle" linestyle = 2;
   style GraphData12 from GraphData12 / markersymbol = "square" linestyle = 1;
   style GraphData13 from GraphData1  / markersymbol = "square" linestyle = 4;
   style GraphData14 from GraphData2  / markersymbol = "square" linestyle = 8;
   style GraphData15 from GraphData3  / markersymbol = "square" linestyle = 5;
   . . .
end;
define style Styles.HTMLBlueM;    parent = styles.htmlbluecml;
   style GraphFit2   from GraphFit2   /                                 linestyle = 1;
   style GraphData1  from GraphData1  / markersymbol = "circle"         linestyle = 1;
   style GraphData2  from GraphData2  / markersymbol = "square"         linestyle = 1;
   style GraphData3  from GraphData3  / markersymbol = "diamond"        linestyle = 1;
   style GraphData4  from GraphData4  / markersymbol = "asterisk"       linestyle = 1;
   style GraphData5  from GraphData5  / markersymbol = "plus"           linestyle = 1;
   style GraphData6  from GraphData6  / markersymbol = "triangle"       linestyle = 1;
   style GraphData7  from GraphData7  / markersymbol = "circlefilled"   linestyle = 1;
   style GraphData8  from GraphData8  / markersymbol = "starfilled"     linestyle = 1;
   style GraphData9  from GraphData9  / markersymbol = "squarefilled"   linestyle = 1;
   style GraphData10 from GraphData10 / markersymbol = "diamondfilled"  linestyle = 1;
   style GraphData11 from GraphData11 / markersymbol = "trianglefilled" linestyle = 1;
   style GraphData12 from GraphData12 / markersymbol = "circle"         linestyle = 4;
   style GraphData13 from GraphData1  / markersymbol = "square"         linestyle = 4;
   style GraphData14 from GraphData2  / markersymbol = "diamond"        linestyle = 4;
   style GraphData15 from GraphData3  / markersymbol = "asterisk"       linestyle = 4;
   . . .
end;
```

```
define style Styles.HTMLBlueFL;    parent = styles.htmlbluecml;
   style GraphFit2   from GraphFit2   /                               linestyle = 1;
   style GraphData1  from GraphData1  / markersymbol = "circlefilled" linestyle = 1;
   style GraphData2  from GraphData2  / markersymbol = "circlefilled" linestyle = 4;
   style GraphData3  from GraphData3  / markersymbol = "circlefilled" linestyle = 8;
   style GraphData4  from GraphData4  / markersymbol = "circlefilled" linestyle = 5;
   style GraphData5  from GraphData5  / markersymbol = "circlefilled" linestyle = 14;
   style GraphData6  from GraphData6  / markersymbol = "circlefilled" linestyle = 26;
   style GraphData7  from GraphData7  / markersymbol = "circlefilled" linestyle = 15;
   style GraphData8  from GraphData8  / markersymbol = "circlefilled" linestyle = 20;
   style GraphData9  from GraphData9  / markersymbol = "circlefilled" linestyle = 41;
   style GraphData10 from GraphData10 / markersymbol = "circlefilled" linestyle = 42;
   style GraphData11 from GraphData11 / markersymbol = "circlefilled" linestyle = 2;
   style GraphData12 from GraphData12 / markersymbol = "starfilled"   linestyle = 1;
   style GraphData13 from GraphData1  / markersymbol = "starfilled"   linestyle = 4;
   style GraphData14 from GraphData2  / markersymbol = "starfilled"   linestyle = 8;
   style GraphData15 from GraphData3  / markersymbol = "starfilled"   linestyle = 5;
   . . .
end;
define style Styles.HTMLBlueFM;    parent = styles.htmlbluecml;
   style GraphFit2   from GraphFit2   /                                 linestyle = 1;
   style GraphData1  from GraphData1  / markersymbol = "circlefilled"   linestyle = 1;
   style GraphData2  from GraphData2  / markersymbol = "starfilled"     linestyle = 1;
   style GraphData3  from GraphData3  / markersymbol = "squarefilled"   linestyle = 1;
   style GraphData4  from GraphData4  / markersymbol = "diamondfilled"  linestyle = 1;
   style GraphData5  from GraphData5  / markersymbol = "trianglefilled" linestyle = 1;
   style GraphData6  from GraphData6  / markersymbol = "circlefilled"   linestyle = 4;
   style GraphData7  from GraphData7  / markersymbol = "starfilled"     linestyle = 4;
   style GraphData8  from GraphData8  / markersymbol = "squarefilled"   linestyle = 4;
   style GraphData9  from GraphData9  / markersymbol = "diamondfilled"  linestyle = 4;
   style GraphData10 from GraphData10 / markersymbol = "trianglefilled" linestyle = 4;
   style GraphData11 from GraphData11 / markersymbol = "circlefilled"   linestyle = 8;
   style GraphData12 from GraphData12 / markersymbol = "starfilled"     linestyle = 8;
   style GraphData13 from GraphData1  / markersymbol = "squarefilled"   linestyle = 8;
   style GraphData14 from GraphData2  / markersymbol = "diamondfilled"  linestyle = 8;
   style GraphData15 from GraphData3  / markersymbol = "trianglefilled" linestyle = 8;
   . . .
end;
```

New **GraphData***n* style elements are created that inherit colors from the **GraphData1** through **GraphData12** style elements. The line styles and markers are explicitly set in the new style templates. The **style GraphFit2 from GraphFit2 / linestyle = 1** statement creates a solid second fit line. You can remove that statement if you prefer a dashed second fit line.

The following statements use SAS macros to generate these four new styles:

```
proc template;
   %let m = circle square diamond asterisk plus triangle circlefilled
            starfilled squarefilled diamondfilled trianglefilled;
   %let ls = 1 4 8 5 14 26 15 20 41 42 2;
   %macro makestyle;
      %let l = %eval(%sysfunc(mod(&k,12))+1);
      %let k = %eval(&k+1);
      style GraphData&k from GraphData&l /
            linestyle=%scan(&ls, &j) markersymbol="%scan(&m, &i)";
      %mend;
   define style styles.HTMLBlueL;
```

```
      parent=styles.htmlbluecml;
      style GraphFit2 from GraphFit2 / linestyle = 1;
      %macro htmlbluel;
         %let k = 0;
         %do i = 1 %to 11; %do j = 1 %to 11; %makestyle %end; %end;
         %mend;
      %htmlbluel
      end;
   define style styles.HTMLBlueM;
      parent=styles.htmlbluecml;
      style GraphFit2 from GraphFit2 / linestyle = 1;
      %macro htmlbluem;
         %let k = 0;
         %do j = 1 %to 11; %do i = 1 %to 11; %makestyle %end; %end;
         %mend;
      %htmlbluem
      end;
   %let m = circlefilled starfilled squarefilled diamondfilled trianglefilled;
   define style styles.HTMLBlueFL;
      parent=styles.htmlbluecml;
      style GraphFit2 from GraphFit2 / linestyle = 1;
      %macro htmlbluel;
         %let k = 0;
         %do i = 1 %to 5; %do j = 1 %to 11; %makestyle %end; %end;
         %mend;
      %htmlbluel
      end;
   define style styles.HTMLBlueFM;
      parent=styles.htmlbluecml;
      style GraphFit2 from GraphFit2 / linestyle = 1;
      %macro htmlbluem;
         %let k = 0;
         %do j = 1 %to 11; %do i = 1 %to 5; %makestyle %end; %end;
         %mend;
      %htmlbluem
   end;
run;
```

The %LET m statement provides the list of markers. The %LET ls statement provides the list of line styles. The MAKESTYLE macro makes the kth style element from the **GraphData***n* style element for $n = \text{mod}(k - 1, 12) + 1$. The remaining macros vary markers and line styles in the appropriate order over the elements in each list.

The following step, which is used in the section "ODS Style Comparisons" on page 673, is used along with the different styles to produce Figure 21.65 through Figure 21.68:

```
proc sgplot data=x2;
   title 'Colors, Markers, Lines Patterns for Groups';
   series  y=y x=x / group=group markers;
   scatter y=y x=z / group=group markerchar=l;
run;
```

Figure 21.65 Markers, Lines, and Colors with Groups in the HTMLBLUEL Style

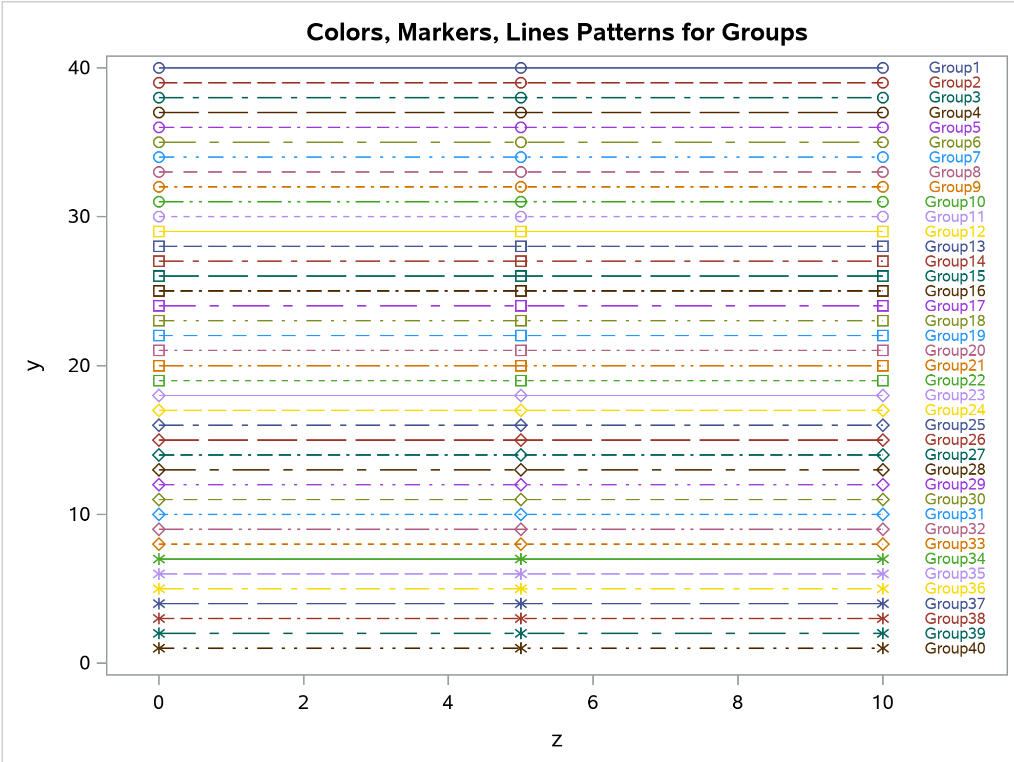


Figure 21.66 Markers, Lines, and Colors with Groups in the HTMLBLUEM Style

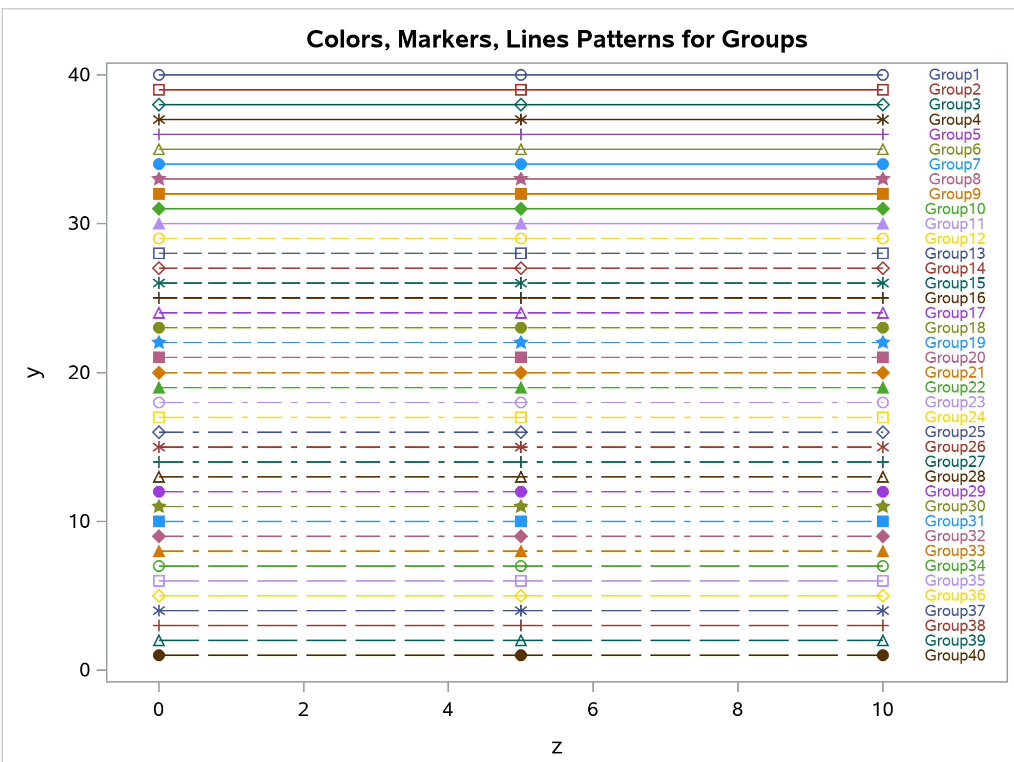


Figure 21.67 Markers, Lines, and Colors with Groups in the HTMLBLUEFL Style

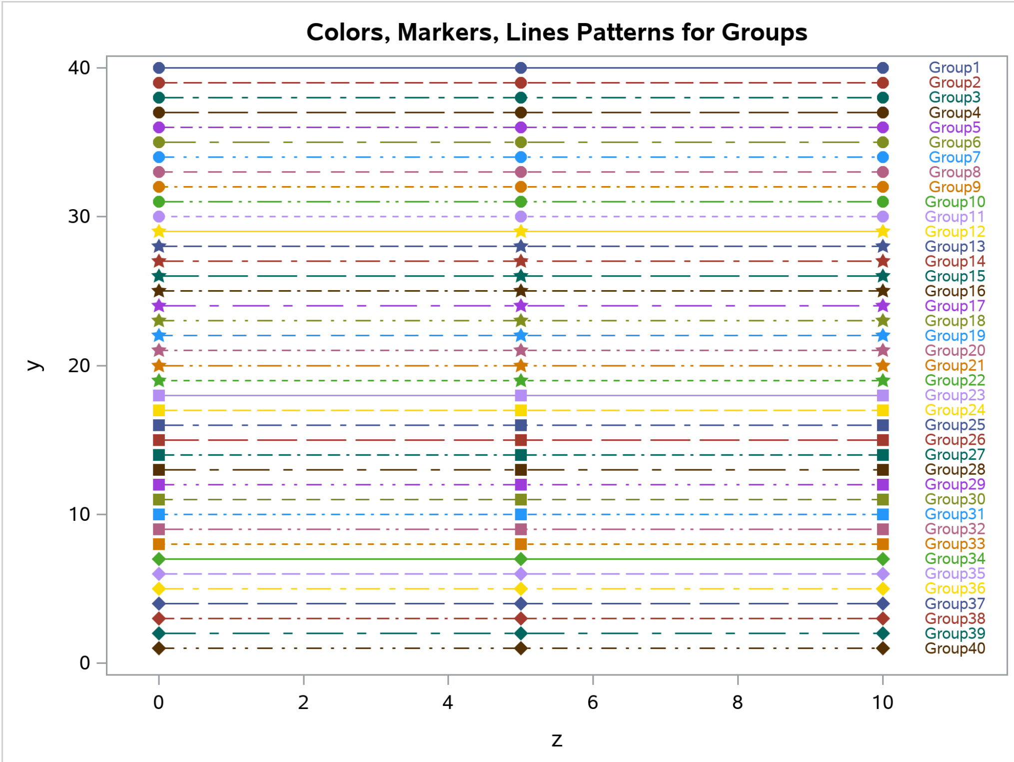
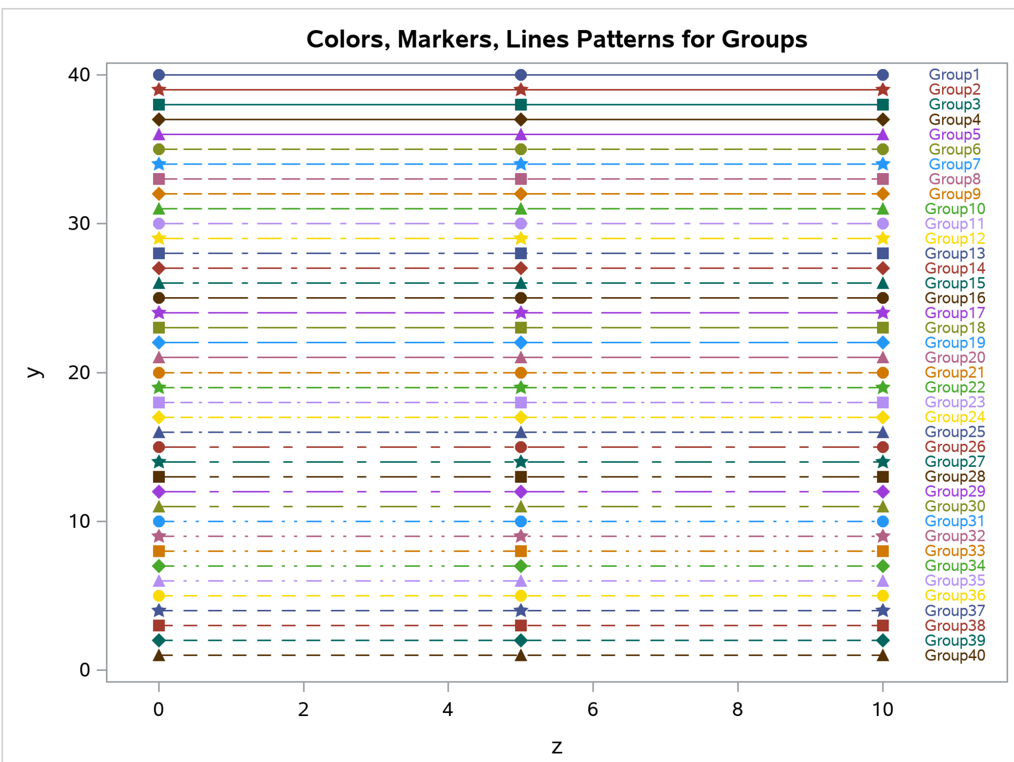


Figure 21.68 Markers, Lines, and Colors with Groups in the HTMLBLUEFM Style



ODS Style Template Modification Macro

The **%ModStyle** macro provides easy ways to customize the style elements (**GraphData1**–**GraphData***n*) that control how groups of observations are distinguished. Examples of using the **%ModStyle** macro can be found in the section "Changing the Default Markers and Lines" on page 705. Also see Kuhfeld (2009) for more information about this macro. **NOTE:** Many of the things that you can do with the **%ModStyle** macro are more conveniently accomplished by using the ATTRPRIORITY= option in the ODS GRAPHICS statement and the options in the STYLEATTRS statement in PROC SGPLOT or the corresponding options in the BEGINGRAPH statement in PROC TEMPLATE. Specify the ATTRPRIORITY=NONE option in the ODS GRAPHICS statement or specify an ATTRPRIORITY=NONE style such as STATISTICAL in the PARENT= option if you want to distinguish groups by marker or line changes. For more information, see the sections "ODS Styles" on page 624 and "Grouped Scatter Plot with PROC SGPLOT" on page 620.

You do not need to include autocall macros (for example, by specifying a **%include** statement). You can call them directly after they are properly installed. If your site has installed the autocall libraries that are supplied by the SAS System and uses the standard configuration of SAS software, you need to ensure that the SAS System option MAUTOSOURCE is in effect before you can begin using the autocall macros. For more information about autocall libraries, see the *SAS Macro Language: Reference*. For information about installing autocall macros, consult your host documentation.

The **%ModStyle** macro has the following options:

COLORS=*color-list*

specifies a space-delimited list of colors for markers and lines. If you do not specify this option, then the colors from the parent style are used. You can specify the colors by using any SAS color notation, such as CX*rrggbb*.

COLORS=GRAYS generates seven distinguishable grayscale colors from blackest to whitest. The colors should be mixed up to be more easily distinguished when you need fewer colors, but you can do that in your own COLORS= list. The HLS (hue/light/saturation) coding generates colors by setting hue and saturation to 0 and incrementing the lightness for each gray. You can also use the keywords BLUES, PURPLES, MAGENTAS, REDS, ORANGES, YELLOWS, GREENS, and CYANS to generate seven colors that have a fixed hue and a saturation of AA (hex).

COLORS=SHADES INT generates seven colors as described previously, except that you specify an integer $0 \leq \text{INT} < 360$. (See *SAS/GRAPH: Reference*.) The available hues are GRAY (or GREY), BLUE=0, PURPLE=30, MAGENTA=60, RED=120, ORANGE=150, YELLOW=180, GREEN=240, and CYAN=300.

DISPLAY=*n*

specifies whether to display the generated template. By default, the template is not displayed. Specify DISPLAY=1 to display the generated template.

FILLCOLORS=*color-list*

specifies a space-delimited list of colors for bands and fills. If you do not specify this option, then the colors from the parent style are used.

Fill colors from the parent style are designed to work well with the colors from the parent style. If you specify a COLORS= list, then you might also want to redefine the FILLCOLORS= list. You need to have at least as many fill colors as you have colors (any extra fill colors are ignored). Two shortcuts are available: FILLCOLORS=COLORS uses the COLORS= colors for the fills (your confidence bands should have transparency for this shortcut to be useful), and FILLCOLORS=LIGHTCOLORS modifies the lightness that is associated with each color generated by COLORS=SHADES (this is allowed only with COLORS=SHADES).

LINESTYLES=*line-style-list*

specifies a space-delimited list of line styles. The default is

```
LineStyles=Solid MediumDash MediumDashShortDash LongDash
           DashDashDot LongDashShortDash DashDotDot Dash
           ShortDashDot MediumDashDotDot ShortDash
```

Line style numbers can range from 1 to 46. Some line styles have names associated with them. You can specify either the name or the number for the following number/name pairs: 1 Solid, 2 ShortDash, 4 MediumDash, 5 LongDash, 8 MediumDashShortDash, 14 DashDashDot, 15 DashDotDot, 20 Dash, 26 LongDashShortDash, 34 Dot, 35 ThinDot, 41 ShortDashDot, and 42 MediumDashDotDot.

MARKERS=*marker-list*

specifies a space-delimited list of marker symbols. By default, **`Markers=Circle Plus X Triangle Square Asterisk Diamond`**. The available marker symbols are listed in the *SAS Graph Template Language: Reference*. Two shortcuts are available: MARKERS=FILLED is an alias for the specification **`Markers=CircleFilled TriangleFilled SquareFilled DiamondFilled StarFilled HomeDownFilled`**, and MARKERS=EMPTY is an alias for the specification **`Markers=Circle Triangle Square Diamond Star HomeDown`**.

NAME=*style-name*

specifies the name of the new style that you are creating. This name is used when you specify the style in an ODS destination statement (for example, ODS HTML STYLE=*style-name*). By default, NAME=NEWSTYLE.

NUMBEROFGROUPS=*n*

specifies *n*, the number of **`GraphData`***n* style elements to create. The **`GraphData1`**–**`GraphData`***n* style elements contain *n* combinations of colors, markers, and line styles. By default, 32 combinations are created.

PARENT=*style-name*

specifies the parent style. The new style inherits most of its attributes from the parent style. By default, PARENT=DEFAULT (which is the top-level parent style for all the styles that are recommended for statistical graphics). If your goal is to change colors or create an all-color style, you can use any style as the parent style. However, if your goal is to change markers or line styles without creating an all-color style, do not use ATTRPRIORITY='Color' style (such as HTMLBLUE, PEARL, PEARLJ, and SAPPHIRE) as a parent.

TYPE=*type-specification*

specifies how your new style cycles through colors, markers, and line styles. The values that are specified in the TYPE= option are case-sensitive ("by" is lowercase and the L, C, and M are uppercase). By default, TYPE=LMbyC.

These first three methods work well in all plots, because cycling line styles and markers together ensures that both scatter-plot markers and series plot lines are distinguishable:

CLM cycles through colors, line styles, and markers simultaneously. The first group uses the first color, line style, and marker; the second group uses the second color, line style, and marker; and so on. This is the method that most styles use; it corresponds to ATTRPRIORITY='None'.

LMbyC fixes line style and marker, cycles through colors, and then moves to the next line style and marker. This is the default; it creates a style where the first groups are distinguished entirely by color, which corresponds to ATTRPRIORITY='Color'.

CbyLM fixes color, cycles through line style and marker, and then moves to the next color. This option uses the smaller of the number of line styles or the number of markers when cycling within a color.

The following two methods might not work well in all plots:

CbyLbyM fixes color and line style, then cycles through markers, increments line style, and then cycles through markers again. After all line styles have been used, this option moves to the next color and continues.

LbyMbyC fixes line style and marker, then cycles through colors, increments marker, and then cycles through colors again. After all markers have been used, this option moves to the next line style and continues. This is closest to the legacy SAS/GRAPH method.

Varying Colors and Markers but Not Lines

Many styles are designed to make color plots where lines, markers, and groups of observations can be distinguished even when the plot is sent to a black-and-white printer. Hence, you can distinguish lines by both their colors and their line patterns. Similarly, you can distinguish markers by both their colors and their symbols. This is not true in the HTMLBLUE, PEARL, PEARLJ, and SAPPHIRE styles, which are all-color styles. You can easily make any style an all-color style by specifying the ATTRPRIORITY= option in the ODS GRAPHICS statement. For more information, see the section “Attribute Priority and Overriding How Groups Are Distinguished” on page 660.

Alternatively, you can make an all-color style by using the %MODSTYLE autocall macro. It creates a new style by modifying a parent style and reordering the colors, line patterns, and marker symbols in the **GraphData***n* style elements (see the section “Some Common ODS Style Elements” on page 668). The macro is documented in the section “ODS Style Template Modification Macro” on page 701.

The following example illustrates how you can use the macro and is taken from the section “Fitting a Curve through a Scatter Plot” on page 10216 in Chapter 123, “The TRANSREG Procedure.” The data come from an experiment in which nitrogen oxide emissions from a single-cylinder engine are measured for various combinations of fuel and equivalence ratio.

The following statements fit separate curves for each group and produce Figure 21.69 and Figure 21.70:

```
ods graphics on;
ods html style=htmlblue; * You can instead specify other destinations
                           such as LISTING, PDF, or RTF;
proc transreg data=sashelp.Gas plots=fit(nocli noclm);
   model identity(nox) = class(Fuel / zero=none) * pbspline(EqRatio);
run;
ods html close;
```

```
%modstyle(parent=statistical, name=StatColor, linestyles=solid, type=CLM)
ods html style=StatColor;
proc transreg data=sashelp.Gas plots=fit(nocli noclm);
   model identity(nox) = class(Fuel / zero=none) * pbspline(EqRatio);
run;
ods html close;
```

The first PROC TRANSREG step uses the HTMLBLUE style (which is an ATTRPRIORITY='Color' style) to create the fit plot in Figure 21.69, which uses only colors to distinguish each group. Then the macro creates a new style called STATCOLOR, which inherits attributes from the STATISTICAL style (which is an ATTRPRIORITY='None' style).

When you specify TYPE=CLM and LINESTYLES=SOLID, the %MODSTYLE macro varies colors and markers but displays only solid lines. In Figure 21.70, which is created by using the modified style, the groups are differentiated by color and marker type.

Figure 21.69 Fit Plot with the HTMLBLUE Style

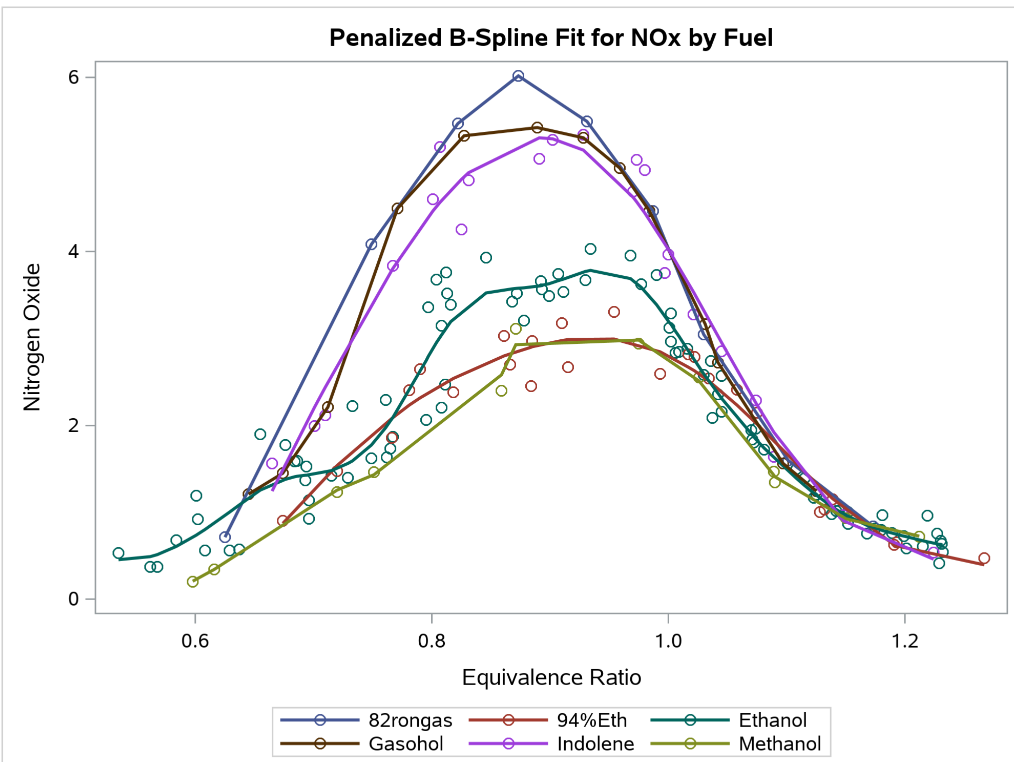
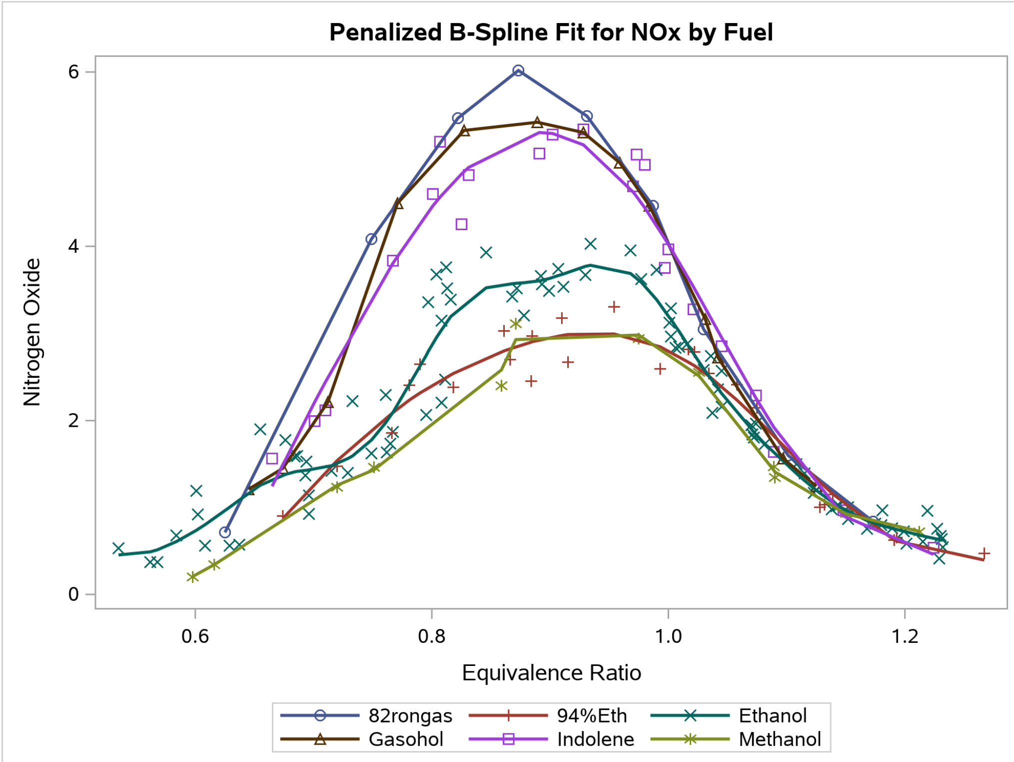


Figure 21.70 Fit Plot with the Modified Style



Changing the Default Markers and Lines

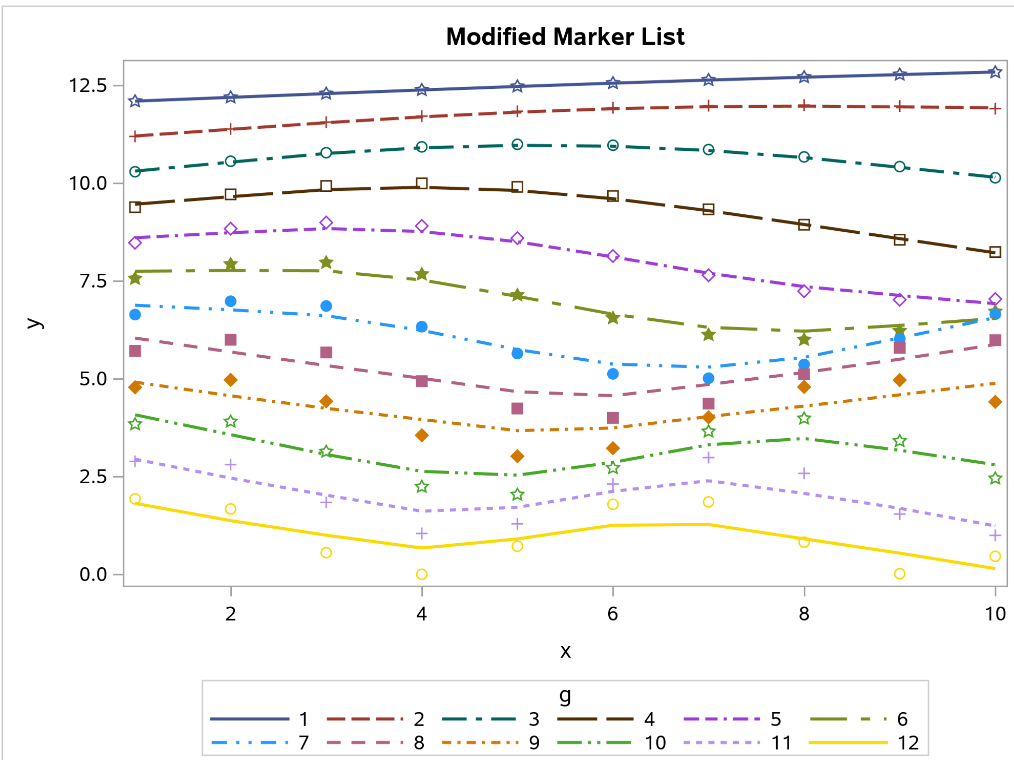
You can use the %MODSTYLE macro to change markers and line styles. This example creates a new style called MARKSTYLE that inherits attributes from the STATISTICAL style but uses a different set of markers. The following statements create artificial data, change the marker list, and display the results:

