SAS/STAT® 14.2 User's Guide
Using the Output Delivery System

This document is an individual chapter from *SAS/STAT® 14.2 User's Guide*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS/STAT® 14.2 User's Guide*. Cary, NC: SAS Institute Inc.

SAS/STAT® 14.2 User's Guide

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

November 2016

Chapter 20
Using the Output Delivery System

Contents

Overview: Using the Output Delivery System

Most SAS procedures use the Output Delivery System (ODS) to manage their output. ODS enables you to do the following:

- display your output in hypertext markup language (HTML), rich text format (RTF), portable document format (PDF), PostScript, SAS listing, or other formats
- create SAS data sets directly from tables or graphs
- select or exclude individual pieces of output
- customize the layout, format, headers, and style of your output
- produce graphs with ODS Graphics (see Chapter 21, "Statistical Graphics Using ODS")

This chapter discusses some typical applications of ODS with SAS software. For complete documentation about the Output Delivery System, see the *SAS Output Delivery System: User's Guide*. For more information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS," and Chapter 22, "ODS Graphics Template Modification."

Output Defaults

HTML output with ODS Graphics enabled is the default in the SAS windowing environment for Microsoft Windows and UNIX. LISTING output with ODS Graphics disabled is the default when you run SAS in batch mode or on the mainframe. Your actual defaults might be different due to your registry, system option, or configuration file settings. The following sections explain these defaults and how to change them.

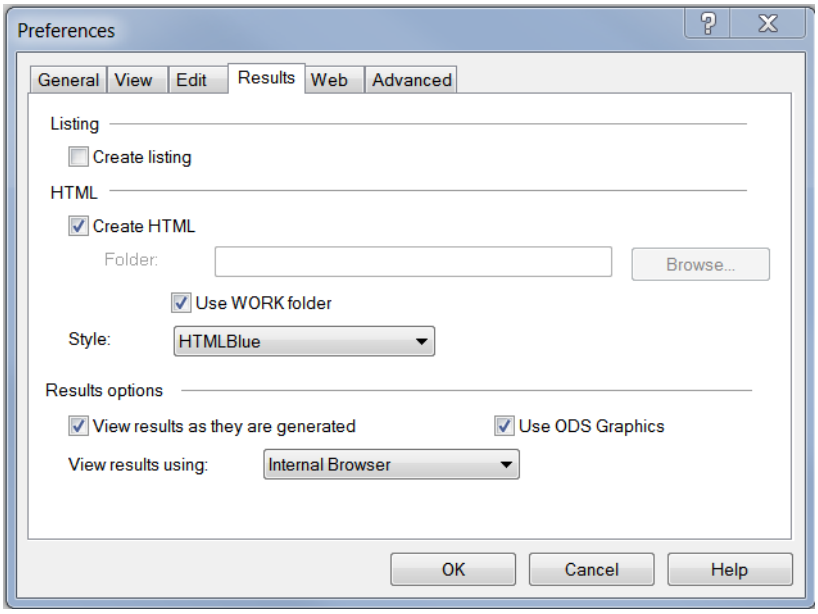
HTML Output in the SAS Windowing Environment

The default destination in the SAS windowing environment is HTML and ODS Graphics is enabled by default.[1] These defaults have several advantages. Graphs are integrated with tables, and all output is displayed in the same HTML file. The HTML destination uses the HTMLBLUE style, which is an all-color style, that is designed to integrate tables and modern statistical graphics.

You can view and modify the default settings by selecting **Tools ► Options ► Preferences** from the menu at the top of the main SAS window. Then click the **Results** tab. You can remember this sequence using the mnemonic TOPR (pronounced "topper"). See Figure 20.1.

[1] HTML output with ODS Graphics enabled is the default in the SAS windowing environment for Microsoft Windows and UNIX, but not on the mainframe.

Figure 20.1 SAS Results Tab with the Default Settings



The default settings are as follows:

- HTML output is created when **Create HTML** is selected, and all output is viewed in the Results Viewer window.
- ODS Graphics is enabled when **Use ODS Graphics** is selected.
- The default style, HTMLBLUE, is selected from the **Style** list.
- Results are viewed in an internal SAS browser when **Internal browser** is selected.
- Graph image files are saved in the Work folder (not in your current folder) when **Use WORK folder** is selected.
- LISTING output is not created when the **Create listing** box is cleared.

In many cases, graphs are an integral part of a data analysis. However, when you run large computational programs (such as when you use procedures with many BY groups), you might not want to create graphs. In those cases, you should disable ODS Graphics, which will improve the performance of your program. You can disable and re-enable ODS Graphics in your SAS programs with the ODS GRAPHICS OFF and ODS GRAPHICS ON statements. You can also change the ODS Graphics default in the **Results** tab.

In the SAS windowing environment, the current folder is displayed in the status line at the bottom of the main SAS window. When **Use WORK folder** is cleared, graph image files are saved in the current folder and are available after your SAS session ends. They can accumulate with time and take up a great deal of space. When **Use WORK folder** is selected, graph image files are stored in the Work folder and are not available after your SAS session ends.

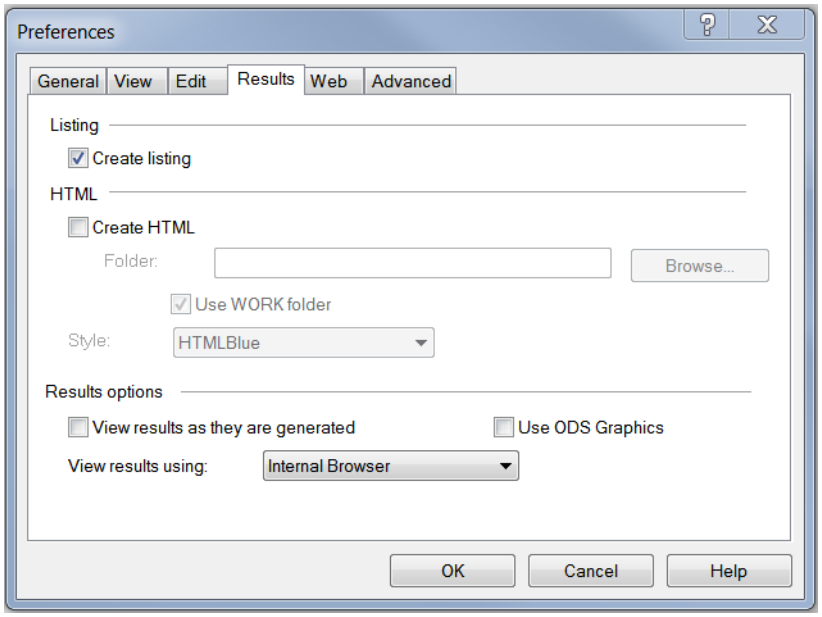
LISTING Output in the SAS Windowing Environment

In the LISTING destination, tables are displayed in monospace, and graphs are not integrated with tables. You can create LISTING output by selecting **Tools ► Options ► Preferences** from the menu at the top of the main SAS window. Then click the **Results** tab. Select **Create listing**, and clear **Create HTML**. See

Figure 20.2. Tabular results are viewed in the Output window. Graphical results are viewed by selecting graphs in the Results window.

You can enable or disable ODS Graphics by default by using the **Use ODS Graphics** check box, and you can use the ODS GRAPHICS ON and ODS GRAPHICS OFF statements to enable and disable ODS Graphics in your SAS programs.

Figure 20.2 SAS Results Tab for LISTING Output



Assumptions about ODS Defaults in This Chapter

Default ODS settings (such as open destinations, styles, and whether or not ODS Graphics is enabled) vary depending on your operating system, registry settings, configuration file settings, system options, and whether you are using the SAS windowing environment or batch mode. By default, SAS/STAT documentation has two forms: PDF and HTML. By default, output is displayed by using the PDF destination and the PEARLJ style (for PDF documentation) or by using the HTML destination and the HTMLBLUE style (for HTML documentation). In most examples, ODS destination statements are not displayed, and you can run the example code to create output for any destination. In contrast, this chapter shows you how to work with destinations and styles, so it sometimes opens and closes destinations. In this chapter, when the open destinations are closed, the PDF and HTML destinations are opened at the end of the step so that destinations are available for subsequent output.

The HTMLBLUE Style

In the SAS windowing environment, the default ODS style for HTML output is the HTMLBLUE style. You can see examples of the HTMLBLUE style in this chapter in Output 20.1.1, Output 20.8.2, and Output 20.8.3. The HTMLBLUE style inherits most of its attributes from the STATISTICAL style, but it has a brighter appearance and color coordination between the tables and graphs. In the HTMLBLUE style, the dominant color is blue; in the DEFAULT style, the dominant color is gray. For a comparison of the HTMLBLUE style and other styles, see Chapter 21, "Statistical Graphics Using ODS."

Default Open Destination

By default, either the LISTING or the HTML destination is open. You can change the default destination three ways, which are described in the next three subsections. The fourth subsection of this section explains how to use ODS statements to open and close destinations.

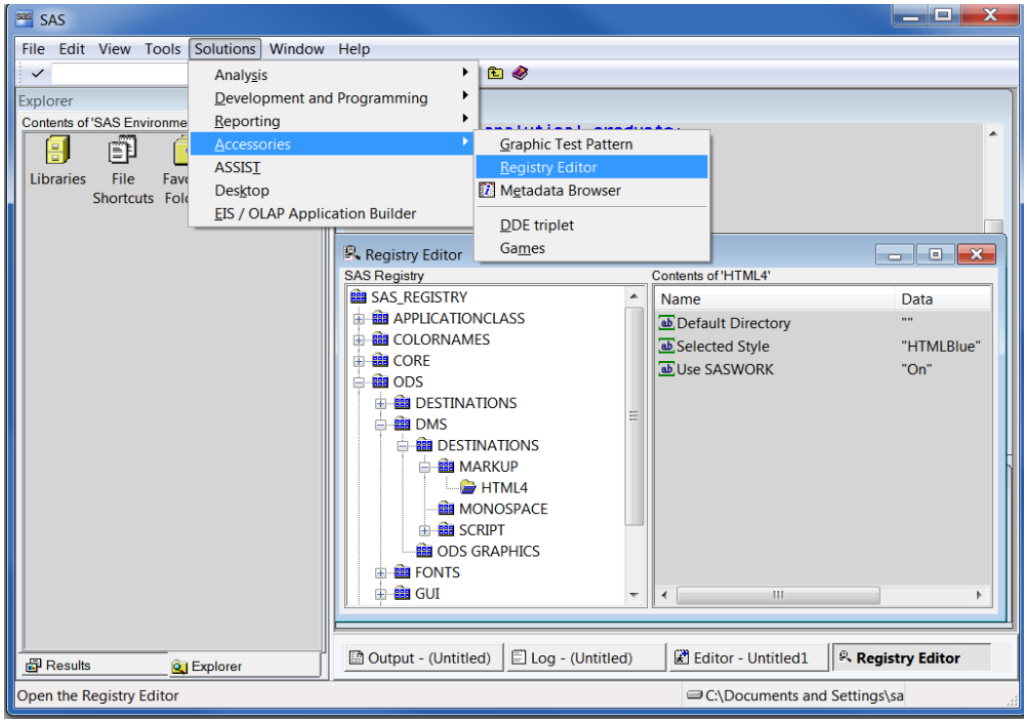
Setting the Default Destination in the Results Tab

You can change the default destination in the SAS windowing environment by selecting **Tools ► Options ► Preferences** from the menu at the top of the main SAS window. Then select the **Results** tab. See sections "HTML Output in the SAS Windowing Environment" on page 520 and "LISTING Output in the SAS Windowing Environment" on page 521. Changing defaults in the SAS windowing environment affects only the SAS windowing environment; it does not affect batch jobs.

Setting the Default Destination in the SAS Registry

You can change the default destination by editing the SAS registry (see Figure 20.3) or by changing system options in your SAS configuration file.

Figure 20.3 SAS Registry Window



Registry customization is generally performed by more advanced users who have experience and knowledge about the SAS System and their operating environment. Incorrect registry entries can corrupt your SAS registry. For more information about SAS configuration files and the SAS registry, see the *SAS Companion* for your operating system.

Setting the Default Destination in SAS System Options

The ODSDEST system option controls the default destination. This option is specified only at SAS start-up time. Other relevant system options that correspond to entries in the **Results** tab in Figure 20.1 include ODSGRAPHICS (which specifies whether ODS Graphics is enabled by default) and ODSSTYLE (which specifies the default style for the HTML destination in the SAS windowing environment). See the *SAS System Options: Reference* for more information.

Setting the Destination in ODS Statements

You can use ODS destination statements to explicitly set destinations. These statements are described in this chapter and in detail in the *SAS Output Delivery System: User's Guide*. When you open a new destination, you should close all other open destinations unless you really need multiple destinations to be open. When multiple destinations are open, each piece of output is created multiple times, once per destination. Closing unneeded destinations increases efficiency.

You can create HTML output in any environment by using the ODS HTML statement as in the following example:

```
ods _all_ close;
ods html file='MyFile.html';

proc reg data=sashelp.class;
   model height=weight;
run; quit;

ods html close;
```

The first statement closes all open destinations. The second statement opens the HTML destination and specifies the HTML output file *MyFile.html*. The last statement closes the HTML destination.

You can create LISTING output in any environment by using the ODS LISTING statement as in the following example:

```
ods _all_ close;
ods listing;

proc reg data=sashelp.class;
   model height=weight;
run; quit;
```

The first statement closes all open destinations. The second statement opens the LISTING destination which sends output to the SAS listing. In this example, the LISTING destination is not closed so that subsequent steps can append more information to the listing.

If the LISTING destination is open, then you can simultaneously create LISTING and HTML output as follows:

```
* The ODS LISTING destination is not closed,
  which is not recommended for efficiency reasons;

ods html file='Reg.htm';

proc reg data=sashelp.class;
   model height=weight;
run; quit;

ods html close;
```

Sometimes you see ODS Graphics notes or warnings multiple times when multiple destinations are open. The messages appear once for each affected graph for each destination.

Output Objects and ODS Destinations

All SAS procedures produce *output objects* that ODS delivers to various *ODS destinations*, according to the default specifications for the procedure or according to your own specifications. Typically, you see the output objects displayed as tables, data sets, or graphs. Underlying all output (for example, a table of parameter estimates) are two component parts:

- the data component, which consists of the results computed by a SAS procedure
- the template, which contains the instructions for formatting and displaying the results

Each output object has an associated template, provided by the SAS System, that defines its presentation format. You can use the TEMPLATE procedure to view or alter these templates or to create new templates by changing the headers, formats, column order, and so on. For more information, see the chapter titled "The Template Procedure" in the *SAS Output Delivery System: User's Guide*.

You define the form that the output should take by specifying an ODS destination. Some supported destinations are as follows:

- LISTING, the standard SAS monospace listing
- HTML, for viewing in a browser
- RTF, for inclusion in Microsoft Word
- PDF, PostScript, and PCL, for high-fidelity printers
- OUTPUT, for saving results to SAS data sets
- DOCUMENT, for saving, modifying, and replaying your output

You can open multiple ODS destinations at the same time so that a single procedure step can produce output for multiple destinations. If you do not supply any ODS statements, ODS delivers all output to the default destination (which is usually LISTING or HTML). See the section "Output Defaults" on page 520 for more information about default destinations. You can specify an output style for each ODS destination. The style controls the foreground, background, colors, lines, fonts, and so on.

The following statements provide an example of temporarily closing all open destinations for PROC REG and then opening the PDF and HTML destinations for PROC PRINT. PROC REG with the ODS OUTPUT statement makes an output data set, Parms, from the parameter estimates table. Closing unneeded open

destinations is not required, but it is done in many examples in this chapter for efficiency. Closing the superfluous destinations suppresses the generation of output that is not needed or used. This is particularly beneficial with graphics. This example uses the Sashelp.Class data set, one of the sample data sets in the Sashelp library that are automatically available for your use. The following statements produce Figure 20.4:

```
title 'Getting Started with ODS';

ods _all_ close;

proc reg data=sashelp.class;
   model height=weight;
   ods output ParameterEstimates=parms;
run; quit;

ods pdf;
ods html;

proc print noobs data=parms;
run;
```

The ODS OUTPUT statement contains an object name, an equal sign, and the name of the output SAS data set to create. You can use the ODS TRACE statement to find the object names. The ODS TRACE statement is described in the section "Paths and Selection" on page 527. Also see Example 20.4 for more information.

Figure 20.4 PROC REG Parameter Estimates Table

Getting Started with ODS

Model	Dependent	Variable	DF	Estimate	StdErr	tValue	Probt
MODEL1	Height	Intercept	1	42.57014	2.67989	15.89	<.0001
MODEL1	Height	Weight	1	0.19761	0.02616	7.55	<.0001

You could accomplish the same thing using ODS SELECT statements as follows:

```
ods select none;

proc reg data=sashelp.class;
   model height=weight;
   ods output ParameterEstimates=parms;
run; quit;

ods select all;

proc print noobs data=parms;
run;
```

You can specify ODS EXCLUDE ALL instead of ODS SELECT NONE and ODS EXCLUDE NONE instead of ODS SELECT ALL. These statements remain in effect until a new ODS SELECT or ODS EXCLUDE statement changes the selection list.

The ODS Statement

You use the ODS statement to provide instructions to ODS. You can use the ODS statement to specify options for different destinations, specify the output style, and select and exclude output. Here are some examples:

```
/* open the HTML destination with the HTMLBlue style */
ods html style=HTMLBlue;

/* select only the parameter estimates table */
ods select ParameterEstimates;

/* output the parameter estimates table to a SAS data set*/
ods output ParameterEstimates=Parms;

/* exclude the number of observations, ANOVA, and fit statistics tables */
ods exclude NObs ANOVA FitStatistics;
```

Paths and Selection

Each output object (tables, graphs, notes, and so on) produced by a SAS procedure has a name and a label. Each name is part of a name path. For example, PROC GLM has a table called ErrorSSCP, and the name path (fully qualified name) is `GLM.Repeated.MANOVA.Model.Error.ErrorSSCP`. Each level in the name path corresponds to a part of the PROC GLM hierarchy of output. Each piece of output also has a label and a label path. For example, the PROC GLM ErrorSSCP table is labeled `'SSCP Matrix'`, and the label path is `'The GLM Procedure'.'Repeated Measures Analysis'.'MANOVA'.'Model'.'Error'.'SSCP Matrix'`.

