SAS/STAT® 9.2 User's Guide
Using the Output Delivery System
(Book Excerpt)

SAS® Documentation

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), PDF, PostScript, SAS listing (the default), or other formats
- create SAS data sets directly from tables or plots
- select or exclude individual pieces of output

- customize the layout, format, headers, and style of your output
- produce graphs with ODS Graphics in many procedures (see Chapter 21, "Statistical Graphics Using ODS")

This chapter discusses some typical applications of ODS with SAS software. For complete documentation on the Output Delivery System, see the *SAS Output Delivery System: User's Guide*.

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

You define the form that the output should take when you specify an ODS destination. Some supported destinations are as follows:

- LISTING destination (the standard SAS listing), which is open by default
- HTML destination, for viewing in a browser
- RTF, for inclusion in Microsoft Word
- PDF, PostScript, and PCL, for high-fidelity printers
- OUTPUT destination, for saving results to SAS data sets
- DOCUMENT destination, 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 LISTING destination. You can specify an output style for each ODS destination. The style controls the foreground, background, colors, lines, fonts, and so on.

Each output object has an associated template, provided by SAS, that defines its presentation format. You can use the TEMPLATE procedure to view or alter these templates or to create new templates, 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*.

The following statements provide an example of temporarily closing the LISTING destination and making an output data set, Parm, from the parameter estimates table from PROC REG. Closing the

LISTING destination is not required, but it is done frequently in the examples in this chapter for efficiency. Closing the LISTING 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 20.1:

```
title 'Getting Started with ODS';

ods listing close;

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

ods listing;

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

Figure 20.1 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
```

The ODS OUTPUT statement contains a table name, an equal sign, and the name of the output SAS data set to create. The table names can be found using the ODS TRACE statement, which is described in the next section. Also see Example 20.4 for more information.

Paths and Selection

You use the ODS statement to provide instructions to ODS. You can use ODS statements 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 statistical style */
ods html style=Statistical;

/* 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;
```

In order to select, exclude, or modify a table, you must first know its name. You can obtain the table names in several ways:

- You can obtain table names from the individual procedure chapter or from the individual procedure section of the SAS online Help system. See the "ODS Table Names" section within the "Details" section of the procedure documentation chapter.
- You can use the SAS Results window to view the names of the tables created in your SAS session (see the section "ODS and the SAS Results Window" on page 440 for more information).
- You can use the ODS TRACE statement to find the names of tables created in your SAS session. The ODS TRACE statement writes identifying information to the SAS log or listing for each generated output table.

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 specific tables created in the REG procedure step:

```
ods trace on;
proc reg;
   model y=x;
   model z=x;
run;
ods trace off;
```

By default, the trace record is written to the SAS log. The output from this step is as follows:

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

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

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

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

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

Output Added:
-------------
Name:       ANOVA
Label:      Analysis of Variance
Template:   Stat.REG.ANOVA
Path:       Reg.MODEL2.Fit.z.ANOVA
-------------

Output Added:
-------------
Name:       FitStatistics
Label:      Fit Statistics
Template:   Stat.REG.FitStatistics
Path:       Reg.MODEL2.Fit.z.FitStatistics
-------------

Output Added:
-------------
Name:       ParameterEstimates
Label:      Parameter Estimates
Template:   Stat.REG.ParameterEstimates
Path:       Reg.MODEL2.Fit.z.ParameterEstimates
-------------
```

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.

The trace record contains the name of each created table and its associated label, template, and path. The label provides a description of the table. The path shows the output hierarchy for the table (see, for example, Figure 20.2). In this example, 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 (y or z), and a level for the table name (**`NObs`**, **`ANOVA`**, **`FitStatistics`**, **`ParameterEstimates`**, **`NObs`**, **`ANOVA`**, **`FitStatistics`**, **`ParameterEstimates`**). The path shown in the trace record is fully qualified. Often, you can omit levels and instead use a partially qualified path. A partially qualified path consists of any part of the full path that begins immediately after a period and continues to the end of the full path. For example, the full path for the parameter estimates for the first model in the preceding regression analysis is **`Reg.Model1.Fit.y.ParameterEstimates`**. This table can be referenced in any of the following ways:

`ParameterEstimates`	name
`y.ParameterEstimates`	partially qualified path
`fit.y.ParameterEstimates`	partially qualified path
`Model1.fit.y.ParameterEstimates`	partially qualified path
`Reg.Model1.fit.y.ParameterEstimates`	fully qualified path