```
data x;
   do g = 1 to 12;
      do x = 1 to 10;
         y = 13 - g + sin(x * 0.1 * g);
         output;
      end;
   end;
run;

%modstyle(name=markstyle, parent=statistical, type=CLM,
          markers=star plus circle square diamond starfilled
                  circlefilled squarefilled diamondfilled)

ods html style=markstyle; * You can instead specify other destinations
                            such as LISTING, PDF, or RTF;
```

```
proc sgplot;
   title 'Modified Marker List';
   loess y=y x=x / group=g;
run;
ods html close;
```

The NAME= option specifies the new style name, and the PARENT= option specifies the parent style. The TYPE= option controls the method of cycling through colors, lines, and markers. The default, TYPE=LMbyC, fixes (holds constant) the line styles and markers while cycling through the color list. This example uses TYPE=CLM to cycle through colors, line styles, and markers (holding none of them constant). Other TYPE= values are described in the section "ODS Style Template Modification Macro" on page 701. The values that are specified in the TYPE= option are case-sensitive ("by" is lowercase and the L, C, and M are uppercase). The new marker list is specified in the MARKERS= option. The results are displayed in Figure 21.71. The marker list is reused in the tenth and subsequent groups because only nine markers are defined.

Figure 21.71 A Modified Style with a New List of Markers

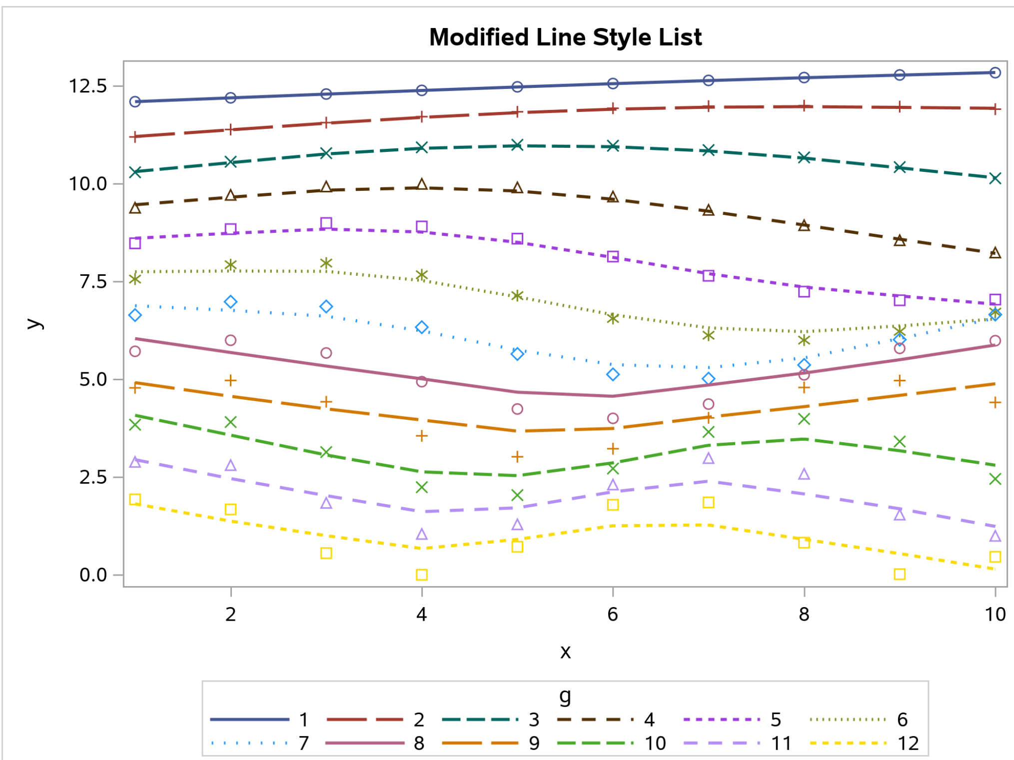


The following statements create a new style called LINESTYLE that inherits attributes from the STATISTICAL style and changes the line list:

```
%modstyle(name=linestyle, parent=statistical, type=CLM,
          linestyles=Solid LongDash MediumDash Dash ShortDash Dot ThinDot)

ods html style=linestyle; * You can instead specify other destinations
                            such as LISTING, PDF, or RTF;
proc sgplot;
   title 'Modified Line Style List';
   loess y=y x=x / group=g;
run;
ods html close;
```

The new line list is specified in the LINESTYLES= option. The results are displayed in Figure 21.72. In this example, each of the first seven groups uses a dash that is shorter than the dash in the previous group. The line list is reused in the eighth and subsequent groups because only seven line patterns are defined.

Figure 21.72 Modified Style with a New List of Line Styles



You can learn more about style modification by examining the new styles, as in the following example:

```
proc template;
   source styles.markstyle;
   source styles.linestyle;
run;
```

The results show the definitions of **GraphData1** through **GraphData32** that the macro created. An abridged listing of the results follows:

```
define style Styles.Markstyle;
   parent = Styles.statistical;
   . . .
   style GraphData1 /
      markersymbol = "star"
      linestyle = 1
      contrastcolor = ColorStyles('c1')
      color = FillStyles('f1');
   . . .
   style GraphData32 /
      markersymbol = "diamond"
      linestyle = 42
      contrastcolor = ColorStyles('c8')
      color = FillStyles('f8');
end;

define style Styles.Linestyle;
   parent = Styles.statistical;
   . . .
   style GraphData1 /
      markersymbol = "circle"
      linestyle = 1
      contrastcolor = ColorStyles('c1')
      color = FillStyles('f1');
   . . .
   style GraphData32 /
      markersymbol = "triangle"
      linestyle = 20
      contrastcolor = ColorStyles('c8')
      color = FillStyles('f8');
end;
```

You can use the NUMBEROFGROUPS= option in the %MODSTYLE macro to control the number of **GraphData***n* style elements that the new style creates.

Modifying Graph Fonts in Styles

You can modify an ODS style to customize the general appearance of plots that ODS Graphics produces, just as you can modify a style to customize the general appearance of ODS tables. This section shows you how to customize fonts that are used in graphs. The following step displays the HTMLBLUE style and its parent styles, STATISTICAL and DEFAULT:

```
proc template;
   source Styles.HTMLBlue / expand;
run;
```

If you search for “font”, you find the style elements that control graph fonts:

```
class GraphFonts /
   'NodeDetailFont' = ("<sans-serif>, <MTsans-serif>",7pt)
   'NodeInputLabelFont' = ("<sans-serif>, <MTsans-serif>",9pt)
   'NodeLabelFont' = ("<sans-serif>, <MTsans-serif>",9pt)
   'NodeTitleFont' = ("<sans-serif>, <MTsans-serif>",9pt)
   'GraphDataFont' = ("<sans-serif>, <MTsans-serif>",7pt)
   'GraphUnicodeFont' = ("<MTsans-serif-unicode>",9pt)
   'GraphValueFont' = ("<sans-serif>, <MTsans-serif>",9pt)
   'GraphLabel2Font' = ("<sans-serif>, <MTsans-serif>",10pt)
   'GraphLabelFont' = ("<sans-serif>, <MTsans-serif>",10pt)
   'GraphFootnoteFont' = ("<sans-serif>, <MTsans-serif>",10pt)
   'GraphTitleFont' = ("<sans-serif>, <MTsans-serif>",11pt,bold)
   'GraphTitle1Font' = ("<sans-serif>, <MTsans-serif>",14pt,bold)
   'GraphAnnoFont' = ("<sans-serif>, <MTsans-serif>",10pt);
```

The fonts `GraphTitle1Font` and `GraphLabel2Font` are not used by ODS Graphics. The following fonts are the ones that are usually used for the text in most graphs:

- `GraphDataFont` is the smallest font. It is used for text that needs to be small (labels for points in scatter plots, labels for contours, and so on).

- `GraphValueFont` is the next-largest font. It is used for axis value (tick mark) labels and legend entry labels.

- `GraphLabelFont` is the next-largest font. It is used for axis labels and legend titles.

- `GraphFootnoteFont` is the next-largest font. It is used for all footnotes.

- `GraphTitleFont` is the largest font. It is used for all titles.

- `GraphUnicodeFont` is used for special characters. (See the section “Unicode and Special Characters” on page 816 in Chapter 22, “ODS Graphics Template Modification.”)

The following statements define a style named NEWSTYLE that replaces the graph fonts in the STATISTICAL style by Times New Roman fonts, which are available in the Windows operating system:

```
proc template;
   define style Styles.NewStyle;
      parent=Styles.Statistical;
      class GraphFonts  /
         'GraphAnnoFont'     = ("<MTserif>, Times New Roman",10pt)
         'GraphDataFont'     = ("<MTserif>, Times New Roman", 7pt)
         'GraphUnicodeFont'  = ("<MTserif>, Times New Roman", 9pt)
         'GraphValueFont'    = ("<MTserif>, Times New Roman", 9pt)
         'GraphLabel2Font'   = ("<MTserif>, Times New Roman",10pt)
         'GraphLabelFont'    = ("<MTserif>, Times New Roman",10pt)
         'GraphFootnoteFont' = ("<MTserif>, Times New Roman",10pt)
         'GraphTitleFont'    = ("<MTserif>, Times New Roman",11pt)
         'GraphTitle1Font'   = ("<MTserif>, Times New Roman",14pt)
         'NodeTitleFont'     = ("<MTserif>, Times New Roman", 9pt)
         'NodeLabelFont'     = ("<MTserif>, Times New Roman", 9pt)
         'NodeLinkLabelFont' = ("<MTserif>, Times New Roman", 9pt)
         'NodeInputLabelFont'= ("<MTserif>, Times New Roman", 9pt)
         'NodeDetailFont'    = ("<MTserif>, Times New Roman", 7pt);
   end;
run;
```

For more information about the DEFINE, PARENT, and CLASS statements, see the *SAS Graph Template Language: Reference*.

The “Getting Started” section in Chapter 104, “The ROBUSTREG Procedure,” creates the following data set to illustrate the use of PROC ROBUSTREG for robust regression:

```
data stack;
   input x1 x2 x3 y @@;
   datalines;
80 27 89 42   80 27 88 37   75 25 90 37   62 24 87 28   62 22 87 18
62 23 87 18   62 24 93 19   62 24 93 20   58 23 87 15   58 18 80 14
58 18 89 14   58 17 88 13   58 18 82 11   58 19 93 12   50 18 89  8
50 18 86  7   50 19 72  8   50 19 79  8   50 20 80  9   56 20 82 15
70 20 91 15
;
```

The following statements create a Q-Q plot that uses the HTMLBLUE style (see Figure 21.73) and the NEWSTYLE style (see Figure 21.74):

```
ods html style=HTMLBlue;
ods graphics on;
proc robustreg data=stack plots=qqplot;
   ods select QQPlot;
   model y = x1 x2 x3;
run;
ods html close;

ods html style=NewStyle;
proc robustreg data=stack plots=qqplot;
   ods select QQPlot;
   model y = x1 x2 x3;
run;
ods html close;
```

Figure 21.73 Q-Q Plot That Uses the HTMLBLUE Style

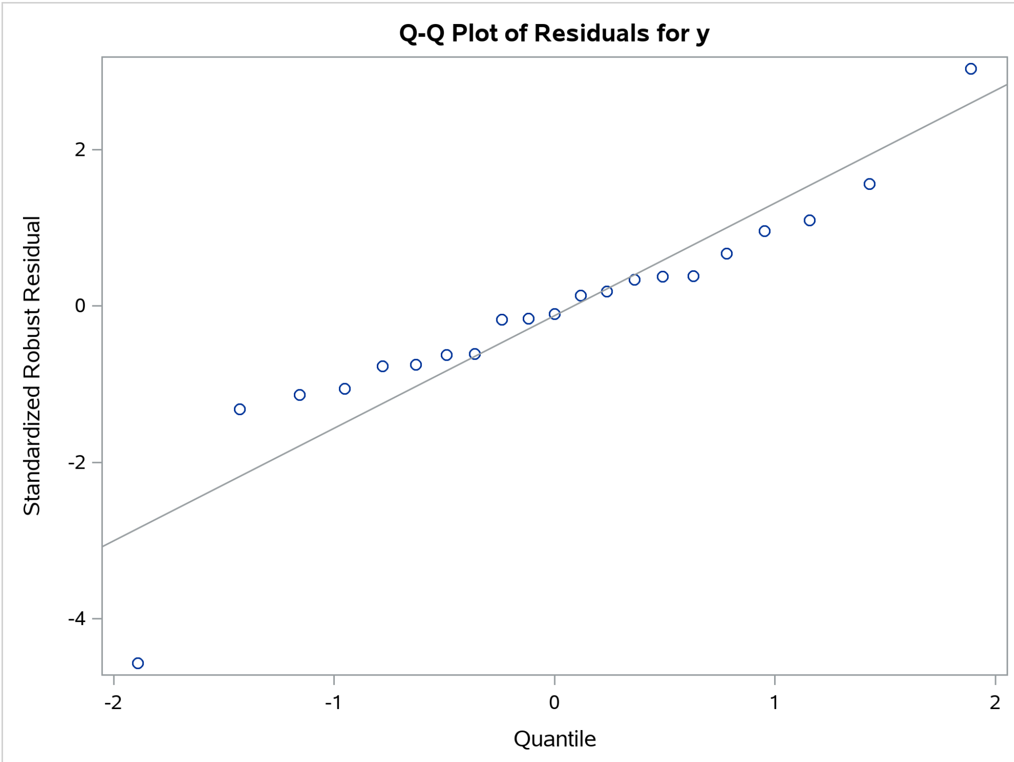
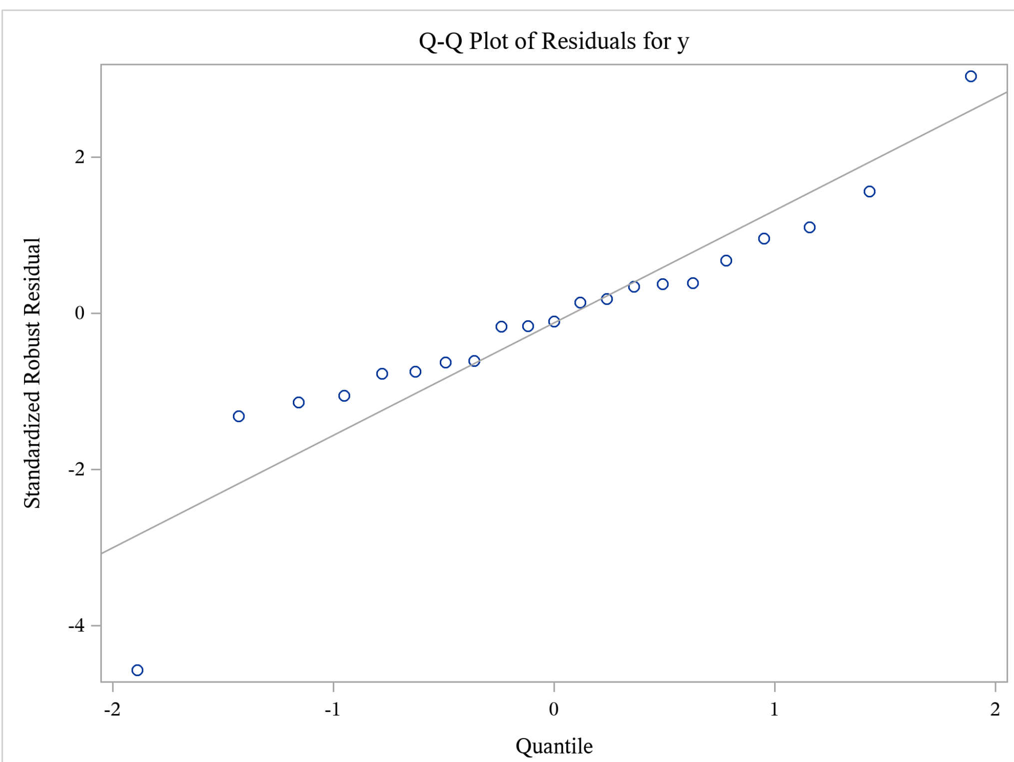


Figure 21.74 Q-Q Plot That Uses the NEWSTYLE Style



Although this example illustrates the use of a style with graphical output from a particular procedure, a style is applied to *all* your output (graphs and tables) in the destination for which you specify the style. For information about specifying a default style for all your output, see the section "Changing the Default Style" on page 714.

Modifying Other Graph Elements in Styles

This section illustrates how to modify other style elements for graphics, specifically the style element **GraphReference**, which controls the attributes of reference lines. You can run the following statements to learn more about the **GraphReference** style element:

```
proc template;
   source styles.HTMLBlue;
run;
```

The following are the first two lines of the source listing:

```
define style Styles.HTMLBlue;
   parent = styles.statistical;
```

There is no mention of **GraphReference** in the template source listing because **GraphReference** is inherited from a parent style. The following step displays the HTMLBLUE style and its parent styles:

```
proc template;
   source Styles.HTMLBlue / expand;
run;
```

The EXPAND option lists the styles in the following order: style of interest, then its parent, and then its grandparent, and so on. The HTMLBLUE style inherits attributes from the STATISTICAL style, which inherits attributes from the DEFAULT style. If you search the results from the top, you will find the most recent specification of a style element first. The **GraphReference** style element is defined as follows:

```
class GraphReference /
   linethickness = 1px
   linestyle = 1
   contrastcolor = GraphColors('greferencelines');
```

To specify a line thickness of 4 pixels for all reference lines, add the following statement to the definition of the NEWSTYLE style in the section "Modifying Graph Fonts in Styles" on page 709:

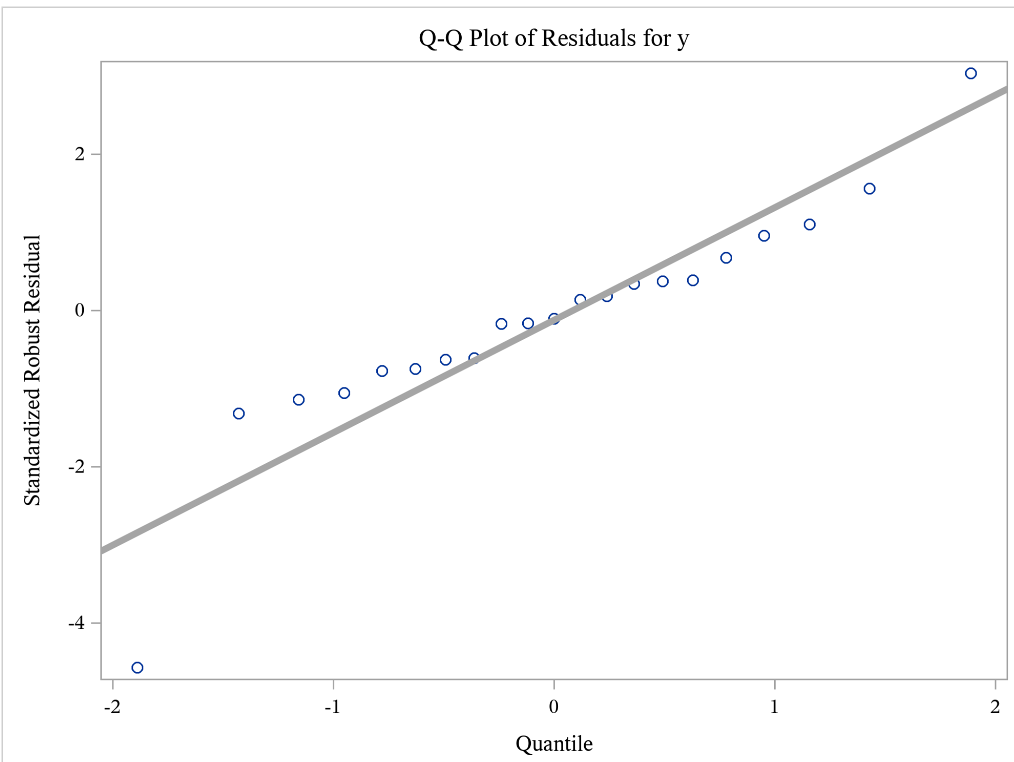
```
class GraphReference / linethickness=4px;
```

The following statements modify the style and produce the Q-Q plot shown in Figure 21.75:

```
proc template;
   define style Styles.NewStyle;
      parent=Styles.Statistical;
      class GraphFonts  /
         'GraphAnnoFont'     = ("<MTserif>, Times New Roman",10pt)
         'GraphDataFont'     = ("<MTserif>, Times New Roman", 7pt)
         'GraphUnicodeFont'  = ("<MTserif>, Times New Roman", 9pt)
         'GraphValueFont'    = ("<MTserif>, Times New Roman", 9pt)
         'GraphLabel2Font'   = ("<MTserif>, Times New Roman",10pt)
         'GraphLabelFont'    = ("<MTserif>, Times New Roman",10pt)
         'GraphFootnoteFont' = ("<MTserif>, Times New Roman",10pt)
         'GraphTitleFont'    = ("<MTserif>, Times New Roman",11pt)
```

```
            'GraphTitle1Font'    = ("<MTserif>, Times New Roman",14pt)
            'NodeTitleFont'      = ("<MTserif>, Times New Roman", 9pt)
            'NodeLabelFont'      = ("<MTserif>, Times New Roman", 9pt)
            'NodeLinkLabelFont' = ("<MTserif>, Times New Roman", 9pt)
            'NodeInputLabelFont'= ("<MTserif>, Times New Roman", 9pt)
            'NodeDetailFont'     = ("<MTserif>, Times New Roman", 7pt);
         class GraphReference / linethickness=4px;
      end;
   run;

   ods html style=NewStyle;
   ods graphics on;

   proc robustreg data=stack plots=qqplot;
      ods select QQPlot;
      model y = x1 x2 x3;
   run;
   ods html close;
```

Figure 21.75 Q-Q Plot That Uses the NEWSTYLE Style with a Thicker Line



You can use this approach to modify other attributes of the line, such as **LineStyle** and **ContrastColor**. These style modifications apply to all graphs that display reference lines, not just Q-Q plots that are produced by PROC ROBUSTREG. You can control the attributes of specific graphs by modifying the graph template, as discussed in the section "Graph Templates" on page 782 in Chapter 22, "ODS Graphics Template Modification." Values that are specified directly in a graph template override style attributes.

When you are done with the NEWSTYLE style, you do not need to restore the HTMLBLUE style template, because you did not modify it. Rather, you inherited its attributes from the HTMLBLUE style.

Changing the Default Style

The default style for each ODS destination is specified in the SAS Registry. For example, the default style for the HTML destination is HTMLBLUE, and the default style for the RTF destination is RTF. You can specify a default style for all your output in a particular ODS destination. This is useful if you want to use a different ODS style, if you have modified one of the styles that the SAS System supplies (see the section "Style Templates and Colors" on page 667), or if you have defined your own style. For example, you can specify the JOURNAL style as the default style for RTF output.

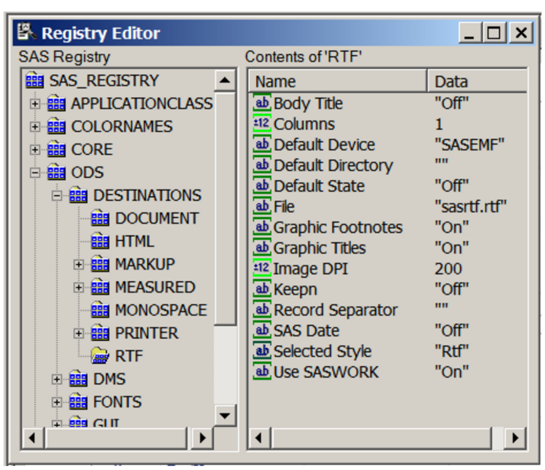
The recommended approach for specifying a default style is as follows. Open the SAS Registry Editor by typing **regedit** on the command line. Expand the node **ODS** ▶ **DESTINATIONS** and select a destination (for example, select **RTF**). Double-click the **Selected Style** item, shown in Figure 21.76, and specify a style. This can be any style that the SAS System supplies or a user-defined style, as long as you can find it in the current template search path (for example, specify **Journal**). You can specify a default style for the other destinations in a similar way.

In a few cases, the default style is specified in more than one place. Assume you are using the SAS windowing environment and Microsoft Windows or UNIX in the following:

- If you expand the node **ODS** ▶ **DESTINATIONS** ▶ **HTML** (which refers to the obsolete HTML3 destination), you see that the **Selected Style** is DEFAULT.
- If you expand the node **ODS** ▶ **MARKUP** ▶ **HTML4**, you see that the **Selected Style** is HTML-BLUE.
- If you expand the node **ODS** ▶ **DMS** ▶ **DESTINATIONS** ▶ **MARKUP** ▶ **HTML4**, you see that the **Selected Style** is HTMLBLUE.

If you want to change the default style for the HTML destination, you need to change both HTML4 entries in the registry.

Figure 21.76 SAS Registry Editor



ODS searches sequentially through each element of the template search path for the first style template that matches the name of the style specified in the SAS Registry. It uses the first style template that it finds. (For more information about the template search path, see the sections "Saving Customized Templates" on page 791, "Using Customized Templates" on page 791, and "Reverting to the Default Templates" on page 792 in Chapter 22, "ODS Graphics Template Modification.") If you are specifying a customized style as your default style, the following are useful suggestions:

- If you save your style in Sasuser.Templat, verify that the name of your default style matches the name of the style specified in the SAS Registry. For example, suppose the RTF style is specified for the RTF destination in the SAS Registry. You can name your style RTF and save it in Sasuser.Templat. This blocks the RTF style in Sashelp.Tmplmst (provided that you did not alter the default template search path).
- If you save your style in a user-defined template store, verify that this template store is the first in the current template search path. Include the ODS PATH statement in your SAS autoexec file so that it is executed at start-up.

For the HTML destination, an alternative approach for specifying a default style is as follows. From the main SAS window, select **Tools ► Options ► Preferences**. On the **Results** tab, select the **Create HTML** check box and select a style from the **Style** list.

Statistical Graphics Procedures

Three Base SAS statistical graphics procedures use ODS Graphics and provide a convenient syntax for creating a variety of graphs from raw data or from procedure output:

SGSCATTER	creates single-cell and multicell scatter plots and scatter plot matrices along with optional fits and ellipses.
SGPLOT	creates single-cell plots along with a variety of plot and chart types.
SGPANEL	creates single-page or multipage panels of plots and charts conditional on classification variables.

You do not need to enable ODS Graphics in order to use these SG (statistical graphics) procedures. In addition, the Base SAS SGRENDER procedure provides a way to create plots from graph templates that you modify or write yourself. For more information about the SG procedures and PROC SGRENDER, see the *SAS ODS Graphics: Procedures Guide* and Kuhfeld (2016).

These procedures do much more than make scatter plots. They can produce density plots, dot plots, needle plots, series plots, horizontal and vertical bar charts, histograms, and box plots. They can also compute and display loess fits, polynomial fits, penalized B-spline fits, reference lines, bands, and ellipses. PROC SGRENDER is the most flexible SG procedure because it uses the Graph Template Language (GTL). The syntax for the other SG procedures is much simpler than that of the GTL, so these procedures are recommended for creating most plots that are commonly required in statistical work.

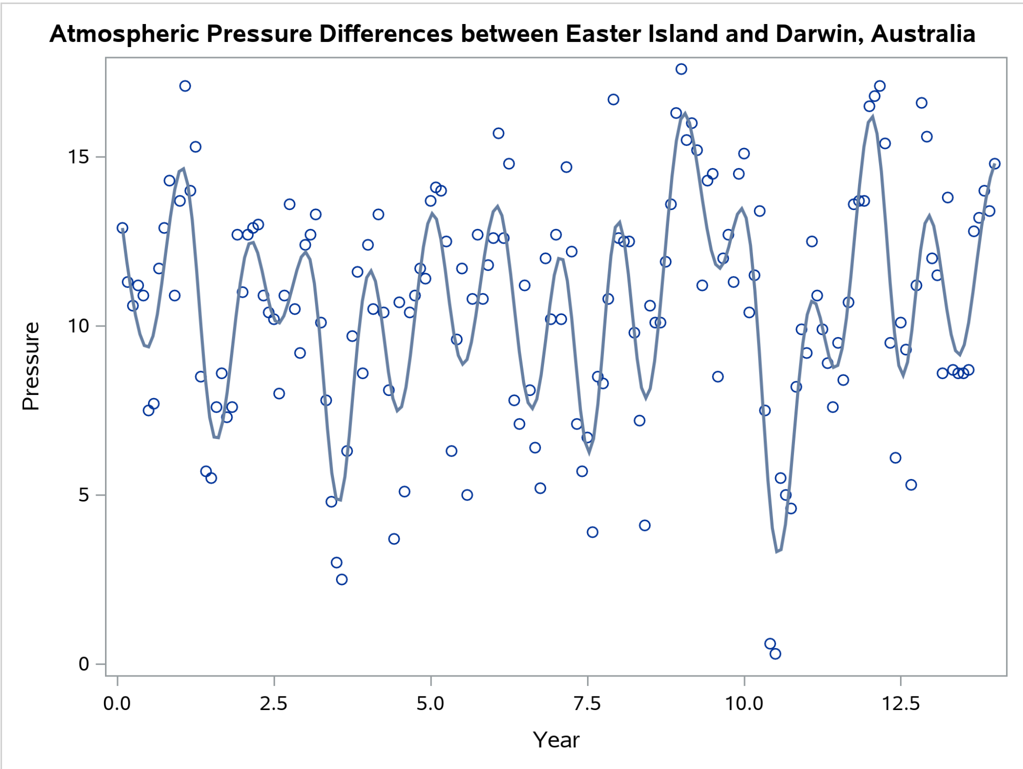
The SGPLOT Procedure

PROC SGPLOT provides a simple way to make a variety of scatter plots. This example is taken from Example 75.4 in Chapter 75, "The LOESS Procedure." The ENSO data set, which contains information about differences in ocean pressure over time, is available from the Sashelp library.

The following statements create a scatter plot of points along with a penalized B-spline fit to the data and produce Figure 21.77:

```
proc sgplot data=sashelp.enso noautolegend;
   title 'Atmospheric Pressure Differences between '
         'Easter Island and Darwin, Australia';
   pbspline y=pressure x=year;
run;
```

Figure 21.77 Penalized B-Spline Fit with PROC SGPLOT



For more information about penalized B-splines, see Chapter 123, "The TRANSREG Procedure." Also see the section "Grouped Scatter Plot with PROC SGPLOT" on page 620 and Figure 21.12 for an example of a scatter plot that has groups of observations.

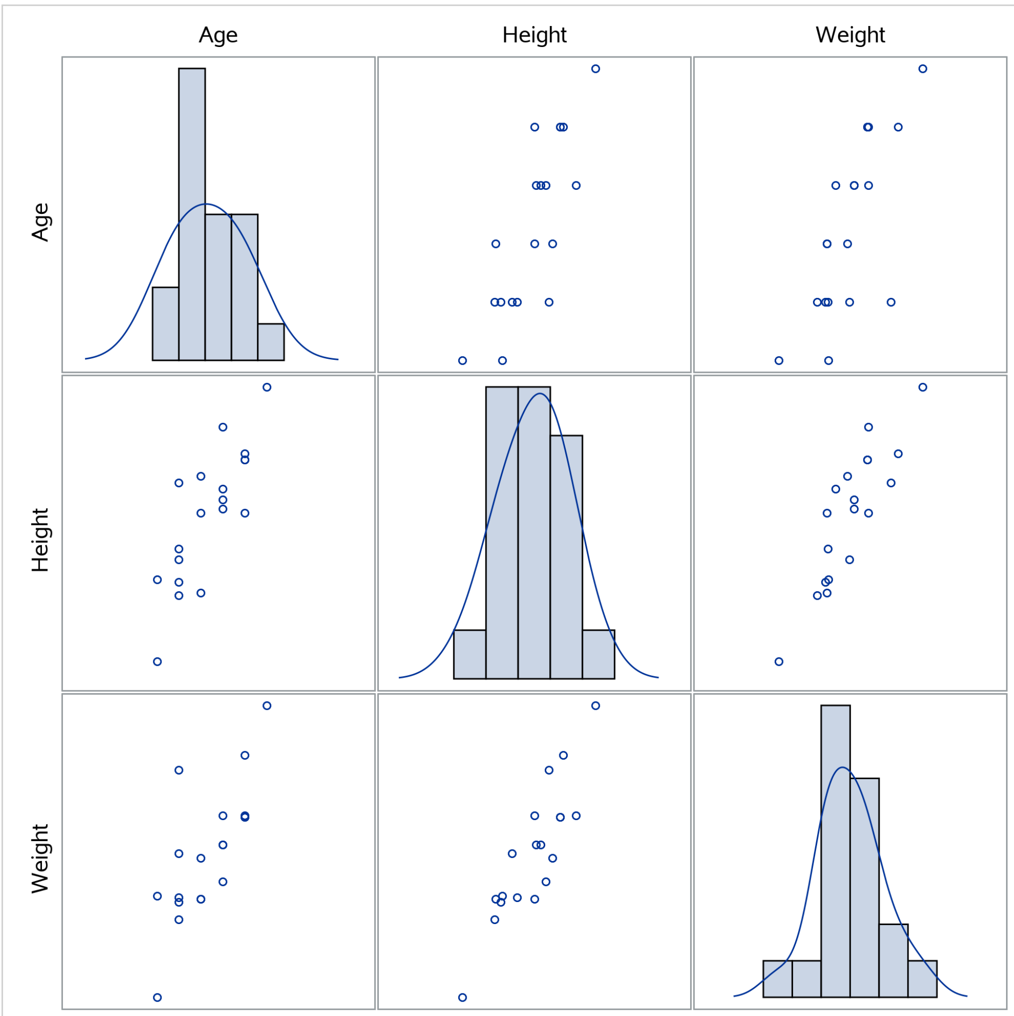
The SGSCATTER Procedure

You can use the SGSCATTER procedure to produce scatter plot matrices. The following step creates a scatter plot matrix from all the numeric variables in the Class data set (available in the Sashelp library) and produces Figure 21.78:

```
proc sgscatter data=sashelp.class;
   matrix _numeric_ / diagonal=(kernel histogram);
run;
```

The diagonal cells of Figure 21.78 contain a histogram and a kernel density fit. The off-diagonal cells contain all pairs of scatter plots.

Figure 21.78 Scatter Plot Matrix with PROC SGSCATTER



The MATRIX statement creates a symmetric $n \times n$ scatter plot matrix. Other statements are also available. The PLOT statement creates a panel that contains one or more individual scatter plots. The COMPARE statement creates a rectangular $m \times n$ scatter plot matrix. Linear and nonlinear fits can be added, and you can request many graphical features by specifying options.

The SGPANEL Procedure

The SGPANEL procedure creates paneled plots and charts that have one or more classification variables. Classification variables can be designated as row or column variables, or there can be multiple classifications.

Graphs are drawn for each combination of the levels of classification variables, showing a subset of the data in each cell.

This example is taken from Example 49.6 in Chapter 49, "The GLIMMIX Procedure." The following statements create the input SAS data sets:

```
data times;
   input time1-time23;
   datalines;
122  150  166  179  219  247  276  296  324  354  380  445
478  508  536  569  599  627  655  668  723  751  781
;

data cows;
   if _n_ = 1 then merge times;
   array t{23} time1 - time23;
   array w{23} weight1 - weight23;
   input cow iron infection weight1-weight23 @@;
   do i=1 to 23;
      weight = w{i};
      tpoint = (t{i}-t{1})/10;
      output;
   end;
   keep cow iron infection tpoint weight;
   datalines;
 1 0 0  4.7    4.905  5.011  5.075  5.136  5.165  5.298  5.323
        5.416  5.438  5.541  5.652  5.687  5.737  5.814  5.799
        5.784  5.844  5.886  5.914  5.979  5.927  5.94
 2 0 0  4.868  5.075  5.193  5.22   5.298  5.416  5.481  5.521

   ... more lines ...

;
```

First, PROC GLIMMIX is run to fit the model, and then the results are prepared for plotting:

```
proc glimmix data=cows;
   t2 = tpoint / 100;
   class cow iron infection;
   model weight = iron infection iron*infection tpoint;
   random t2 / type=rsmooth subject=cow
                    knotmethod=kdtree(bucket=100 knotinfo);
   output out=gmxout pred(blup)=pred;
   nloptions tech=newrap;
run;

data plot;
   set gmxout;
   length Group $ 26;
   if      (iron=0) and (infection=0) then group='Control Group (n=4)';
   else if (iron=1) and (infection=0) then group='Iron - No Infection (n=3)';
   else if (iron=0) and (infection=1) then group='No Iron - Infection (n=9)';
   else group = 'Iron - Infection (n=10)';
run;
```

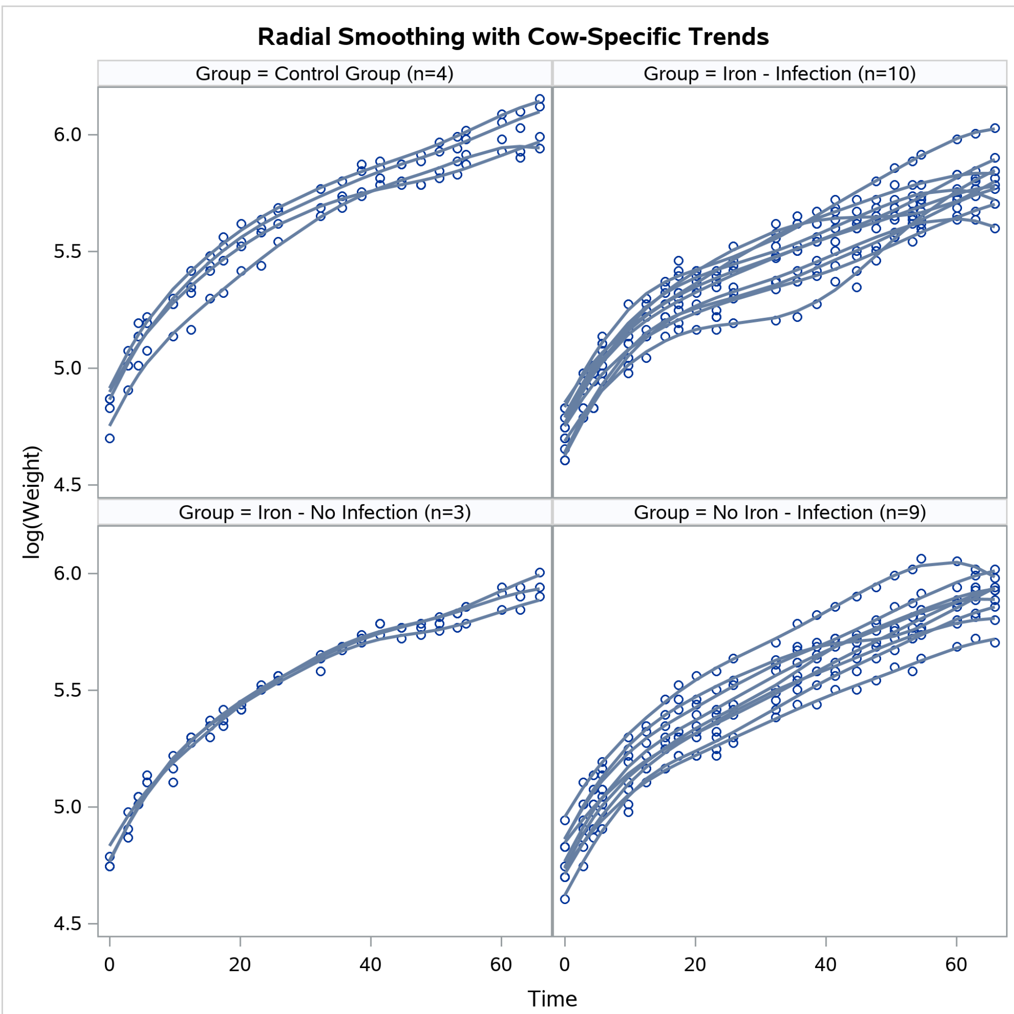
```
proc sort data=plot; by group cow;
run;
```

The following statements produce graphs of the observed data and fitted profiles in the four groups:

```
proc sgpanel data=plot noautolegend;
   title 'Radial Smoothing with Cow-Specific Trends';
   label tpoint='Time' weight='log(Weight)';
   panelby group / columns=2 rows=2;
   scatter x=tpoint y=weight;
   series  x=tpoint y=pred / group=cow lineattrs=GraphFit;
run;
```

The results are shown in Figure 21.79.

Figure 21.79 Fit Using PROC SGPANEL



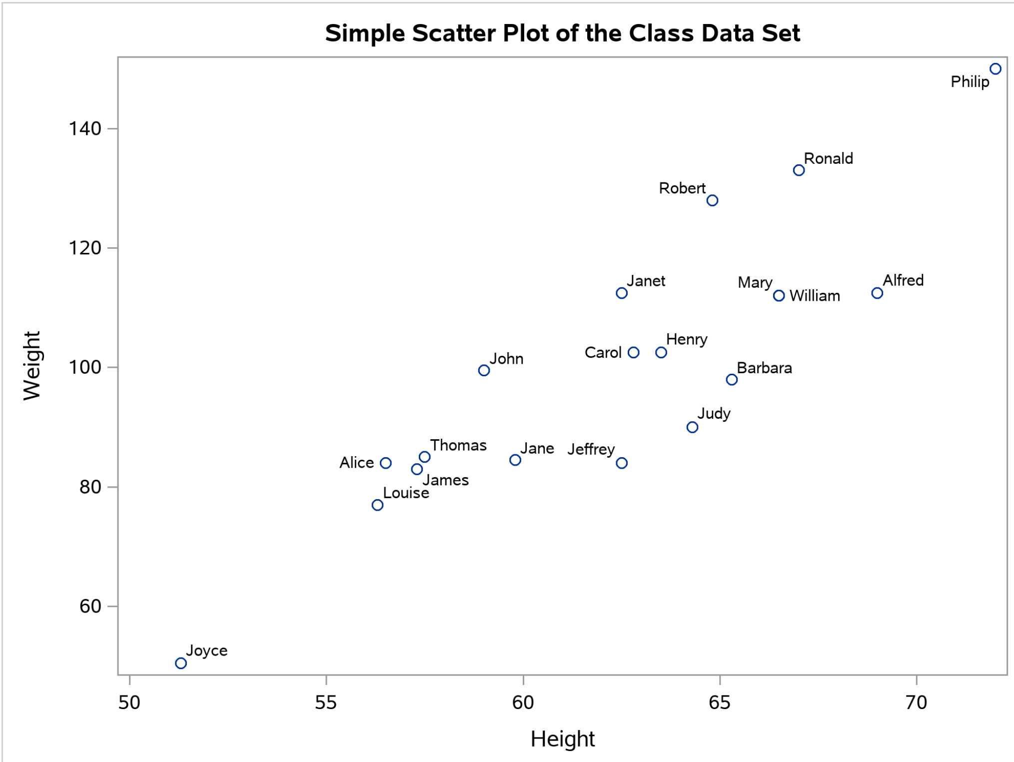
The SGRENDER Procedure

The SGRENDER procedure produces a graph from an input SAS data set and an ODS graph template. By using PROC SGRENDER and the Graph Template Language, you can create highly customized graphs. The following steps create a simple scatter plot of the Class data set (available in the Sashelp library) and produce Figure 21.80:

```
proc template;
   define statgraph Scatter;
      begingraph;
         entrytitle "Simple Scatter Plot of the Class Data Set";
         layout overlay;
            scatterplot y=weight x=height / datalabel=name;
         endlayout;
      endgraph;
   end;
run;

proc sgrender data=sashelp.class template=scatter;
run;
```

The template definition consists of an outer block that begins with a DEFINE statement and ends with an END statement. Inside that is a BEGINGRAPH/ENDGRAPH block. Inside that block, the ENTRYTITLE statement provides the plot title, and the LAYOUT OVERLAY block contains the statement or statements that define the graph. In this case, there is just a single SCATTERPLOT statement that names the Y-axis (vertical) variable, the X-axis (horizontal) variable, and an optional variable that contains labels for the points. The PROC SGRENDER statement simply specifies the input data set and the template. The real work in using PROC SGRENDER is writing the template.

Figure 21.80 Scatter Plot of Labeled Points with PROC SGRENDER



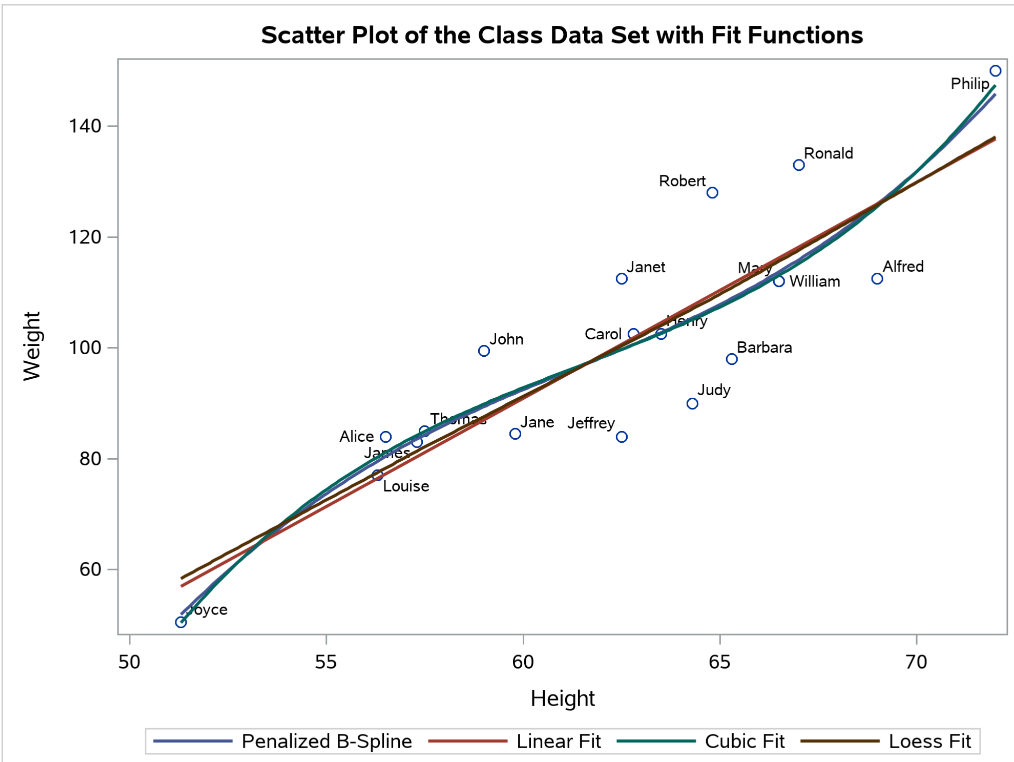
The following steps add a series of fit functions to the scatter plot and create a legend by adding statements to the **Scatter** template:

```
proc template;
   define statgraph Scatter;
      begingraph;
         entrytitle "Scatter Plot of the Class Data Set with Fit Functions";
         layout overlay;
            scatterplot y=weight x=height / datalabel=name;
            pbsplineplot y=weight x=height / name='pbs'
                         legendlabel='Penalized B-Spline'
                         lineattrs=GraphData1;
            regressionplot y=weight x=height / degree=1 name='line'
                         legendlabel='Linear Fit'
                         lineattrs=GraphData2;
            regressionplot y=weight x=height / degree=3 name='cubic'
                         legendlabel='Cubic Fit'
                         lineattrs=GraphData3;
            loessplot y=weight x=height / name='loess'
                         legendlabel='Loess Fit'
                         lineattrs=GraphData4;
            discretelegend 'pbs' 'line' 'cubic' 'loess';
         endlayout;
      endgraph;
   end;
run;
```

```
proc sgrender data=sashelp.class template=scatter;
run;
```

The line attributes for each function are specified in different style elements, **GraphData1** through **GraphData4**, so that the functions are adequately identified in the legend. The preceding statements create Figure 21.81.

Figure 21.81 Scatter Plot and Fit Functions with PROC SGRENDER



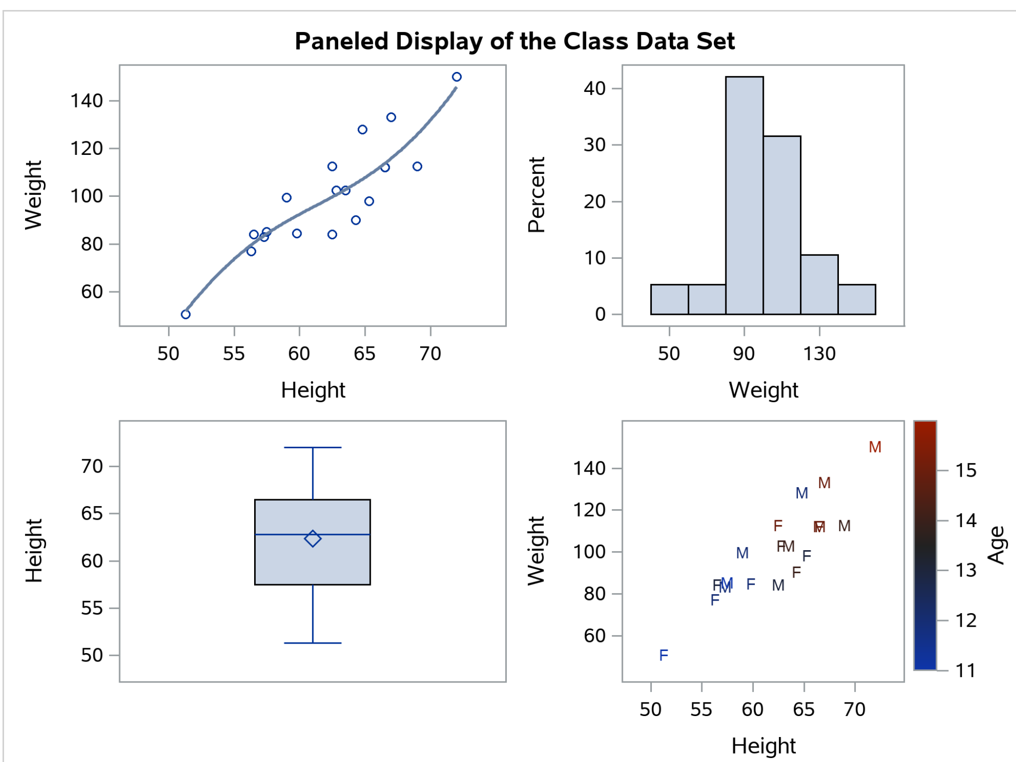
The following statements create a four-panel display of the Class data set and produce Figure 21.82:

```
proc template;
   define statgraph Panel;
      begingraph;
         entrytitle "Paneled Display of the Class Data Set";

         layout lattice / rows=2 columns=2 rowgutter=10 columngutter=10;

            layout overlay;
               scatterplot y=weight x=height;
               pbsplineplot y=weight x=height;
            endlayout;

            layout overlay / xaxisopts=(label='Weight');
               histogram weight;
            endlayout;

            layout overlay / yaxisopts=(label='Height');
```

```
            boxplot y=height;
         endlayout;

         layout overlay / xaxisopts=(offsetmin=0.1 offsetmax=0.1)
                          yaxisopts=(offsetmin=0.1 offsetmax=0.1);
            scatterplot  y=weight x=height / markercharacter=sex
               name='color' markercolorgradient=age;
            continuouslegend 'color'/ title='Age';
         endlayout;

      endlayout;
   endgraph;
end;
run;

proc sgrender data=sashelp.class template=panel;
run;
```

In this template, the outermost layout is a LAYOUT LATTICE block. It creates a 2×2 panel of plots that have a 10-pixel separation (or gutter) between pairs of plots. Inside the lattice are four LAYOUT OVERLAY blocks, each defining one of the graphs. The first is a simple scatter plot that contains a nonlinear penalized B-spline fit. The second is a histogram of the dependent variable Weight. The third is a box plot of the independent variable Height. The fourth simultaneously shows the height, weight, age, and sex of the students in the class. Each axis has an offset that is added at both the maximum and the minimum. This provides padding between the axes and the data.

Figure 21.82 Multiple Panels Using PROC SGRENDER



Many other types of graphs are available in the SG procedures. However, even the few examples that are provided here show the power and flexibility available for making professional-quality statistical graphics. For more information, see the *SAS Graph Template Language: User's Guide* and the *SAS ODS Graphics: Procedures Guide*.

Examples of ODS Statistical Graphics

Example 21.1: Creating Graphs with Tooltips in HTML

This example demonstrates how to request graphs in HTML that are enhanced with tooltip displays, which appear when you move a mouse pointer over certain features of the graph. When you specify the HTML destination and IMAGEMAP=ON in the ODS GRAPHICS statement, an image map of coordinates for tooltips is generated along with the HTML output file. Individual graphs are saved as PNG files.

Example 81.2 and Example 81.8 in Chapter 81, "The MIXED Procedure," analyze a data set that has repeated growth measurements for 27 children. The following step creates the data set:

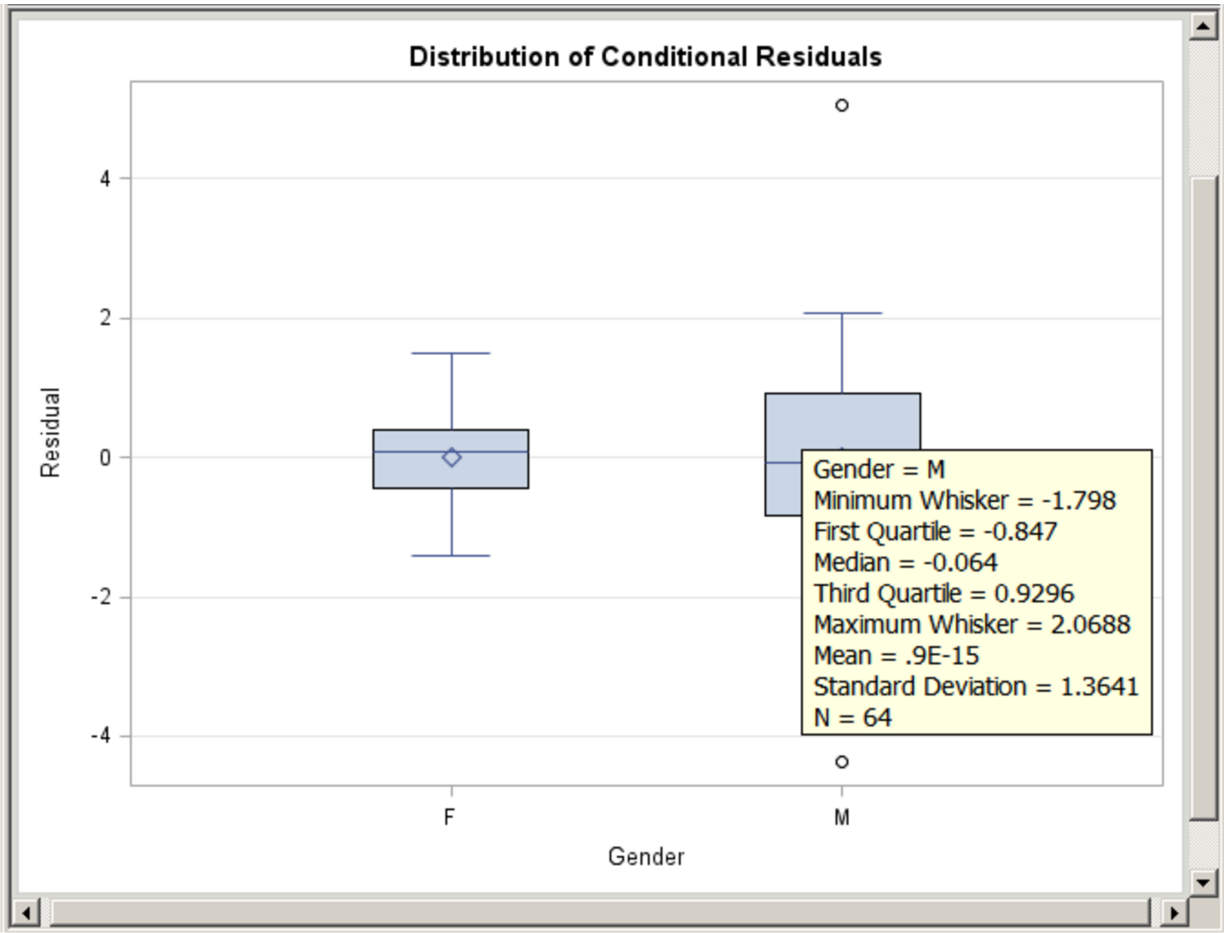
```
data pr;
   input Person Gender $ y1 y2 y3 y4 @@;
   y=y1; Age=8;  output;
   y=y2; Age=10; output;
   y=y3; Age=12; output;
   y=y4; Age=14; output;
   drop y1-y4;
   datalines;
 1  F  21.0  20.0  21.5  23.0      2  F  21.0  21.5  24.0  25.5
 3  F  20.5  24.0  24.5  26.0      4  F  23.5  24.5  25.0  26.5

   ... more lines ...

;
```

The following statements fit a mixed model that has random intercepts and slopes for each child:

```
ods _all_ close;
ods html body='b.html' style=HTMLBlue;
ods graphics on / imagemap=on;

proc mixed data=pr method=ml plots=boxplot;
   ods select 'Conditional Residuals by Gender';
   class Person Gender;
   model y = Gender Age Gender*Age;
   random intercept Age / type=un subject=Person;
run;

ods html close;
```

The PLOTS=BOXPLOT option in the PROC MIXED statement requests box plots of observed values and residuals for each classification main effect in the model (Gender and Person). Only the by-gender box plots are actually created because of the ODS SELECT statement, which uses the plot label to select the plot. Output 21.1.1 displays the results. Moving the mouse pointer over a box plot displays a tooltip that contains summary statistics for the class level. Graphics with tooltips are supported for only the HTML destination.

Output 21.1.1 Box Plot with Tooltips



Example 21.2: Creating Graphs for a Presentation

The RTF destination provides an easy way to create graphs for inclusion in a paper or presentation. You can specify the ODS RTF statement to create a file that is easily imported into a document (such as Microsoft Word or WordPerfect) or a presentation (such as Microsoft PowerPoint).

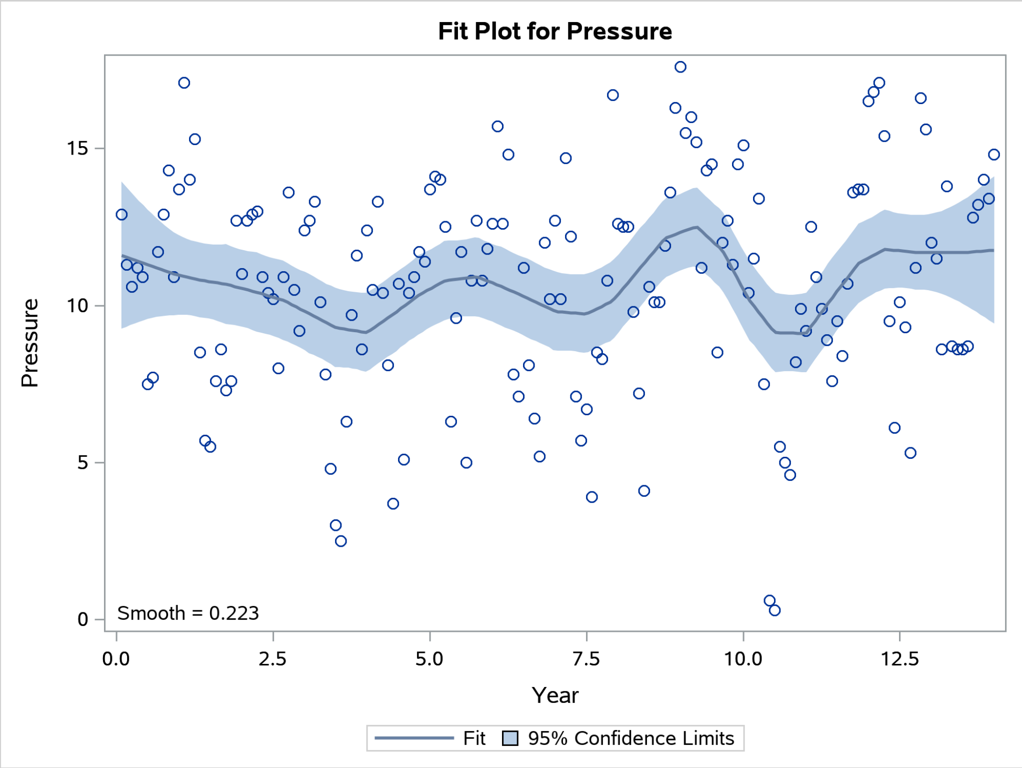
The following statements request a loess fit and save the output in the file *loess.rtf*:

```
ods _all_ close;
ods rtf file="loess.rtf" style=HTMLBlue;
ods graphics on;

proc loess data=sashelp.enso;
   model pressure = year / clm residual;
run;
```

```
ods rtf close;
ods listing;
```

The output file includes various tables and the following plots: a plot of the selection criterion versus smoothing parameter, a fit plot with 95% confidence bands, a plot of residual by regressors, and a diagnostics panel. The fit plot is produced by using the HTMLBLUE style and is shown in Output 21.2.1.

Output 21.2.1 Loess Fit Plot with the HTMLBLUE Style



If you are running SAS on the Microsoft Windows operating system, you can open the RTF file in Microsoft Word and simply copy and paste the graphs into Microsoft PowerPoint. In general, RTF output is convenient for exchange of graphical results between Windows applications through the clipboard.

Alternatively, if you use the LISTING or HTML destination, then your individual graphs are created as PNG files by default. You can insert these files in a Microsoft PowerPoint presentation. For information about how the image files are named and saved, see the sections "Naming Graphic Image Files" on page 645 and "Saving Graphic Image Files" on page 647.

Example 21.3: Creating Graphs in PostScript Files

This example illustrates how to create individual graphs in PostScript files. This is particularly useful when you want to include them in a LaTeX document.

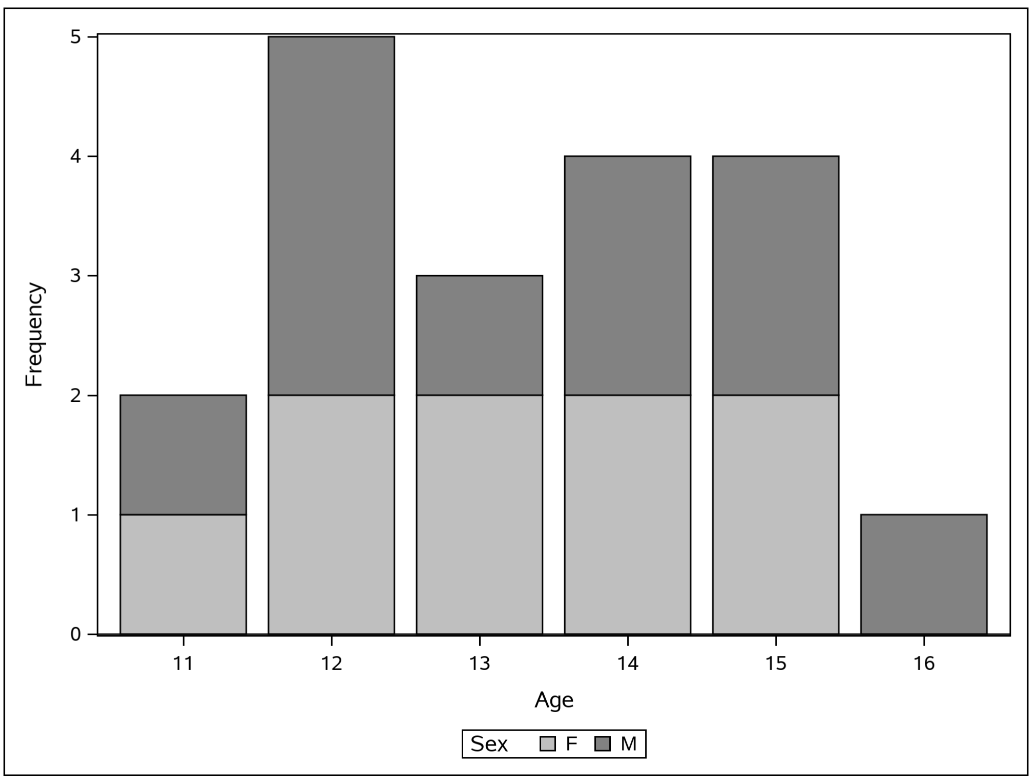
The following statements close all open destinations, open the LATEX[4] destination with the JOURNAL style, and request a grouped bar chart for the Sashelp.Class data set:

[4] The LATEX destination is experimental.