You need to know the name or label to select, exclude, or modify a table or graph. You can obtain this information in several ways:

- You can obtain names from the individual procedure documentation chapter or from the individual procedure section of the SAS online Help system. See the sections "ODS Table Names" and "ODS Graphics" from within the "Details" section of the procedure documentation chapter.
- You can use the SAS Results window to view the name of each piece of output that is created in your SAS session (see the section "The SAS Results Window" on page 531 for more information).
- You can use the ODS TRACE statement to find the name and label of each piece of output that is created in your SAS session. The ODS TRACE statement writes identifying information to the SAS log or listing for each generated output object.

If you are working interactively with reasonably small data sets, then the ODS TRACE statement is usually the most convenient way to find the names. Specify the ODS TRACE ON statement prior to the procedure statements that create the output for which you want information. For example, the following statements write the trace record for the output created in the REG procedure step:

```
ods trace on;
ods graphics on;
proc reg data=sashelp.class;
   model weight=height;
   model age=height;
run; quit;
ods trace off;
```

By default, the trace output is written to the SAS log. Some of the trace output from the previous step is as follows:

```
Output Added:
-------------
Name:        NObs
Label:       Number of Observations
Template:    Stat.Reg.NObs
Path:        Reg.MODEL1.Fit.Weight.NObs
-------------

Output Added:
-------------
Name:        ANOVA
Label:       Analysis of Variance
Template:    Stat.REG.ANOVA
Path:        Reg.MODEL1.Fit.Weight.ANOVA
-------------

Output Added:
-------------
Name:        FitStatistics
Label:       Fit Statistics
Template:    Stat.REG.FitStatistics
Path:        Reg.MODEL1.Fit.Weight.FitStatistics
-------------

Output Added:
-------------
Name:        ParameterEstimates
Label:       Parameter Estimates
Template:    Stat.REG.ParameterEstimates
Path:        Reg.MODEL1.Fit.Weight.ParameterEstimates
-------------

Output Added:
-------------
Name:        DiagnosticsPanel
Label:       Fit Diagnostics
Template:    Stat.REG.Graphics.DiagnosticsPanel
Path:        Reg.MODEL1.ObswiseStats.Weight.DiagnosticPlots.DiagnosticsPanel
-------------
```

```
Output Added:
-------------
Name:        ResidualPlot
Label:       Height
Template:    Stat.REG.Graphics.ResidualPlot
Path:        Reg.MODEL1.ObswiseStats.Weight.ResidualPlots.ResidualPlot
-------------

Output Added:
-------------
Name:        FitPlot
Label:       Fit Plot
Template:    Stat.REG.Graphics.Fit
Path:        Reg.MODEL1.ObswiseStats.Weight.FitPlot
-------------

Output Added:
-------------
Name:        NObs
Label:       Number of Observations
Template:    Stat.Reg.NObs
Path:        Reg.MODEL2.Fit.Age.NObs
-------------

.
.
.

Output Added:
-------------
Name:        FitPlot
Label:       Fit Plot
Template:    Stat.REG.Graphics.Fit
Path:        Reg.MODEL2.ObswiseStats.Age.FitPlot
-------------
```

Alternatively, you can specify the LISTING option (`ods trace on / listing;`), which writes the trace record, interleaved with the procedure output, to the LISTING destination (if it is open).

The trace record contains the name of each output object created and its associated label, template, and fully qualified name path. The label provides a description of the table or graph. The fully qualified name path shows the output hierarchy. (An example of the hierarchy is shown in Figure 20.5. The SAS Results window displays the labels, rather than the names of objects, but the hierarchy is the same for both names and labels.) The hierarchy has a level for the REG procedure, a level for the model (MODEL1 or MODEL2), a level for the fit results, a level for the dependent variable (Weight or Age), and a level for the name (for example, NObs, ANOVA, FitStatistics, ParameterEstimates).

When you specify ODS objects in an ODS statement, you can often omit the first few levels and instead use a partially qualified name path. A partially qualified name path consists of any part of the fully qualified name path that begins immediately after a period and continues to the end of the fully qualified name path. For example, the table **`Reg.Model1.Fit.Weight.ParameterEstimates`** can be referenced in any of the following ways:

ParameterEstimates	name
Weight.ParameterEstimates	partially qualified name path
Fit.Weight.ParameterEstimates	partially qualified name path
Model1.Fit.Weight.ParameterEstimates	partially qualified name path
Reg.Model1.Fit.Weight.ParameterEstimates	fully qualified name path

When a procedure creates multiple output objects that have the same name, as shown in the preceding trace output, you have several selection options for referring to the object. You can specify the name, a fully qualified name path, or a partially qualified name path in ODS statements such as ODS SELECT, ODS EXCLUDE, or ODS OUTPUT. You can also specify a WHERE clause. For example, you can specify any of the following statements (in addition to other possibilities) to display both tables of parameter estimates:

```
ods select ParameterEstimates;

ods select Weight.ParameterEstimates Age.ParameterEstimates;

ods select Reg.Model1.Fit.Weight.ParameterEstimates
           Reg.Model2.Fit.Age.ParameterEstimates;

ods select where = (_path_ ? 'Parameter');
```

The first ODS SELECT statement specifies the object name, which is shared by both tables. The second statement specifies a partially qualified name path for both tables. The third statement specifies the fully qualified name path for each table. The fourth statement selects every object that contains the string **'Parameter'** anywhere in its path.

In the first three statements, selection is case insensitive. Any combination of uppercase and lowercase letters works. This is not true in the fourth statement, which uses an ordinary SAS comparison of character strings. For case insensitivity in WHERE clause selection, use the LOWCASE function as in the following example:

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

You can also select objects based on a WHERE clause and the label path. The following statements turn on the trace record, display a label path in addition to the name path, and select all objects that have the string **'var'** in the label:

```
ods trace on / label;
ods select where = (lowcase(_label_) ? 'var');
```

A subset of the trace record for PROC REG with this ODS SELECT list, showing just the name path and label path, is as follows:

```
Path:       Reg.MODEL1.Fit.Weight.ANOVA
Label Path: 'The Reg Procedure'.'MODEL1'.'Fit'.Weight.'Analysis of Variance'
Path:       Reg.MODEL2.Fit.Age.ANOVA
Label Path: 'The Reg Procedure'.'MODEL2'.'Fit'.Age.'Analysis of Variance'
```

The ODS SELECT statement selects the ANOVA tables, because they have the string **'Analysis of Variance'** (which when lowercased contains **'var'**) in their labels. WHERE clause selection is also useful for selecting all of the objects within a group or level of the path hierarchy (the group **'MODEL2'** or **'Fit'**). You can specify any part of the name path or label path—for example, **'.Age.'** matches the variable Age and ignores any **'Age'** that might be in the middle of a word, **'2.F'** matches Model 2 fit tables and any other object that has the string **'2.F'** in its path, and so on.

ODS records the specified object names in its internal selection or exclusion list, and then it processes the output it receives. ODS maintains an overall selection or exclusion list that pertains to all ODS destinations, and it maintains a separate selection or exclusion list for each ODS destination. The list for a specific destination provides the primary filtering step. The restrictions that you specify in the overall list are added to the destination-specific lists.

Suppose, for example, that your LISTING exclusion list (that is, the list of objects you want to exclude from the LISTING destination) contains the FitStatistics table, which you specify with the following statement:

```
ods listing exclude FitStatistics;
```

Suppose also that your overall selection list (that is, the list of objects you want to select for all destinations) contains the tables ParameterEstimates and FitStatistics, which you specify with the following statement:

```
ods select ParameterEstimates FitStatistics;
```

ODS then sends only the ParameterEstimates and FitStatistics tables to all open destinations except the LISTING destination. It sends only the ParameterEstimates table to the LISTING destination because the table FitStatistics is excluded from that destination.

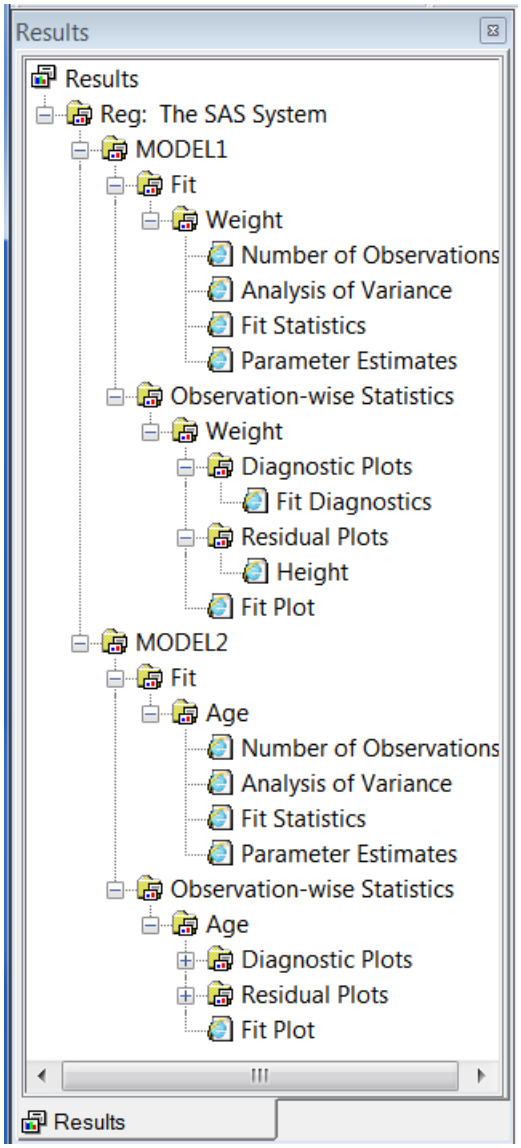
RUN-Group Processing

Some SAS procedures, such as PROC REG and PROC GLM, support RUN-group processing, which means that a RUN statement does not end the procedure. A QUIT statement explicitly ends such procedures. If you omit the QUIT statement, a PROC or a DATA statement implicitly ends such procedures. When you use ODS with procedures that support RUN-group processing, it is good programming practice to specify a QUIT statement at the end of the procedure. This causes ODS to clear the selection or exclusion list, and you are less likely to encounter unexpected results. See Example 20.6 for more information about RUN-group processing with interactive procedures.

The SAS Results Window

The SAS Results window contains a running record of the output from your SAS session. In the SAS windowing environment, select **View ► Results** to open the Results window. Figure 20.5 displays the Results window from the PROC REG step shown previously.

Figure 20.5 The Results Window from the SAS Explorer



When you click the output names in the Results window, you link directly to the output in the Output Results window (for the HTML destination) or the Output window or graph viewer window (for the LISTING destination). The Results window contains an entry for each level of the label path and for each object. You can also use the Results window to determine the names of the templates that are associated with each object. Right-click the name and select **Properties**. You can see all the templates from the Results window by selecting **View ► Templates**. For SAS/STAT, select **Sashelp.Tmplstat**. Then select a procedure such as **REG** and a template such as **ParameterEstimates**.

The ODS PATH Statement

The ODS PATH statement controls where ODS stores new templates that you create and where ODS finds the templates that your programs use.[2] 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.)

[2] Other types of paths include the name path and label path, which are discussed in the section "Paths and Selection" on page 527.

By default, the templates that you write are stored in Sasuser.Templat, and the templates that the SAS System provides are stored in Sashelp.Tmplmst and other Sashelp template stores. By default, ODS retrieves templates from Sashelp template stores unless you compile and store copies in Sasuser.Templat.

You can see the list of active template stores by submitting the following statement:

```
ods path show;
```

By default, the results are as follows:

```
Current ODS PATH list is:

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

You can see a list of all of the templates in a template store as follows:

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

See the sections "The Master Template Store" on page 533 and "Controlling Output Appearance with Templates" on page 534 for more information about the template search path and template stores.

The Master Template Store

By default, the ODS path includes the name of the template store that provides the templates that are shipped with the SAS system. You can use the ODS PATH SHOW statement to see the list of active template stores:

```
ods path show;
```

By default, the results are as follows:

```
Current ODS PATH list is:

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

The ODS PATH statement template search path name Sashelp.Tmplmst refers to the template stores that are shipped with the SAS System. More precisely, the ODS PATH name Sashelp.Tmplmst refers to multiple template store files including Sashelp.Tmplmst, Sashelp.Tmplstat, Sashelp.Tmplets, Sashelp.Tmplqc, Sashelp.Tmpliml, and others. The name Sashelp.Tmplmst refers to both the entire template store that is shipped with the SAS System and one particular file in that template store. In earlier releases of the SAS System, there was just one template store file, namely Sashelp.Tmplmst, and there was a one-to-one correspondence between the ODS PATH statement name and the template store file name. Now there are multiple files (because certain products have their own template store files), but the ODS PATH statement syntax for selecting all of them is unchanged. You do not need to be concerned about this, and you should not specify any of these template stores individually. You simply specify Sashelp.Tmplmst to get the item stores that are shipped with the SAS System. However, you will see these other names sometimes in the SAS log and output when you are working with templates, so you need to know what that means.

If you see a template store name of the form Sashelp followed by a period and a name, it refers to part of the overall template store that you can reference with the name Sashelp.Tmplmst. For example, submit the following step:

```
proc template;
   source Stat.REG.ANOVA;
run;
```

The following is displayed in the SAS log file:

```
NOTE: Path 'Stat.Reg.ANOVA' is in: SASHELP.TMPLSTAT (via SASHELP.TMPLMST).
```

Submit the following step:

```
proc template;
   list Stat.REG;
   list ETS.ARIMA;
   list QC.Shewhart;
run;
```

The names Sashelp.Tmplstat, Sashelp.Tmplets, and Sashelp.Tmplqc are displayed in the headers of the tables that list the procedure templates.

Controlling Output Appearance with Templates

A template is a description of how output should appear when it is formatted. Templates describe several characteristics of the output, including headers, column ordering, style information, justification, and formats. Each object in the output has a template, and all SAS templates are stored in the Sashelp library. You can find the template associated with a particular output object or table column by using the ODS TRACE statement or the SAS Results window. You can create or modify a template with the TEMPLATE procedure. For example, you can specify different column headings or different orders of columns in a table.

There are a number of different types of templates including column and table templates, graphical templates, and style templates. A column or table template applies to the specific columns or tables that refer to the template. Graphical templates are discussed in more detail in Chapter 21, "Statistical Graphics Using ODS." A style template applies to an entire SAS program, including all tables and graphs, and can be specified with the STYLE= option in a valid ODS destination, such as HTML, RTF, or PDF. You can specify a style as follows:

```
ods html style=HTMLBlue;
```

A style template controls stylistic elements such as colors, fonts, and presentation attributes. You can change the style to give your output different looks and color schemes. You can also refer to style information in table templates for individual headers and data cells. You can modify all types of templates with PROC TEMPLATE. For information about creating your own styles, see the *SAS Output Delivery System: User's Guide*.

You can display the contents of a template by running PROC TEMPLATE with a SOURCE statement and a template name, as in the following example:

```
proc template;
   source Stat.REG.ANOVA;
   source Stat.GLM.OverallANOVA;
run;
```

In many cases, a template definition is based at least in part on another template. When you see the PARENT=*template* option in a template definition, you need to look at the specified template to learn more about the rest of the template definition. To illustrate, consider the following PROC GLM step:

```
proc glm data=sashelp.class;
   model height=weight;
run; quit;
```

The ANOVA table from this step is displayed in Figure 20.6.

Figure 20.6 PROC GLM ANOVA Table with the Default Template

Getting Started with ODS

The GLM Procedure

Dependent Variable: Height

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	364.5762619	364.5762619	57.08	<.0001
Error	17	108.5879486	6.3875264		
Corrected Total	18	473.1642105			

The sums of squares and mean squares are presented with eight decimal places. You can change the templates to change the formats of those columns to use fewer decimal places. First, you can use the ODS TRACE statement when you run PROC GLM to determine the name of the template:

```
ods trace output;
proc glm data=sashelp.class;
   model height=weight;
run; quit;
ods trace off;
```

The trace output results include the following:

```
Output Added:
-------------
Name:        OverallANOVA
Label:       Overall ANOVA
Template:    stat.GLM.OverallANOVA
Path:        GLM.ANOVA.Height.OverallANOVA
-------------
```

From this, you can see that the template for the overall ANOVA table is **`stat.GLM.OverallANOVA`**. You can submit the following statements to see the overall ANOVA table template:

```
proc template;
   source stat.glm.overallanova;
run;
```

The results are as follows:

```
define table Stat.GLM.Overallanova;
   notes "Over-all ANOVA";
   top_space = 1;
   parent = Stat.GLM.ANOVA;
   double_space;
end;
```

The results show that this template inherits its definition from a parent template named **Stat.GLM.ANOVA**. Submit the following statements to see the parent template:

```
proc template;
   source stat.glm.anova;
run;
```

Some of the results are as follows:

```
define SS;
   parent = Stat.GLM.SS;
end;

define MS;
   parent = Stat.GLM.MS;
end;
```

These columns inherit their definitions from the parent columns named **Stat.GLM.SS** and **Stat.GLM.MS**. This is all of the information that you need to redefine these columns, but you can run PROC TEMPLATE again as follows to see more information about how these templates are defined:

```
proc template;
   source Stat.GLM.SS;
   source Stat.GLM.MS;
run;
```

The results are as follows:

```
define column Stat.GLM.Ss;
   notes "Parent for GLM ANOVA Sums of Squares columns";
   parent = Common.ANOVA.SS;
end;
define column Stat.GLM.Ms;
   notes "Parent for GLM ANOVA Mean Squares columns";
   parent = Common.ANOVA.MS;
end;
```

These columns inherit their definitions from the columns named **Common.ANOVA.SS** and **Common.ANOVA.MS**. You can run PROC TEMPLATE as follows to see their definitions:

```
proc template;
   source Common.ANOVA.SS;
   source Common.ANOVA.MS;
run;
```

The results are as follows:

```
define column Common.ANOVA.Ss;
   notes "Default ANOVA Sum of squares column";
   header = "Sum of Squares";
```

```
      translate _val_=._ into "";
   end;
   define column Common.ANOVA.Ms;
      notes "Default ANOVA Mean square column";
      header = "Mean Square";
      translate _val_=._ into "";
   end;
```

You can redefine **Common.ANOVA.SS** and **Common.ANOVA.MS** to change all **SS** and **MS** columns in ANOVA tables. This would be the most general redefinition. More specifically, you can redefine **Stat.GLM.SS** and **Stat.GLM.MS** to change **SS** and **MS** columns in ANOVA tables produced by PROC GLM. Finally, and most specifically, you can change the **SS** and **MS** columns in just the overall ANOVA table template.