When a procedure creates multiple tables that have the same name, as in the preceding example, you have several selection options. You can specify the name, a full path, or a partially qualified path when you refer to a table (for example, when you select it or exclude it from display). You can also specify a WHERE clause. For example, you can specify any of the following statements to display both tables of parameter estimates:

```
ods select ParameterEstimates;

ods select y.ParameterEstimates z.ParameterEstimates;

ods select Reg.Model1.Fit.y.ParameterEstimates
           Reg.Model2.Fit.z.ParameterEstimates;

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

There are other possibilities as well. The first ODS SELECT statement specifies the single name, which is shared by both tables. The second statement specifies a partially qualified path for both tables. The third statement specifies the full path for both tables. The fourth statement selects every object (table or graph) that contains the string **`Parameter`** anywhere in its path. Note that in the first three statements, selection is case insensitive. Any combination of uppercase and lowercase letters will work. This is not true in the fourth statement, which uses an ordinary SAS comparison of character strings. The following WHERE clause selection is case insensitive:

```
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 tables 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 select list, showing just the name and label paths, is as follows:

```
Path:       Reg.MODEL1.Fit.y.ANOVA
Label Path: 'The Reg Procedure'.'MODEL1'.'Fit'.y.'Analysis of Variance'
Path:       Reg.MODEL2.Fit.z.ANOVA
Label Path: 'The Reg Procedure'.'MODEL2'.'Fit'.z.'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 path or label path—for example, '.z.', the variable z ignoring any 'z' that might be in the middle of a word; '2.F', model 2 fit tables and any other table that has the string '2.F' in its path; and so on.

ODS records the specified table names in its internal selection or exclusion list, and then it processes the output it receives. Note that 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 tables 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 tables you want to select for all destinations) contains the tables **`ParameterEstimates`** and **`FitStatistics`**, which you specify with the following statement:

```
   ods select ParameterEstimates FitStatistics;
```

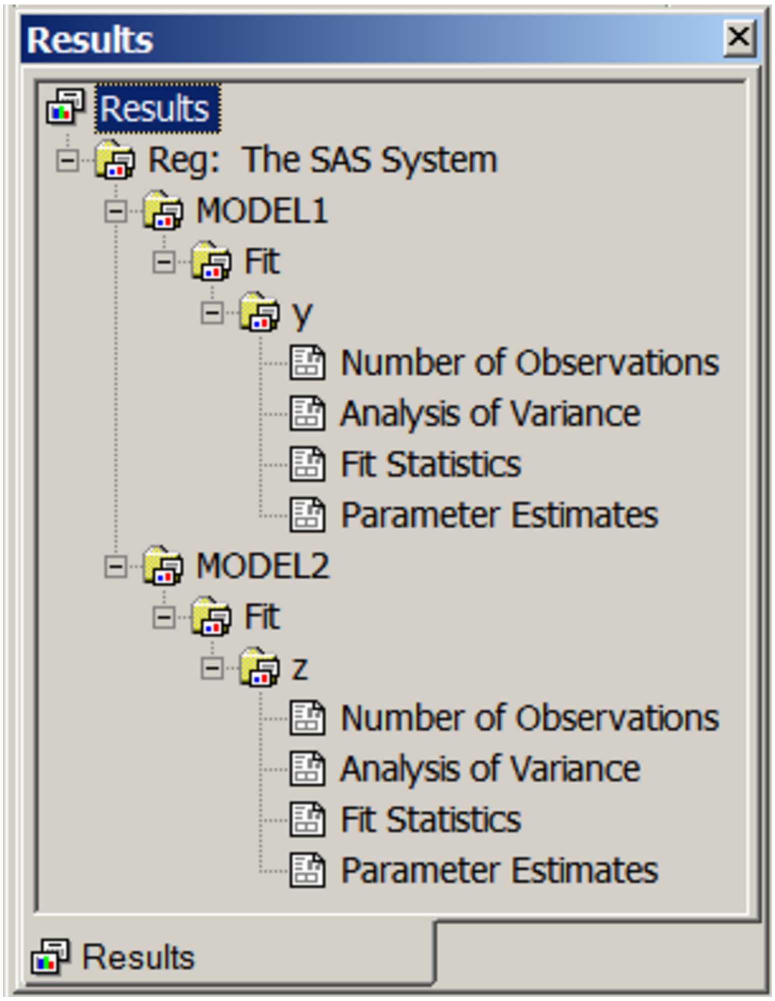
ODS then sends only the **`ParameterEstimates`** and **`FitStatistics`** tables to all open destinations except the LISTING destination. It sends only the **`ParameterEstimates`** table to the LISTING destination because the table **`FitStatistics`** is excluded from that destination.

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.

ODS and the SAS Results Window

The SAS Results window contains a running record of the output from your SAS session. In the display manager, select **View** ► **Results** to open the Results window. Figure 20.2 displays the Results window from the PROC REG step shown previously.

Figure 20.2 The Results Window from the SAS Explorer