```
ods graphics on / reset=index;
ods _all_ close;
ods latex style=Journal;

proc sgplot data=sashelp.class;
   vbar age / group=sex;
run;

ods latex close;
ods listing;
```

The JOURNAL style displays grayscale graphs that are suitable for a journal. When you specify the ODS LATEX destination, ODS creates a PostScript file for each individual graph in addition to a LaTeX source file that includes the tabular output and references to the PostScript files. By default, these files are saved in the SAS current folder. The bar chart shown in Output 21.3.1 is saved by default in a file named *SGPlot.ps*. For information about how graphic image files are named, see the section "Naming Graphic Image Files" on page 645. If both the default destination (LISTING or HTML) and the LATEX destination are open, then two files are created: *SGPlot.png* and *SGPlot1.ps*. If RESET=INDEX is not specified in the ODS GRAPHICS statement and you run the step again, the next names are based on an incremented index (*SGPlot2.png* and *SGPlot3.ps*).

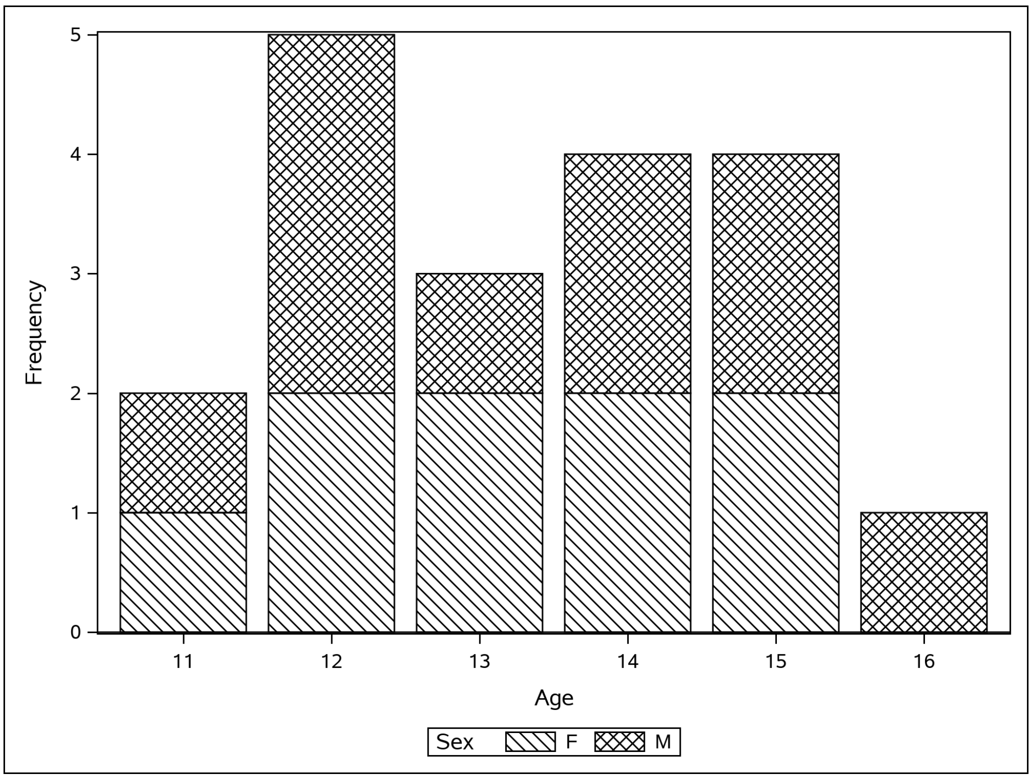
Output 21.3.1 Bar Chart Using the JOURNAL Style



You can use the JOURNAL2 style for a different appearance—the bars are not shaded. Crosshatching is used to indicate group membership. The following step produces Output 21.3.2:

```
ods graphics on / reset=index;
ods _all_ close;
ods latex style=Journal2;

proc sgplot data=sashelp.class;
   vbar age / group=sex;
run;

ods latex close;
ods listing;
```

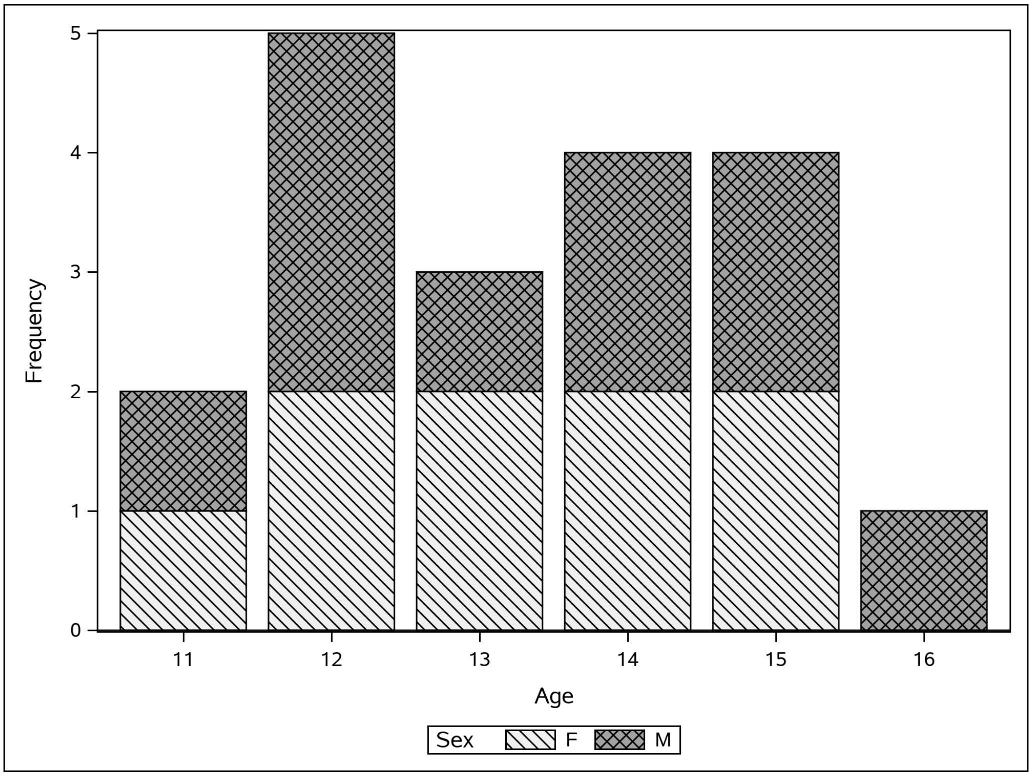
Output 21.3.2 Bar Chart Using the JOURNAL2 Style



You can use the JOURNAL3 style for a different kind of appearance from the JOURNAL style. A mix of filled areas and crosshatching is used in grouped bar charts. The following step produces Output 21.3.3:

```
ods graphics on / reset=index;
ods _all_ close;
ods latex style=Journal3;

proc sgplot data=sashelp.class;
   vbar age / group=sex;
run;

ods latex close;
ods listing;
```

Output 21.3.3 Bar Chart Using the JOURNAL3 Style



If you are writing a paper, you can include the graphs in your own LaTeX source file by referencing the names of the individual PostScript graphics files. In this situation, you might not find it necessary to use the LaTeX source file created by the SAS System. Alternatively, you can include PNG files in a LaTeX document, after using some other ODS destination (such as HTML) to create the PNG files.

Example 21.4: Displaying Graphs Using the DOCUMENT Procedure

This example illustrates the use of the ODS DOCUMENT destination and the DOCUMENT procedure to display your ODS graphs. You can use this approach whenever you want to generate and save your output (both tables and graphs) and then display it later, potentially in subsets or more than once. This approach is particularly useful when you want to display your output in multiple ODS destinations or when you want to use different styles without rerunning your SAS program. This approach is also useful when you want to break your output in separate parts for inclusion in different parts of a document such as a LaTeX file.

Consider again the data set Stack created by the following statements:

```
data stack;
   input x1 x2 x3 y @@;
   datalines;
80 27 89 42   80 27 88 37   75 25 90 37   62 24 87 28   62 22 87 18
62 23 87 18   62 24 93 19   62 24 93 20   58 23 87 15   58 18 80 14
58 18 89 14   58 17 88 13   58 18 82 11   58 19 93 12   50 18 89  8
50 18 86  7   50 19 72  8   50 19 79  8   50 20 80  9   56 20 82 15
70 20 91 15
;
```

The following statements request a Q-Q plot from PROC ROBUSTREG to analyze the Stack data:

```
ods graphics on;
ods _all_ close;
ods document name=QQDoc(write);

proc robustreg data=stack plots=qqplot;
   model y = x1 x2 x3;
quit;

ods document close;
ods listing;
```

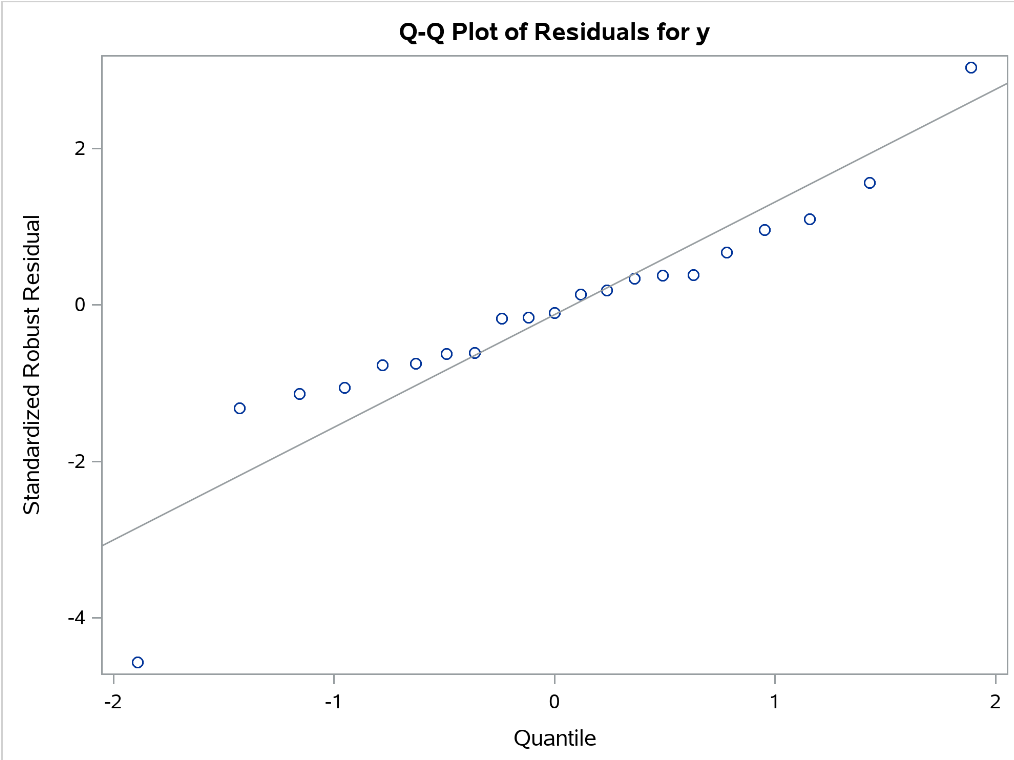
The ODS DOCUMENT statement opens an ODS document named **QQDoc**. All the results—tables, graphs, titles, notes, footnotes, headings—are stored in the ODS document. None of them are displayed because no other destination is open. To display the Q-Q plot by using PROC DOCUMENT, you first need to determine its name. You can do this by specifying the ODS TRACE ON statement before the procedure statements (for more information, see the section "Determining Graph Names and Labels" on page 640). Alternatively, you can enter **odsdocuments** (or **odsd** for short) on the command line to open the Documents window, which you can then use to manage your ODS documents.

The following statements specify an HTML destination and display the residual Q-Q plot by using the REPLAY statement in PROC DOCUMENT:

```
ods html body='b.htm';

proc document name=QQDoc;
   ods select QQPlot;
   replay;
quit;

ods html close;
```

Subsequent steps can replay one or more objects from the same ODS document. By default, the REPLAY statement attempts to display every output object stored in the ODS document, but here only the Q-Q plot is displayed because it is specified by the ODS SELECT statement. The plot is displayed in Output 21.4.1.

Output 21.4.1 Q-Q Plot Displayed by PROC DOCUMENT



As an alternative to running PROC DOCUMENT along with an ODS SELECT statement, you can run PROC DOCUMENT and use a *document path* for the Q-Q plot in the REPLAY statement. This approach is preferable when the ODS document contains a large volume of output, so that PROC DOCUMENT does not attempt to process every piece of output that is stored in the ODS document.

You can determine the ODS document path for the Q-Q plot by specifying the LIST statement and LEVELS=ALL in PROC DOCUMENT as follows:

```
proc document name=QQDoc;
   list / levels=all;
quit;
```

The contents of the ODS document **QQDoc** are shown in Output 21.4.2.

Output 21.4.2 Contents of the ODS Document `QQDoc`

Listing of: \Work.Qqdoc
Order by: Insertion
Number of levels: All

Obs	Path	Type
1	\Robustreg#1	Dir
2	\Robustreg#1\ModelInfo#1	Table
3	\Robustreg#1\NObs#1	Table
4	\Robustreg#1\ParmInfo#1	Table
5	\Robustreg#1\SummaryStatistics#1	Table
6	\Robustreg#1\ParameterEstimates#1	Table
7	\Robustreg#1\DiagSummary#1	Table
8	\Robustreg#1\DiagnosticPlots#1	Dir
9	\Robustreg#1\DiagnosticPlots#1\QQPlot#1	Graph
10	\Robustreg#1\GoodFit#1	Table

The ODS document path of the `QQPlot` entry in the `QQDoc` ODS document, as shown in Output 21.4.2, is `\Robustreg#1\DiagnosticPlots#1\QQPlot#1`.

You can use this path to display the residual Q-Q plot by using PROC DOCUMENT as follows:

```
proc document name=QQDoc;
   replay \Robustreg#1\DiagnosticPlots#1\QQPlot#1;
quit;
```

You can also determine the ODS document path from the Results window or Documents window. Right-click the object icon and select **Properties**.

The SAS/STAT documentation preparation process uses the ODS document. SAS output is saved in an ODS document that is then used in sections of the documentation, which is prepared using LaTeX. In general, when you send your output to the DOCUMENT destination, you can use PROC DOCUMENT to rearrange, duplicate, or remove output from the results of a procedure or a database query. For more information, see the ODS DOCUMENT statement in the section "Dictionary of ODS Language Statements" and the chapter "The DOCUMENT Procedure" in the *SAS Output Delivery System: User's Guide*.

Example 21.5: Customizing the Style for Box Plots

This example demonstrates how to modify the style for box plots. This example is taken from Example 21.1. The following step creates the data set:

```
data pr;
   input Person Gender $ y1 y2 y3 y4 @@;
   y=y1; Age=8;  output;
   y=y2; Age=10; output;
   y=y3; Age=12; output;
   y=y4; Age=14; output;
   drop y1-y4;
   datalines;
 1  F  21.0  20.0  21.5  23.0       2  F  21.0  21.5  24.0  25.5
 3  F  20.5  24.0  24.5  26.0       4  F  23.5  24.5  25.0  26.5

   ... more lines ...

;
```

The following step displays the HTMLBLUE style and its parent styles, STATISTICAL and DEFAULT:

```
proc template;
   source Styles.HTMLBlue / expand;
run;
```

If you search for “box”, you find the style element that controls some aspects of the box plot:

```
class GraphBox /
   capstyle = "serif"
   connect = "mean"
   displayopts = "fill caps median mean outliers";
```

You can learn more about the **GraphBox** style element and its attributes in the section on the BOXPLOT statement in the *SAS Graph Template Language: Reference* and in the section “ODS Style Elements” in the *SAS Output Delivery System: User’s Guide*.

The following statements create two new styles by modifying attributes of the **GraphBox** style element. The first style is a sparse style; the box is outlined (not filled), and the median is shown, but not the mean. In contrast, the second style produces a filled box, with caps on the whiskers that shows the mean, median, and outliers. In addition, the box is notched.

The following statements create the two styles:

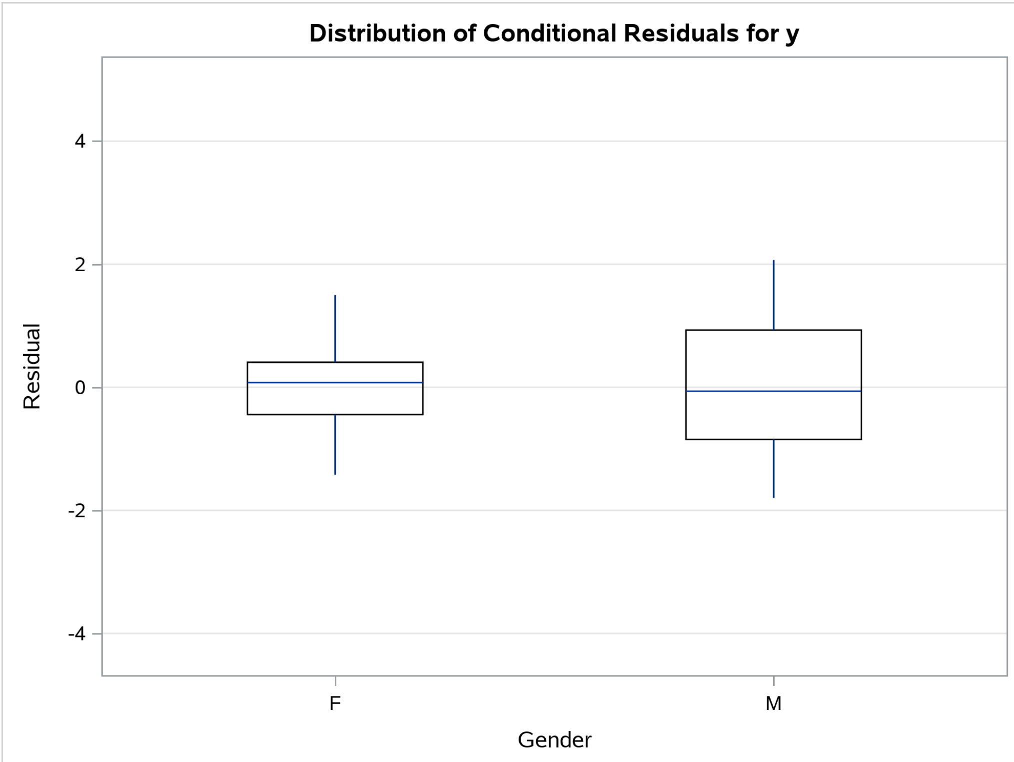
```
proc template;
   define style BoxStyleSparse;
      parent=styles.HTMLBlue;
      style GraphBox / capstyle = "line" displayopts = "median";
   end;
   define style BoxStyleRich;
      parent=styles.HTMLBlue;
      style GraphBox / capstyle = "bracket"
            displayopts = "fill caps median mean outliers notches";
   end;
run;
```

The following steps run PROC MIXED and create box plots that use the two styles:

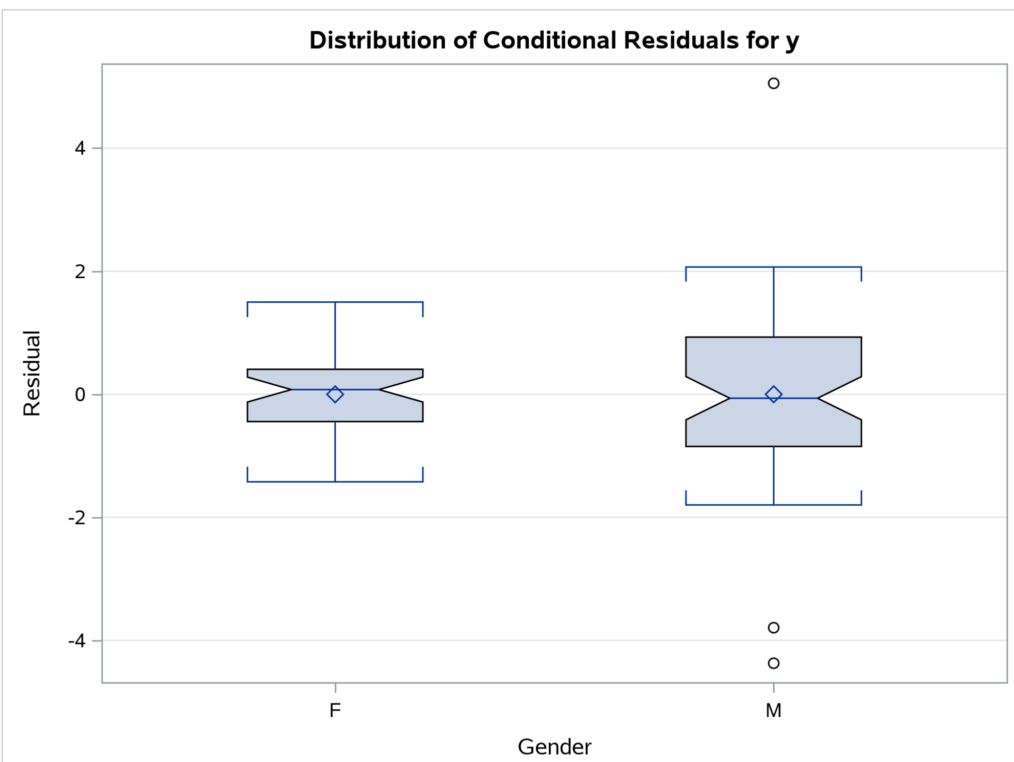
```
ods graphics on;
ods html style=boxstylesparse;

proc mixed data=pr method=ml plots=boxplot;
   ods select 'Conditional Residuals by Gender';
   class Person Gender;
   model y = Gender Age Gender*Age;
   random intercept Age / type=un subject=Person;
run;
ods html close;

ods html style=boxstylerich;

proc mixed data=pr method=ml plots=boxplot;
   ods select 'Conditional Residuals by Gender';
   class Person Gender;
   model y = Gender Age Gender*Age;
   random intercept Age / type=un subject=Person;
run;
ods html close;
```

The results from using the sparse style are displayed in Output 21.5.1, and the results from using the richer style are displayed in Output 21.5.2. See Output 21.1.1 in Example 21.1 for the results of using the HTMLBLUE style.

Output 21.5.1 Box Plot with the Sparse Style



Output 21.5.2 Box Plot with the Richer Style



Example 21.6: Splines

This example illustrates several types of spline fit functions that are available in ODS Graphics and SAS/STAT. It begins by showing you how to use PROC SGPLOT to fit a single function through a scatter plot. Later examples use PROC SGPLOT to fit multiple functions by specifying a group variable. Subsequent sections show how to fit spline functions in SAS/STAT procedures, concentrating on PROC TRANSREG and the spline functionality in the EFFECT and EFFECTPLOT statements. PROC ADAPTIVEREG is also discussed. Topics include polynomials, polynomial splines (Smith 1979), B-splines (De Boor 1978), penalized B-splines (Eilers and Marx 1996), monotone splines (Winsberg and Ramsay 1980), natural cubic splines and restricted splines (Hastie, Tibshirani, and Friedman 2009), smoothing splines (Reinsch 1967), multivariate adaptive regression splines (Friedman 1991), and loess (Cleveland, Devlin, and Grosse 1988).

Single Fit Function Using PROC SGPLOT

Polynomial Fit Function

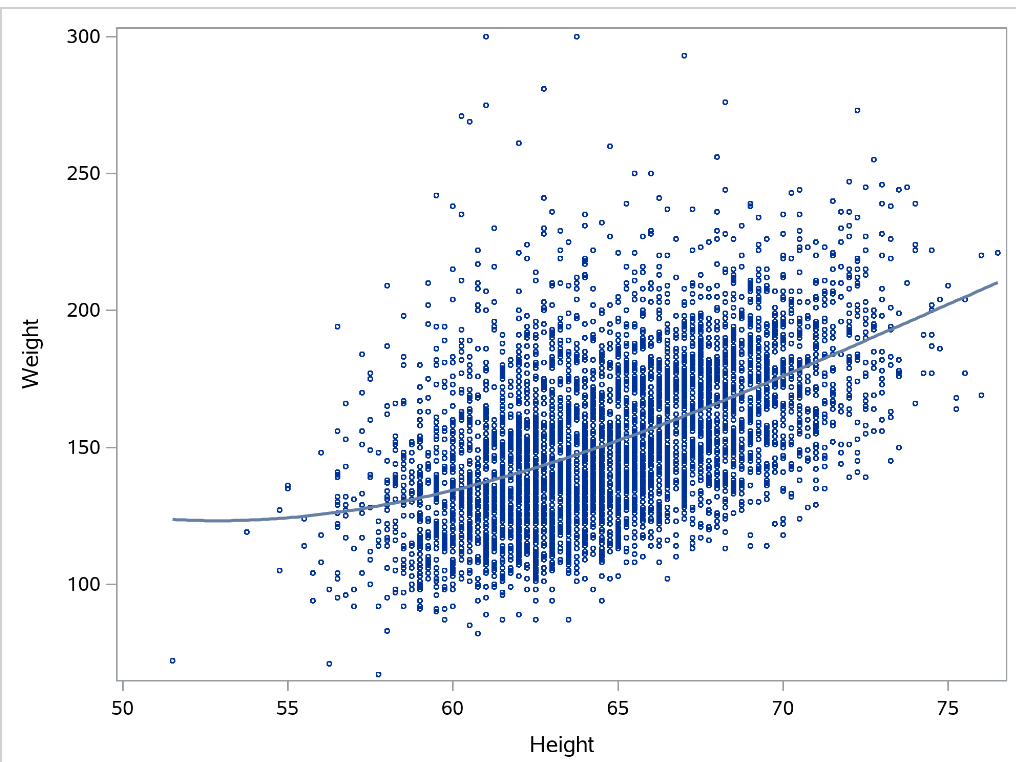
You can use PROC SGPLOT and the REG statement along with the DEGREE=3 option to fit a cubic polynomial function to your data. A cubic polynomial is smooth and has little freedom to follow nonlinear trends in the data. The model is

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$$

The following step creates the plot in Output 21.6.1:

```
proc sgplot data=sashelp.heart noautolegend;
   reg y=weight x=height / markerattrs=(size=3px) degree=3;
run;
```

Output 21.6.1 Cubic Polynomial Fit Plot

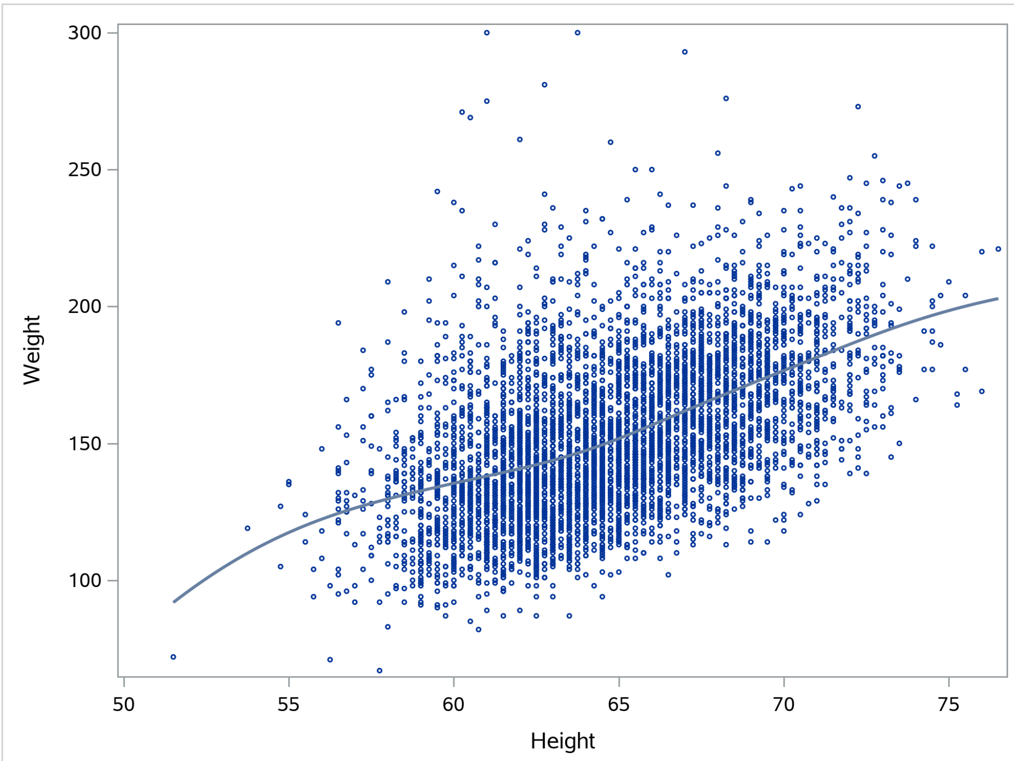


Penalized B-Spline Fit Function

You can use the PBSPLINE statement to fit penalized B-splines (Eilers and Marx 1996). Penalized B-splines draw a smooth curve through a scatter plot by using an automatic selection of the smoothing parameter. The following step creates the plot in Output 21.6.2:

```
ods graphics on / antialiasmax=6000;
proc sgplot data=sashelp.heart noautolegend;
   pbspline y=weight x=height / markerattrs=(size=3px);
run;
```

Output 21.6.2 Penalized B-Spline Fit Plot



The resulting fit function is smooth and nonlinear. You do not need to know anything about the shape of the scatter plot. The PBSPLINE statement automatically finds a smooth fit while trying to guard against overfitting. It is not possible to write a simple equation for the model. For more information about how penalized B-splines work, see Chapter 123, "The TRANSREG Procedure."

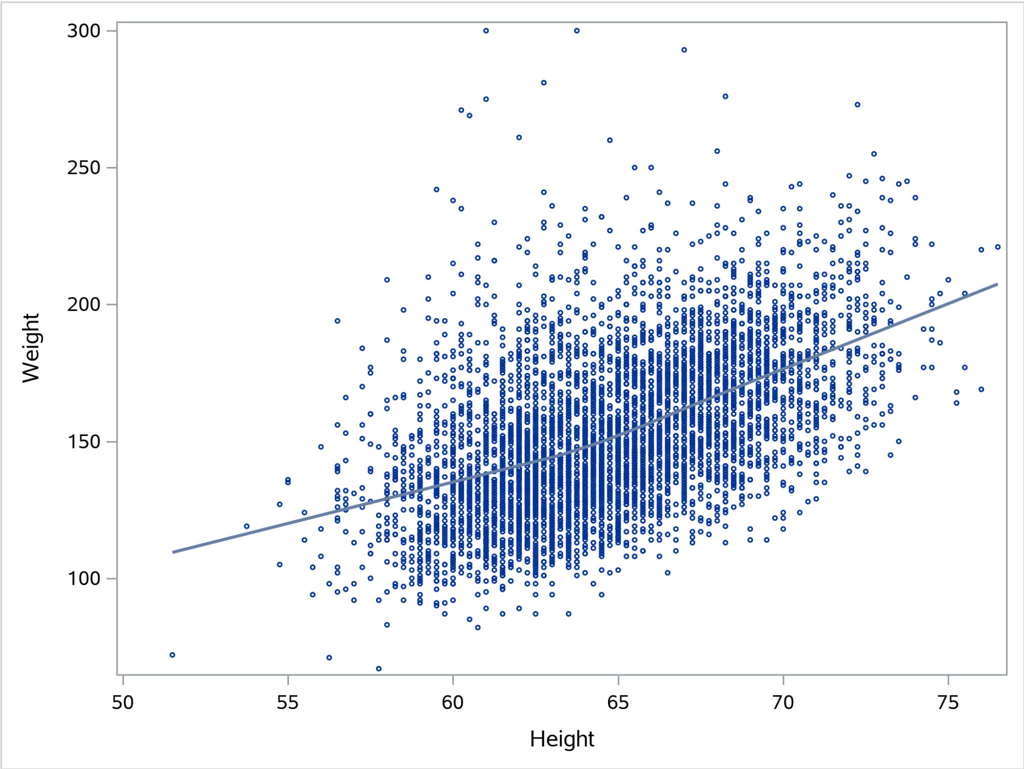
NOTE: Antialiasing smooths the elements of a graph. The ANTIALIASMAX=6000 option enables antialiasing through 6,000 elements. By default, antialiasing is disabled after 4,000 elements.

Loess Fit Function

You can use the LOESS statement to find a loess fit function (Cleveland, Devlin, and Grosse 1988). Loess is a locally weighted scatter plot smoothing. The following step creates the plot in Output 21.6.3:

```
ods graphics on / loessmaxobs=6000;
proc sgplot data=sashelp.heart noautolegend;
   loess y=weight x=height / markerattrs=(size=3px);
run;
```

Output 21.6.3 Loess Fit Plot



The loess fit is not a spline fit, but loess is similar to penalized B-splines in that it automatically tries to find a smooth fit while trying to guard against overfitting. It is not possible to write a simple equation for the model. For more information about loess, see Chapter 75, "The LOESS Procedure."

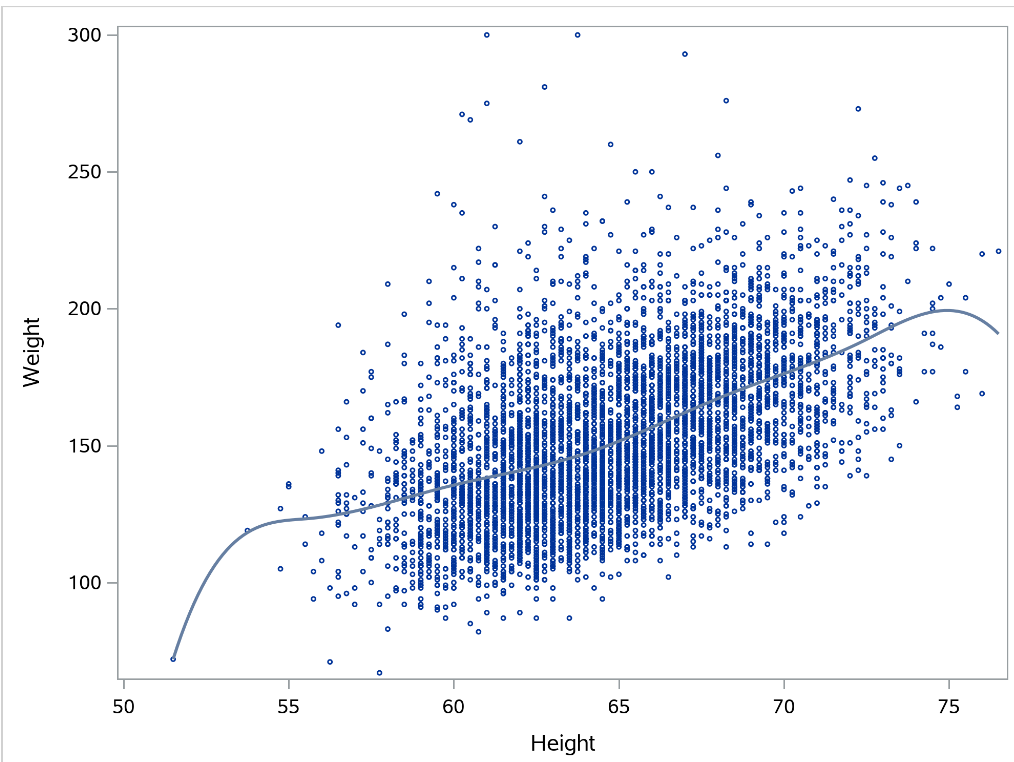
NOTE: Loess becomes computationally expensive with larger data sets. The LOESSOBSMAX=6000 option enables loess fits through 6,000 observations. By default, loess fits are disabled after 5,000 observations.

B-Spline Fit Function

You can use the PBSPLINE statement along with the option SMOOTH=0 to fit B-splines (De Boor 1978), which are equivalent to piecewise-polynomial splines. You specify SMOOTH=0 to disable all automatic smoothing. You specify the number of knots in the NKNOTS= option.[5] You use fewer knots to create smoother plots and more knots to enable greater curvature. The following step creates the plot in Output 21.6.4:

```
proc sgplot data=sashelp.heart noautolegend;
   pbspline y=weight x=height / smooth=0 nknots=5 markerattrs=(size=3px);
run;
```

Output 21.6.4 B-Spline Fit Plot



The resulting fit function is equivalent to those that you can obtain by using SPLINE (spline transformation), PSPLINE (polynomial-spline basis), or BSPLINE (B-spline basis) in the MODEL statement in PROC TRANSREG. Of all the functions shown in Output 21.6.2 through Output 21.6.4, the B-spline fit in Output 21.6.4 is most influenced by the extreme X values. The polynomial-spline model is

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \beta_4 (x - k_1)_+^3 + \beta_5 (x - k_2)_+^3 + \beta_6 (x - k_3)_+^3 + \beta_8 (x - k_4)_+^3 + \beta_9 (x - k_5)_+^3 + \epsilon$$

The values k_1 through k_5 are the knots, which fall in the range of x. When $(x - k_i)$ is negative, $(x - k_i)_+ = 0$; otherwise, $(x - k_i)_+ = (x - k_i)$. The expression $(x - k_i)_+^p$ is called a truncated power function. Each knot term changes the cubic part of the polynomial as x advances beyond each knot.

For an introduction to piecewise-polynomial splines, see Smith (1979). For more information about how splines work, see Chapter 123, "The TRANSREG Procedure."

[5] "Knots," without qualification, refer to interior knots—points inside the range of the x variable. Exterior knots, which are outside the range of the data, are explained in the section "Interior and Exterior Knots" on page 764.

Understanding Splines and Knots

This section is optional; it shows some of the mathematical details of polynomial-spline models. As shown previously, the following is a cubic-polynomial regression model:

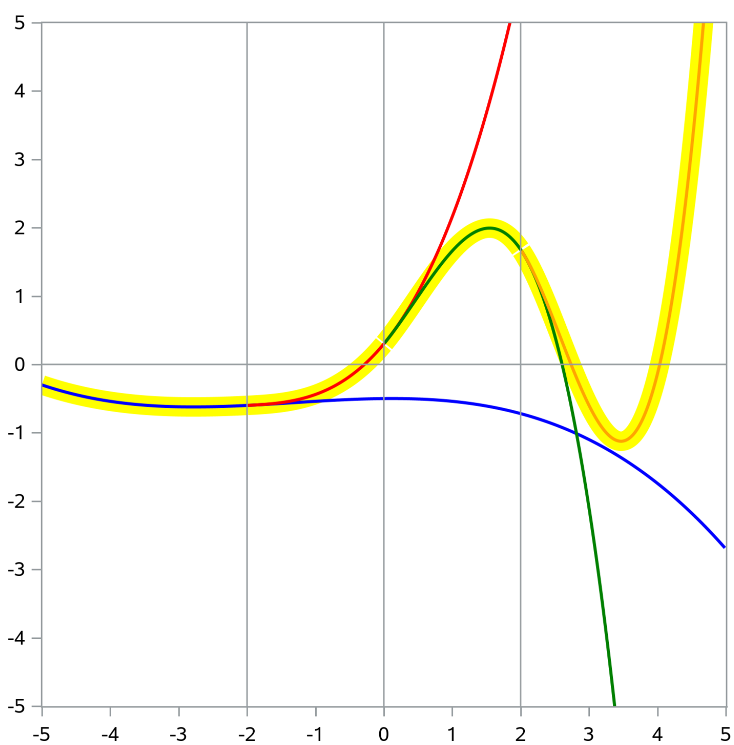
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$$

If you add n_k knots, it becomes a polynomial-spline regression model:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{i=1}^{n_k} \beta_{3+i}(x - k_i)_+^3 + \epsilon$$

Polynomial splines are easy to understand and describe. A curve has an overall intercept, linear portion, quadratic portion, and cubic portion. Then the cubic portion changes at each knot. Output 21.6.5 illustrates a spline that has knots at –2, 0, and 2. The blue function, $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$, extends from –5 to 5. The blue and red function, $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4(x - -2)_+^3$, extends from –5 to almost 2 as it heads toward $y = \infty$. The red component first contributes to the overall function when $x - -2$ is positive. The blue, red, and green function, $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4(x - -2)_+^3 + \beta_5(x - 0)_+^3$, extends from –5 to just beyond 3 as it heads toward $y = -\infty$. The green component first contributes to the overall function when $x - 0$ is positive. The blue, red, green, and orange function, $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4(x - -2)_+^3 + \beta_5(x - 0)_+^3 + \beta_6(x - 2)_+^3$, extends from –5 to almost 5 as it heads toward $y = \infty$. The orange component first contributes to the overall function when $x - 2$ is positive. Thus, $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4(x - -2)_+^3 + \beta_5(x - 0)_+^3 + \beta_6(x - 2)_+^3$, which is highlighted in yellow, is the spline function, and it is composed of four component functions. The coefficients β_4, β_5, and β_6 are the change in the cubic portion of the spline. The intercept does not change; this makes the spline continuous. The linear and quadratic terms do not change; this makes the spline smooth.

Output 21.6.5 Polynomial-Spline Components



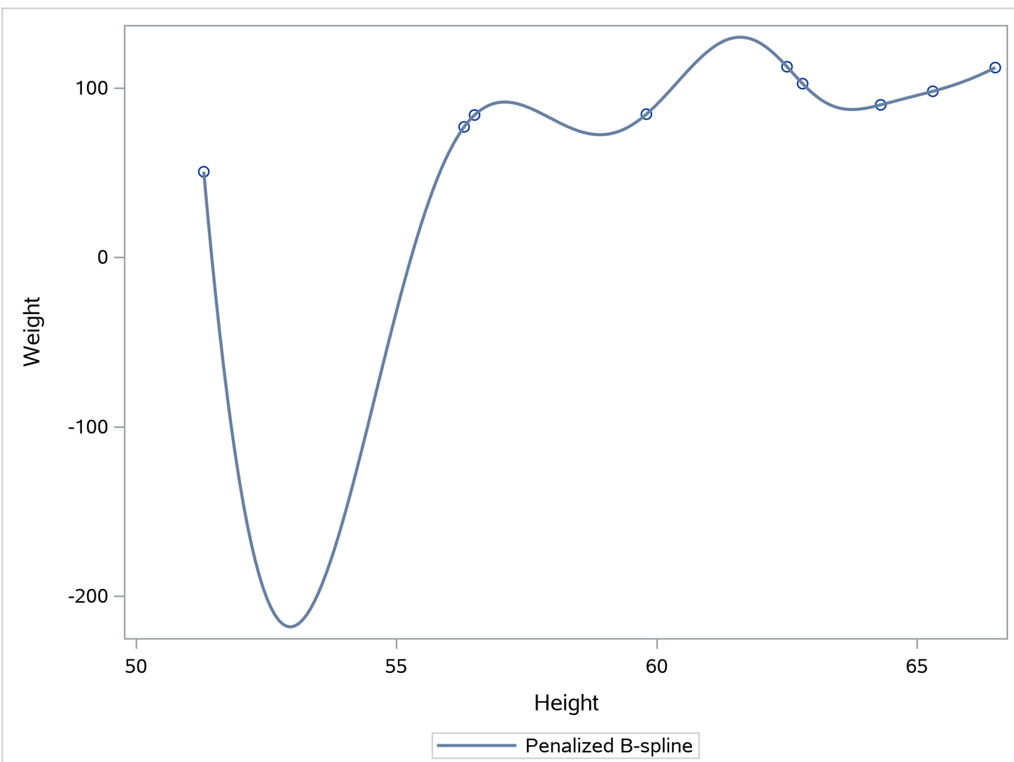
Mathematically, the cubic-polynomial spline is continuous, as are its first and second derivatives. Computationally, cubic-polynomial splines might be problematic, particularly for large data sets or when there are many knots. This is because some terms might be highly correlated, resulting in an unstable model. In practice, B-splines are preferred over cubic-polynomial splines, although the two types of splines are equivalent. If $\mathbf{X}_p$ is a full-rank polynomial-spline basis and $\mathbf{X}_B$ is the corresponding full-rank B-spline basis, then there exists a matrix $\mathbf{T}$ such that $\mathbf{X}_B = \mathbf{X}_p\mathbf{T}$ and $\mathbf{X}_B\mathbf{T}^{-1} = \mathbf{X}_p$. For an illustration, see the section "B-Spline Basis" on page 766. The overall fit and R-square are the same, but because the basis columns of the $\mathbf{X}$ matrices are different, the regression coefficients are different. Regression coefficients for B-spline models are usually not interpretable.

Sparse Data and Overfitting

Penalized B-splines (and many other methods) do not work well with sparse data. For example, data sets might be small and sparse when you are following the results of a medical test for a single subject over time. You might want to fit a smooth function through the results rather than simply connect the dots. The following step creates the plot in Output 21.6.6:

```
proc sgplot data=sashelp.class(where=(sex='F'));
   pbspline y=weight x=height;
run;
```

Output 21.6.6 Overfit Sparse Data



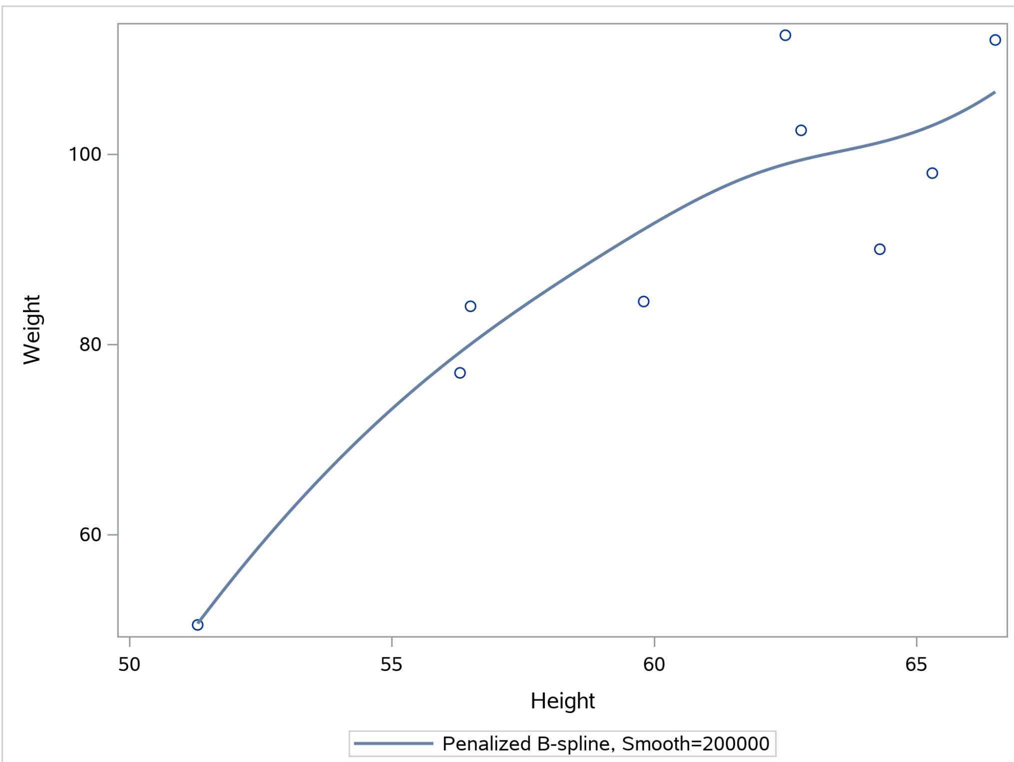
The X and Y coordinates of the fit function consist of 201 equally spaced X values that range from the minimum of X to its maximum. The Y values are computed from those X values and the coefficients. Penalized B-splines (and other spline methods) do not use the interpolation data when they compute the smoothing parameter and coefficients. First they analyze the actual data values, and then they compute the interpolated values.

The data are so sparse and the model has so many parameters that the computations fail to produce a reasonable nonlinear fit function. You can see that the interpolated Y values extend well beyond the range of the data. You see this again in the section “Multiple Fit Functions Using PROC SGPLOT” on page 743, where you also see other remedies, including restricted splines. For information about interpolation, also see the section “Interpolation” on page 747.

A perfect connect-the-dots fit instead of the fit plot in Output 21.6.6 would look reasonable. In contrast, displaying the interpolated values more fully shows the spline function and shows that the results are overfit and not generalizable. You can manually set the smoothing parameter to get a better fit function. Larger values create a smoother fit function. The following step creates the plot in Output 21.6.7:

```
proc sgplot data=sashelp.class(where=(sex='F'));
   pbspline y=weight x=height / smooth=2e5;
run;
```

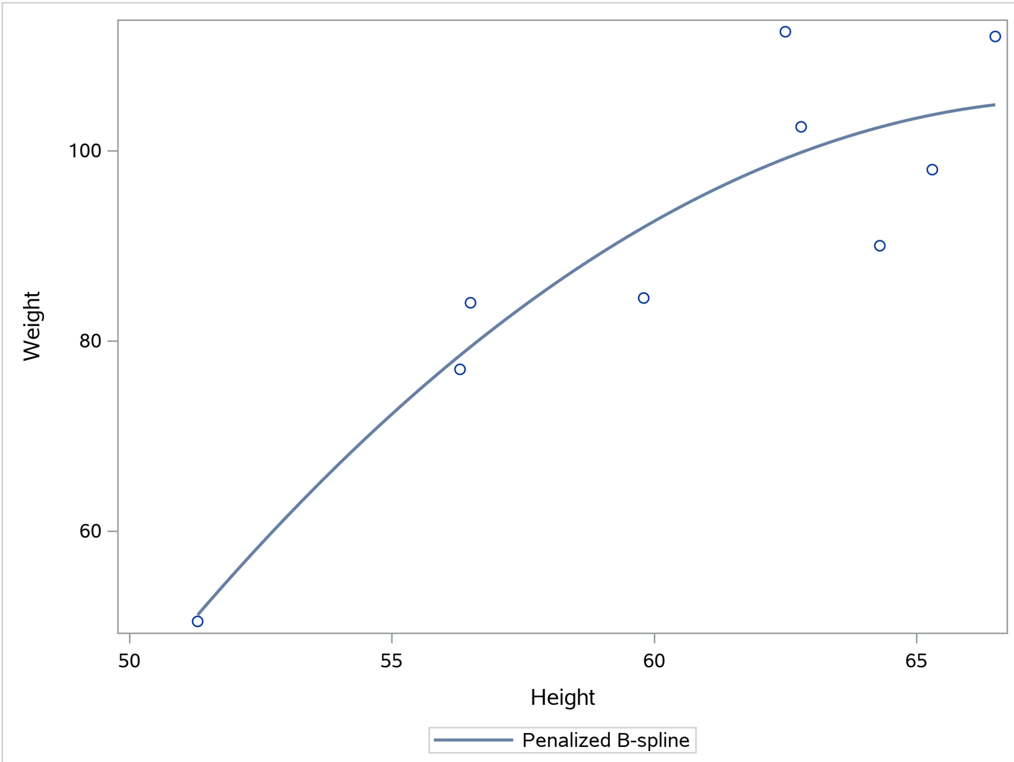
Output 21.6.7 Manually Set Smoothing Parameter



Usually, it takes some trial and error to set the smoothing parameter. You can use the penalized B-spline functionality in PROC TRANSREG to see which smoothing parameters it considers. PROC SGPLOT and PROC TRANSREG use the same code. Alternatively, you can specify fewer knots. By default, there are 100 knots. The following step specifies NKNOTS=20 and creates the plot in Output 21.6.8:

```
proc sgplot data=sashelp.class(where=(sex='F'));
   pbspline y=weight x=height / nknots=20;
run;
```

Output 21.6.8 Twenty Knots



Multiple Fit Functions Using PROC SGPLOT

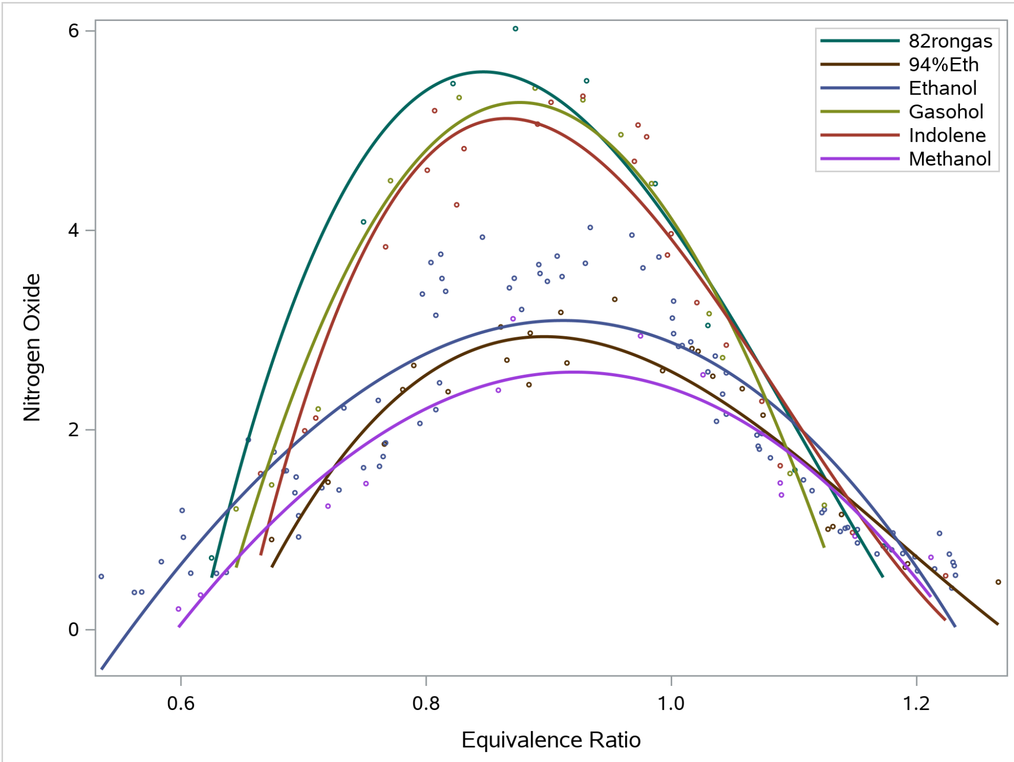
The next set of steps shows the same methods that were shown in the section "Single Fit Function Using PROC SGPLOT" on page 736 but this time using different data and a group variable.

Polynomial Fit Functions

You can use the REG statement with the DEGREE= and GROUP= options to fit multiple polynomial functions. The following step creates the plot in Output 21.6.9:

```
proc sgplot data=sashelp.gas;
   reg y=nox x=eqratio / degree=3 group=fuel markerattrs=(size=3px) name='a';
   keylegend 'a' / location=inside position=topright across=1;
run;
```

Output 21.6.9 Grouped Cubic Polynomial Fit Plot



The plot in Output 21.6.9 contains six cubic polynomial functions, each independent of the others. Each function has the form

$$y_i = \beta_{i0} + \beta_{i1}x + \beta_{i2}x^2 + \beta_{i3}x^3 + \epsilon$$

If c_i is a binary indicator variable (i = 1 to 6) such that $\sum_{i=1}^{6} c_i = 1$, c_1 is 1 for `'82rongas'` and 0 for all other fuels, c_2 is 1 for `'94%Eth'` and 0 for all other fuels, and so on, then

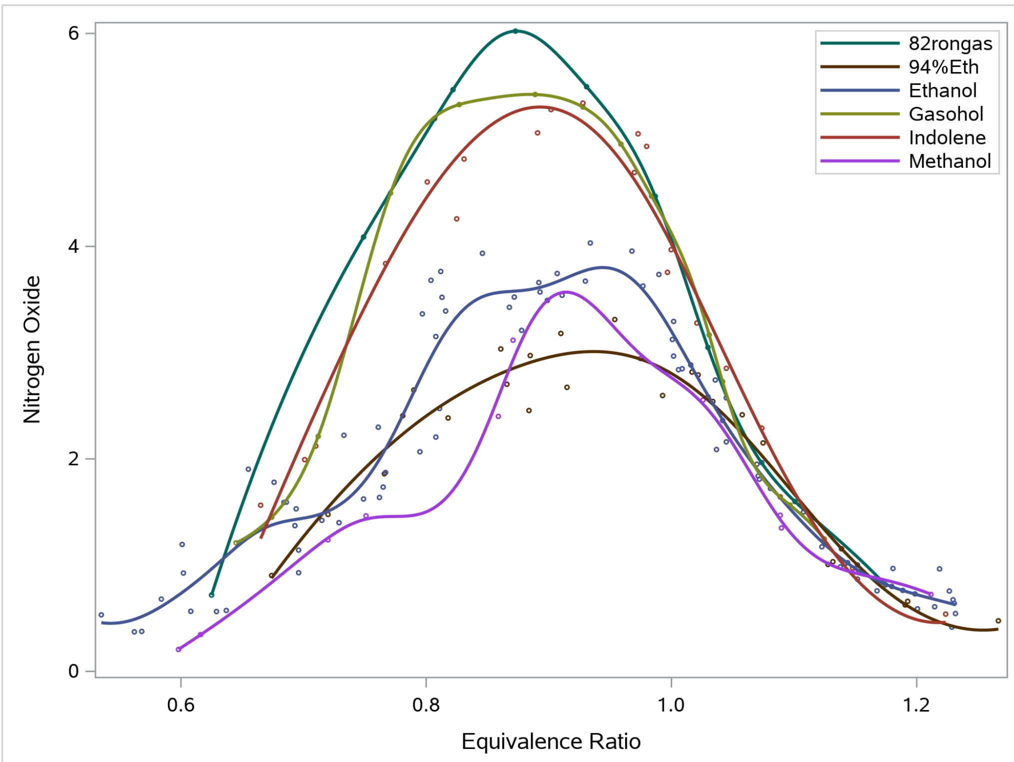
$$y = \sum_{i=1}^{6} c_i \left[\beta_{i0} + \beta_{i1}x + \beta_{i2}x^2 + \beta_{i3}x^3\right] + \epsilon$$

Penalized B-Spline Fit Functions

You can use the PBSPLINE statement with the GROUP= option to fit multiple penalized B-splines. The following step creates the plot in Output 21.6.10:

```
proc sgplot data=sashelp.gas;
   pbspline y=nox x=eqratio / group=fuel markerattrs=(size=3px) name='a';
   keylegend 'a' / location=inside position=topright across=1;
run;
```

Output 21.6.10 Grouped Penalized B-Spline Fit Plot



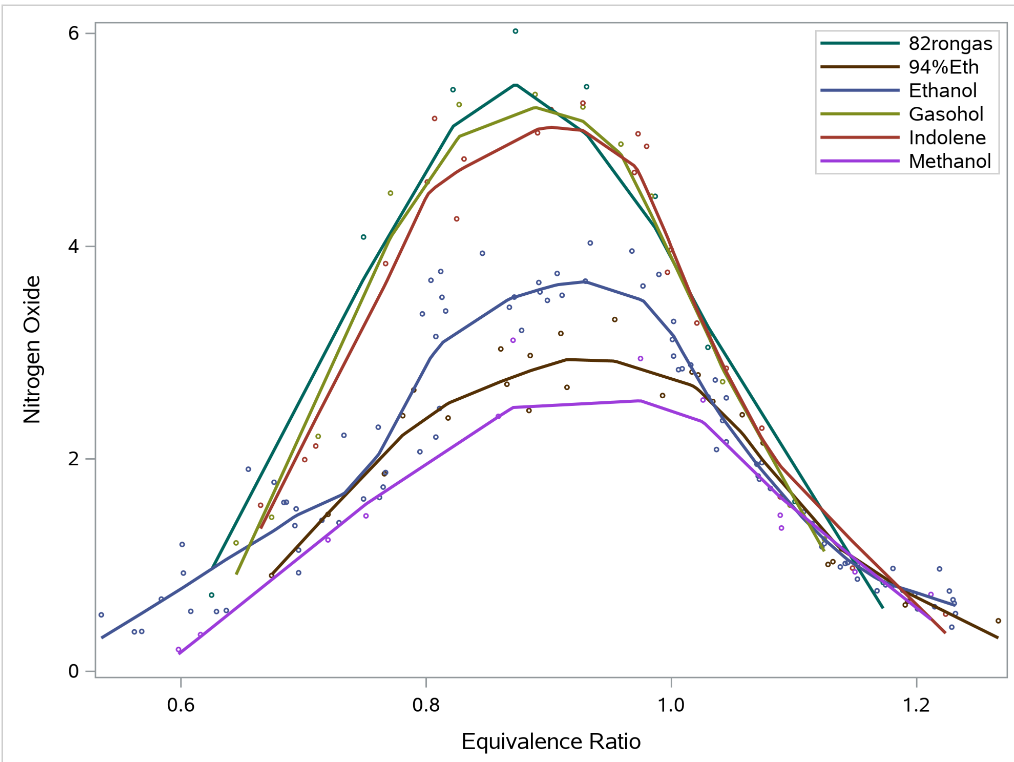
The penalized B-spline fit functions are less smooth than the polynomials.

Loess Fit Functions

You can use the LOESS statement with the GROUP= option to fit multiple loess curves. The following step creates the plot in Output 21.6.11:

```
proc sgplot data=sashelp.gas;
   loess y=nox x=eqratio / group=fuel markerattrs=(size=3px) name='a';
   keylegend 'a' / location=inside position=topright across=1;
run;
```

Output 21.6.11 Grouped Loess Fit Plot



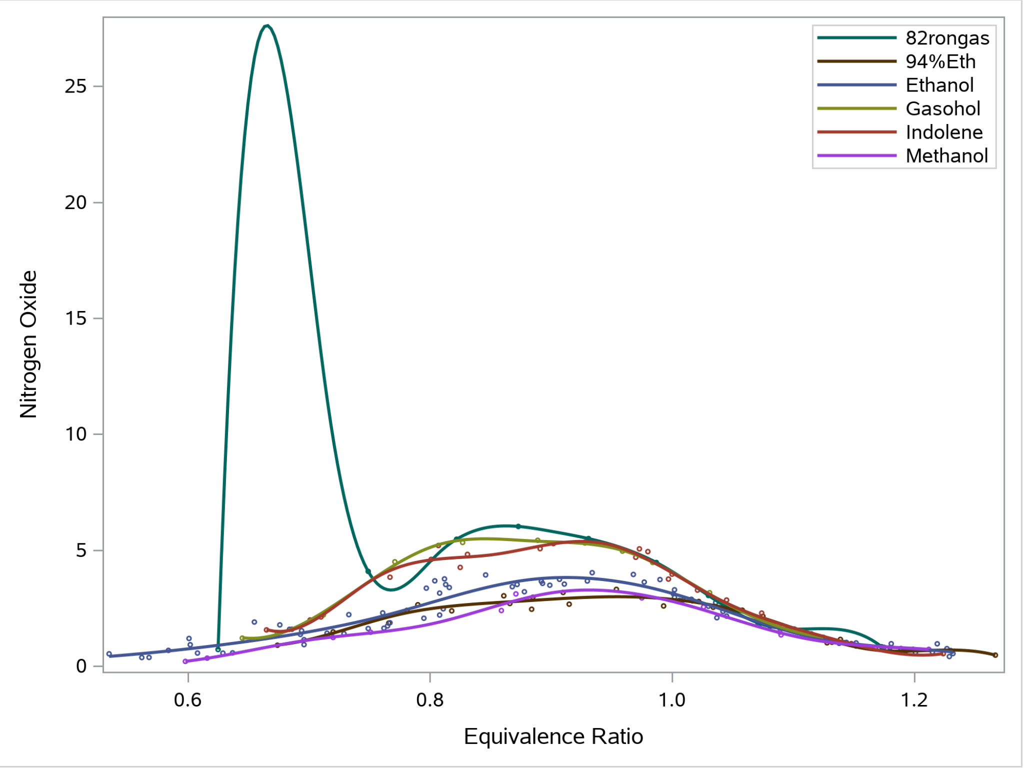
The loess spline fit functions are less smooth than the polynomials.

B-Spline Fit Functions

You can use the PBSPLINE statement, the SMOOTH=0 and NKNOTS= options, and the GROUP= option to fit multiple B-spline functions. The following step creates the plot in Output 21.6.12:

```
proc sgplot data=sashelp.gas;
   pbspline y=nox x=eqratio / group=fuel smooth=0 nknots=5
                             markerattrs=(size=3px) name='a';
   keylegend 'a' / location=inside position=topright across=1;
run;
```

Output 21.6.12 Grouped B-Spline Fit Plot



The polynomial-spline function is unsatisfactory for the `'82rongas'` function. This is because there is a large gap between the first two points. The function connects the dots between the first several points, and it is free to deviate from the data between points. This is a frequent concern with splines. Having fixed knots that apply to all functions, no matter how sparse they are, is problematic. In contrast, penalized B-splines and loess automatically smooth that section of that function. The polynomial functions do not have enough parameters to enable that level of nonlinearity.

Interpolation

You can examine the underlying ODS data object to better understand how PROC SGPLOT constructs the fit functions. The following step creates the plot, this time creating an ODS data set:

```
proc sgplot data=sashelp.gas;
   ods output sgplot=sg;
   pbspline y=nox x=eqratio / group=fuel smooth=0 nknots=5
                              markerattrs=(size=3px) name='a';
   keylegend 'a' / location=inside position=topright across=1;
run;
```

The next steps create and display a subset of the data:

```
data subset(drop=SORT_FUEL_RETAIN_ALL_);
   set sg;
   Obs = _n_;
   by PBSPLINE_EQRATIO_NOX_GROUP_S__GP fuel;
   if _N_ gt 169 then do; fuel = '_'; eqratio = ._; nox = ._; end;
```

```
      if first.fuel or last.fuel or first.PBSPLINE_EQRATIO_NOX_GROUP_S__GP or
         last.PBSPLINE_EQRATIO_NOX_GROUP_S__GP or obs = 169 then output;
      if lag(first.fuel) or lag(first.PBSPLINE_EQRATIO_NOX_GROUP_S__GP) then do;
         call missing(of PBSPLI: Fuel EqRatio NOx obs);
         if _n_ gt 169 then do; fuel = '_'; eqratio = ._; nox = ._; end;
         output; output; output;
      end;
   run;

   proc print noobs; id obs; run;
```

The results are shown in Output 21.6.13.

Output 21.6.13 Grouped Fit Function Data Object

Obs	PBSPLINE_ EQRATIO_NOX_ GROUP_S___X	PBSPLINE_ EQRATIO_NOX_ GROUP_S___Y	PBSPLINE_ EQRATIO_NOX_ GROUP_S__GP	Fuel	Eq Ratio	NOx
1	0.62500	0.7160	82rongas	82rongas	0.749	4.084
.	.	.			.	.
.	.	.			.	.
.	.	.			.	.
9	0.64692	22.8547	82rongas	82rongas	1.173	0.835
10	0.64966	24.2451	82rongas	94%Eth	0.993	2.593
.	.	.			.	.
.	.	.			.	.
.	.	.			.	.
34	0.71542	11.3943	82rongas	94%Eth	0.674	0.900
35	0.71816	10.4285	82rongas	Ethanol	1.152	0.866
.	.	.			.	.
.	.	.			.	.
.	.	.			.	.
124	0.96202	5.0358	82rongas	Ethanol	0.693	1.369
125	0.96476	4.9835	82rongas	Gasohol	0.645	1.207
.	.	.			.	.
.	.	.			.	.
.	.	.			.	.
137	0.99764	4.1538	82rongas	Gasohol	0.712	2.209
138	1.00038	4.0667	82rongas	Indolene	1.224	0.537
.	.	.			.	.
.	.	.			.	.
.	.	.			.	.
159	1.05792	2.1655	82rongas	Indolene	1.089	1.640
160	1.06066	2.0945	82rongas	Methanol	0.598	0.204
.	.	.			.	.
.	.	.			.	.
.	.	.			.	.
169	1.08532	1.6622	82rongas	Methanol	1.150	0.934

Output 21.6.13 *continued*

Obs	PBSPLINE_ EQRATIO_NOX_ GROUP_S___X	PBSPLINE_ EQRATIO_NOX_ GROUP_S___Y	PBSPLINE_ EQRATIO_NOX_ GROUP_S__GP	Fuel	Eq Ratio	NOx
201	1.173	0.83500	82rongas	_	_	_
202	0.674	0.90805	94%Eth	_	_	_
.	.	.		_	_	_
.	.	.		_	_	_
.	.	.		_	_	_
402	1.267	0.47396	94%Eth	_	_	_
403	0.535	0.41974	Ethanol	_	_	_
.	.	.		_	_	_
.	.	.		_	_	_
.	.	.		_	_	_
603	1.232	0.61021	Ethanol	_	_	_
604	0.645	1.22616	Gasohol	_	_	_
.	.	.		_	_	_
.	.	.		_	_	_
.	.	.		_	_	_
804	1.125	1.24539	Gasohol	_	_	_
805	0.665	1.58007	Indolene	_	_	_
.	.	.		_	_	_
.	.	.		_	_	_
.	.	.		_	_	_
1005	1.224	0.53708	Indolene	_	_	_
1006	0.598	0.20678	Methanol	_	_	_
.	.	.		_	_	_
.	.	.		_	_	_
.	.	.		_	_	_
1206	1.212	0.72281	Methanol	_	_	_

Observations at the beginning and end of data groups are displayed. Missing values (ellipses) are displayed for other values. The first 169 observations contain the scatter plot variables Fuel, EqRatio, and NOx. After that, underscores indicate that those values are ignored. In the actual data set, which is too large to print in this example, observations 170 and beyond are excluded from the scatter plot because of missing values. All $6 \times 201 = 1{,}206$ observations contain interpolated coordinates for the six fit functions. The manufactured variable PBSPLINE_EQRATIO_NOX_GROUP_S__GP contains 201 copies of each of the six fuel values. The other manufactured variables, PBSPLINE_EQRATIO_NOX_GROUP_S___X and PBSPLINE_EQRATIO_NOX_GROUP_S___Y, provide the X and Y coordinates, respectively, for the curve for each fuel group.

The results of the following step (which are not shown) show that `'82rongas'` has only nine values:

```
proc freq data=sashelp.gas(where=(n(eqratio, nox) eq 2));
   tables fuel;
run;
```

Interpolation creates 201 interpolated values (200 line segments) from the minimum to the maximum by (maximum – minimum) / 200. You can specify the MAXPOINTS= option in the REG and PBSPLINE statements to change the number of interpolated values. Interpolation enables splines like the ones in Output 21.6.6 and Output 21.6.12—splines that have too many knots and too few values—to vary substantially from the original data. The automatic smoothing in penalized B-splines often prevents this variation from happening, but not always. In most cases, it is good that ODS Graphics automatically interpolates, but not always. In the next section, you will see examples that do not use interpolation.

PROC TRANSREG

PROC TRANSREG was the first SAS/STAT modeling procedure to incorporate splines. Its syntax is different from that of other modeling procedures. It predates the EFFECT statement and options in the CLASS statement that other procedures now support.

B-Spline Fit Function

The following step creates a fit plot that has both a classification variable and a spline variable:

```
proc transreg data=sashelp.gas nomiss solve ss2 plots=fit(nocli noclm);
   ods select anova fitstatistics fitplot;
   model identity(nox) = spline(eqratio / nknots=5 evenly after) |
                         class(fuel / zero=none);
run;
```

The MODEL statement in PROC TRANSREG lists transformations, expansions, variables, and options. The IDENTITY transformation leaves the dependent variable unchanged. The CLASS expansion replaces the variable Fuel with a set of six binary variables, one for each type of fuel. The vertical bar in the MODEL statement creates an interaction between the SPLINE and CLASS variables. If there were no interaction, the SPLINE transformation would replace the variable EqRatio by a single new variable, which is a linear combination of the columns in a spline basis. Because there is an interaction between the SPLINE and CLASS variables, PROC TRANSREG finds a separate spline transformation for groups of values of the EqRatio variable. There is one group for each level of the CLASS variable (six total).

The model has an implicit intercept, because the ZERO=NONE option creates a binary variable for every level of the CLASS variable. The model has one parameter for each level of the CLASS variable (six total) and eight parameters (degree 3 plus 5 knots) for each spline transformation ($6 \times 8 = 48$ total), for a grand total of 54 parameters (53 model *df*). The AFTER option creates the knots for each term after the SPLINE and CLASS variables are combined to form the interaction term. The structural zeros (those that come from a zero in the binary CLASS variables) are ignored when the knots are found. The EVENLY option spaces the knots evenly; by default, knots are placed at the percentiles. The NOCLI and NOCLM options suppress the default prediction and confidence limits. The results are displayed in Output 21.6.14. Statistical procedures such as PROC TRANSREG display fit statistics in addition to plots. However, PROC TRANSREG does not automatically plot interpolated values. The functions are smoother when there are more data (such as for `'Ethanol'`) and less smooth for sparser functions (such as `'82rongas'`). The advantage of not interpolating is that the splines are less likely to leave the range of the data.

PROC SGPLOT finds separate fit functions by fitting separate models for each group. In contrast, PROC TRANSREG fits a single model, the one shown previously:

$$y = \sum_{i=1}^{6} c_i \left[\beta_{i0} + \beta_{i1}x + \beta_{i2}x^2 + \beta_{i3}x^3\right] + \epsilon$$

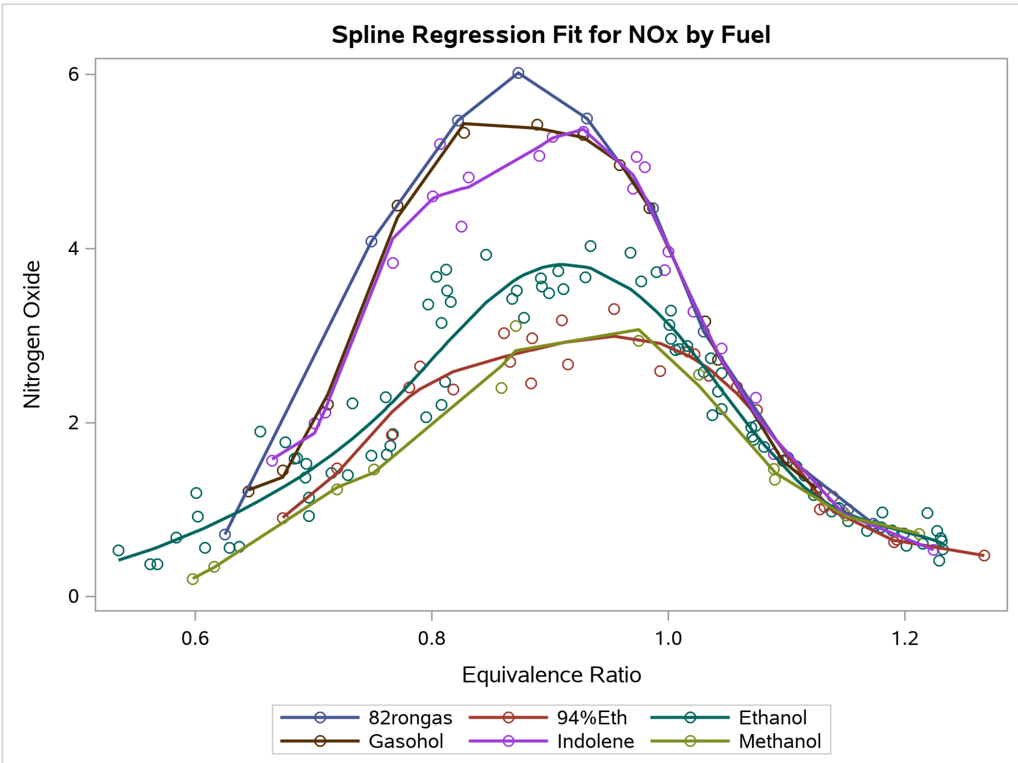
The two approaches are equivalent as long as the knots are the same. However, a procedure such as PROC TRANSREG gives you control over the model beyond anything that PROC SGPLOT gives you. For example, you can force the curves to be the same for each group by omitting the vertical bar from the MODEL statement (in other words, by omitting the interaction of the SPLINE and CLASS variables).

Output 21.6.14 Grouped Fit Function, Fit Statistics, and Plot

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	53	329.1232	6.209871	64.69	<.0001
Error	115	11.0388	0.095989		
Corrected Total	168	340.1619			

Root MSE	0.30982	**R-Square**	0.9675
Dependent Mean	2.34593	**Adj R-Sq**	0.9526
Coeff Var	13.20676		



The following example shows how you can set up a data set for interpolation:

```
proc means data=sashelp.gas(where=(n(nox, eqratio))) noprint;
   class fuel;
   var eqratio;
   output out=m(where=(_type_ eq 1 and trim(_stat_) in ('MIN', 'MAX')));
run;

proc transpose data=m out=m2(drop=_:);
   by fuel;
   id _stat_;
   var eqratio;
run;
```

```
data gas(drop=min max);
   if _n_ = 1 then do i = 1 to n;
      set m2 nobs=n point=i;
      if fuel ne '82rongas' then
         do eqratio = min to max by (max - min) / 200; output; end;
   end;
   set sashelp.gas(where=(n(nox, eqratio)));
   output;
run;

proc transreg data=gas solve ss2 plots(interpolate)=fit(nocli noclm);
   ods select anova fitstatistics fitplot;
   model identity(nox) = class(fuel / zero=none) |
                         spline(eqratio / nknots=5 after evenly);
run;
```

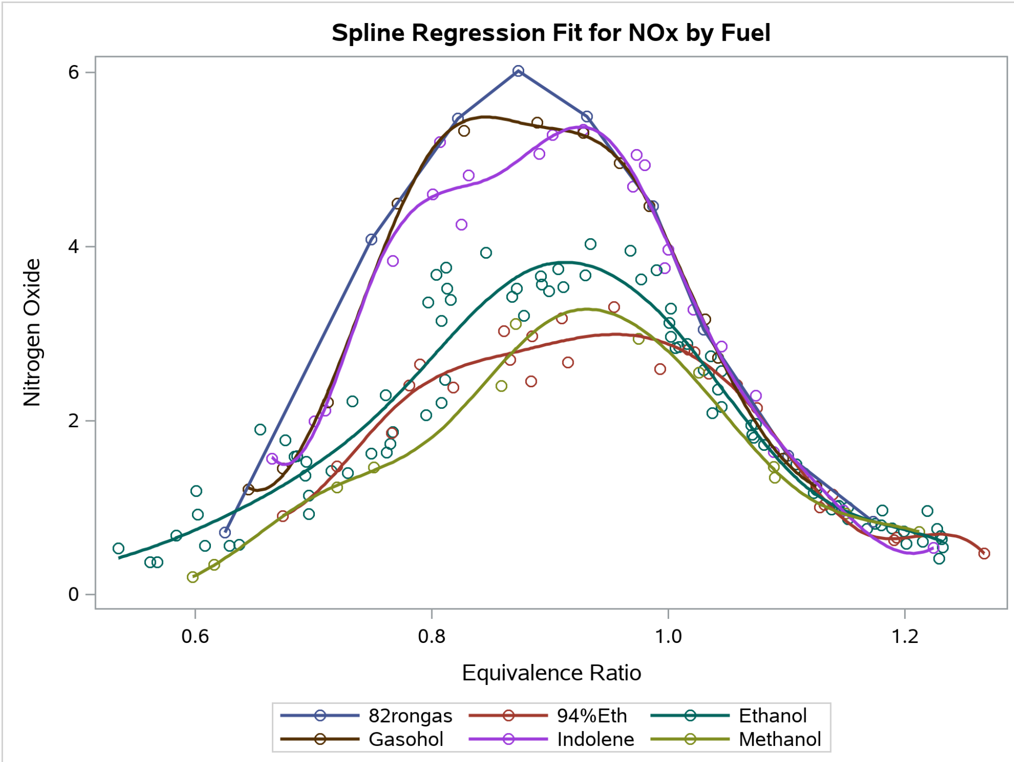
PROC MEANS finds the minimum and maximum for each type of fuel. The WHERE= data set option ensures that only the observations without missing values in the quantitative variables are used. PROC TRANSPOSE arranges the minimum and maximum data set so that there is one observation for each fuel type. The Gas data set adds observations to provide interpolated fit, just as PROC SGPLOT does, except that no interpolated values are generated for the `'82rongas'` group. The results are displayed in Output 21.6.15.

Output 21.6.15 Interpolation

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	53	329.1232	6.209871	64.69	<.0001
Error	115	11.0388	0.095989		
Corrected Total	168	340.1619			

Root MSE	0.30982	**R-Square**	0.9675
Dependent Mean	2.34593	**Adj R-Sq**	0.9526
Coeff Var	13.20676		



The first PROC TRANSREG step excludes observations that have missing values by using the NOMISS option and analyzes a data set that has 171 observations. The second PROC TRANSREG step excludes observations that have missing values by using a WHERE clause in a preceding DATA step and analyzes a data set that has 1,007 observations. All fit statistics match, because the two data sets contain exactly the same 171 observations that are analyzed. The second data set contains missing values in the variable NOx. Those observations are excluded in IDENTITY transformations. PROC TRANSREG has options for analyzing observations that have missing data, but they are not used for IDENTITY variables. PROC TRANSREG and many other procedures score observations that they exclude from the analysis when computing predicted values, residuals, confidence limits, and so on, as long as the relevant data are there to do the computations, even when those observations are not used in computing the sums of squares, degrees of freedom, coefficients, and so on. The PLOTS(INTERPOLATE) option specifies that those scored observations should be used in plotting the regression functions. When you compare the plots in Output 21.6.14 and Output 21.6.15, you see that the fit functions are the same for `'82rongas'` and are smoother in Output 21.6.15 for the other fuel types.

```
data gas(drop=min max);
   if _n_ = 1 then do i = 1 to n;
      set m2 nobs=n point=i;
      do eqratio = min to max by (max - min) / 200; output; end;
   end;
   set sashelp.gas(where=(n(nox, eqratio)));
   output;
run;

proc transreg data=gas solve ss2 plots(interpolate)=fit(nocli noclm);
   ods select anova fitstatistics fitplot;
   model identity(nox) = class(fuel / zero=none)
                         spline(eqratio / nknots=5 after evenly);
run;
```

Parallel Curves

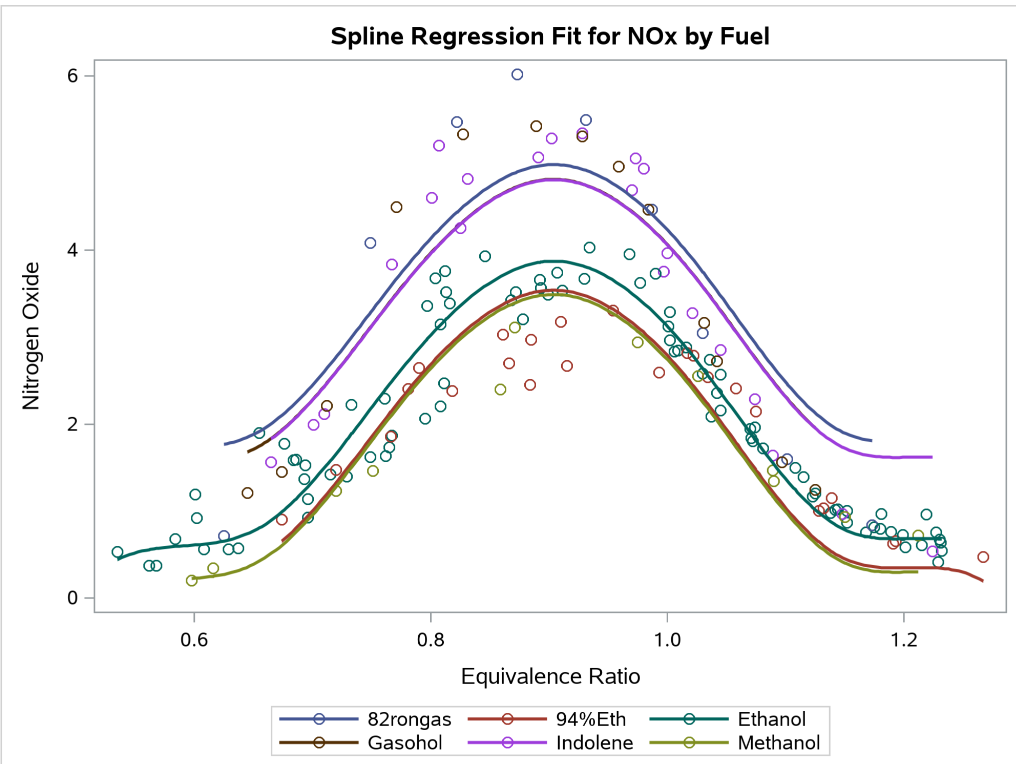
The following steps fit a model without interactions and create the parallel fit functions that are displayed in Output 21.6.16.

Output 21.6.16 Interpolation

Univariate ANOVA Table Based on the Usual Degrees of Freedom

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	13	303.1972	23.32286	97.80	<.0001
Error	155	36.9647	0.23848		
Corrected Total	168	340.1619			

Root MSE	0.48835	**R-Square**	0.8913
Dependent Mean	2.34593	**Adj R-Sq**	0.8822
Coeff Var	20.81676		



Because the plot has one spline curve and six intercepts, there is no need to guard against the lack of smoothness in the `'82rongas'` fuel type. R-square is smaller in this model because you did not fit a separate curve for each group. You cannot display results like this by using the statistical calculations in PROC SGPLOT; you must first compute the predicted values by using a procedure such as PROC TRANSREG.

Penalized B-Spline Fit Functions

The following step uses penalized B-splines and displays the results in Output 21.6.17:

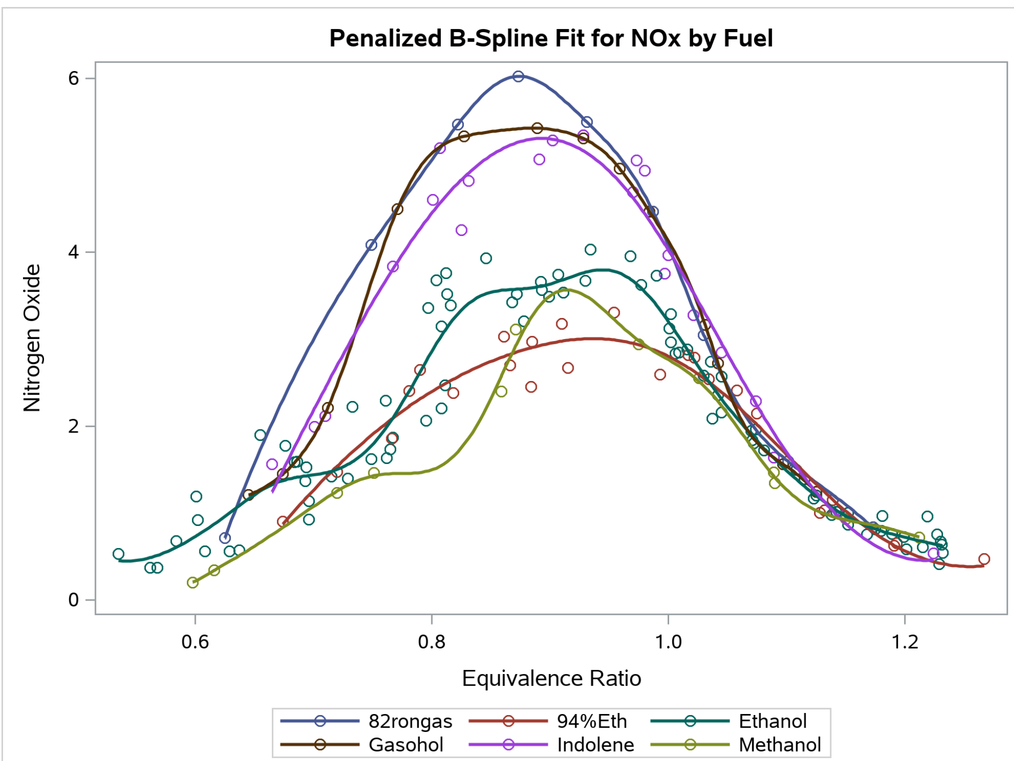
```
proc transreg data=gas ss2 plots(interpolate)=fit(nocli noclm);
   ods select anova fitstatistics fitplot;
   model identity(nox) = class(fuel / zero=none) * pbspline(eqratio / after);
run;
```

Output 21.6.17 Interpolation

Univariate ANOVA Table, Penalized B-Spline Transformation

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	54.288	330.1742	6.081921	69.24	<.0001
Error	113.71	9.9878	0.087834		
Corrected Total	168	340.1619			

Root MSE	0.29637	**R-Square**	0.9706
Dependent Mean	2.34593	**Adj R-Sq**	0.9566
Coeff Var	12.63327		



The MODEL statement uses an asterisk rather than a vertical bar to specify the interaction. PROC TRANSREG fits multiple types of models. There are linear models such as those shown previously where you can specify either a vertical bar or an asterisk, depending on the type of model that you want. There are also models whose results are computed by preprocessing. For these models, which include penalized

B-spline, smoothing spline, and Box-Cox models, you have no freedom to control the intercept terms. Each penalized B-spline has an intercept as part of the model.

Smoothing Spline Functions

PROC TRANSREG has other types of splines, including smoothing splines (Reinsch 1967). This type of spline was first available in PROC GPLOT. The following step fits separate smoothing splines for each fuel and displays the results in Output 21.6.18:

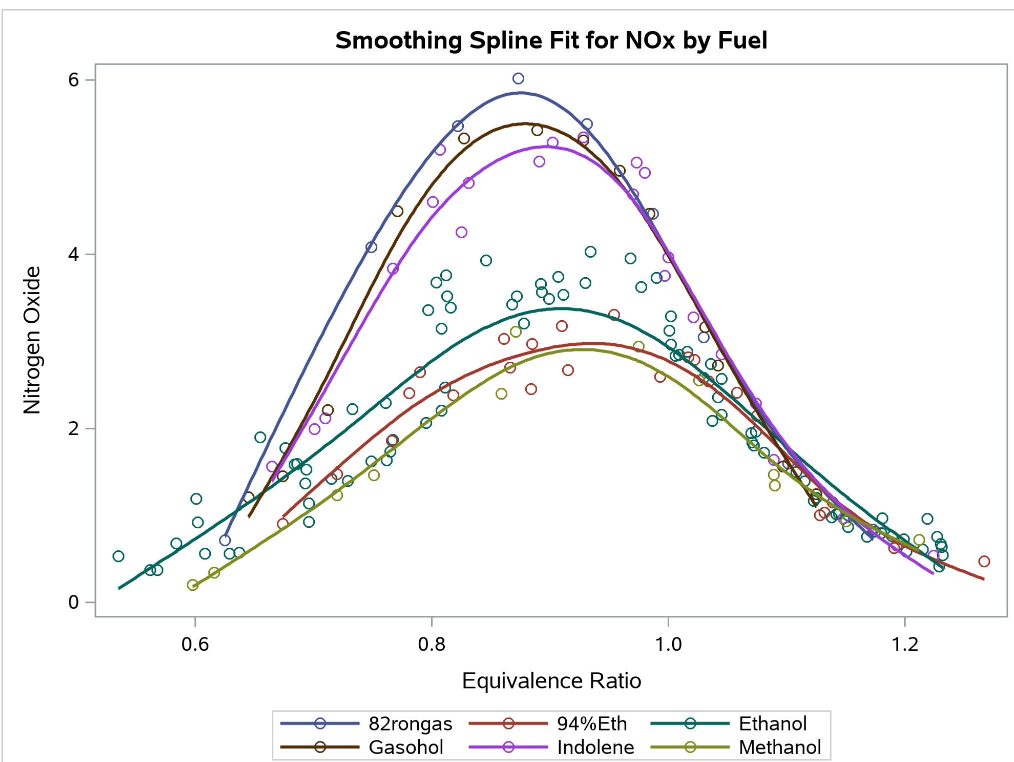
```
proc transreg data=gas solve ss2 plots(interpolate)=fit(nocli noclm);
   ods select anova fitstatistics fitplot;
   model identity(nox) = class(fuel / zero=none) *
                         smooth(eqratio / after sm=60);
run;
```

Output 21.6.18 Interpolation

Univariate ANOVA Table, Smooth Transformation

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	29.774	322.8643	10.84389	86.65	<.0001
Error	138.23	17.2977	0.12514		
Corrected Total	168	340.1619			

Root MSE	0.35375	**R-Square**	0.9491
Dependent Mean	2.34593	**Adj R-Sq**	0.9382
Coeff Var	15.07939		



You use the SM= option to specify a smoothing parameter that ranges from 0 to 100. PROC TRANSREG does not pick an optimal smoothing parameter for you.

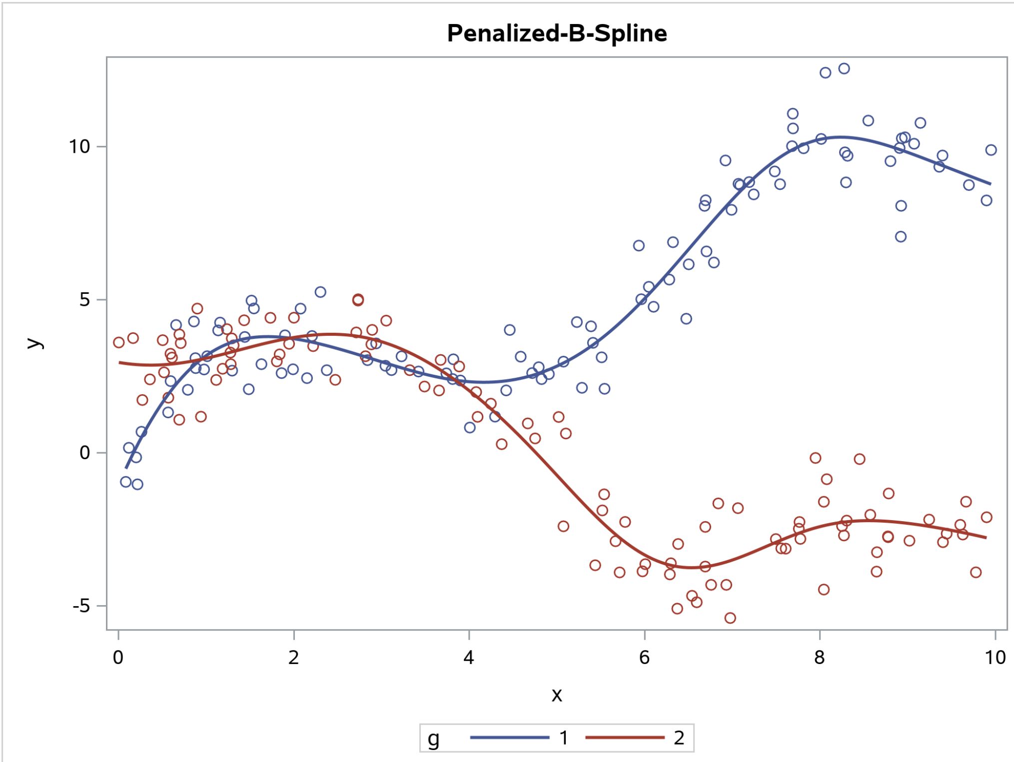
Monotone Splines

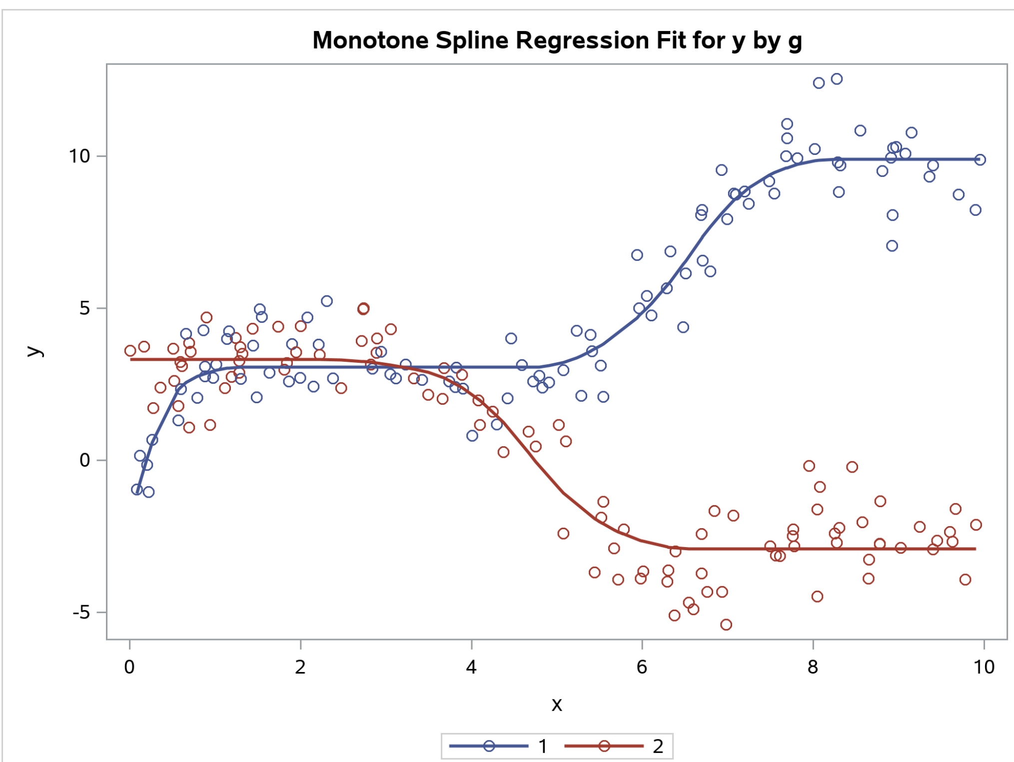
This example uses artificial data to illustrate monotone splines (Winsberg and Ramsay 1980):