In this example, the **Stat.GLM.SS** and **Stat.GLM.MS** columns are redefined as follows, so that results are displayed with fewer decimal places:

```
proc template;
   edit Stat.GLM.SS;
      choose_format=max format_width=8;
   end;
   edit Stat.GLM.MS;
      choose_format=max format_width=8;
   end;
run;
```

The CHOOSE_FORMAT=MAX option along with the FORMAT_WIDTH=8 option chooses the format for each column based on the maximum value in that column and an overall width of eight. You are editing and not replacing the definition, so the column header and other information in the definition is not lost. The following step uses the new templates:

```
proc glm data=sashelp.class;
   model height=weight;
run; quit;
```

The new ANOVA results, using the edited templates, are shown in Figure 20.7. You can see that the original results in Figure 20.6 have eight decimal places, whereas the new results in Figure 20.7 have only five decimal places and an overall format width of eight.

Figure 20.7 PROC GLM ANOVA Table after Template Customization

Getting Started with ODS

The GLM Procedure

Dependent Variable: Height

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	364.5763	364.5763	57.08	<.0001
Error	17	108.5879	6.3875		
Corrected Total	18	473.1642			

The preceding PROC TEMPLATE step produces the following notes:

```
NOTE: Overwriting existing template/link: Stat.GLM.Ss
NOTE: COLUMN 'Stat.GLM.Ss' has been saved to: SASUSER.TEMPLAT
NOTE: Overwriting existing template/link: Stat.GLM.Ms
NOTE: COLUMN 'Stat.GLM.Ms' has been saved to: SASUSER.TEMPLAT
```

When you run PROC TEMPLATE to modify or edit a template, the template is stored by default in your Sasuser library. You can delete your custom template and restore the default template as follows:

```
proc template;
   delete Stat.GLM.SS / store=sasuser.templat;
   delete Stat.GLM.MS / store=sasuser.templat;
run;
```

The preceding PROC TEMPLATE step produces the following notes:

```
NOTE: 'Stat.GLM.SS' has been deleted from: SASUSER.TEMPLAT
NOTE: 'Stat.GLM.MS' has been deleted from: SASUSER.TEMPLAT
```

It is good practice to delete any template redefinitions that you do not want to be permanent, because otherwise they persist beyond the duration of your SAS session. The option 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.

You can modify the template search path with the ODS PATH statement—for example, so you can access these new templates in a later SAS session. This enables you to create a new default set of templates to modify the display format for all of your SAS output. You can specify the SHOW option in the ODS PATH statement to determine the current template search path. The following statements illustrate the template search path:

```
ods path show;
libname mytpls '.';
ods path (prepend) mytpls.template(update);
ods path show;

proc template;
   edit Stat.GLM.SS;
      choose_format=max format_width=8;
   end;
   edit Stat.GLM.MS;
      choose_format=max format_width=8;
   end;
run;
```

The results of the first statement are as follows:

```
Current ODS PATH list is:

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

This shows that the Sasuser.Templat template store is open for storing new templates and retrieving templates for use. After that, the Sashelp.Tmplmst template store is used, but it is open only for read access.[3] The LIBNAME and second ODS PATH statements add a template store to the front of this list in the current directory. The final ODS PATH SHOW statement shows the new template search path, which is as follows:

```
Current ODS PATH list is:

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

The PROC TEMPLATE step produces the following notes, which show that the templates are now stored in MYTPLS.TEMPLATE:

```
NOTE: Overwriting existing template/link: Stat.GLM.Ss
NOTE: COLUMN 'Stat.GLM.Ss' has been saved to: MYTPLS.TEMPLATE
NOTE: Overwriting existing template/link: Stat.GLM.Ms
NOTE: COLUMN 'Stat.GLM.Ms' has been saved to: MYTPLS.TEMPLATE
```

In all cases, the original template definitions in Sashelp.Tmplmst are not changed. You can delete your custom template and restore the default template as follows:

```
proc template;
   delete Stat.GLM.SS / store=mytpls.template;
   delete Stat.GLM.MS / store=mytpls.template;
run;
```

If you would like all template modifications to be automatically deleted at the end or your SAS session, you can modify the template search path so that an updatable template store is placed in the Work directory in the front of the current path in either of the two equivalent ways:

```
ods path (prepend) work.templat(update);
ods path work.templat(update) sasuser.templat(update) sashelp.tmplmst(read);
```

Alternatively, you can replace Sasuser.Templat with Sashelp.Templat as follows:

```
ods path work.templat(update) sashelp.tmplmst(read);
```

When you are done, you can reset the default template search path as follows:

```
ods path reset;
```

ODS and the NOPRINT Option

Many SAS procedures support a NOPRINT option that you can use when you want to create an output data set without displaying any output. You use an option (such as the OUTEST= option or an OUTPUT statement with an OUT= option) in addition to the procedure's NOPRINT option to create a data set and suppress displayed output.

[3] Most SAS users cannot modify templates in Sashelp. However, if you have computer administrator privileges, you might be able to modify templates in Sashelp, so you should be careful to not do so.

You can also use the ODS OUTPUT statement to create output data sets. However, if you specify the NOPRINT option, the procedure might not send any output to ODS. In most procedures that support a NOPRINT option, NOPRINT means no ODS. (However, there are a few procedures that for historical reasons still might produce some output even when NOPRINT is specified.) When you want to create output data sets through the ODS OUTPUT statement and you want to suppress the display of all output, specify the following statement instead of using the NOPRINT option:

```
ods select none;
```

Alternatively, you can close the active ODS destinations like this:

```
ods _all_ close;
```

ODS statements do not instruct a procedure to generate output. Instead, they specify how ODS should manage output after it is created. You must ensure that the proper procedure options are in effect, or the output is not generated. For example, the following statements do not create the requested data set Parms because the SOLUTION option is not specified in the MODEL statement:

```
proc glm data=sashelp.class;
   ods output ParameterEstimates=Parms;
   class sex;
   model height=sex;
run; quit;
```

Since PROC GLM did not create the table, ODS cannot make the output data set. When you execute these statements, the following message is displayed in the log:

```
WARNING: Output 'ParameterEstimates' was not created.
```

The following step creates the output data set:

```
proc glm data=sashelp.class;
   ods output ParameterEstimates=Parms;
   class sex;
   model height=sex / solution;
run; quit;
```

Examples: Using the Output Delivery System

This section provides examples of creating HTML output, selecting and excluding output, tracing ODS output, using the Results window, creating ODS output data sets, modifying templates, creating hyperlinks, and using ODS Graphics.

Example 20.1: Creating HTML Output with ODS

This example demonstrates how you can use the ODS HTML statement to display your output in HTML. The following statements create the data set Scores, which contains the golf scores of boys and girls in a physical education class:

```
title 'Comparing Group Means';

data Scores;
   input Gender $ Score @@;
   datalines;
f 75  f 76  f 80  f 77  f 80  f 77  f 73
m 82  m 80  m 85  m 85  m 78  m 87  m 82
;
```

The TTEST procedure is used to compare the scores. The ODS HTML statement specifies the name of the file to contain the HTML output. The following statements create the HTML file *ttest.htm*:

```
ods html body='ttest.htm' style=HTMLBlue;

proc ttest;
   class Gender;
   var Score;
run;

ods html close;
```

In many cases, the LISTING destination is open by default. See the section "Output Defaults" on page 520 for more information about default destinations. When the LISTING destination is open, the LISTING destination receives all output generated during your SAS session. In this example, the ODS HTML statement also opens the HTML destination, and both destinations receive the generated output. If you are in the SAS windowing environment and are using the internal browser, you do not need to close the HTML destination before viewing your output. However, when you write to an HTML file, you must specify the following statement before you can view your output in an external browser:

```
ods html close;
```

If you do not close the HTML destination, your HTML file might contain no output or incomplete output, or you might experience other unexpected results.

The following statements use ODS to display the output in HTML with a table of contents:

```
ods _all_ close;
ods html body='ttest.htm' contents='ttestc.htm' frame='ttestf.htm'
         style=HTMLBlue;
ods graphics on;

proc ttest;
   class Gender;
   var Score;
run;

ods html close;
ods html;
ods pdf;
```

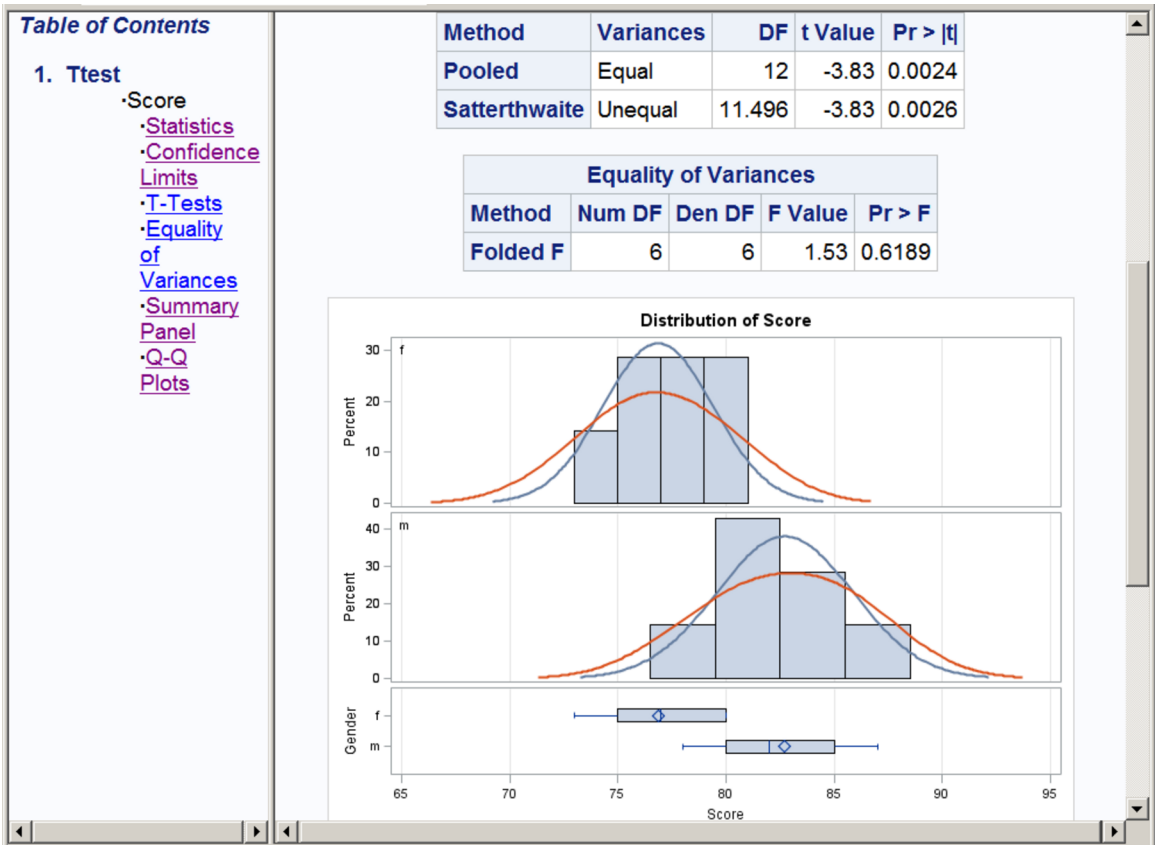
The ODS _ALL_ CLOSE statement closes all open destinations. The ODS HTML statement specifies three files and the HTMLBLUE style of output. The BODY= option specifies the file that contains the SAS output. The CONTENTS= option specifies the file that contains the table of contents. The FRAME= option specifies the file that displays both the table of contents and the output. You can open the FRAME= file (*ttestf.htm*) in

your browser to view the table of contents together with the generated output (see Output 20.1.1). By default, the HTML files are generated in your current working directory. You can instead specify a path, such as `frame='html/ttestf.htm'`, to store a file in a subdirectory.

If you specify the ODS HTML statement with only the BODY= argument, no table of contents is created. The table of contents contains the descriptive label for each output object produced in the PROC TTEST step. You can select any label in the table of contents, and the corresponding output is displayed on the right side of the browser window.

The ODS GRAPHICS ON statement enables ODS Graphics, which creates the graph displayed in Output 20.1.1. For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS."

Output 20.1.1 HTML Output with a Table of Contents and a Frame



Example 20.2: Selecting ODS Tables for Display

You can use the ODS SELECT statement to deliver only a subset of the tables or graphs to ODS destinations. The following statements create an input SAS data set and use PROC GLM to perform an analysis of an unbalanced two-way experimental design:

```
title 'Unbalanced Two-way Design';
data twoway;
   input Treatment Block y @@;
   datalines;
```

```
1 1 17   1 1 28   1 1 19   1 1 21   1 1 19   1 2 43
1 2 30   1 2 39   1 2 44   1 2 44   1 3 16
2 1 21   2 1 21   2 1 24   2 1 25   2 2 39   2 2 45
2 2 42   2 2 47   2 3 19   2 3 22   2 3 16
3 1 22   3 1 30   3 1 33   3 1 31   3 2 46   3 3 26
3 3 31   3 3 26   3 3 33   3 3 29   3 3 25
;

proc glm data=twoway;
   class Treatment Block;
   model y = Treatment | Block;
   means Treatment;
   lsmeans Treatment;
   ods select ModelANOVA Means;
   ods trace on;
   ods show;
run;
```

The ODS SELECT statement selects only two tables (ModelANOVA and Means) for display in the ODS destinations. In this example, no ODS destinations are explicitly opened. Therefore, only the default destination (usually LISTING or HTML) receives the procedure output. See the section "Output Defaults" on page 520 for more information about default destinations. The ODS SHOW statement displays the current overall selection list in the SAS log. The ODS SHOW statement is not required; it is used here simply to show the effects of the ODS SELECT statement. The results of the ODS SHOW statement are as follows:

```
Current OVERALL select list is:
1. ModelANOVA
2. Means
```

The ODS TRACE statement writes the trace record of the ODS output objects to the SAS log. The trace record is as follows:

```
Output Added:
-------------
Name:        ModelANOVA
Label:       Type I Model ANOVA
Template:    stat.GLM.Tests
Path:        GLM.ANOVA.y.ModelANOVA
-------------

Output Added:
-------------
Name:        ModelANOVA
Label:       Type III Model ANOVA
Template:    stat.GLM.Tests
Path:        GLM.ANOVA.y.ModelANOVA
-------------

Output Added:
-------------
Name:        Means
Label:       Means
Template:    stat.GLM.Means
Path:        GLM.Means.Treatment.Means
-------------
```

There are two tables with the name ModelANOVA. One contains the "Type I Model ANOVA" table, and the other contains the "Type III Model ANOVA" table. If you want to select only one of them, you can specify either of the labels in the ODS SELECT statement instead of the name. You specify one of the following:

```
ods select 'Type I Model ANOVA' Means;
ods select 'Type III Model ANOVA' Means;
```

In the following statements, the ODS SHOW statement writes the current overall selection list to the SAS log, the QUIT statement ends the PROC GLM step, and the second ODS SHOW statement writes the selection list to the log after PROC GLM terminates:

```
ods show;
quit;
ods show;
```

The results of these statements are as follows:

```
ods show;

 Current OVERALL select list is:
 1. ModelANOVA
 2. Means


quit;
ods show;

 Current OVERALL select list is: ALL
```

PROC GLM supports interactive RUN-group processing. Before the QUIT statement is executed, PROC GLM is active and the ODS selection list remains at its previous setting. The list includes only the two tables, ModelANOVA and Means. After the QUIT statement, when PROC GLM is no longer active, the selection list is reset to ALL. The displayed output, shown in Output 20.2.1, consists of the three selected tables (two ModelANOVA tables and the Means table). The LS-means results are not displayed even though an LSMEANS statement was specified. This is because the LS-means table, named LSMeans, is not specified in the ODS SELECT statement. Other tables are suppressed as well.

Output 20.2.1 Selected Tables from PROC GLM

Unbalanced Two-way Design

The GLM Procedure

Dependent Variable: y

Source	DF	Type I SS	Mean Square	F Value	Pr > F
Treatment	2	8.060606	4.030303	0.24	0.7888
Block	2	2621.864124	1310.932062	77.95	<.0001
Treatment*Block	4	32.684361	8.171090	0.49	0.7460

Output 20.2.1 *continued*

Source	DF	Type III SS	Mean Square	F Value	Pr > F
Treatment	2	266.130682	133.065341	7.91	0.0023
Block	2	1883.729465	941.864732	56.00	<.0001
Treatment*Block	4	32.684361	8.171090	0.49	0.7460

Unbalanced Two-way Design

The GLM Procedure

		y	
Level of Treatment	N	Mean	Std Dev
1	11	29.0909091	11.5104695
2	11	29.1818182	11.5569735
3	11	30.1818182	6.3058414

For more information about ODS exclusion and selection lists, see the section "The ODS Statement" on page 527.

Example 20.3: Excluding ODS Tables from Display

The following example demonstrates how you can use the ODS EXCLUDE statement to exclude particular objects from ODS destinations. This example also creates a SAS data set from the excluded table and uses it to create a specialized plot.

The data are from Hemmerle and Hartley (1973). The response variable consists of measurements from an oven experiment, and the model contains a fixed effect a and random effects b and a*b. The following statements create the input SAS data set:

```
title 'Oven Measurements';

data hh;
   input a b y @@;
   datalines;
1 1 237   1 1 254    1 1 246
1 2 178   1 2 179
2 1 208   2 1 178    2 1 187
2 2 146   2 2 145    2 2 141
3 1 186   3 1 183
3 2 142   3 2 125    3 2 136
;
```

The following ODS statements are submitted before the analysis, which will be done with the MIXED procedure:

```
ods _all_ close;
ods html body='mixed.htm' contents='mixedc.htm' frame='mixedf.htm'
         style=HTMLBlue;
```

```
ods exclude ParmSearch(persist);
ods show;
```

The ODS HTML statement specifies the filenames to contain the output generated from the statements that follow. The ODS EXCLUDE statement excludes the table ParmSearch from display. Although the table is excluded from the displayed output, the information contained in the ParmSearch table is graphically summarized in a later step.

The PERSIST option in the ODS EXCLUDE statement excludes the object for the entire SAS session or until you execute an ODS SELECT statement or an ODS EXCLUDE NONE statement. If you omit the PERSIST option, the exclusion list is cleared when the procedure terminates. The resulting exclusion list is displayed next:

```
Current OVERALL exclude list is:
1. ParmSearch(PERSIST)
```

The MIXED procedure is run to fit the model:

```
proc mixed data=hh;
   class a b;
   model y = a;
   random b a*b;
   parms (17 to 20 by 0.1) (.3 to .4 by .005) (1.0);
   ods output ParmSearch=parms;
run;

ods show;
```

All output from PROC MIXED, except the ParmSearch table, is delivered to the HTML destination. The ODS OUTPUT statement outputs the table ParmSearch to a SAS data set called Parms.

The ODS SHOW statement again displays the overall current exclusion list after PROC MIXED has terminated. The results of the ODS SHOW statement are displayed next:

```
Current OVERALL exclude list is:
1. ParmSearch(PERSIST)
```

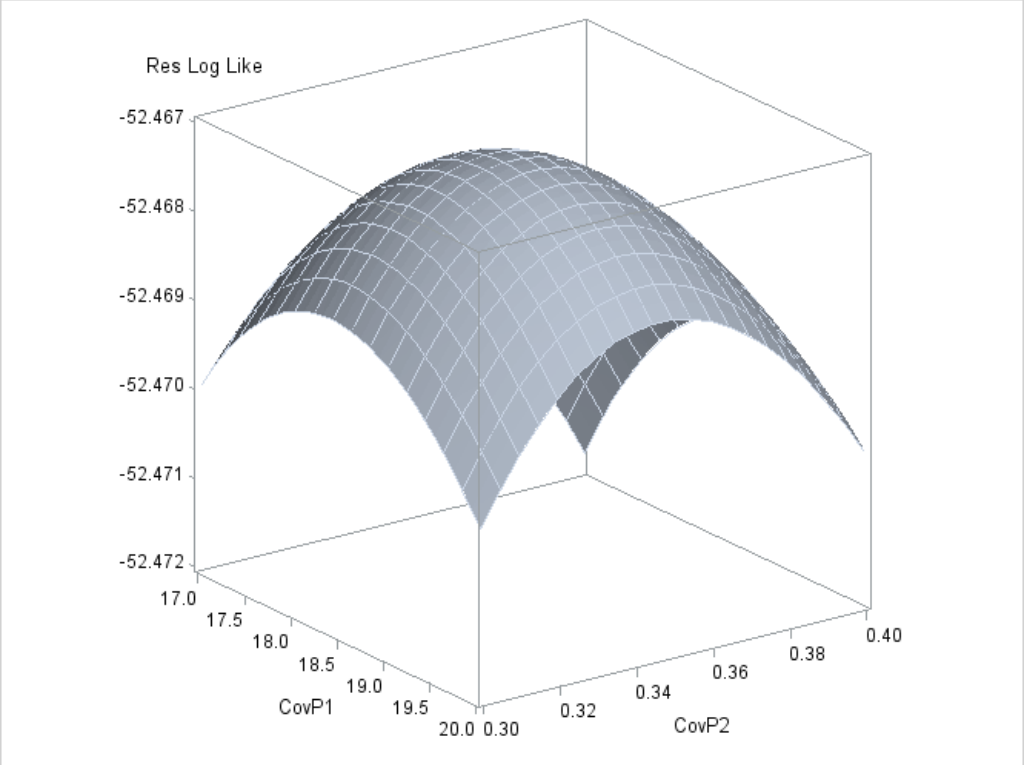
The ParmSearch table is saved in the Parms data set (as specified in the ODS OUTPUT statement). The following steps plot the surface of the residual log likelihood as a function of the covariance parameters and produce Output 20.3.1:

```
proc template;
   define statgraph surface;
      begingraph;
         layout overlay3d;
            surfaceplotparm x=CovP1 y=CovP2 z=ResLogLike;
         endlayout;
      endgraph;
   end;
run;

proc sgrender data=parms template=surface;
```

```
run;

ods html close;
```

PROC TEMPLATE is used to create a template for displaying the data as a three-dimensional surface plot. The plot is displayed with the ODS Graphics procedure SGRENDER. For more information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS."

Output 20.3.1 HTML Output from PROC MIXED



Example 20.4: Creating an Output Data Set from an ODS Table

In this example, the GENMOD procedure is used to perform Poisson regression, and part of the resulting procedure output is written to a SAS data set with the ODS OUTPUT statement. Insurance claims data are classified by two factors: age group (with two levels) and car type (with three levels). The following statements create the data set Insure:

```
title 'Insurance Claims';

data Insure;
   input n c Car $ Age;
   ln = log(n);
   datalines;
 500   42  Small  1
1200   37  Medium 1
 100    1  Large  1
 400  101  Small  2
 500   73  Medium 2
 300   14  Large  2
;
```

The variable n represents the number of insurance policyholders, and the variable c represents the number of insurance claims. The variable Car represents the type of car involved (classified into three groups), and the variable Age is the age of a policyholder (classified into two groups).

You can use PROC GENMOD to perform a Poisson regression analysis of these data with a log link function. Assume that the number-of-claims variable, c, has a Poisson probability distribution and the log of its mean, μ_i, is related to the factors Car and Age.

The following statements obtain the names of the objects produced by this PROC GENMOD run. The ODS TRACE statement lists the trace record. If you already know the names, such as by looking them up in the procedure documentation, you do not have to run this step. The following step displays the trace information:

```
ods trace on;

proc genmod data=insure;
   class car age;
   model c = car age / dist=poisson link=log offset=ln obstats;
run;

ods trace off;
```

The trace record from the SAS log is displayed next:

```
Output Added:
-------------
Name:        ModelInfo
Label:       Model Information
Template:    Stat.Genmod.ModelInfo
Path:        Genmod.ModelInfo
-------------

Output Added:
-------------
Name:        NObs
Label:       Number of observations summary
Template:    Stat.Genmod.NObs
Path:        Genmod.NObs
-------------

Output Added:
-------------
Name:        ClassLevels
Label:       Class Level Information
Template:    Stat.Genmod.Classlevels
Path:        Genmod.ClassLevels
-------------
```

```
Output Added:
-------------
Name:       ParmInfo
Label:      Parameter Information
Template:   Stat.Genmod.Parminfo
Path:       Genmod.ParmInfo
-------------

Output Added:
-------------
Name:       ModelFit
Label:      Criteria For Assessing Goodness Of Fit
Template:   stat.genmod.ModelFit
Path:       Genmod.ModelFit
-------------

Output Added:
-------------
Name:       ConvergenceStatus
Label:      Convergence Status
Template:   Stat.Genmod.ConvergenceStatus
Path:       Genmod.ConvergenceStatus
-------------

Output Added:
-------------
Name:       ParameterEstimates
Label:      Analysis Of Parameter Estimates
Template:   stat.genmod.parameterestimates
Path:       Genmod.ParameterEstimates
-------------

Output Added:
-------------
Name:       ObStats
Label:      Observation Statistics
Template:   Stat.Genmod.Obstats
Path:       Genmod.ObStats
-------------
```

In the following step, no output is displayed because the ODS SELECT NONE statement is included. The ODS OUTPUT statement writes the ODS table ObStats to a SAS data set named myObStats. All of the usual data set options, such as the KEEP= or RENAME= option, can be used in the ODS OUTPUT statement. Thus, to create the myObStats data set so that it contains only certain columns from the ObStats table, you can use the data set options as follows:

```
ods select none;
proc genmod data=insure;
   class car age;
   model c = car age / dist=poisson link=log offset=ln obstats;
   ods output ObStats=myObStats(keep=car age pred
                                rename=(pred=PredictedValue));
run;
```

The KEEP= data set option in the ODS OUTPUT statement specifies that only the variables Car, Age, and Pred are written to the data set. The RENAME= data set option changes the name of variable Pred to PredictedValue. The following statements sort the output data set myObStats, select all output, and produce Output 20.4.1:

```
proc sort data=myObStats;
   by descending PredictedValue;
run;

ods select all;
proc print data=myObStats noobs;
   title2 'Values of Car, Age, and the Predicted Values';
run;
```

The ODS SELECT NONE statement remains in effect until it is explicitly canceled (for example, with the ODS SELECT ALL statement).

Output 20.4.1 The ObStats Table Created as a SAS Data Set

Insurance Claims
Values of Car, Age, and the Predicted Values

Car	Age	PredictedValue
Small	2	107.2011
Medium	2	67.025444
Medium	1	42.974556
Small	1	35.798902
Large	2	13.773459
Large	1	1.2265414

Example 20.5: Creating an Output Data Set: Subsetting the Data

This example demonstrates how you can create an output data set with the ODS OUTPUT statement and also use data set selection keywords to limit the output that ODS writes to a SAS data set. The data set, called Color, contains the eye color and hair color of children from two different regions of Europe. The data are recorded as cell counts, where the variable Count contains the number of children who exhibit each of the 15 combinations of eye and hair color. The following statements create the SAS data set:

```
title 'Hair Color of European Children';

data Color;
   input Region Eyes $ Hair $ Count @@;
   label Eyes  ='Eye Color'
         Hair  ='Hair Color'
         Region='Geographic Region';
   datalines;
1 blue  fair   23  1 blue  red     7  1 blue  medium 24
1 blue  dark   11  1 green fair   19  1 green red     7
1 green medium 18  1 green dark   14  1 brown fair   34
1 brown red     5  1 brown medium 41  1 brown dark   40
```

```
1 brown black   3  2 blue  fair   46  2 blue  red    21
2 blue  medium 44  2 blue  dark   40  2 blue  black   6
2 green fair   50  2 green red    31  2 green medium 37
2 green dark   23  2 brown fair   56  2 brown red    42
2 brown medium 53  2 brown dark   54  2 brown black  13
;
```

The following statements exclude all output and sort the observations in the Color data set by the Region variable:

```
ods select none;

proc sort data=Color;
   by Region;
run;
```

The following ODS OUTPUT statement creates the ChiSq table as a SAS data set named myStats:

```
ods output ChiSq=myStats(drop=Table
                         where=(Statistic =: 'Chi' or
                                Statistic =: 'Like'));
```

You specify the table name in the ODS OUTPUT statement.[4] The DROP= data set option excludes variables from the new data set. The WHERE= data set option selects observations for output to the new data set myStats—specifically, those that begin with `'Chi'` or `'Like'`.

The following statements create Output 20.5.1:

```
proc freq data=Color order=data;
   weight Count;
   tables Eyes*Hair / testp=(30 12 30 25 3);
   by Region;
run;

ods select all;
proc print data=myStats noobs;
run;
```

The FREQ procedure is used to create and analyze a crosstabulation table from the two categorical variables Eyes and Hair, for each value of the variable Region.

Output 20.5.1 Output Data Set from PROC FREQ and ODS

Hair Color of European Children

Region	Statistic	DF	Value	Prob
1	Chi-Square	8	12.6331	0.1251
1	Likelihood Ratio Chi-Square	8	14.1503	0.0779
2	Chi-Square	8	18.2839	0.0192
2	Likelihood Ratio Chi-Square	8	23.3021	0.0030

[4] You can obtain the names of the objects created by any procedure in the individual procedure documentation chapter or from the individual procedure section of the SAS online Help system. (See the "ODS Table Names" section in the "Details" section of the documentation.) You can also determine the names of objects with the ODS TRACE statement (see Example 20.4 and Example 20.2).

Example 20.6: RUN-Group Processing

Some SAS procedures, such as PROC REG and PROC GLM, are interactive. They enable you to submit a group of statements that end with a RUN statement, and then submit more statement groups, each of which ends with a RUN statement. Each group of statements with its RUN statement is called a RUN group. For example, the following PROC REG step has two RUN groups:

```
proc reg data=sashelp.class;
   var Age;
   model Weight = Height;
run;

   model Weight = Height Age;
run;
quit;
```

Interactive procedures can produce several blocks of output for each of several RUN groups. The procedure stays active until it processes a QUIT statement, it encounters a DATA or PROC statement, or the SAS session ends. However, ODS settings are cleared by default at RUN-group boundaries. You can specify the PERSIST= option to maintain ODS settings across RUN statements for procedures that support RUN-group processing.

Consider a data set that contains US population growth trends:

```
title1 'US Population Study';
title2 'Concatenating Two Tables into One Data Set';

data USPopulation;
   input Population @@;
   retain Year 1780;
   Year=Year+10;
   YearSq=Year*Year;
   Population=Population/1000;
   datalines;
3929 5308 7239 9638 12866 17069 23191 31443 39818 50155
62947 75994 91972 105710 122775 131669 151325 179323 203211
;
```

In the following analysis, which has two RUN groups, PROC REG is used to compute the covariance matrix of the estimates for two different models, and the covariance matrices are saved in a single SAS data set. The PERSIST=RUN option in the ODS OUTPUT statement is required to make this happen. The first RUN group creates a data set that contains the CovB table (the covariance matrix of the estimates):

```
proc reg data=USPopulation;
   ods output covb(persist=run)=Bmatrix;
   var YearSq;
   model Population = Year / covb;
run;
```

The MODEL statement defines the regression model and requests the CovB table. The RUN statement executes PROC REG and the model is fit, producing a covariance matrix of the estimates. The covariance matrix has two rows and two columns and is shown in Output 20.6.1.

Output 20.6.1 CovB Table for the First Model

US Population Study
Concatenating Two Tables into One Data Set

The REG Procedure
Model: MODEL1
Dependent Variable: Population

Covariance of Estimates		
Variable	**Intercept**	**Year**
Intercept	20393.138485	-10.83821461
Year	-10.83821461	0.0057650078

In the next RUN group, the YearSq variable is added to the model and the model is fit again, producing a covariance matrix of the estimates that has three rows and three columns:

```
   add YearSq;
   print;
run; quit;
```

The CovB table for the second RUN group is displayed in Output 20.6.2.

Output 20.6.2 CovB Table for the Second Model

Covariance of Estimates			
Variable	**Intercept**	**Year**	**YearSq**
Intercept	711450.62602	-757.2493826	0.2013282694
Year	-757.2493826	0.8061328943	-0.000214361
YearSq	0.2013282694	-0.000214361	5.7010894E-8

If the PERSIST=RUN option is omitted, the selection list is cleared when the RUN statement is encountered, and only the first CovB table is selected. But because the PERSIST=RUN option is specified, the selection list remains in effect throughout the PROC REG step. This ensures that each of the CovB tables is selected and output. The following statements display the ODS OUTPUT data set and create Output 20.6.3:

```
proc print;
   id _run_;
   by _run_;
run;
```

Output 20.6.3 Results of the ODS OUTPUT Statement When the PERSIST Option Is Specified

US Population Study
Concatenating Two Tables into One Data Set

Run Group Number=1

Run	Model	Dependent	Variable	Intercept	Year	YearSq
1	MODEL1	Population	Intercept	20393.138485	-10.83821461	.
	MODEL1	Population	Year	-10.83821461	0.0057650078	.

Output 20.6.3 *continued*

Run	Model	Dependent	Variable	Intercept	Year	YearSq
2	MODEL1.1	Population	Intercept	711450.62602	-757.2493826	0.2013282694
	MODEL1.1	Population	Year	-757.2493826	0.8061328943	-0.000214361
	MODEL1.1	Population	YearSq	0.2013282694	-0.000214361	5.7010894E-8

Even though the two CovB tables do not have the same rows or columns, ODS automatically combines the two tables into one data set.

ODS Statement Placement with Interactive Procedures

Where you place a statement in an interactive procedure is critical. The following examples demonstrate the results of various placements of statements:

- If you submit the following steps to a new SAS session, to an existing SAS session after a QUIT statement, or to an existing SAS session after a noninteractive procedure has completed, the ODS OUTPUT statement creates a SAS data set that contains the PROC GLM parameter estimates:

```
ods output parameterestimates=pm;
proc glm data=sashelp.class;
   class sex;
   model weight = sex | height / solution;
run;
```

- If you submit the following steps, no ODS OUTPUT SAS data set is created:

```
proc reg data=sashelp.class;
   model weight = height;
run;

ods output parameterestimates=pm;
proc glm data=sashelp.class;
   class sex;
   model weight = sex | height / solution;
run;
```

  The preceding steps work as follows:

  – The first three statements (PROC REG, MODEL, and RUN) perform a simple regression analysis and display the parameter estimates table. This completes one RUN group.
  – PROC REG is still active when it encounters the ODS OUTPUT statement. But the first RUN group has completed, so the first parameter estimates table is no longer available for ODS to output.
  – PROC REG terminates because of the PROC GLM statement.