```
data x;
   do i = 1 to 100;
      g = 1;
      x = 10 * uniform(17);
      y = x + 2 * sin(x) + normal(17);
      output;
      g = 2;
      x = 10 * uniform(17);
      y = 5 - x - 2 * cos(x) + normal(17);
      output;
   end;
run;

proc sgplot data=x;
   title 'Penalized-B-Spline';
   pbspline y=y x=x / group=g;
run;
title;

proc transreg data=x ss2 plots=fit(nocli noclm) maxiter=100;
   model identity(y) = class(g / zero=none) | mspline(x / nknots=10);
run;
```

The results are displayed in Output 21.6.19. The first plot, which is produced by PROC SGPLOT, shows the penalized B-spline fit functions, which are not monotonic. The second plot, which is produced by PROC TRANSREG, shows the monotone spline fit functions, which increase for the first group and decrease for the second group. Monotone splines are flat in areas where other splines increase and decrease. A monotone spline transformation of a variable x is always at least weakly increasing (that is, it is increasing or flat). The monotone spline fit function is always (at least weakly) increasing or decreasing, depending on the relationship between y and x. PROC TRANSREG is an iterative procedure. In many cases, you can specify the SOLVE option to directly compute a solution without iterations. However, iterations are required for monotone splines. Monotone splines are quadratic by default (DEGREE=2); most other splines are cubic by default.

Output 21.6.19 Penalized B-Splines and Monotone Splines





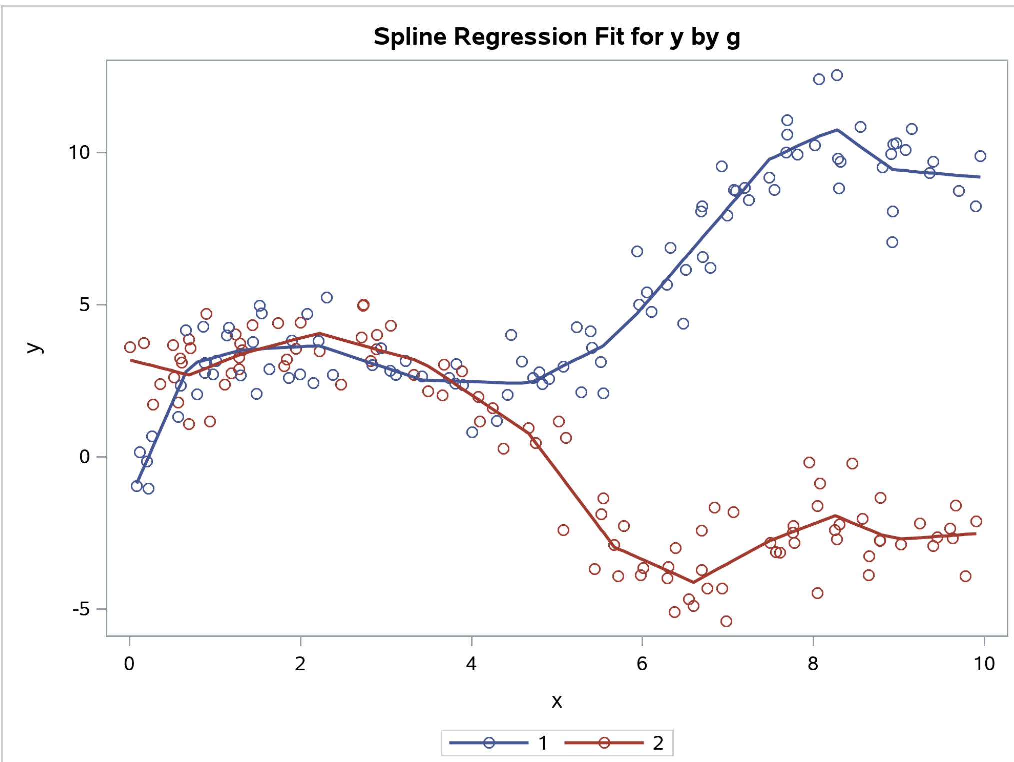
Piecewise-Linear Splines

You can specify the option DEGREE=1 along with splines or monotone splines for a piecewise-linear fit function:

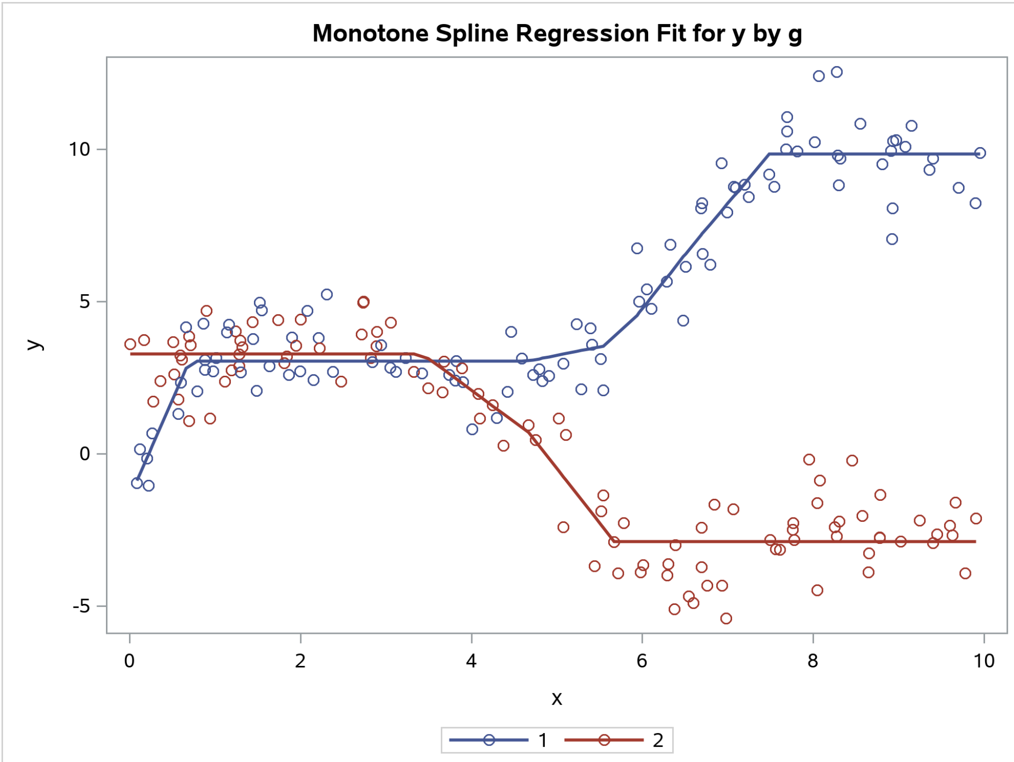
```
proc transreg data=x ss2 plots=fit(nocli noclm) solve;
   model identity(y) = class(g / zero=none) | spline(x / nknots=10 degree=1);
run;

proc transreg data=x ss2 plots=fit(nocli noclm) maxiter=100;
   model identity(y) = class(g / zero=none) | mspline(x / nknots=10 degree=1);
run;
```

The results are shown in Output 21.6.20.

Output 21.6.20 Piecewise-Linear Splines



Output 21.6.20 *continued*

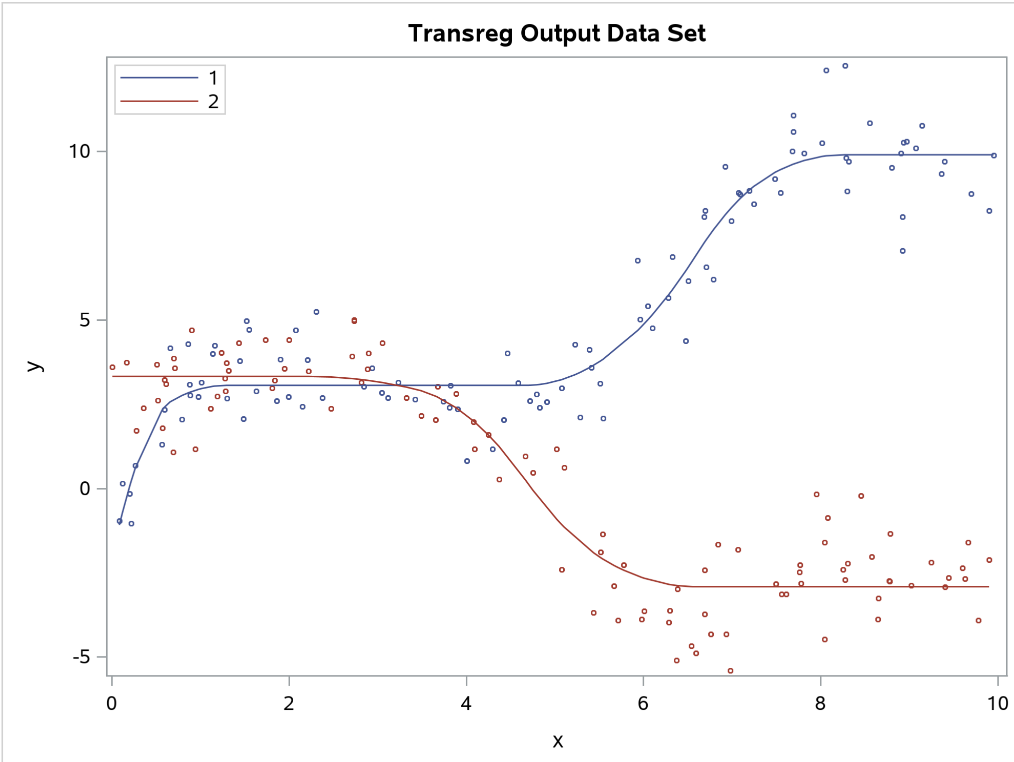


Outputting Splines

You can create an output data set from a SAS/STAT procedure such as PROC TRANSREG and then use PROC SGPLOT to display the results. This enables you to use the full power of the statistical procedure to fit models and customize the results without modifying a template. The following steps create the plot in Output 21.6.21, which has legend and marker customizations:

```
proc transreg data=x ss2 plots=fit(nocli noclm) maxiter=100;
   model identity(y) = class(g / zero=none) | mspline(x / nknots=10);
   output out=msp p replace;
run;

proc sort data=msp; by g x; run;

proc sgplot data=msp;
   title 'Transreg Output Data Set';
   scatter y=y  x=x / group=g markerattrs=(size=3px);
   series  y=py x=x / group=g name='a';
   keylegend 'a' / location=inside position=topleft across=1;
run;
title;
```

Output 21.6.21 Monotone Splines Displayed by PROC SGPLOT



The EFFECT and EFFECTPLOT Statements

Splines are available in many other procedures when you use the EFFECT statement. The following procedures provide an EFFECT statement: GEE, GENMOD, GLIMMIX, GLMSELECT, HPMIXED, LOGISTIC, ORTHOREG, PHREG, PLM, PROBIT, QUANTLIFE, QUANTREG, QUANTSELECT, ROBUSTREG, SURVEYLOGISTIC, and SURVEYREG.

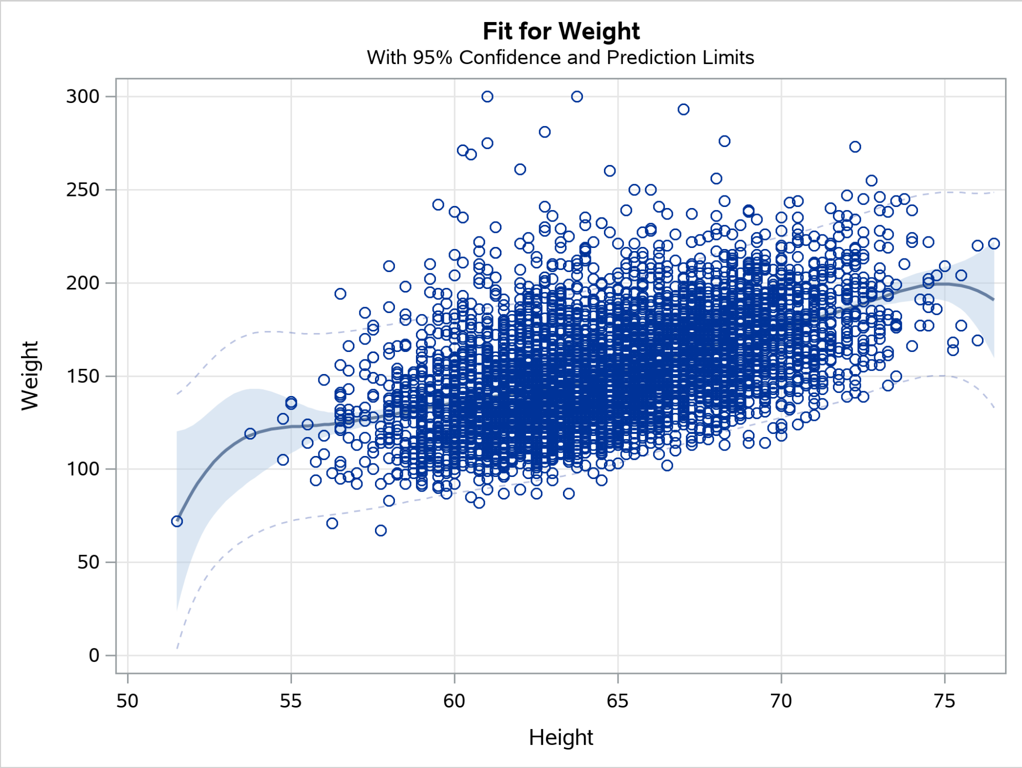
B-Spline Fit Function

The following step fits a spline that has five equally spaced knots:

```
proc orthoreg data=sashelp.heart;
   effect spl = spline(height / knotmethod=equal(5));
   model weight = spl;
   effectplot / obs;
run;
```

The EFFECT statement names the variable Height as a SPLINE variable and requests five evenly spaced knots. The name in the EFFECT statement, Spl, is then specified in the MODEL statement. Notice that the method is similar to the method in PROC TRANSREG, even though elements of the syntax are different. The EFFECTPLOT statement displays the fit function, and the OBS option adds the scatter plot to the graph. Note that the *plot-type*, which can appear before the slash, is not specified. The results are displayed in Output 21.6.22.

Output 21.6.22 EFFECT and EFFECTPLOT Statements

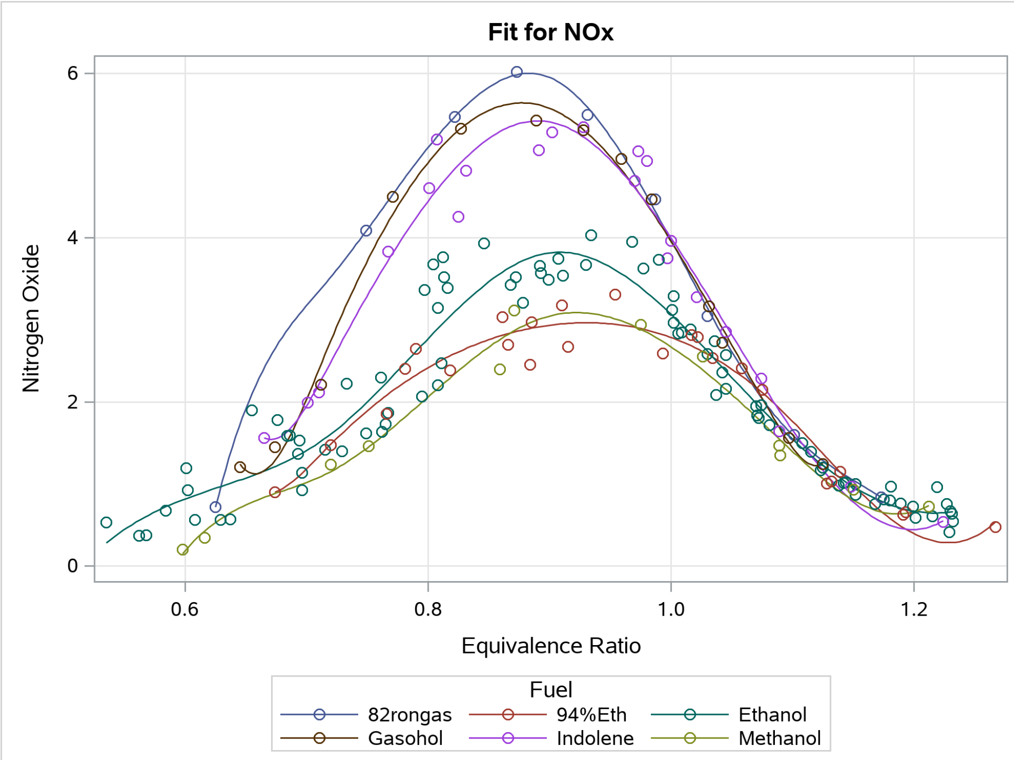


B-Spline Fit Functions

The EFFECTPLOT statement acts like PROC SGPLOT in that it automatically adds observations for interpolation (and in some cases extrapolation). The following step creates the plot in Output 21.6.23:

```
proc orthoreg data=sashelp.gas;
   effect spl = spline(eqratio / knotmethod=equal(3));
   class fuel;
   model nox = spl | fuel;
   effectplot / obs extend=data;
   ods output SliceFitPlot=sp;
run;
```

Output 21.6.23 EFFECT and EFFECTPLOT Statements with a CLASS Variable



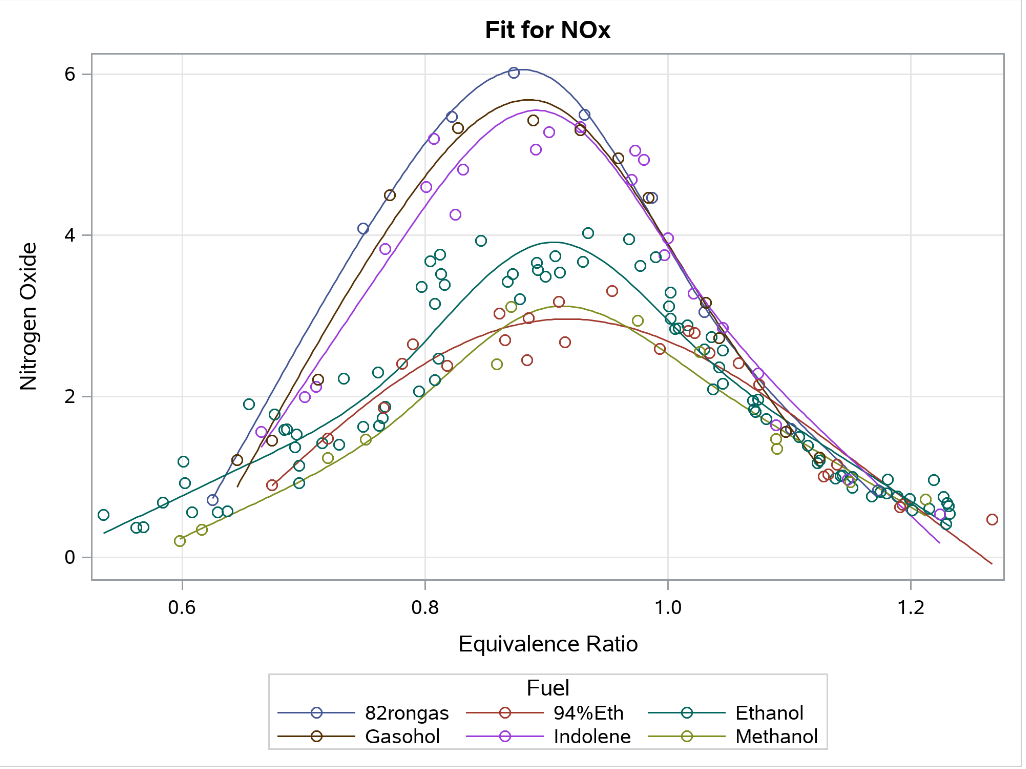
Note that the *plot-type*, which can appear before the slash, must not be specified for an interaction model like this. Furthermore, you must specify the EXTEND=DATA option for this type of plot. It causes the interpolation for plotted values to be between the minimum and maximum for each CLASS level, as is shown in the example that created the plot in Output 21.6.15. Without this option, the function for each group is generated through the range of the minimum and maximum of the spline variable EqRatio. Splines often rapidly head to plus or minus infinity as you extrapolate beyond the range of your data. The option does not affect the analysis; it affects only the plot construction. The number of knots was changed to three in this example. Five knots illustrate an even more extreme version of the same problem that was shown in Output 21.6.12. Because the `'82rongas'` function is sparse, that function is not well behaved when there are five knots.

Natural Cubic Splines (Restricted Splines)

You can get much better results by using natural cubic splines (Hastie, Tibshirani, and Friedman 2009), which are also known as restricted splines. The following step creates the plot in Output 21.6.24:

```
proc orthoreg data=sashelp.gas;
   effect spl = spline(eqratio / naturalcubic knotmethod=equal(5));
   class fuel;
   model nox = spl | fuel;
   effectplot / obs;
run;
```

Output 21.6.24 Natural Cubic (Restricted) Splines



Restricted splines are linear beyond the interior knots, so they tend to be better behaved at the tails, particularly with a sparse CLASS level such as `'82rongas'`. There are 30 parameters in this restricted-spline model. In contrast, the full spline model has 51 parameters.

B-Splines

This section provides details about how B-splines work and the ways in which you can use them.

Interior and Exterior Knots

Polynomial splines have interior knots (knots that are inside the range of the minimum and maximum x). B-splines have interior knots and exterior knots (knots that are outside the range of the minimum and maximum x). For example, if x ranges from –2 to 2, a reasonable knot vector for three interior knots is $\mathbf{k} = (-4-\epsilon, -3-\epsilon, -2-\epsilon, -1, 0, 1, 2+\epsilon, 3+\epsilon, 4+\epsilon)$, where ϵ is a small value such as 10^{-12}. These knots are all but evenly spaced. The addition and subtraction of ϵ ensure that the first three exterior knots are less than the minimum and the last three exterior knots are greater than the maximum. Statistical procedures such as PROC TRANSREG and the procedures that support the EFFECT statement enable you to explicitly control both interior and exterior knots. Simpler options enable you to specify the number of knots and let the procedure determine their values. In many cases, fit functions based on slightly different knots are only slightly different. However, when you fit a model to one set of data and then score another set of data, you must ensure that you use the same knots for both purposes. The following procedure steps all fit the same model and use the knot vector $\mathbf{k}$:

```
data x2;
   do x = -2 to 2 by 0.1;
      y = x + sin(x) + normal(17);
      output;
   end;
run;

proc transreg data=x2 details ss2;
   model identity(y) = bspline(x / nknots=3 evenly=3);
run;

proc transreg data=x2 details ss2;
   model identity(y) = spline(x / nknots=3 evenly=3);
run;

%let k = -4.000000000001 -3.000000000001 -2.000000000001 -1 0 1
          2.000000000001  3.000000000001  4.000000000001;
proc transreg data=x2 details ss2;
   model identity(y) = bspline(x / knots=&k);
run;

proc orthoreg data=x2;
   effect spl = spline(x / knotmethod=equal(3));
   model  y = spl / noint;
run;

proc orthoreg data=x2;
   effect spl = spline(x / knotmethod=listwithboundary(&k));
   model  y = spl / noint;
run;
```

Although the models are the same, how the effects are grouped can vary. PROC TRANSREG with the DETAILS option displays the knots shown in Output 21.6.25. PROC TRANSREG with the BSPLINE expansion fits a model that has no intercept and seven 1-*df* B-spline effects. PROC TRANSREG with the SPLINE transformation fits a model that has an intercept and one 6-*df* B-spline effect. PROC ORTHOREG fits a model that has no intercept and seven 1-*df* B-spline effects.

PROC TRANSREG, because it has more than a 30-year history, does not create evenly spaced knots by default. By default, in the data set X2, the knot list is $\mathbf{k} = (-2-\epsilon, -2-\epsilon, -2-\epsilon, -1, 0, 1, 2+\epsilon, 2+\epsilon, 2+\epsilon)$. (Exterior knots are repeated.) Both repeated and all-but-equally-spaced exterior knots work, but the latter method better matches later software developments.

Output 21.6.25 Interior and Exterior Knots

Model Statement Specification Details				
Type	**DF**	**Variable**	**Description**	**Value**
Dep	1	Identity(y)	DF	1
Ind	6	Spline(x)	Degree	3
			Number of Knots	3
			Exterior Knots	-4.000000000001000E+00
				-3.000000000001000E+00
				-2.000000000001000E+00
			Interior Knots	-1.000000000000000E+00
				8.8817841970012000E-16
				1.0000000000000000E+00
			Exterior Knots	2.0000000000010000E+00
				3.0000000000010000E+00
				4.0000000000010000E+00
			Coefficients	-17.07078951
				0.20741011
				-1.77174923
				0.58192568
				0.54636248
				3.33130267
				-8.42096356

B-Spline Basis

The following four steps show four equivalent methods of putting the B-spline basis into a SAS data set:

```
proc transreg data=x2 details ss2;
   model identity(y) = bspline(x / nknots=3 evenly=3);
   output out=b1(keep=x_:) replace;
run;

proc glimmix data=x2 outdesign=b2(drop=x y
   rename=(%macro ren; %do i = 1 %to 7; _x&i=x_%eval(&i-1) %end; %mend; %ren));
   effect spl = spline(x / knotmethod=equal(3));
   model y = spl / noint;
run;

%let k = -4.000000000001 -3.000000000001 -2.000000000001 -1 0 1
          2.000000000001  3.000000000001  4.000000000001;

proc iml;
   use x2(keep=x); read all into x;
   b = bspline(x, 3, {&k});
   vname = 'x_0' : rowcatc('x_' || char(ncol(b)-1));
   create b3 from b[colname=vname]; append from b;
quit;
```

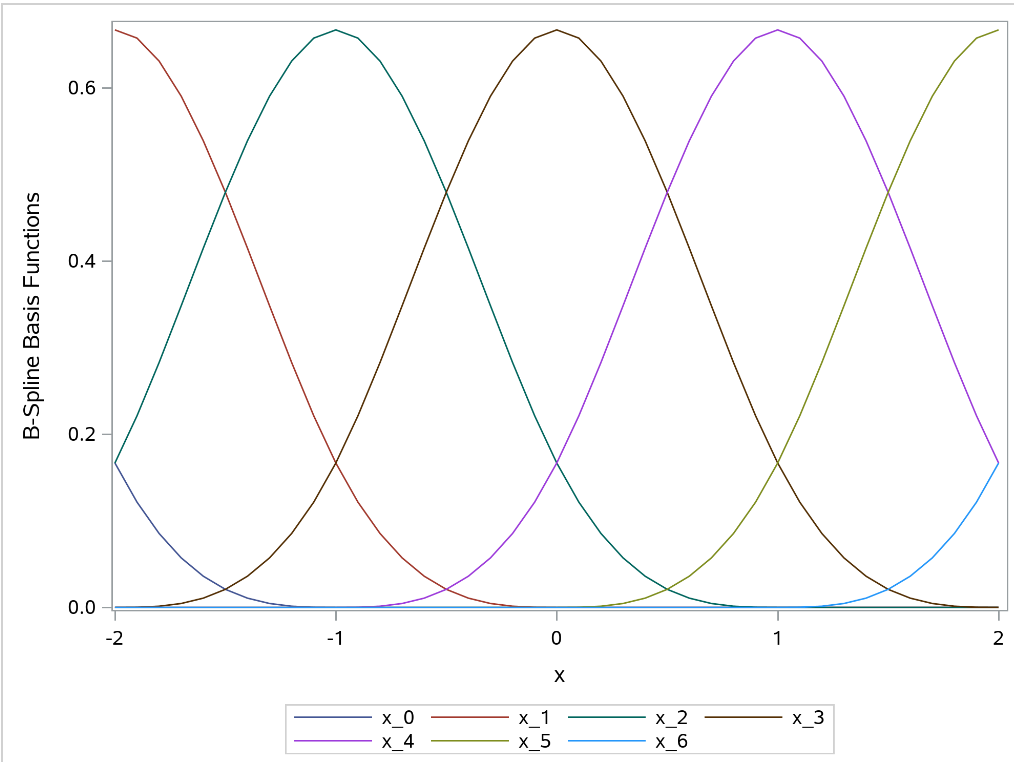
```
data b4(keep=x:);
   %let d   = 3;                        /* degree                  */
   %let nkn = %eval(&d * 2 + 3);        /* total number of knots   */
   %let nb  = %eval(&d + 1 + 3);        /* number of cols in basis */
   array k[&nkn] _temporary_ (&k);      /* knots                   */
   array b[&nb] x_0 - x_%eval(&nb - 1); /* basis                   */
   array w[%eval(2 * &d)];              /* work                    */
   set x2;
   do i = 1 to &nb; b[i] = 0; end;

   * find the index of first knot greater than current data value;
   do ki = 1 to &nkn while(k[ki] le x); end;
   kki = ki - &d - 1;

   * make the basis;
   b[1 + kki] = 1;
   do j = 1 to &d;
      w[&d + j] = k[ki + j - 1] - x;
      w[j] = x - k[ki - j];
      s = 0;
      do i = 1 to j;
         t = w[&d + i] + w[j + 1 - i];
         if t ne 0.0 then t = b[i + kki] / t;
         b[i + kki] = s + w[&d + i] * t;
         s = w[j + 1 - i] * t;
      end;
      b[j + 1 + kki] = s;
   end;
run;
```

The %REN macro is used to rename the PROC GLIMMIX variables, because different procedures have different rules for naming variables. Each data set has seven columns (intercept, degree 3, and three knots, although no column corresponds directly to these terms). The B-spline basis has at most order 4 (degree + 1) nonzero values in each row. This sparseness can be exploited by modeling code, which is one of the many appealing characteristics of B-splines. The following steps compare and display the bases:

```
options nolabel;
proc compare error note briefsummary criterion=1e-12
   data=b1 compare=b2 method=relative(1);
run;

proc compare error note briefsummary criterion=1e-12
   data=b1 compare=b3 method=relative(1);
run;

proc compare error note briefsummary criterion=1e-12
   data=b1 compare=b4(drop=x) method=relative(1);
run;
options label;

proc sgplot data=b4;
   %macro s; %do i = 0 %to &nb-1; series y=x_&i x=x; %end; %mend; %s
   yaxis label='B-Spline Basis Functions';
run;
```

The PROC COMPARE output (not displayed here) shows that the matrices are identical. The B-spline basis is plotted in Output 21.6.26.

Output 21.6.26 B-Spline Basis



The next steps show that the polynomial-spline basis and the B-spline basis are linear transformations of each other:

```
proc transreg data=x2 details ss2;
   model identity(y) = pspline(x / name=(p) nknots=3);
   output out=p1(keep=p_:) replace;
run;

data both; merge p1 b1; run;

proc print data=both noobs;
   format _numeric_ bestd6.5 p_1 p_2 best6.;
run;

proc cancorr data=both;
  var x:;
  with p:;
run;
```

The polynomial spline (variables that begin with p_) and B-spline (variables that begin with x_) bases are displayed in Output 21.6.27. The variable p_1 matches the original X variable. All canonical correlations in Output 21.6.28 are 1.0. This shows that every vector in the space that is defined by the columns in the B-spline basis can be constructed from a linear combination of the columns of the polynomial-spline basis.

Output 21.6.27 Polynomial-Spline and B-Spline Bases

p_1	p_2	p_3	p_4	p_5	p_6	x_0	x_1	x_2	x_3	x_4	x_5	x_6
-2	4	-8	0	0	0	0.1667	0.6667	0.1667	0	0	0	0
-1.9	3.61	-6.859	0	0	0	0.1215	0.6572	0.2212	0.0002	0	0	0
-1.8	3.24	-5.832	0	0	0	0.0853	0.6307	0.2827	0.0013	0	0	0
-1.7	2.89	-4.913	0	0	0	0.0572	0.5902	0.3482	0.0045	0	0	0
-1.6	2.56	-4.096	0	0	0	0.0360	0.5387	0.4147	0.0107	0	0	0
-1.5	2.25	-3.375	0	0	0	0.0208	0.4792	0.4792	0.0208	0	0	0
-1.4	1.96	-2.744	0	0	0	0.0107	0.4147	0.5387	0.0360	0	0	0
-1.3	1.69	-2.197	0	0	0	0.0045	0.3482	0.5902	0.0572	0	0	0
-1.2	1.44	-1.728	0	0	0	0.0013	0.2827	0.6307	0.0853	0	0	0
-1.1	1.21	-1.331	0	0	0	0.0002	0.2212	0.6572	0.1215	0	0	0
-1	1	-1	0	0	0	0	0.1667	0.6667	0.1667	0	0	0
-0.9	0.81	-0.729	0.0010	0	0	0	0.1215	0.6572	0.2212	0.0002	0	0
-0.8	0.64	-0.512	0.0080	0	0	0	0.0853	0.6307	0.2827	0.0013	0	0
-0.7	0.49	-0.343	0.0270	0	0	0	0.0572	0.5902	0.3482	0.0045	0	0
-0.6	0.36	-0.216	0.0640	0	0	0	0.0360	0.5387	0.4147	0.0107	0	0
-0.5	0.25	-0.125	0.1250	0	0	0	0.0208	0.4792	0.4792	0.0208	0	0
-0.4	0.16	-0.064	0.2160	0	0	0	0.0107	0.4147	0.5387	0.0360	0	0
-0.3	0.09	-0.027	0.3430	0	0	0	0.0045	0.3482	0.5902	0.0572	0	0
-0.2	0.04	-0.008	0.5120	0	0	0	0.0013	0.2827	0.6307	0.0853	0	0
-0.1	0.01	-0.001	0.7290	0	0	0	0.0002	0.2212	0.6572	0.1215	0	0
0	0	0	1	0	0	0	0	0.1667	0.6667	0.1667	0	0
0.1	0.01	0.0010	1.3310	0.0010	0	0	0	0.1215	0.6572	0.2212	0.0002	0
0.2	0.04	0.0080	1.7280	0.0080	0	0	0	0.0853	0.6307	0.2827	0.0013	0
0.3	0.09	0.0270	2.1970	0.0270	0	0	0	0.0572	0.5902	0.3482	0.0045	0
0.4	0.16	0.0640	2.7440	0.0640	0	0	0	0.0360	0.5387	0.4147	0.0107	0
0.5	0.25	0.1250	3.3750	0.1250	0	0	0	0.0208	0.4792	0.4792	0.0208	0
0.6	0.36	0.2160	4.0960	0.2160	0	0	0	0.0107	0.4147	0.5387	0.0360	0
0.7	0.49	0.3430	4.9130	0.3430	0	0	0	0.0045	0.3482	0.5902	0.0572	0
0.8	0.64	0.5120	5.8320	0.5120	0	0	0	0.0013	0.2827	0.6307	0.0853	0
0.9	0.81	0.7290	6.8590	0.7290	0	0	0	0.0002	0.2212	0.6572	0.1215	0
1	1	1	8	1	0	0	0	0	0.1667	0.6667	0.1667	0
1.1	1.21	1.3310	9.2610	1.3310	0.0010	0	0	0	0.1215	0.6572	0.2212	0.0002
1.2	1.44	1.7280	10.648	1.7280	0.0080	0	0	0	0.0853	0.6307	0.2827	0.0013
1.3	1.69	2.1970	12.167	2.1970	0.0270	0	0	0	0.0572	0.5902	0.3482	0.0045
1.4	1.96	2.7440	13.824	2.7440	0.0640	0	0	0	0.0360	0.5387	0.4147	0.0107
1.5	2.25	3.3750	15.625	3.3750	0.1250	0	0	0	0.0208	0.4792	0.4792	0.0208
1.6	2.56	4.0960	17.576	4.0960	0.2160	0	0	0	0.0107	0.4147	0.5387	0.0360
1.7	2.89	4.9130	19.683	4.9130	0.3430	0	0	0	0.0045	0.3482	0.5902	0.0572
1.8	3.24	5.8320	21.952	5.8320	0.5120	0	0	0	0.0013	0.2827	0.6307	0.0853
1.9	3.61	6.8590	24.389	6.8590	0.7290	0	0	0	0.0002	0.2212	0.6572	0.1215
2	4	8	27	8	1	0	0	0	0	0.1667	0.6667	0.1667

Output 21.6.28 Canonical Correlations

Canonical Correlation Analysis

	Canonical Correlation	Adjusted Canonical Correlation	Approximate Standard Error	Squared Canonical Correlation	Eigenvalues of Inv(E)*H = CanRsq/(1-CanRsq) Eigenvalue	Difference	Proportion	Cumulative
1	1.000000	.	0.000000	1.000000	Infty	.	.	.
2	1.000000	.	0.000000	1.000000	Infty	.	.	.
3	1.000000	.	0.000000	1.000000	Infty	.	.	.
4	1.000000	.	0.000000	1.000000	Infty	.	.	.
5	1.000000	.	0.000000	1.000000	Infty	.	.	.
6	1.000000	.	0.000000	1.000000	Infty		.	.

Test of H0: The canonical correlations in the current row and all that follow are zero

	Likelihood Ratio	Approximate F Value	Num DF	Den DF	Pr > F
1	0.00000000	Infty	36	130.11	<.0001
2	0.00000000	Infty	25	112.95	<.0001
3	0.00000000	Infty	16	95.344	<.0001
4	0.00000000	Infty	9	78.03	<.0001
5	0.00000000	Infty	4	66	<.0001
6	0.00000000	Infty	1	34	<.0001

Scoring

Many procedures score observations that have missing values or have zero, missing, or invalid weights or frequencies. If the independent variables are all valid, then the procedure can compute predicted values. When the independent and dependent variables are all valid, then the procedure can also compute residuals. The following steps illustrate by using two simple regression models:

```
data class;
   set sashelp.class(rename=(height=Height1)) nobs=n;
   output;
   if _n_ = n then do;
      call missing(name, sex, height1);
      do age = 10 to 17; output; end;
   end;
run;

proc reg data=class;
   model height1 = age;
   output out=p1 p=p1y r=r1y;
run;

data class;
   f = 1;
   set sashelp.class(rename=(height=Height2)) nobs=n;
   output;
   if _n_ = n then do;
      call missing(f, name, sex);
```

```
      do age = 10 to 17; output; end;
   end;
run;

proc reg data=class;
   freq f;
   model height2 = age;
   output out=p2 p=p2y r=r2y;
run;

data all; merge p1 p2; run;

proc print data=all;
   var f name sex age height: p: r:;
   format p1y p2y r1y r2y 6.3;
run;
```

The results are displayed in Output 21.6.29.

Output 21.6.29 Scoring in a Simple Regression Model

Obs	f	Name	Sex	Age	Height1	Height2	p1y	p2y	r1y	r2y
1	1	Alfred	M	14	69.0	69.0	64.244	64.244	4.756	4.756
2	1	Alice	F	13	56.5	56.5	61.457	61.457	-4.957	-4.957
3	1	Barbara	F	13	65.3	65.3	61.457	61.457	3.843	3.843
4	1	Carol	F	14	62.8	62.8	64.244	64.244	-1.444	-1.444
5	1	Henry	M	14	63.5	63.5	64.244	64.244	-0.744	-0.744
6	1	James	M	12	57.3	57.3	58.670	58.670	-1.370	-1.370
7	1	Jane	F	12	59.8	59.8	58.670	58.670	1.130	1.130
8	1	Janet	F	15	62.5	62.5	67.031	67.031	-4.531	-4.531
9	1	Jeffrey	M	13	62.5	62.5	61.457	61.457	1.043	1.043
10	1	John	M	12	59.0	59.0	58.670	58.670	0.330	0.330
11	1	Joyce	F	11	51.3	51.3	55.882	55.882	-4.582	-4.582
12	1	Judy	F	14	64.3	64.3	64.244	64.244	0.056	0.056
13	1	Louise	F	12	56.3	56.3	58.670	58.670	-2.370	-2.370
14	1	Mary	F	15	66.5	66.5	67.031	67.031	-0.531	-0.531
15	1	Philip	M	16	72.0	72.0	69.818	69.818	2.182	2.182
16	1	Robert	M	12	64.8	64.8	58.670	58.670	6.130	6.130
17	1	Ronald	M	15	67.0	67.0	67.031	67.031	-0.031	-0.031
18	1	Thomas	M	11	57.5	57.5	55.882	55.882	1.618	1.618
19	1	William	M	15	66.5	66.5	67.031	67.031	-0.531	-0.531
20	.			10	.	66.5	53.095	53.095	.	13.405
21	.			11	.	66.5	55.882	55.882	.	10.618
22	.			12	.	66.5	58.670	58.670	.	7.830
23	.			13	.	66.5	61.457	61.457	.	5.043
24	.			14	.	66.5	64.244	64.244	.	2.256
25	.			15	.	66.5	67.031	67.031	.	-0.531
26	.			16	.	66.5	69.818	69.818	.	-3.318
27	.			17	.	66.5	72.605	72.605	.	-6.105

The first model excludes observations that have missing values. The second model excludes observations that have missing frequencies. The first model uses the variable Height1, which has missing values in the extra observations that are to be scored. The predicted values and residual for the first model are displayed in the variables p1y and r1y, respectively. The second model uses the variable Height2, which has no missing values. The predicted values and residual for the second model are displayed in the variables p1y and r1y, respectively. The predicted values match for both models. The residuals match for the observations that do not have a missing height. For both models, the predicted values for the scored observations match the predicted values for the analysis observations that have the same age. Additionally, the procedure creates predicted values for heights that are not in the data set.

The following steps create observations for each fuel type that are to be scored:

```
proc means min max data=sashelp.gas;
   class fuel;
   var eqratio;
   output out=m(where=(_type_ eq 1 and _stat_ in ('MIN', 'MAX')));
run;

proc transpose data=m out=m2(drop=_:);
   var eqratio;
   by fuel;
   id _stat_;
run;

data score(drop=min max);
   set m2;
   do eqratio = min to max by (max - min) / 200; output; end;
run;
```

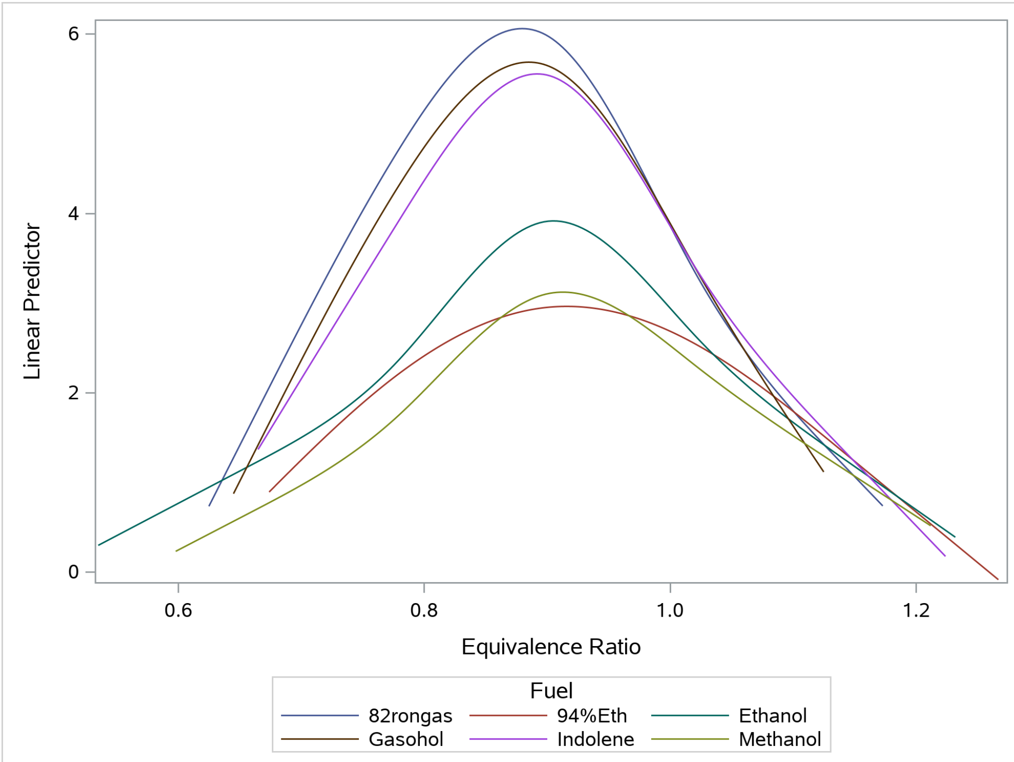
The following steps concatenate those observations to the data set and use PROC GLIMMIX to score them:

```
data gas;
   set sashelp.gas(where=(n(nox))) score;
run;

proc glimmix data=gas;
   effect spl = spline(eqratio / naturalcubic knotmethod=equal(5));
   class fuel;
   model nox = spl | fuel;
   output out=scored(where=(nmiss(nox))) pred=py;
run;
```

The next step displays the scores:

```
proc sgplot data=scored;
   series y=py x=eqratio / group=fuel;
run;
```

Output 21.6.30 Scored Observations

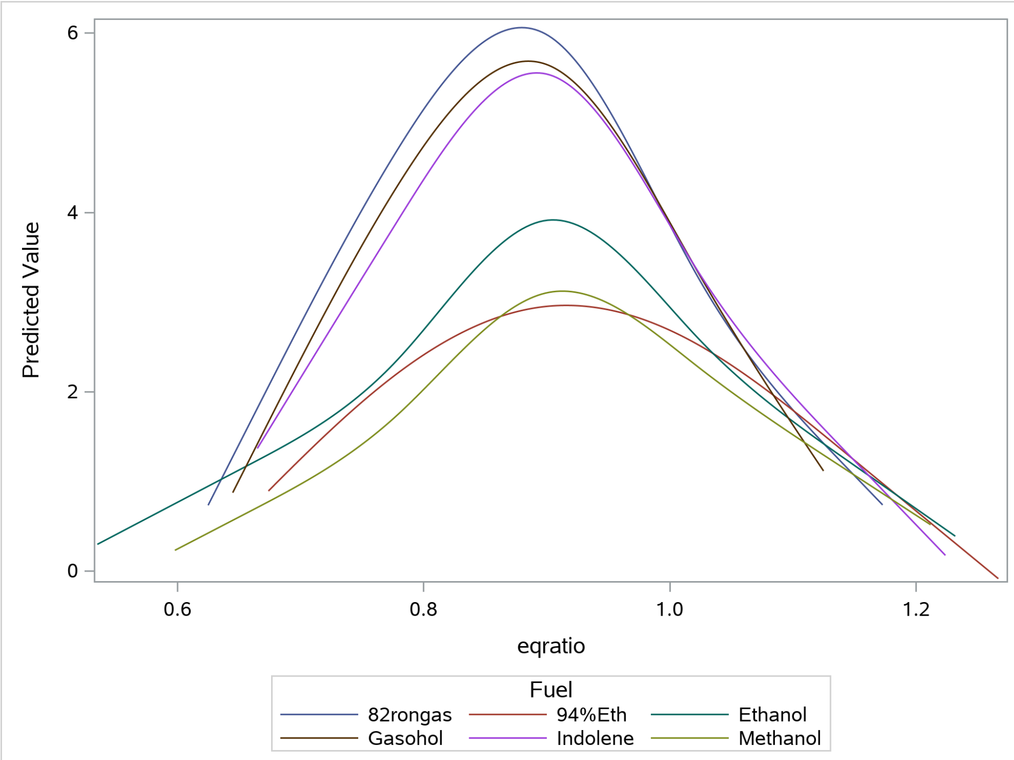


The following steps score the same observations by using the PLM procedure:

```
proc glimmix data=sashelp.gas;
   effect spl = spline(eqratio / naturalcubic knotmethod=equal(5));
   class fuel;
   model nox = spl | fuel;
   store SplineModel;
run;

proc plm restore=SplineModel;
   score data=score out=scored2 predicted=py;
run;

proc sgplot data=scored2;
   series y=py x=eqratio / group=fuel;
run;
```

The scores are displayed in Output 21.6.31. For more information about PROC PLM, see Chapter 91, "The PLM Procedure."

Output 21.6.31 Observations Scored by PROC PLM



Nonlinear Models

The preceding examples all show linear models. The EFFECT statement and the SPLINE effect are available for both linear and nonlinear models. The following procedures provide an EFFECT statement: GEE, GENMOD, GLIMMIX, GLMSELECT, HPMIXED, LOGISTIC, ORTHOREG, PHREG, PLM, PROBIT, QUANTLIFE, QUANTREG, QUANTSELECT, ROBUSTREG, SURVEYLOGISTIC, and SURVEYREG. The following steps show how to use the EFFECT statement in logistic regression:

```
data Neuralgia;
   input Treatment $ Sex $ Age Duration Pain $ @@;
   datalines;
P  F  68   1  No   B  M  74  16  No   P  F  67  30  No   P  M  66  26  Yes
B  F  67  28  No   B  F  77  16  No   A  F  71  12  No   B  F  72  50  No
B  F  76   9  Yes  A  M  71  17  Yes  A  F  63  27  No   A  F  69  18  Yes
B  F  66  12  No   A  M  62  42  No   P  F  64   1  Yes  A  F  64  17  No
P  M  74   4  No   A  F  72  25  No   P  M  70   1  Yes  B  M  66  19  No
B  M  59  29  No   A  F  64  30  No   A  M  70  28  No   A  M  69   1  No
B  F  78   1  No   P  M  83   1  Yes  B  F  69  42  No   B  M  75  30  Yes
P  M  77  29  Yes  P  F  79  20  Yes  A  M  70  12  No   A  F  69  12  No
B  F  65  14  No   B  M  70   1  No   B  M  67  23  No   A  M  76  25  Yes
P  M  78  12  Yes  B  M  77   1  Yes  B  F  69  24  No   P  M  66   4  Yes
P  F  65  29  No   P  M  60  26  Yes  A  M  78  15  Yes  B  M  75  21  Yes
A  F  67  11  No   P  F  72  27  No   P  F  70  13  Yes  A  M  75   6  Yes
B  F  65   7  No   P  F  68  27  Yes  P  M  68  11  Yes  P  M  67  17  Yes
B  M  70  22  No   A  M  65  15  No   P  F  67   1  Yes  A  M  67  10  No
P  F  72  11  Yes  A  F  74   1  No   B  M  80  21  Yes  A  F  69   3  No
;
```

```
proc logistic data=Neuralgia outdesign=coded;
   class Treatment Sex;
   effect Age2      = spline(age / degree=2 knotmethod=equal(0));
   effect Duration2 = polynomial(duration / degree=2);
   model Pain= Treatment Sex Treatment*Sex Age2 Duration2 / expb;
run;

proc print data=coded(obs=10); run;
```

The parameter estimates are displayed in Output 21.6.32. There are three rows for the variable Age, which is designated as a quadratic spline with no knots (in other words, as a quadratic polynomial). There are three rows and two parameter estimates because the B-spline basis had $d + 1$ columns for a polynomial of degree d. The variable Duration is directly specified as a quadratic polynomial, and it has two rows in the table. Output 21.6.33 displays the first 10 coded observations. You can see that Age has values between 0 and 1, which is characteristic of a B-spline basis, and Duration is coded as an ordinary polynomial (duration and duration squared).

Output 21.6.32 Grouped Fit Function, Fit Statistics, and Plot

Analysis of Maximum Likelihood Estimates

Parameter		DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq	Exp(Est)
Intercept		1	-22.1944	11.0096	4.0639	0.0438	0.000
Treatment	**A**	1	0.6601	0.6187	1.1381	0.2861	1.935
Treatment	**B**	1	2.0464	0.8893	5.2954	0.0214	7.740
Sex	**F**	1	0.8436	0.4179	4.0736	0.0436	2.325
Treatment*Sex	**A F**	1	-0.2576	0.5914	0.1897	0.6632	0.773
Treatment*Sex	**B F**	1	0.5762	0.7017	0.6743	0.4116	1.779
Age2	**1**	1	16.8907	6.6152	6.5194	0.0107	21654611
Age2	**2**	1	29.1886	14.6180	3.9871	0.0459	4.747E12
Age2	**3**	0	0	.	.	.	.
Duration		1	-0.2274	0.1667	1.8615	0.1724	0.797
Duration^2		1	0.00803	0.00562	2.0431	0.1529	1.008

Output 21.6.33 The First 10 Observations of the OUTDESIGN= Data Set

Obs	Pain	Intercept	TreatmentA	TreatmentB	SexF	TreatmentASexF	TreatmentBSexF	Age21	Age22	Age23	Duration	Duration_2
1	No	1	-1	-1	1	-1	-1	0.19531	0.73437	0.07031	1	1
2	No	1	0	1	-1	0	-1	0.07031	0.73437	0.19531	16	256
3	No	1	-1	-1	1	-1	-1	0.22222	0.72222	0.05556	30	900
4	Yes	1	-1	-1	-1	1	1	0.25087	0.70660	0.04253	26	676
5	No	1	0	1	1	0	1	0.22222	0.72222	0.05556	28	784
6	No	1	0	1	1	0	1	0.03125	0.68750	0.28125	16	256
7	No	1	1	0	1	1	0	0.12500	0.75000	0.12500	12	144
8	No	1	0	1	1	0	1	0.10503	0.74826	0.14670	50	2500
9	Yes	1	0	1	1	0	1	0.04253	0.70660	0.25087	9	81
10	Yes	1	1	0	-1	-1	0	0.12500	0.75000	0.12500	17	289

PROC ADAPTIVEREG

The ADAPTIVEREG procedure fits multivariate adaptive regression splines (Friedman 1991). Multivariate adaptive regression splines extend linear models to analyze nonlinear dependencies and produce parsimonious models that do not overfit the data and thus have good predictive power. This method is a nonparametric regression technique that combines both regression splines and model selection. It constructs spline basis functions in an adaptive way by automatically selecting appropriate knot values for different variables, and it obtains reduced models by applying model selection techniques. The method does not assume parametric model forms and does not require specification of knot values. For more information about PROC ADAPTIVEREG, see Kuhfeld and Cai (2013) and Chapter 25, "The ADAPTIVEREG Procedure." The following step displays the results in Output 21.6.34:

```
proc adaptivereg data=sashelp.gas plots=all details=bases;
   class fuel;
   model nox = eqratio | fuel;
run;
```

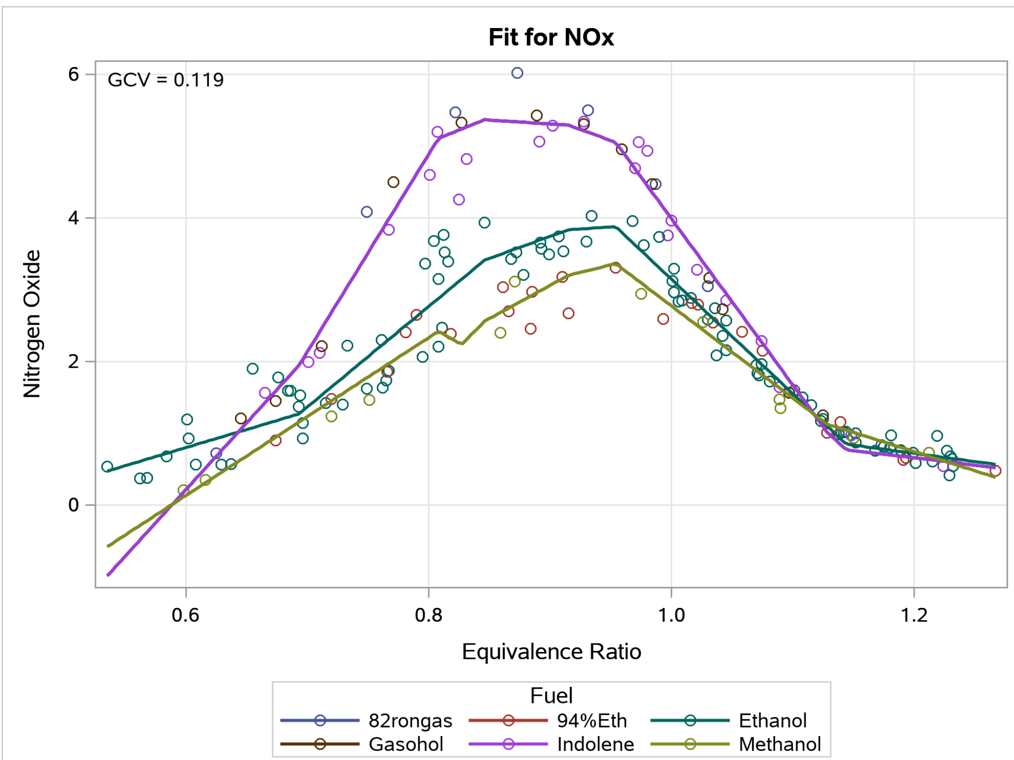
Output 21.6.34 Grouped Fit Function, Fit Statistics, and Plot

Basis Information

Name	Transformation
Basis0	1
Basis1	Basis0*MAX(EqRatio - 0.915,0)
Basis2	Basis0*MAX(0.915 - EqRatio,0)
Basis3	Basis0*(Fuel = 'Indolene' OR Fuel = '82rongas' OR Fuel = 'Gasohol' OR Fuel = 'Ethanol')
Basis4	Basis0*NOT(Fuel = 'Indolene' OR Fuel = '82rongas' OR Fuel = 'Gasohol' OR Fuel = 'Ethanol')
Basis5	Basis0*(Fuel = 'Ethanol')
Basis6	Basis0*NOT(Fuel = 'Ethanol')
Basis7	Basis6*MAX(EqRatio - 0.808,0)
Basis8	Basis6*MAX(0.808 - EqRatio,0)
Basis9	Basis4*MAX(EqRatio - 0.827,0)
Basis10	Basis4*MAX(0.827 - EqRatio,0)
Basis11	Basis3*MAX(EqRatio - 1.144,0)
Basis12	Basis3*MAX(1.144 - EqRatio,0)
Basis13	Basis0*MAX(EqRatio - 0.954,0)
Basis14	Basis0*MAX(0.954 - EqRatio,0)
Basis15	Basis6*MAX(EqRatio - 1.128,0)
Basis16	Basis6*MAX(1.128 - EqRatio,0)
Basis17	Basis3*MAX(EqRatio - 0.693,0)
Basis18	Basis3*MAX(0.693 - EqRatio,0)
Basis19	Basis0*MAX(EqRatio - 0.846,0)
Basis20	Basis0*MAX(0.846 - EqRatio,0)

Output 21.6.34 *continued*

Regression Spline Model after Backward Selection					
Name	**Coefficient**	**Parent**	**Variable**	**Knot**	**Levels**
Basis0	2.8148		Intercept		
Basis2	-5.0396	Basis0	EqRatio	0.9150	
Basis3	1.8016	Basis0	Fuel		4 0 3 2
Basis5	-2.2310	Basis0	Fuel		2
Basis7	-7.2268	Basis6	EqRatio	0.8080	
Basis8	-13.5265	Basis6	EqRatio	0.8080	
Basis9	19.3139	Basis4	EqRatio	0.8270	
Basis10	7.5643	Basis4	EqRatio	0.8270	
Basis11	13.6667	Basis3	EqRatio	1.1440	
Basis13	-17.0561	Basis0	EqRatio	0.9540	
Basis15	7.4962	Basis6	EqRatio	1.1280	
Basis17	8.9758	Basis3	EqRatio	0.6930	
Basis19	-7.8762	Basis0	EqRatio	0.8460	



It is obvious from the plot in Output 21.6.34 that this analysis is different from those shown previously. Three splines are displayed even though there are still six types of fuel. Also, the functions are not smooth; they are piecewise linear. The first table shows that the terms that can enter the model include the following:

Basis0	is an intercept.
Basis1	is a linear truncated power function with a knot at 0.915. Like a hockey stick, this term is flat (0) up through 0.915 and then linearly increases as *x* increases beyond 0.915.
Basis2	is a linear truncated power function with a knot at 0.915. Like a reflection of the preceding hockey stick, this term linearly decreases as *x* increases to 0.915 and is flat (0) beyond 0.915.
Basis3	is a binary variable that is constructed by combining levels of the CLASS variable.
Basis4	is 0 when Basis3 is 1 and 0 otherwise.
Basis5	is a binary variable that corresponds to the `'Ethanol'` level of the CLASS variable.
Basis6	is 0 when Basis5 is 1 and 0 otherwise.

The remaining terms are interactions of preceding terms and other hockey-stick functions. Forward and backward selection creates a final model, which consists of a subset of the full set of basis functions. A model such as this, which is less smooth and treats groups of fuels the same, is likely to do better in scoring additional observations than a model that has more parameters (as many of the models shown previously have).

For More Information

The following procedures support the EFFECT statement:

GEE	Chapter 47, "The GEE Procedure"
GENMOD	Chapter 48, "The GENMOD Procedure"
GLIMMIX	Chapter 49, "The GLIMMIX Procedure"
GLMSELECT	Chapter 53, "The GLMSELECT Procedure"
HPMIXED	Chapter 59, "The HPMIXED Procedure"
LOGISTIC	Chapter 76, "The LOGISTIC Procedure"
ORTHOREG	Chapter 88, "The ORTHOREG Procedure"
PHREG	Chapter 89, "The PHREG Procedure"
PLM	Chapter 91, "The PLM Procedure"
PROBIT	Chapter 97, "The PROBIT Procedure"
QUANTLIFE	Chapter 99, "The QUANTLIFE Procedure"
QUANTREG	Chapter 100, "The QUANTREG Procedure"
QUANTSELECT	Chapter 101, "The QUANTSELECT Procedure"
ROBUSTREG	Chapter 104, "The ROBUSTREG Procedure"
SURVEYLOGISTIC	Chapter 117, "The SURVEYLOGISTIC Procedure"
SURVEYREG	Chapter 120, "The SURVEYREG Procedure"

The following procedures support splines by means other than the EFFECT statement:

ADAPTIVEREG	Chapter 25, "The ADAPTIVEREG Procedure"
GAM	Chapter 45, "The GAM Procedure"
TPSPLINE	Chapter 122, "The TPSPLINE Procedure"
TRANSREG	Chapter 123, "The TRANSREG Procedure"

For more information, also see the following sections in Chapter 19, “Shared Concepts and Topics”:

“Parameterization of Model Effects” on page 393
“EFFECT Statement” on page 403
“Splines and Spline Bases” on page 417
“ODS Graphics: EFFECTPLOT Statement” on page 435

For more information, also see the following sections in Chapter 123, “The TRANSREG Procedure”:

“Using Splines and Knots” on page 10292
“Scoring Spline Variables” on page 10304
“Penalized B-Splines” on page 10319
“Smoothing Splines” on page 10322
“SPLINE and MSPLINE Transformations” on page 10359
“SPLINE, BSPLINE, and PSPLINE Comparisons” on page 10361

For more information, also see the conference papers and other resources: Harrell (1988); Cai (2008); Bilenas and Herat (2016); Croxford (2016); Wicklin (2017).

References

Bilenas, J. V., and Herat, N. (2016). “Using Regression Splines in SAS STAT Procedures.” In *Proceedings of SESUG 2016 Conference,* Paper BF-140. Cary, NC: SAS Institute Inc. `http://analytics.ncsu.edu/sesug/2016/BF-140_Final_PDF.pdf`.

Cai, W. (2008). “Fitting Generalized Additive Models with the GAM Procedure in SAS 9.2.” In *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. `http://www2.sas.com/proceedings/forum2008/378-2008.pdf`.

Cleveland, W. S., Devlin, S. J., and Grosse, E. (1988). “Regression by Local Fitting.” *Journal of Econometrics* 37:87–114.

Croxford, R. (2016). “Restricted Cubic Spline Regression: A Brief Introduction.” In *Proceedings of the SAS Global Forum 2016 Conference*. Cary, NC: SAS Institute Inc. `http://support.sas.com/resources/papers/proceedings16/5621-2016.pdf`.

De Boor, C. (1978). *A Practical Guide to Splines*. New York: Springer-Verlag.

Eilers, P. H. C., and Marx, B. D. (1996). “Flexible Smoothing with *B*-Splines and Penalties.” *Statistical Science* 11:89–121. With discussion.

Friedman, J. H. (1991). “Multivariate Adaptive Regression Splines.” *Annals of Statistics* 19:1–67.

Harrell, F. E. (1988). “SAS Macros for Assisting with Survival and Risk Analysis, and Some SAS Procedures Useful for Multivariable Modeling.” `http://biostat.mc.vanderbilt.edu/wiki/Main/SasMacros`.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer-Verlag.

Kaplan, E. L., and Meier, P. (1958). “Nonparametric Estimation from Incomplete Observations.” *Journal of the American Statistical Association* 53:457–481.

Kuhfeld, W. F. (2009). “Modifying ODS Statistical Graphics Templates in SAS 9.2.” `http://support.sas.com/rnd/app/papers/ods/modtmplt.pdf`. Revision of paper by the same title that was published in the *Proceedings of the SAS Global Forum 2009 Conference*.

Kuhfeld, W. F. (2015). *Advanced ODS Graphics Examples*. Cary, NC: SAS Institute Inc. `http://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsadvg.pdf`.

Kuhfeld, W. F. (2016). *Basic ODS Graphics Examples*. Cary, NC: SAS Institute Inc. `http://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsbasicg.pdf`.

Kuhfeld, W. F., and Cai, W. (2013). “Introducing the New ADAPTIVEREG Procedure for Adaptive Regression.” In *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc. `https://support.sas.com/resources/papers/proceedings13/457-2013.pdf`.

Reinsch, C. H. (1967). “Smoothing by Spline Functions.” *Numerische Mathematik* 10:177–183.

Smith, P. L. (1979). “Splines as a Useful and Convenient Statistical Tool.” *American Statistician* 33:57–62.

Wicklin, R. (2017). “Regression with Restricted Cubic Splines in SAS.” April. `https://blogs.sas.com/content/iml/2017/04/19/restricted-cubic-splines-sas.html`.

Winsberg, S., and Ramsay, J. O. (1980). “Monotonic Transformations to Additivity Using Splines.” *Biometrika* 67:669–674.

Subject Index

Syntax Index