  - The ODS OUTPUT statement has not created an output data set when PROC REG terminates because no model was fit after the first RUN group.
  - SAS prints warning messages because the ODS OUTPUT statement could not output a parameter estimates table.
  - The ODS OUTPUT statement corresponds to the PROC REG step because PROC REG is still active when the statement is encountered. If PROC REG had ended first, then the ODS OUTPUT statement would apply to the following PROC GLM step.

- The ODS OUTPUT statement in the following example outputs the PROC GLM parameter estimates because the QUIT statement ends the PROC REG step:

```
proc reg data=sashelp.class;
   model weight = height;
run; quit;

ods output parameterestimates=pm;
proc glm data=sashelp.class;
   class sex;
   model weight = sex | height / solution;
run;
```

- The ODS OUTPUT statement in the following example outputs the PROC GLM parameter estimates because the ODS OUTPUT statement appears within the PROC GLM step:

```
proc reg data=sashelp.class;
   model weight = height;
run;

proc glm data=sashelp.class;
   ods output parameterestimates=pm;
   class sex;
   model weight = sex | height / solution;
run;
```

Explicitly Ending an Interactive Procedure

You can end interactive procedures such as PROC GLM and PROC REG by submitting a RUN statement and then a QUIT statement:

```
proc reg data=sashelp.class;
   model weight = height;
run; quit;
```

Alternatively, you can end PROC GLM and PROC REG by submitting only a QUIT statement:

```
proc reg data=sashelp.class;
   model weight = height;
quit;
```

Some interactive procedures behave differently from PROC GLM and PROC REG. The GPLOT procedure does not produce results when you submit a QUIT statement but no RUN statement. You should end the IML procedure by submitting only a QUIT statement, because a RUN statement in PROC IML runs a module.

Example 20.7: ODS Output Data Sets and Using PROC TEMPLATE to Customize Output

You can use ODS statements, the DATA step, and PROC TEMPLATE to modify the appearance of your displayed tables or to display results in forms that are not directly produced by any procedure. The following example, similar to that given in Olinger and Tobias (1998), runs an analysis with PROC GLM. This example has several parts. It creates output data sets with the ODS OUTPUT statement, combines and manipulates those data sets, displays the results by using a standard SAS template, modifies a template by using PROC TEMPLATE, and displays the output data sets by using the modified template. Each step works toward the final goal of taking multiple tables and creating a custom display of those tables in a way that cannot be done directly by PROC GLM.

The following statements create a SAS data set named Histamine that contains the experimental data:

```
title1 'Histamine Study';

data Histamine;
   input Drug $12. Depleted $ hist0 hist1 hist3 hist5;
   logHist0 = log(hist0); logHist1 = log(Hist1);
   logHist3 = log(hist3); logHist5 = log(Hist5);
   datalines;
Morphine      N  .04  .20  .10  .08
Morphine      N  .02  .06  .02  .02
Morphine      N  .07 1.40  .48  .24
Morphine      N  .17  .57  .35  .24
Morphine      Y  .10  .09  .13  .14
Morphine      Y  .07  .07  .06  .07
Morphine      Y  .05  .07  .06  .07
Trimethaphan  N  .03  .62  .31  .22
Trimethaphan  N  .03 1.05  .73  .60
Trimethaphan  N  .07  .83 1.07  .80
Trimethaphan  N  .09 3.13 2.06 1.23
Trimethaphan  Y  .10  .09  .09  .08
Trimethaphan  Y  .08  .09  .09  .10
Trimethaphan  Y  .13  .10  .12  .12
Trimethaphan  Y  .06  .05  .05  .05
;
```

The data set comes from a preclinical drug experiment (Cole and Grizzle 1966). In order to study the effect of two different drugs on histamine levels in the blood, researchers administer the drugs to 13 animals and

measure the levels of histamine in the animals' blood after 0, 1, 3, and 5 minutes. The response variable is the logarithm of the histamine level.

In the analysis that follows, PROC GLM is used to perform a repeated measures analysis, naming the drug and depletion status as between-subject factors in the MODEL statement and naming post-administration measurement time as the within-subject factor. For more information about this study and its analysis, see Example 47.7 in Chapter 47, "The GLM Procedure."

The following PROC GLM statements begin the analysis:

```
ods graphics off;
ods trace output;

proc glm data=Histamine;
   class Drug Depleted;
   model LogHist0--LogHist5 = Drug Depleted Drug*Depleted / nouni;
   repeated Time 4 (0 1 3 5) polynomial / summary printe;
run; quit;
```

The portion of the trace output that contains the fully qualified name paths is shown next:

```
Path:       GLM.Data.ClassLevels
Path:       GLM.Data.NObs
Path:       GLM.Repeated.RepeatedLevelInfo
Path:       GLM.Repeated.PartialCorr
Path:       GLM.Repeated.MANOVA.Model.Error.ErrorSSCP
Path:       GLM.Repeated.MANOVA.Model.Error.PartialCorr
Path:       GLM.Repeated.MANOVA.Model.Error.Sphericity
Path:       GLM.Repeated.MANOVA.Model.Time.MultStat
Path:       GLM.Repeated.MANOVA.Model.Time_Drug.MultStat
Path:       GLM.Repeated.MANOVA.Model.Time_Depleted.MultStat
Path:       GLM.Repeated.MANOVA.Model.Time_Drug_Depleted.MultStat
Path:       GLM.Repeated.BetweenSubjects.ModelANOVA
Path:       GLM.Repeated.WithinSubject.ModelANOVA
Path:       GLM.Repeated.WithinSubject.Epsilons
Path:       GLM.Repeated.Summary.Time_1.ModelANOVA
Path:       GLM.Repeated.Summary.Time_2.ModelANOVA
Path:       GLM.Repeated.Summary.Time_3.ModelANOVA
```

The goal here is to output the within-subjects multivariate statistics and the between-subjects ANOVA table to SAS data sets for use in subsequent steps. The following statements run the analysis and save the desired results to output data sets:

```
ods select none;

proc glm data=Histamine;
   class Drug Depleted;
   model LogHist0--LogHist5 = Drug Depleted Drug*Depleted / nouni;
   repeated Time 4 (0 1 3 5) polynomial / summary printe;
   ods output MultStat                    = HistWithin
              BetweenSubjects.ModelANOVA = HistBetween;
run; quit;

ods select all;
```

No output is displayed due to the ODS SELECT statements. The ODS OUTPUT statement creates two SAS data sets, named HistWithin and HistBetween, from the two ODS tables. This analysis creates the following tables:

```
Path:       GLM.Repeated.MANOVA.Model.Time.MultStat
Path:       GLM.Repeated.MANOVA.Model.Time_Drug.MultStat
Path:       GLM.Repeated.MANOVA.Model.Time_Depleted.MultStat
Path:       GLM.Repeated.MANOVA.Model.Time_Drug_Depleted.MultStat
Path:       GLM.Repeated.BetweenSubjects.ModelANOVA
```

Here is the full trace output for the model ANOVA table:

```
Output Added:
-------------
Name:       ModelANOVA
Label:      Type III Model ANOVA
Template:   stat.GLM.Tests
Path:       GLM.Repeated.BetweenSubjects.ModelANOVA
-------------
```

All of the multivariate test results are routed to the HistWithin data set because all multivariate test tables are named MultStat, even though they occur in different directories in the output directory hierarchy. Only the between-subject ANOVA table appears in the HistBetween data set, even though there are also other tables named ModelANOVA. ODS selects just the one specific table for the HistBetween data set because of the partial name path (**BetweenSubjects.ModelANOVA**) in the second specification. For more information about names and qualified path names, see the discussion in the section "The ODS Statement" on page 527.

The following statements show the names and the variable labels for the two data sets and produce Output 20.7.1:

```
proc contents data=HistBetween varnum;
   ods select position;
run;

proc contents data=HistWithin varnum;
   ods select position;
run;
```

Output 20.7.1 Variable Names and Labels for the Two Data Sets

Histamine Study

The CONTENTS Procedure

Variables in Creation Order

#	Variable	Type	Len	Format	Label
1	Dependent	Char	15		
2	HypothesisType	Num	8	BEST8.	
3	Source	Char	20		
4	DF	Num	8	BEST6.	
5	SS	Num	8	12.8	Type III SS
6	MS	Num	8	12.8	Mean Square
7	FValue	Num	8	7.2	F Value
8	ProbF	Num	8	PVALUE6.4	Pr > F

Histamine Study

The CONTENTS Procedure

Variables in Creation Order

#	Variable	Type	Len	Format	Label
1	Hypothesis	Char	32		
2	Error	Char	55		
3	Statistic	Char	22		
4	Value	Num	8	12.8	
5	FValue	Num	8	7.2	F Value
6	NumDF	Num	8	BEST6.	Num DF
7	DenDF	Num	8	BEST6.	Den DF
8	ProbF	Num	8	PVALUE6.4	Pr > F
9	PValue	Num	8	PVALUE6.4	P-Value

The following statements create a new data set that contains the two data sets created in the preceding PROC GLM step and display the results in Output 20.7.2:

```
title2 'The Combined Data Set';

data temp1;
   set HistBetween HistWithin;
run;

proc print label;
run;
```

Output 20.7.2 Listing of the Combined Data Set: Histamine Study

Histamine Study
The Combined Data Set

Obs	Dependent	HypothesisType	Source	DF
1	BetweenSubjects	3	Drug	1
2	BetweenSubjects	3	Depleted	1
3	BetweenSubjects	3	Drug*Depleted	1
4	BetweenSubjects	3	Error	11
5		.		.
6		.		.
7		.		.
8		.		.
9		.		.
10		.		.
11		.		.
12		.		.
13		.		.
14		.		.
15		.		.
16		.		.
17		.		.
18		.		.
19		.		.
20		.		.

Output 20.7.2 *continued*

Histamine Study
The Combined Data Set

Obs	Type III SS	Mean Square	F Value	Pr > F	Hypothesis	Error
1	5.99336243	5.99336243	2.71	0.1281		
2	15.44840703	15.44840703	6.98	0.0229		
3	4.69087508	4.69087508	2.12	0.1734		
4	24.34683348	2.21334850	_	_		
5	.	.	24.03	0.0001	Time	Error SSCP Matrix
6	.	.	24.03	0.0001	Time	Error SSCP Matrix
7	.	.	24.03	0.0001	Time	Error SSCP Matrix
8	.	.	24.03	0.0001	Time	Error SSCP Matrix
9	.	.	5.78	0.0175	Time_Drug	Error SSCP Matrix
10	.	.	5.78	0.0175	Time_Drug	Error SSCP Matrix
11	.	.	5.78	0.0175	Time_Drug	Error SSCP Matrix
12	.	.	5.78	0.0175	Time_Drug	Error SSCP Matrix
13	.	.	21.31	0.0002	Time_Depleted	Error SSCP Matrix
14	.	.	21.31	0.0002	Time_Depleted	Error SSCP Matrix
15	.	.	21.31	0.0002	Time_Depleted	Error SSCP Matrix
16	.	.	21.31	0.0002	Time_Depleted	Error SSCP Matrix
17	.	.	12.48	0.0015	Time_Drug_Depleted	Error SSCP Matrix
18	.	.	12.48	0.0015	Time_Drug_Depleted	Error SSCP Matrix
19	.	.	12.48	0.0015	Time_Drug_Depleted	Error SSCP Matrix
20	.	.	12.48	0.0015	Time_Drug_Depleted	Error SSCP Matrix

Output 20.7.2 *continued*

Histamine Study
The Combined Data Set

Obs	Statistic	Value	Num DF	Den DF	P-Value
1		.	.	.	.
2		.	.	.	.
3		.	.	.	.
4		.	.	.	.
5	Wilks' Lambda	0.11097706	3	9	.
6	Pillai's Trace	0.88902294	3	9	.
7	Hotelling-Lawley Trace	8.01087137	3	9	.
8	Roy's Greatest Root	8.01087137	3	9	.
9	Wilks' Lambda	0.34155984	3	9	.
10	Pillai's Trace	0.65844016	3	9	.
11	Hotelling-Lawley Trace	1.92774470	3	9	.
12	Roy's Greatest Root	1.92774470	3	9	.
13	Wilks' Lambda	0.12339988	3	9	.
14	Pillai's Trace	0.87660012	3	9	.
15	Hotelling-Lawley Trace	7.10373567	3	9	.
16	Roy's Greatest Root	7.10373567	3	9	.
17	Wilks' Lambda	0.19383010	3	9	.
18	Pillai's Trace	0.80616990	3	9	.
19	Hotelling-Lawley Trace	4.15915732	3	9	.
20	Roy's Greatest Root	4.15915732	3	9	.

The next steps are designed to produce a more parsimonious display of the most important information in Output 20.7.2. The next step creates a data set named HistTests. Only the observations from the input data sets that are needed for interpretation are included. The variable Hypothesis in the HistWithin data set is renamed Source, and the NumDF variable is renamed DF. The renamed variables correspond to the variable names found in the HistBetween data set. These names are chosen since the template for the ModelANOVA table is used in subsequent steps. An explicit length for the new variable Source is provided since the input variables, Hypothesis and Source, have different lengths. The following statements produce Output 20.7.3:

```
data HistTests;
   length Source $ 20;
   set HistBetween(where =(Source    ^= 'Error'))
       HistWithin (rename=(Hypothesis =  Source NumDF=DF)
                   where =(Statistic  = 'Hotelling-Lawley Trace'));
run;

proc print label;
   title2 'Listing of the Combined Data Set';
run;
```

Output 20.7.3 Listing of the HistTests Data Set: Histamine Study

Histamine Study
Listing of the Combined Data Set

Obs	Source	Dependent	HypothesisType	Num DF	Type III SS
1	Drug	BetweenSubjects	3	1	5.99336243
2	Depleted	BetweenSubjects	3	1	15.44840703
3	Drug*Depleted	BetweenSubjects	3	1	4.69087508
4	Time		.	3	.
5	Time_Drug		.	3	.
6	Time_Depleted		.	3	.
7	Time_Drug_Depleted		.	3	.

Obs	Mean Square	F Value	Pr > F	Error	Statistic	Value	Den DF	P-Value
1	5.99336243	2.71	0.1281			.	.	.
2	15.44840703	6.98	0.0229			.	.	.
3	4.69087508	2.12	0.1734			.	.	.
4	.	24.03	0.0001	Error SSCP Matrix	Hotelling-Lawley Trace	8.01087137	9	.
5	.	5.78	0.0175	Error SSCP Matrix	Hotelling-Lawley Trace	1.92774470	9	.
6	.	21.31	0.0002	Error SSCP Matrix	Hotelling-Lawley Trace	7.10373567	9	.
7	.	12.48	0.0015	Error SSCP Matrix	Hotelling-Lawley Trace	4.15915732	9	.

The amount of information contained in the HistTests data set is appropriate for interpreting the analysis; however, there is still extra information, and the information of interest is not being displayed in a compact or useful form. This data set consists of multiple tables, an ANOVA table with between-subjects information, and multivariate statistics tables with the variables renamed to match the names in the ANOVA table. This form was chosen so that the data set could be displayed using PROC GLM's ANOVA template. A template specifies how the data set should be displayed and which columns should be displayed. The output from the ODS TRACE statements shows that the template associated with PROC GLM's ANOVA table is named **`Stat.GLM.Tests`**. You can use the **`Stat.GLM.Tests`** template to display the SAS data set HistTests, as follows:

```
title2 'Listing of the Selections, Using a Standard Template';
proc sgrender data=histtests template=Stat.GLM.Tests;
run;
```

The SGRENDER procedure displays the DATA= data set with the specified TEMPLATE= template. (You can use PROC SGRENDER to display both graphs and tables.) The results are displayed in Output 20.7.4.

Output 20.7.4 Listing of the Data Set Using a Standard PROC GLM ANOVA Template

Histamine Study
Listing of the Selections, Using a Standard Template

Source	DF	SS	Mean Square	F Value	Pr > F
Drug	1	5.99336243	5.99336243	2.71	0.1281
Depleted	1	15.44840703	15.44840703	6.98	0.0229
Drug*Depleted	1	4.69087508	4.69087508	2.12	0.1734
Time	3	.	.	24.03	0.0001
Time_Drug	3	.	.	5.78	0.0175
Time_Depleted	3	.	.	21.31	0.0002
Time_Drug_Depleted	3	.	.	12.48	0.0015

Alternatively, you could display the results by using a DATA step as follows:

```
title2 'Listing of the Selections, Using a Standard Template';

data _null_;
   set histtests;
   file print ods=(template='Stat.GLM.Tests');
   put _ods_;
run;
```

The next steps create a final display of these results, this time by using a custom template. This example shows you how to use PROC TEMPLATE to do the following:

- redefine the format for the SS and Mean Square columns
- include the table title and footnote in the body of the table
- translate the missing values for SS and Mean Square in the rows that correspond to multivariate tests to asterisks
- add a footnote to a table
- add a column that depicts the level of significance of each effect

The following statements create a custom template:

```
proc template;
   define table CombinedTests;
      parent=Stat.GLM.Tests;

      header '#Histamine Study##';
      footer '#* - Test computed using Hotelling-Lawley trace';

      column Source DF SS MS FValue ProbF Star;

      define Source; width=20; end;
      define DF; format=bestd3.; end;
      define SS;
         parent=Stat.GLM.SS
         choose_format=max format_width=7;
         translate _val_ = . into '  *';
      end;
      define MS;
```

```
         parent=Stat.GLM.MS
         choose_format=max format_width=7;
         translate _val_ = . into '  *';
      end;
      define Star;
         compute as ProbF;
         translate _val_ <= 0.001 into 'xxx',
                   _val_ <= 0.01  into 'xx',
                   _val_ <= 0.05  into 'x',
                   _val_ >  0.05  into '';
         pre_space=1 width=3 just=l;
      end;
   end;
run;
```

The CHOOSE_FORMAT=MAX option along with FORMAT_WIDTH=7 chooses the format for each column based on the maximum value and an overall width of 7. Alternatively, you could have specified a format directly by specifying, for example, FORMAT=7.2 or FORMAT=D8.3. The TRANSLATE statements provide values to display in place of the original values. The first two TRANSLATE statements display missing values as an asterisk with leading blanks added to ensure alignment with the decimal place. The third TRANSLATE statement displays *p*-values greater than 0.05 as a blank, values greater than 0.01 but less than or equal to 0.05 as a single `'x'`, and so on. The ProbF column is printed twice—once in the usual way as a numeric column with a PVALUE format and once with a column of blanks or x's. For detailed information about PROC TEMPLATE, see the section "The Template Procedure" in the *SAS Output Delivery System: User's Guide*. The following statements use the customized template to display the HistTests data set:

```
title2 'Listing of the Selections, Using a Customized Template';

proc sgrender data=HistTests template=CombinedTests;
run;
```

The results are displayed in Output 20.7.5.

Output 20.7.5 Display of the Data Sets Using a Customized Template: Histamine Study

Histamine Study
Listing of the Selections, Using a Customized Template

Histamine Study					
Source	Num DF	Sum of Squares	Mean Square	F Value	Pr > F	
Drug	1	5.9934	5.9934	2.71	0.1281	
Depleted	1	15.4484	15.4484	6.98	0.0229	x
Drug*Depleted	1	4.6909	4.6909	2.12	0.1734	
Time	3	*	*	24.03	0.0001	xxx
Time_Drug	3	*	*	5.78	0.0175	x
Time_Depleted	3	*	*	21.31	0.0002	xxx
Time_Drug_Depleted	3	*	*	12.48	0.0015	xx
* - Test computed using Hotelling-Lawley trace						

These next steps display the same table, but this time changing the background color for the entire row to highlight effects with *p*-values less than 0.001 and also those with *p*-values less than 0.01. The table is displayed three times. Output 20.7.6 displays the results by using bold green and yellow backgrounds and a

bold font. Output 20.7.7 displays the results by using subtler cyan and yellow backgrounds and a bold font. Output 20.7.8 displays the results by using very subtle cyan and gray backgrounds and a normal font. This control is provided by the CELLSTYLE statement in PROC TEMPLATE. You can do many things with the CELLSTYLE statement to enhance your output. Several more are shown in other examples in this chapter. These next steps create the custom template with varying colors and fonts and display the results by using PROC SGRENDER:

```
%macro hilight(c1,c2);
   proc template;
      define table CombinedTests;
         parent=Stat.GLM.Tests;

         header '#Histamine Study##';
         footer '#* - Test computed using Hotelling-Lawley trace';

         column Source DF SS MS FValue ProbF;

         cellstyle probf <= 0.001 as {background=&c1},
                   probf <= 0.01  as {background=&c2};

         define DF; format=bestd3.; end;
         define SS;
            parent=Stat.GLM.SS
            choose_format=max format_width=7;
            translate _val_ = . into '  *';
         end;
         define MS;
            parent=Stat.GLM.MS
            choose_format=max format_width=7;
            translate _val_ = . into '  *';
         end;
      end;
   run;

   proc sgrender data=HistTests template=CombinedTests;
   run;
%mend;

title2;
ods _all_ close;
ods html style=HTMLBlue;

%hilight(CX22FF22 fontweight=bold, CXFFFF22 fontweight=bold)
%hilight(CXAAFFFF fontweight=bold, CXFFFFDD fontweight=bold)
%hilight(CXEEFAFA, CXEEEEEE)

ods html close;
ods html;
ods pdf;
```

Output 20.7.6 Rows Boldly Highlighted: Histamine Study

Histamine Study

Histamine Study

Source	Num DF	Sum of Squares	Mean Square	F Value	Pr > F
Drug	1	5.9934	5.9934	2.71	0.1281
Depleted	1	15.4484	15.4484	6.98	0.0229
Drug*Depleted	1	4.6909	4.6909	2.12	0.1734
Time	**3**	*	*	**24.03**	**0.0001**
Time_Drug	3	*	*	5.78	0.0175
Time_Depleted	**3**	*	*	**21.31**	**0.0002**
Time_Drug_Depleted	**3**	*	*	**12.48**	**0.0015**

* - Test computed using Hotelling-Lawley trace

Output 20.7.7 Rows Subtly Highlighted: Histamine Study

Histamine Study

Histamine Study

Source	Num DF	Sum of Squares	Mean Square	F Value	Pr > F
Drug	1	5.9934	5.9934	2.71	0.1281
Depleted	1	15.4484	15.4484	6.98	0.0229
Drug*Depleted	1	4.6909	4.6909	2.12	0.1734
Time	**3**	*	*	**24.03**	**0.0001**
Time_Drug	3	*	*	5.78	0.0175
Time_Depleted	**3**	*	*	**21.31**	**0.0002**
Time_Drug_Depleted	**3**	*	*	**12.48**	**0.0015**

* - Test computed using Hotelling-Lawley trace

Output 20.7.8 Rows Very Subtly Highlighted: Histamine Study

Histamine Study

Histamine Study

Source	Num DF	Sum of Squares	Mean Square	F Value	Pr > F
Drug	1	5.9934	5.9934	2.71	0.1281
Depleted	1	15.4484	15.4484	6.98	0.0229
Drug*Depleted	1	4.6909	4.6909	2.12	0.1734
Time	3	*	*	24.03	0.0001
Time_Drug	3	*	*	5.78	0.0175
Time_Depleted	3	*	*	21.31	0.0002
Time_Drug_Depleted	3	*	*	12.48	0.0015

* - Test computed using Hotelling-Lawley trace

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 (or 0 to 255 base 10). You can run the following step to see the correspondence between the integer and HEX formatting of values in the range 0 to 255:

```
data _null_;
   do color = 0 to 255;
      put color 3. +1 color hex2.;
   end;
run;
```

The results of this step are not shown. Hexadecimal values 0 through F represent the numbers 0 to 15. A hex value xy can be converted to an integer as follows: $16x + y$. For example, BC is $16 \times 11 + 12 = 188$. Common colors are CXFF0000 (red), CX00FF00 (green), CX0000FF (blue), CXFFFF00 (yellow, a mix of red and green), CXFF00FF (magenta, a mix of red and blue), CX00FFFF (cyan, a mix of green and blue), CXFFFFFF (white, a mix of red, green, and blue), CX000000 (black, no color), CXDDDDDD (very light gray), CX222222 (very dark gray), and so on. Colors become lighter as the RGB values increase and darker as they decrease. For example, cyan (CX00FFFF) can be lightened by increasing the red component from 00 to FF until eventually it becomes indistinguishable from white. It can be darkened by jointly decreasing the green and blue values until it becomes indistinguishable from black.

The three CELLSTYLE statements that set the colors after the macro variables are substituted are as follows:

```
cellstyle probf <= 0.001 as {background=CX22FF22 fontweight=bold},
          probf <= 0.01  as {background=CXFFFF22 fontweight=bold};
cellstyle probf <= 0.001 as {background=CXAAFFFF fontweight=bold},
          probf <= 0.01  as {background=CXFFFFDD fontweight=bold};
cellstyle probf <= 0.001 as {background=CXEEFAFA},
          probf <= 0.01  as {background=CXEEEEEE};
```

The first color, CX22FF22, for the smallest p-values in the first table is a bold green color. The first table uses almost pure green and pure yellow, but a little red and blue are added to slightly lighten the colors. The second table uses a cyan and yellow that are very light due to the addition of AA (170) red and DD (221) blue, respectively. The third table uses a cyan that is not much different from light gray, and a light gray that is not much different from white.

Example 20.8: HTML Output with Hyperlinks between Tables

This example demonstrates how you can use ODS to provide links between different parts of your HTML procedure output. This example creates a table where each row contains a link to another table with more information about that row.

Suppose that you are analyzing a 4×4 factorial experiment for an industrial process, testing for differences in the number of defective products that are manufactured by different machines and use different sources of raw material. The data set Experiment is created as follows:

```
title 'Product Defects Experiment';

data Experiment;
   do Supplier = 'A', 'B', 'C', 'D';
      do Machine = 1 to 4;
         do rep = 1 to 5;
            input Defects @@;
            output;
         end;
      end;
   end;
```

```
   datalines;
 2  6  3  3  6  8  6  6  4  4  4  2  4  0  4  5  5  7  8  5
13 12 12 11 12 16 15 14 14 13 11 10 12 12 10 13 13 14 15 12
 2  6  3  6  6  6  4  4  6  6  0  3  2  0  2  4  6  7  6  4
20 19 18 21 22 22 24 23 20 20 17 19 18 16 17 23 20 20 22 21
;
```

Suppose that you are interested in fitting a model to determine the effect that the supplier of raw material and machine type have on the number of defects in the products. If the F test for a factor is significant, you might want to follow up with a multiple-comparison test for the levels of that factor. The tables of interest are the model ANOVA and the multiple-comparison output. Since this is a balanced experiment, the ANOVA procedure computes the appropriate analysis. The following statements produce these tables and Figure 20.8.1:

```
ods _all_ close;
ods html body='anovab.htm' style=HTMLBlue anchor='anova1';
ods trace output;

proc anova data=Experiment;
   ods select ModelANOVA MCLines;
   class Supplier Machine;
   model Defects = Supplier Machine;
   means Supplier Machine / tukey;
run; quit;

ods html close;
ods html;
ods pdf;
```

All destinations are first closed to avoid generating the output multiple times. ODS writes the HTML output to the file *anovab.htm*. The ANCHOR= option specifies *anova1* as the root name for the HTML anchor tags. This means that within the HTML document, the URL for the first table will be *anova1*, the URL for the second table will be *anova2*, and so on.

Output 20.8.1 ANOVA and Multiple-Comparison Results: Histamine Study

Product Defects Experiment

The ANOVA Procedure

Dependent Variable: Defects

Source	DF	Anova SS	Mean Square	F Value	Pr > F
Supplier	3	3441.637500	1147.212500	580.72	<.0001
Machine	3	163.137500	54.379167	27.53	<.0001

Output 20.8.1 *continued*

Product Defects Experiment

The ANOVA Procedure

Tukey's Studentized Range (HSD) Test for Defects

Means with the same letter are not significantly different.

Tukey Grouping	Mean	N	Supplier
A	20.1000	20	D
B	12.7000	20	B
C	4.6000	20	A
C			
C	4.1500	20	C

Product Defects Experiment

The ANOVA Procedure

Tukey's Studentized Range (HSD) Test for Defects

Means with the same letter are not significantly different.

Tukey Grouping	Mean	N	Machine
A	11.7500	20	2
A			
A	11.5000	20	4
B	10.1500	20	1
C	8.1500	20	3

The ODS trace output (not shown) shows that PROC ANOVA uses the **Stat.GLM.Tests** template to format the ANOVA table. The following statements demonstrate how you can link a row of the ANOVA table to the corresponding multiple-comparison table by modifying the table template, using the original values and the URLs for the second and third tables (*anova2* and *anova3*):

```
proc template;
   edit Stat.GLM.Tests;
      edit Source;
         cellstyle _val_ = 'Supplier' as {url="#ANOVA2"},
                   _val_ = 'Machine'  as {url="#ANOVA3"};
      end;
   end;
run;
```

This template uses the CELLSTYLE statement to alter the values in the **Source** column (**'Supplier'** and **'Machine'**) of the ANOVA tests table. The values of **'Supplier'** and **'Machine'** are displayed as hyperlinks in the HTML, and clicking them takes you to the links *anova2* and *anova3*, which are the multiple-comparison tables.

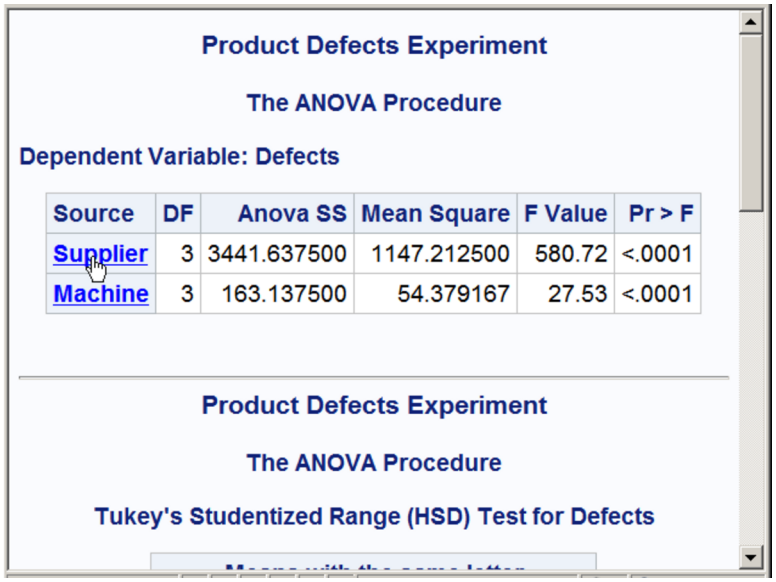
You can find the value to use in the URL by viewing the HTML source file, *anovab.htm*. You can either open the HTML file in a text editor or view it in a browser window and select **View ► Source**. Search for **'<a name='** to find the URL names. The first table is *anova1*, the second is *anova2*, the third is *anova3*, and so on. If the ANCHOR= option had not been used in the ODS HTML statement, the names would have been *IDX*, *IDX1*, *IDX2*, and so on. If you do not use the ODS SELECT statement or if you do anything to change the tables that are produced, the names will be different. The statements create the *Supplier* label as a link that enables you to open the table of means from the "Tukey's Studentized Range Test for Defects" associated with the Supplier variable. Similarly, *Machine* provides a link to the table of means from the "Tukey's Studentized Range Test for Defects" associated with the Machine variable.

Next, the analysis is run again, this time using the modified template. The following statements produce the results:

```
ods _all_ close;
ods html body='anovab.htm' style=HTMLBlue anchor='anova1';

proc anova data=Experiment;
   ods select ModelANOVA MCLines;
   class Supplier Machine;
   model Defects = Supplier Machine;
   means Supplier Machine / tukey;
run; quit;

ods html close;
ods html;
ods pdf;
```

The ANOVA table is displayed in Output 20.8.2.

Output 20.8.2 HTML Output from PROC ANOVA: Linked Output



Product Defects Experiment

The ANOVA Procedure

Dependent Variable: Defects

Source	DF	Anova SS	Mean Square	F Value	Pr > F
Supplier	3	3441.637500	1147.212500	580.72	<.0001
Machine	3	163.137500	54.379167	27.53	<.0001

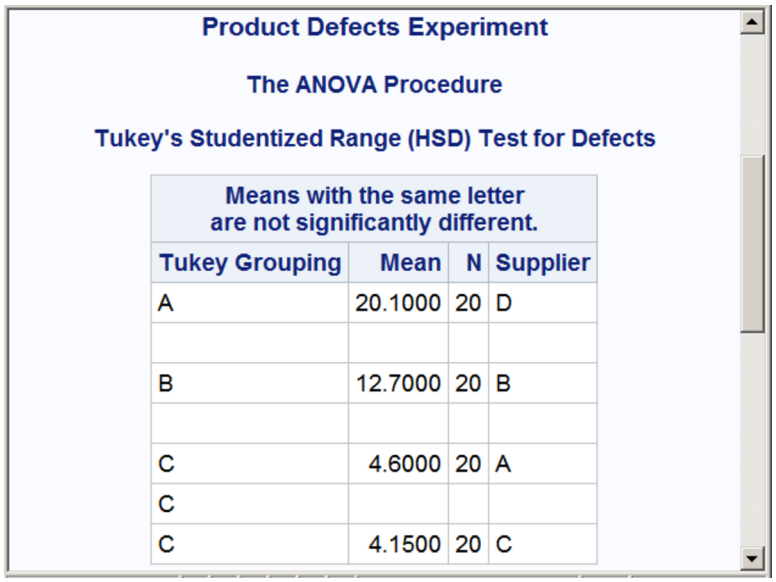
Product Defects Experiment

The ANOVA Procedure

Tukey's Studentized Range (HSD) Test for Defects

The underlined text displayed in Output 20.8.2 shows the links, *Supplier* and *Machine*, that you created with the modified template. When you click a link, the appropriate multiple-comparison table opens in your browser. Output 20.8.3 shows the table from the *Supplier* link.

Output 20.8.3 Linked Output: Multiple-Comparison Table from PROC ANOVA



Product Defects Experiment

The ANOVA Procedure

Tukey's Studentized Range (HSD) Test for Defects

Means with the same letter are not significantly different.

Tukey Grouping	Mean	N	Supplier
A	20.1000	20	D
B	12.7000	20	B
C	4.6000	20	A
C			
C	4.1500	20	C

When you run the PROC TEMPLATE step shown previously, the following note is printed in the SAS log:

```
NOTE: TABLE 'Stat.GLM.Tests' has been saved to: SASUSER.TEMPLAT
```

You can see that there is a version of the template in Sasuser by running the following statements:

```
proc template;
   list Stat.GLM.Tests;
run;
```

These statements produce Output 20.8.4.

Output 20.8.4 Templates

Product Defects Experiment

Listing of: SASUSER.TEMPLAT		
Path Filter is: Stat.GLM.Tests		
Sort by: PATH/ASCENDING		
Obs	Path	Type
1	Stat.GLM.Tests	Table

Listing of: SASHELP.TMPLSTAT		
Path Filter is: Stat.GLM.Tests		
Sort by: PATH/ASCENDING		
Obs	Path	Type
1	Stat.GLM.Tests	Table

You can delete your custom template and restore the default template as follows:

```
proc template;
   delete Stat.GLM.Tests / store=sasuser.templat;
run;
```

The following note is printed in the SAS log:

```
NOTE: 'Stat.GLM.Tests' has been deleted from: SASUSER.TEMPLAT
```

Example 20.9: HTML Output with Graphics and Hyperlinks

This example demonstrates how you can use ODS to create links between each bar in a bar chart (Output 20.9.1) and other parts of the analysis (Output 20.9.2). The data in this example are selected from a larger experiment on the use of drugs in the treatment of leprosy (Snedecor and Cochran 1967, p. 422). Variables in the study are as follows:

Drug	two antibiotics (‘a’ and ‘d’) and a control (‘f’)
PreTreatment	a pretreatment score of leprosy bacilli
PostTreatment	a posttreatment score of leprosy bacilli

The data set is created as follows:

```
title 'Treatment of Leprosy';

data drugtest;
   input Drug $ PreTreatment PostTreatment @@;
   datalines;
a 11  6  a  8  0  a  5  2  a 14  8  a 19 11
a  6  4  a 10 13  a  6  1  a 11  8  a  3  0
d  6  0  d  6  2  d  7  3  d  8  1  d 18 18
d  8  4  d 19 14  d  8  9  d  5  1  d 15  9
f 16 13  f 13 10  f 11 18  f  9  5  f 21 23
f 16 12  f 12  5  f 12 16  f  7  1  f 12 20
;
```

The following statement opens the HTML destination:

```
ods _all_ close;
ods html body='glmb.htm' contents='glmc.htm' frame='glmf.htm' style=HTMLBlue;
```

The ODS HTML statement specifies the body filename, generates a table of contents for the output, and generates a frame to contain the body and table of contents. The following statements perform the analysis:

```
proc glm data=drugtest;
   class drug;
   model PostTreatment = drug | PreTreatment / solution;
   lsmeans drug / stderr pdiff;
   ods output LSMeans=lsmeans;
run; quit;
```

The ODS OUTPUT statement writes the table of LS-means to the data set named lsmeans. PROC GLM performs an analysis of covariance and computes LS-means for the variable Drug.

The following steps demonstrate how you can create links to connect the results of different analyses. In this example, the table of LS-means is graphically summarized in a horizontal bar chart. Each bar is linked to a plot that displays the relationship between the PostTreatment response variable and the PreTreatment variable for the drug that corresponds to the bar.

NOTE: PROC GLM can use ODS Graphics to create LS-means graphs that are different from the one constructed here. You do not have to run the following steps to get PROC GLM's standard LS-means plots.

The following DATA step creates a new variable named DrugClick that matches each drug value with an HTML file:

```
data lsmeans;
   set lsmeans;
   if drug='a' then DrugClick='drug1.htm';
   if drug='d' then DrugClick='drug2.htm';
   if drug='f' then DrugClick='drug3.htm';
run;
```

The variable DrugClick is used in the chart. The variable provides the connection information for linking the two parts of the analysis together. The files referred to in these statements are created in a later step. The following statements create the chart:

```
ods graphics / imagemap=yes height=2in width=6.4in;

proc sgplot data=lsmeans;
   title 'Chart of LS-Means for Drug Type';
   hbar drug / response=lsmean stat=mean
               url=drugclick;
   footnote j=l 'Click on the bar to see a plot of PostTreatment '
                'versus PreTreatment for the corresponding drug.';
   format lsmean 6.3;
run;

ods graphics off;
footnote;
ods html close;
```

The chart is created with the ODS Graphics procedure SGPLOT. For more information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS." The ODS GRAPHICS statement is not required before you run SG procedures. However, in this case, it is necessary to specify IMAGEMAP=YES so that the URL= option works properly. The size of the graph is also specified with the HEIGHT= and WIDTH= options. PROC SGPLOT is used, and the HBAR statement requests a horizontal bar chart for the variable Drug. The lengths of the bars represent the values of the LSMean variable. The URL= option specifies the variable DrugClick as the HTML linking variable. The FOOTNOTE statement provides text that indicates how to use the links in the graph.

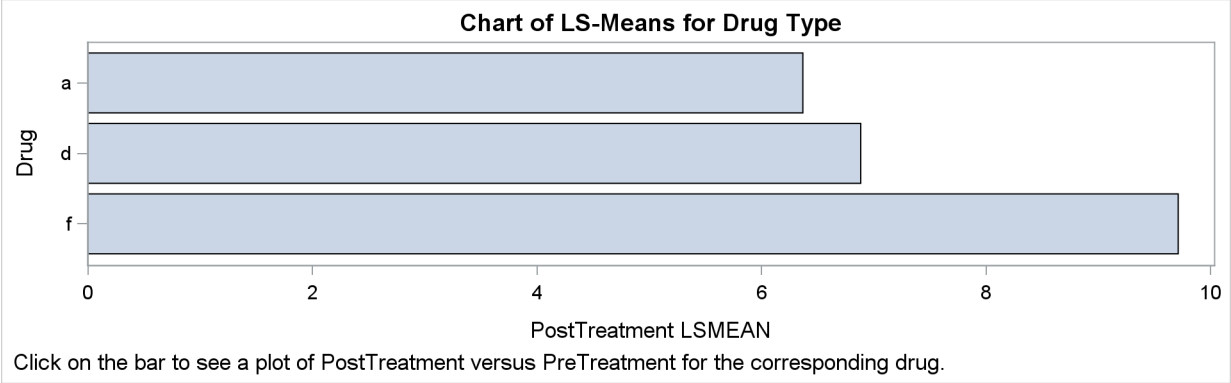
The following statements provide the second analysis. The three files referred to by the DrugClick variable are created as follows:

```
ods html body='drug1.htm' newfile=page style=HTMLBlue;

proc sgplot data=drugtest;
   title 'Plot of PostTreatment versus PreTreatment';
   scatter y=PostTreatment x=PreTreatment;
   by drug notsorted;
run;
ods html close;
```

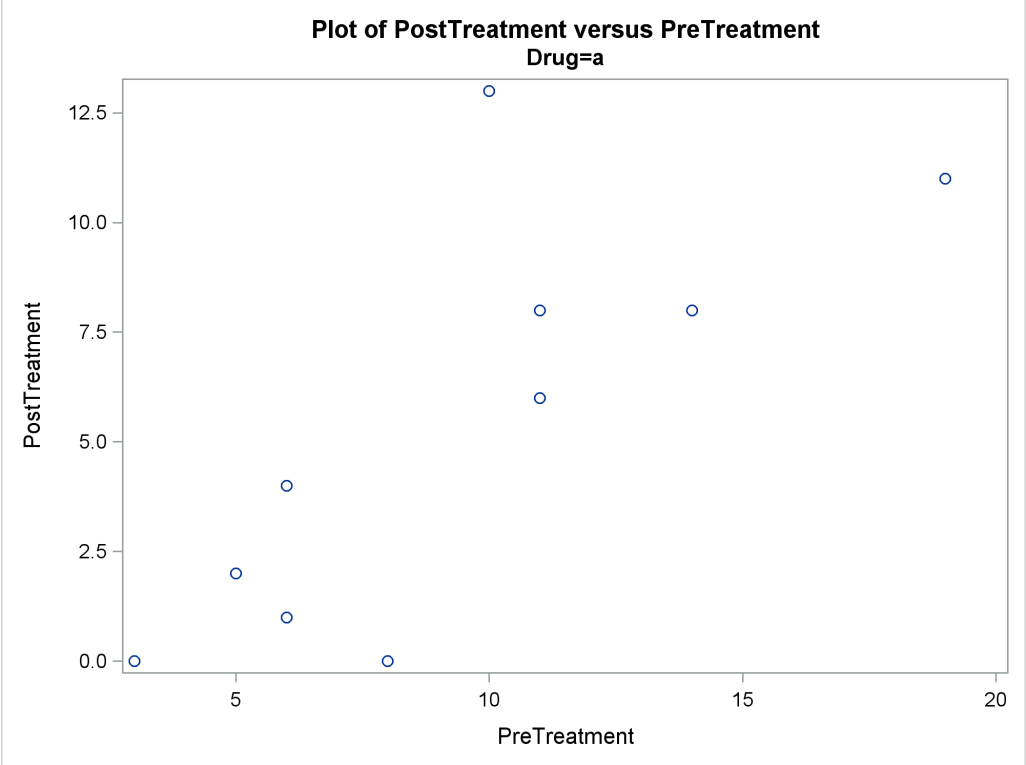
The NEWFILE= option in the ODS HTML statement creates a new HTML file for each page of output. (Page breaks occur only when a procedure explicitly starts a new page.) The NEWFILE= option also increments the filename numeric suffix for each new HTML file created, with the first filename corresponding to that given in the BODY= option, *drug1.htm*.

PROC SGPLOT is used, producing a plot of the variable PostTreatment versus the variable PreTreatment for each value of the Drug variable. Three plots are created, and each plot is contained in a separate HTML file. The files are named *drug1.htm*, *drug2.htm*, and *drug3.htm*. The filenames match those filenames specified as values of the DrugClick variable. By default, the HTML files are generated in your current working directory. You can instead specify a path, such as **frame='html/drug2.htm'**, to put a file in a subdirectory. The chart in Output 20.9.1 displays the difference in LS-means for each drug type. When you click on a bar that represents a value of the variable Drug, the browser opens the plot of PostTreatment versus PostTreatment variables that corresponds to that value of the variable Drug. Output 20.9.2 displays the plots for each drug type.

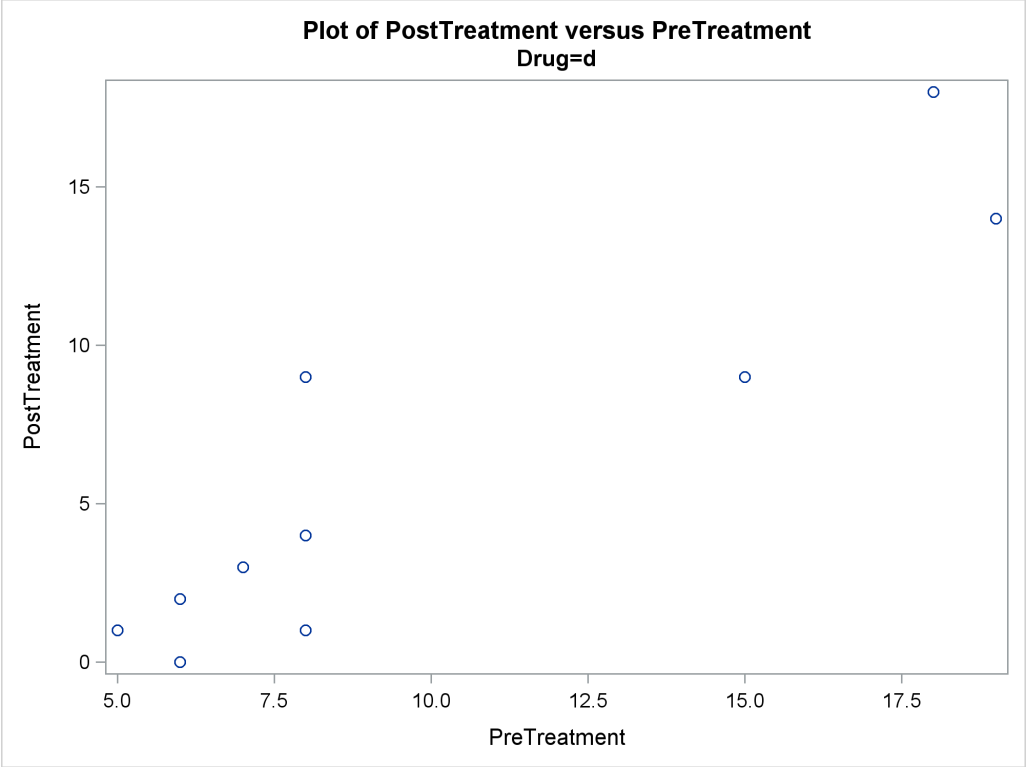
Output 20.9.1 Bar Chart of LS-Means by Drug Type with Links to Plots

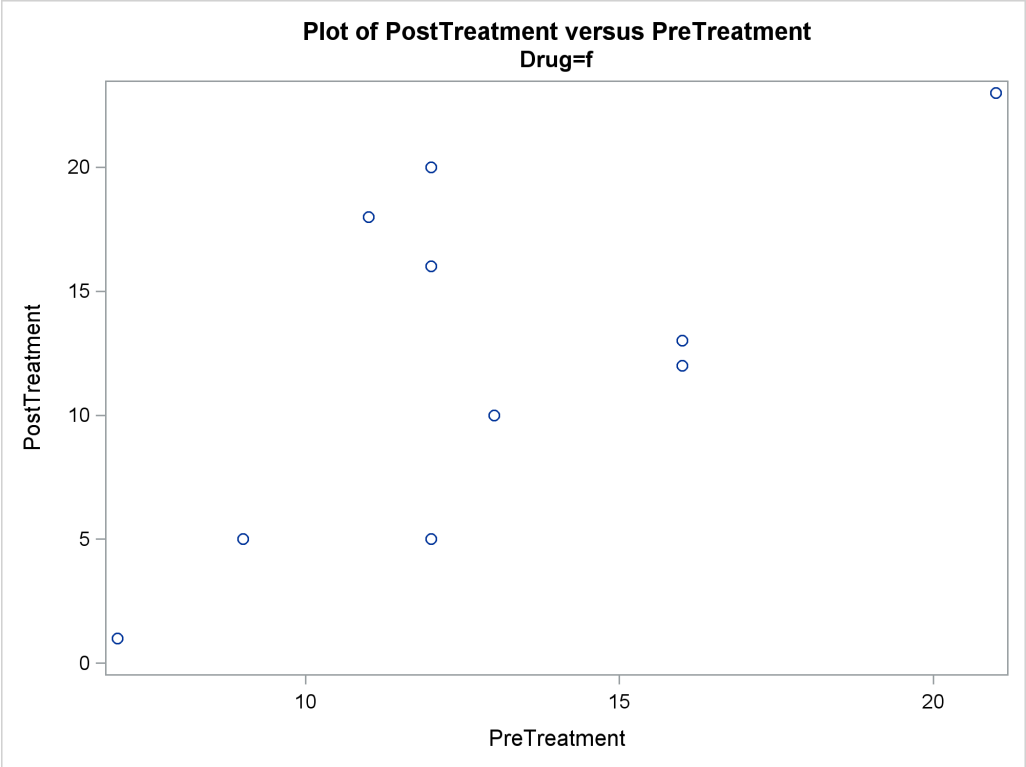


Output 20.9.2 Plots by Drug Type



Output 20.9.2 *continued*





Example 20.10: Correlation and Covariance Matrices

This example demonstrates how you can use ODS to set the background color of individual cells in a table. The color is set to reflect the magnitude of the value in the cell. You can use color to call attention to larger values and to see the pattern in the data in a way that is hard to visualize just by looking at the numbers. This is illustrated with correlation and covariance matrices. The data for this first part of this example are ratings of automobiles. The following statements create the data set:

```
title 'Rating of Automobiles';

data cars;
   input Origin $ 1-8 Make $ 10-19 Model $ 21-36
         (MPG Reliability Acceleration Braking Handling Ride
          Visibility Comfort Quiet Cargo) (1.);
   datalines;
GMC       Buick      Century          3334444544
GMC       Buick      Electra          2434453555
GMC       Buick      Lesabre          2354353545

   ... more lines ...

GMC       Pontiac    Sunbird          3134533234
;
```

The following steps edit the template that PROC CORR uses to display the correlation matrix. The CELLSTYLE statement sets the background color to light gray for correlations equal to 1 or –1. Values less than –0.75 or greater than 0.75 are set to red. Values less than –0.50 or greater than 0.50 are set to blue. Values less than –0.25 or greater than 0.25 are set to cyan. Values in the range –0.25 to 0.25 are set to white. PROC CORR is then run using the custom template. Finally, the default template is restored. The following statements produce Output 20.10.1:

```
proc template;
   edit Base.Corr.StackedMatrix;
      column (RowName RowLabel) (Matrix) * (Matrix2);
      edit matrix;
         cellstyle _val_  = -1.00 as {backgroundcolor=CXEEEEEE},
                   _val_ <= -0.75 as {backgroundcolor=red},
                   _val_ <= -0.50 as {backgroundcolor=blue},
                   _val_ <= -0.25 as {backgroundcolor=cyan},
                   _val_ <=  0.25 as {backgroundcolor=white},
                   _val_ <=  0.50 as {backgroundcolor=cyan},
                   _val_ <=  0.75 as {backgroundcolor=blue},
                   _val_ <   1.00 as {backgroundcolor=red},
                   _val_  =  1.00 as {backgroundcolor=CXEEEEEE};
      end;
   end;
run;

ods _all_ close;
ods html body='corr.html' style=HTMLBlue;
```

```
proc corr data=cars noprob;
   ods select PearsonCorr;
run;

ods html close;
ods html;
ods pdf;

proc template;
   delete Base.Corr.StackedMatrix / store=sasuser.templat;
run;
```

Output 20.10.1 Correlation Matrix from PROC CORR

Rating of Automobiles

The CORR Procedure

Pearson Correlation Coefficients, N = 50

	MPG	Reliability	Acceleration	Braking	Handling	Ride	Visibility	Comfort	Quiet	Cargo
MPG	1.00000	0.22003	-0.19454	0.41475	0.25594	-0.23705	0.67924	-0.06567	-0.49128	-0.03075
Reliability	0.22003	1.00000	-0.08512	0.25881	-0.09443	0.27406	0.33356	0.36607	0.45302	0.35261
Acceleration	-0.19454	-0.08512	1.00000	0.06688	0.07119	0.33888	-0.13280	0.06369	0.00934	-0.12112
Braking	0.41475	0.25881	0.06688	1.00000	0.22335	0.30309	0.44938	0.26165	0.00164	0.20880
Handling	0.25594	-0.09443	0.07119	0.22335	1.00000	0.12435	0.12599	-0.07516	-0.02418	-0.14274
Ride	-0.23705	0.27406	0.33888	0.30309	0.12435	1.00000	0.16114	0.75173	0.48498	0.39108
Visibility	0.67924	0.33356	-0.13280	0.44938	0.12599	0.16114	1.00000	0.29830	-0.18347	0.35585
Comfort	-0.06567	0.36607	0.06369	0.26165	-0.07516	0.75173	0.29830	1.00000	0.44917	0.53836
Quiet	-0.49128	0.45302	0.00934	0.00164	-0.02418	0.48498	-0.18347	0.44917	1.00000	0.33846
Cargo	-0.03075	0.35261	-0.12112	0.20880	-0.14274	0.39108	0.35585	0.53836	0.33846	1.00000

The preceding statements used a small number of discrete colors to show the range of values. In contrast, the following statements use a color gradient. The SAS autocall macro **Paint** is available for generating the CELLSTYLE colors list with a list of interpolated colors. If your site has installed the autocall libraries supplied by the SAS System and uses the standard configuration of software supplied by the SAS System, you need to ensure that the SAS System option MAUTOSOURCE is in effect before you begin using autocall macros. The macros do not have to be included (for example, with a %INCLUDE statement). They can be called directly once they are properly installed. For more information about autocall libraries, see *SAS Macro Language: Reference*.

Usually, you can use the **Paint** macro by specifying a list of values and a list of colors. Here is an example for values that range from 0 to 10:

```
%paint(values=0 to 10 by 0.5,
       colors=white cyan blue magenta red)

proc print data=colors;
run;
```

The **Paint** macro prints the following information to the SAS log:

```
Legend:
        0 = White
      2.5 = Cyan
        5 = Blue
      7.5 = Magenta
       10 = Red
```

A value of 0 maps to white, a value of 2.5 maps to cyan, values in the range 0 to 2.5 map to colors in the range from white to cyan, and so on. The **Paint** macro for this step creates an output data set, Colors, which is shown in Output 20.10.2.

Output 20.10.2 Color Interpolation

Rating of Automobiles

Obs	Start	_RGB_
1	0.0	CXFFFFFF
2	0.5	CXCBFFFF
3	1.0	CX97FFFF
4	1.5	CX63FFFF
5	2.0	CX2FFFFF
6	2.5	CX05FFFF
7	3.0	CX00D1FF
8	3.5	CX009CFF
9	4.0	CX0068FF
10	4.5	CX0034FF
11	5.0	CX0000FF
12	5.5	CX3400FF
13	6.0	CX6800FF
14	6.5	CX9C00FF
15	7.0	CXD100FF
16	7.5	CXFA00FF
17	8.0	CXFF00D1
18	8.5	CXFF009C
19	9.0	CXFF0068
20	9.5	CXFF0034
21	10.0	CXFF0000

This shows the color interpolation for a series of points. You could use a smaller BY value in the **Paint** macro to get more points along the color gradient. However, a few dozen colors are usually sufficient for most purposes.

The following steps use the **Paint** macro to create a color gradient for a correlation matrix, edit the template, display the results, and restore the default template:

```
%paint(values=-1 to 1 by 0.05, macro=setstyle,
       colors=CXEEEEEE red magenta blue cyan white
              cyan blue magenta red CXEEEEEE
              -1 -0.99 -0.75 -0.5 -0.25 0 0.25 0.5 0.75 0.99 1)

proc template;
   edit Base.Corr.StackedMatrix;
      column (RowName RowLabel) (Matrix) * (Matrix2);
      edit matrix;
         %setstyle(backgroundcolor)
      end;
   end;
run;

ods _all_ close;
ods html body='corr.html' style=HTMLBlue;
proc corr data=cars noprob;
   ods select PearsonCorr;
run;
ods html close;
ods html;
ods pdf;

proc template;
   delete Base.Corr.StackedMatrix / store=sasuser.templat;
run;
```

The VALUES= option creates a range of values from –1 to 1 with an increment of 0.05. The **Paint** macro generates a CELLSTYLE **_val_ <=** *value* **as {backgroundcolor=** *color***},** line for each value in the list. Specifically, it generates a macro named SETSTYLE (from the MACRO= option) that contains the entire CELLSTYLE statement for use in PROC TEMPLATE. The argument to the macro is the option that you want to set. In this case, it is the background color. You could specify **foreground** instead to set the color of the numbers themselves. The first part of the generated statement is as follows:

```
cellstyle _val_<=-1 as {backgroundcolor=CXEFEEEE},
          _val_<=-0.95 as {backgroundcolor=CXFF0020},
          _val_<=-0.9 as {backgroundcolor=CXFF0062},
          _val_<=-0.85 as {backgroundcolor=CXFF008D},
          _val_<=-0.8 as {backgroundcolor=CXFF00CF},
```

The color mapping for a correlation matrix can be a bit more involved than it is for most tables. This is because you might want the maximum correlations, 1 and –1, to be displayed using colors outside the gradient that is used for other values. Usually, you specify the color list, and the **Paint** macro maps the first color to the minimum value, the last color to the maximum value, and colors in between using equal increments and values based on the minimum and maximum. Alternatively, you can provide these values, as shown in this example. The legend, displayed in the SAS log, is as follows for the **Paint** macro step:

```
Legend:
      -1 = CXEEEEEE
   -0.99 = Red
   -0.75 = Magenta
    -0.5 = Blue
   -0.25 = Cyan
       0 = White
    0.25 = Cyan
```

```
  0.5 = Blue
 0.75 = Magenta
 0.99 = Red
    1 = CXEEEEEE
```

Values in the range –0.99 to 0.99 follow the interpolation red to magenta to blue to cyan to white to cyan to blue to magenta to red. Of course, the actual correlations for these data do not span this entire range, so a pure red background does not appear in the matrix. Correlations of 1 and –1 are displayed as light gray. The resulting correlation matrix is displayed in Output 20.10.3. Notice that there are now a number of shades of colors, particularly shades of blues, not just a few discrete colors. The largest values are displayed in shades of purple and magenta.

Output 20.10.3 Correlation Matrix from PROC CORR with a Color Gradient

Rating of Automobiles

The CORR Procedure

Pearson Correlation Coefficients, N = 50

	MPG	Reliability	Acceleration	Braking	Handling	Ride	Visibility	Comfort	Quiet	Cargo
MPG	1.00000	0.22003	-0.19454	0.41475	0.25594	-0.23705	0.67924	-0.06567	-0.49128	-0.03075
Reliability	0.22003	1.00000	-0.08512	0.25881	-0.09443	0.27406	0.33356	0.36607	0.45302	0.35261
Acceleration	-0.19454	-0.08512	1.00000	0.06688	0.07119	0.33888	-0.13280	0.06369	0.00934	-0.12112
Braking	0.41475	0.25881	0.06688	1.00000	0.22335	0.30309	0.44938	0.26165	0.00164	0.20880
Handling	0.25594	-0.09443	0.07119	0.22335	1.00000	0.12435	0.12599	-0.07516	-0.02418	-0.14274
Ride	-0.23705	0.27406	0.33888	0.30309	0.12435	1.00000	0.16114	0.75173	0.48498	0.39108
Visibility	0.67924	0.33356	-0.13280	0.44938	0.12599	0.16114	1.00000	0.29830	-0.18347	0.35585
Comfort	-0.06567	0.36607	0.06369	0.26165	-0.07516	0.75173	0.29830	1.00000	0.44917	0.53836
Quiet	-0.49128	0.45302	0.00934	0.00164	-0.02418	0.48498	-0.18347	0.44917	1.00000	0.33846
Cargo	-0.03075	0.35261	-0.12112	0.20880	-0.14274	0.39108	0.35585	0.53836	0.33846	1.00000

Next, the same technique is used to display the covariance and correlation matrices of a heteroscedastic autoregressive model. The data are based on the famous growth measurement data of Pothoff and Roy (1964), but are modified here to illustrate the technique of painting the entries of a matrix. The data consist of four repeated growth measurements of 11 girls and 16 boys. The measurements from two adjacent children in the original data were combined and rearranged here to emulate a repeated measures sequence with eight observations. The following statements create the data set:

```
title 'Analysis of Repeated Growth Measures';

data pr;
   input Person Gender $ y1 y2 y3 y4 y5 y6 y7 y8;
   array y{8};
   do time=5,7,8,4,3,2,1;
      Response = y{time};
      Age      = time+7;
      output;
   end;
   datalines;
 1   F   21.0  20.0  21.5  23.0  21.0  21.5  24.0  25.5
 2   F   20.5  24.0  24.5  26.0  23.5  24.5  25.0  26.5
```

```
 3   F   21.5  23.0  22.5  23.5  20.0  21.0  21.0  22.5
 4   F   21.5  22.5  23.0  25.0  23.0  23.0  23.5  24.0
 5   F   20.0  21.0  22.0  21.5  16.5  19.0  19.0  19.5
 6   F   24.5  25.0  28.0  28.0  26.0  25.0  29.0  31.0
 7   M   21.5  22.5  23.0  26.5  23.0  22.5  24.0  27.5
 8   M   25.5  27.5  26.5  27.0  20.0  23.5  22.5  26.0
 9   M   24.5  25.5  27.0  28.5  22.0  22.0  24.5  26.5
10   M   24.0  21.5  24.5  25.5  23.0  20.5  31.0  26.0
11   M   27.5  28.0  31.0  31.5  23.0  23.0  23.5  25.0
12   M   21.5  23.5  24.0  28.0  17.0  24.5  26.0  29.5
13   M   22.5  25.5  25.5  26.0  23.0  24.5  26.0  30.0
;
```

The following statements create a macro that sets colors for the covariance matrix (SETSTYLE1), create a macro that sets colors for the correlation matrix (SETSTYLE2), edit the templates, run the analysis with PROC GLIMMIX, and restore the default templates:

```
* You need to run the analysis once to know that 20 is a good maximum;
%paint(values=0 to 20 by 0.25,
       colors=cyan blue magenta red, macro=setstyle1)

%paint(values=0 to 1 by 0.05,
       colors=cyan blue magenta red, macro=setstyle2)

proc template;
   edit Stat.Glimmix.V;
      column Subject Index Row Col;
      edit Col;
         %setstyle1(backgroundcolor)
      end;
   end;
   edit Stat.Glimmix.VCorr;
      column Subject Index Row Col;
      edit Col;
         %setstyle2(backgroundcolor)
      end;
   end;
run;

ods _all_ close;
ods html body='ar1.html' style=HTMLBlue;
proc glimmix data=pr;
   class person gender time;
   model response = gender age gender*age;
   random _residual_ / sub=person type=arh(1) v residual vcorr;
   ods select v vcorr;
run;
ods html close;
ods html;
ods pdf;

proc template;
   delete Stat.Glimmix.V / store=sasuser.templat;
   delete Stat.Glimmix.VCorr / store=sasuser.templat;
run;
```

The results are displayed in Output 20.10.4 and Output 20.10.5. Both the covariance and correlation matrices have a structure that is more obvious when colors are added to the display. In particular, the colors clearly show the banded structure of the correlation matrix.

Output 20.10.4 Heteroscedastic AR(1) Covariance Matrix

Analysis of Repeated Growth Measures

The GLIMMIX Procedure

Estimated V Matrix for Person 1

Row	Col1	Col2	Col3	Col4	Col5	Col6	Col7
1	19.1973	10.5505	7.3707	4.5756	2.4983	1.4814	0.9697
2	10.5505	11.8104	8.2508	5.1220	2.7966	1.6582	1.0855
3	7.3707	8.2508	11.7407	7.2885	3.9795	2.3596	1.5446
4	4.5756	5.1220	7.2885	9.2159	5.0318	2.9836	1.9530
5	2.4983	2.7966	3.9795	5.0318	5.5959	3.3181	2.1720
6	1.4814	1.6582	2.3596	2.9836	3.3181	4.0075	2.6232
7	0.9697	1.0855	1.5446	1.9530	2.1720	2.6232	3.4976

Output 20.10.5 Heteroscedastic AR(1) Correlation Matrix

Estimated V Correlation Matrix for Person 1

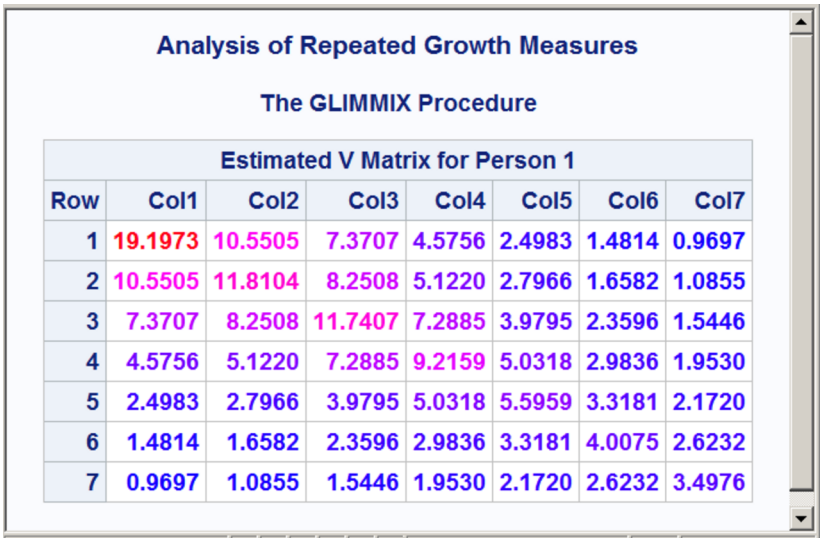
Row	Col1	Col2	Col3	Col4	Col5	Col6	Col7
1	1.0000	0.7007	0.4910	0.3440	0.2410	0.1689	0.1183
2	0.7007	1.0000	0.7007	0.4910	0.3440	0.2410	0.1689
3	0.4910	0.7007	1.0000	0.7007	0.4910	0.3440	0.2410
4	0.3440	0.4910	0.7007	1.0000	0.7007	0.4910	0.3440
5	0.2410	0.3440	0.4910	0.7007	1.0000	0.7007	0.4910
6	0.1689	0.2410	0.3440	0.4910	0.7007	1.0000	0.7007
7	0.1183	0.1689	0.2410	0.3440	0.4910	0.7007	1.0000

Alternatively, you could just use the **`Paint`** macro to do the color interpolation and use its output data set to create other types of style effects. The following statements show one way to set the font to bold and set the foreground color based on the values of the covariances:

```
%let inc = 0.25;

%paint(values=0 to 20 by &inc, colors=blue magenta red)

data cntlin;
   set colors;
   fmtname = 'paintfmt';
   label = _rgb_;
   end = start + &inc;
   keep start end label fmtname;
run;

proc format cntlin=cntlin;
run;

proc template;
   edit Stat.Glimmix.V;
      column Subject Index Row Col;
      edit Col;
         style = {foreground=paintfmt8. font_weight=bold};
      end;
   end;
run;

ods _all_ close;
ods html body='ar1.html' style=HTMLBlue;
proc glimmix data=pr;
   class person gender time;
   model response = gender age gender*age;
   random _residual_ / sub=person type=arh(1) v residual;
   ods select v;
run;
ods html close;
ods html;
ods pdf;

proc template;
   delete Stat.Glimmix.V / store=sasuser.templat;
run;
```

The **Paint** macro creates the SAS data set Colors with the result of the interpolation. This data set can be processed to create a format. The DATA step creates a range of values from Start to End and assigns a color to Label based on the color computed by the **Paint** macro. This data set is input to PROC FORMAT to create the format PAINTFMT. PROC TEMPLATE uses this format to set the color of the values in the table. The cell value is evaluated using the specified FOREGROUND= format for every cell in the table, and the appropriate color is assigned. PROC GLIMMIX does the analysis, and the results are displayed in Output 20.10.6.

Output 20.10.6 Heteroscedastic AR(1) Covariance Matrix



Analysis of Repeated Growth Measures

The GLIMMIX Procedure

Estimated V Matrix for Person 1

Row	Col1	Col2	Col3	Col4	Col5	Col6	Col7
1	19.1973	10.5505	7.3707	4.5756	2.4983	1.4814	0.9697
2	10.5505	11.8104	8.2508	5.1220	2.7966	1.6582	1.0855
3	7.3707	8.2508	11.7407	7.2885	3.9795	2.3596	1.5446
4	4.5756	5.1220	7.2885	9.2159	5.0318	2.9836	1.9530
5	2.4983	2.7966	3.9795	5.0318	5.5959	3.3181	2.1720
6	1.4814	1.6582	2.3596	2.9836	3.3181	4.0075	2.6232
7	0.9697	1.0855	1.5446	1.9530	2.1720	2.6232	3.4976

Many other effects could be achieved by using this approach and different options in the STYLE= specification.

References

Cole, J. W. L., and Grizzle, J. E. (1966). "Applications of Multivariate Analysis of Variance to Repeated Measures Experiments." *Biometrics* 22:810–828.

Hemmerle, W. J., and Hartley, H. O. (1973). "Computing Maximum Likelihood Estimates for the Mixed AOV Model Using the W-Transformation." *Technometrics* 15:819–831.

Olinger, C. R., and Tobias, R. D. (1998). "It Chops, It Dices, It Makes Julienne Slices! ODS for Data Analysis—Output As-You-Like-It in Version 7." In *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*, 1271–1291. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/sugi23/Stats/p233.pdf.

Pothoff, R. F., and Roy, S. N. (1964). "A Generalized Multivariate Analysis of Variance Model Useful Especially for Growth Curve Problems." *Biometrika* 51:313–326.

Snedecor, G. W., and Cochran, W. G. (1967). *Statistical Methods*. 6th ed. Ames: Iowa State University Press.

Subject Index

Syntax Index