```
Results
  Reg: The SAS System
    MODEL1
      Fit
        y
          Number of Observations
          Analysis of Variance
          Fit Statistics
          Parameter Estimates
    MODEL2
      Fit
        z
          Number of Observations
          Analysis of Variance
          Fit Statistics
          Parameter Estimates
```

When you click on the output table names in the Results window, you link directly to the output in the output window (or HTML browser if you specify the HTML destination). The Results window contains a row for each level of the path and for each table. You can also use the Results window to determine the names of the templates associated with each output table. Right-click on the table name and then select **Properties.** You can see all of the templates from the Results window by selecting **View** ► **Templates** ► **Sashelp.Tmplmst**. Then click on a product such as `Stat`, a procedure such as `REG`, and a template such as `ParameterEstimates`.

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 table 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 table or 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 reference the template. Graphical templates are discussed in more detail in Chapter 21, "Statistical Graphics Using ODS." A style template applies to an entire SAS job, 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=Statistical;
```

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 reference 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;
```

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

```
proc glm data=x;
   model y = x;
run; quit;
```

The ANOVA table from this step is displayed in Figure 20.3.

Figure 20.3 PROC GLM ANOVA Table with the Default Template

```
                              Getting Started with ODS

                                 The GLM Procedure

Dependent Variable: y

                                           Sum of
 Source                        DF         Squares     Mean Square    F Value    Pr > F

 Model                          1     77.05005975     77.05005975     190.53    <.0001

 Error                          8      3.23526554      0.40440819

 Corrected Total                9     80.28532529
```

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=x;
   model y = x;
   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.y.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;
```

Here are the results:

```
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 there 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 there 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 FORMAT_WIDTH=8 chooses the format for each column based on the maximum value in that column and an overall width of 8. Note that you are editing and not replacing the definition, so column header and other information in the definition is not lost. The following step uses the new templates:

```
proc glm data=x;
   model y = x;
run; quit;
```

The new ANOVA results, using the edited templates, are shown in Figure 20.4. You can see that the original results in Figure 20.3 have eight decimal places, whereas the new results in Figure 20.4 have only five decimal places and an overall format width of eight.

Figure 20.4 PROC GLM ANOVA Table after Template Customization

```
                          Getting Started with ODS

                             The GLM Procedure

Dependent Variable: y

                                               Sum of          Mean
  Source                            DF        Squares        Square    F Value    Pr > F

  Model                              1       77.05006      77.05006     190.53    <.0001

  Error                              8        3.23527       0.40441

  Corrected Total                    9       80.28533
```

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 by default stored in your SASUSER library. You can then modify the path that ODS uses to look up templates with the ODS PATH statement—for example, to 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 path. The following statements illustrate the ODS 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 file is open for storing new templates and retrieving templates for use. After that, the SASHELP.TMPLMST file is used, but it is open only for Read access. You cannot modify templates in SASHELP. The second and third statements add a template library to the front of this list in the current directory. The fourth statement shows the new path list, 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 showing 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;
   delete Stat.GLM.MS;
run;
```

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

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 but without displaying any output. You use an option (such as OUTEST= 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.

You can also use ODS to create output data sets by using the ODS OUTPUT statement. 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, for example, like this:

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

Note that ODS statements do not instruct a procedure to generate output. Instead, they specify how ODS should manage output once it is created. You must ensure that the proper procedure options are in effect, or the output will not be generated. For example, the following statements do not create the requested data set Parms:

```
proc glm;
   ods output ParameterEstimates=Parms;
   class x;
   model y=x;
run; quit;
```

This is because the SOLUTION option was not specified in the MODEL statement. 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;
   ods output ParameterEstimates=Parms;
   class x;
   model y=x / solution;
run; quit;
```

Examples: Using the Output Delivery System

This section provides examples of making 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=statistical;

proc ttest;
   class Gender;
   var Score;
run;

ods html close;
```

By default, the LISTING destination receives all output generated during your SAS session. In this example, the ODS HTML statement opens the HTML destination as well, and both destinations receive the generated output. Note that you must specify the following statement before you can view your output in a 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.

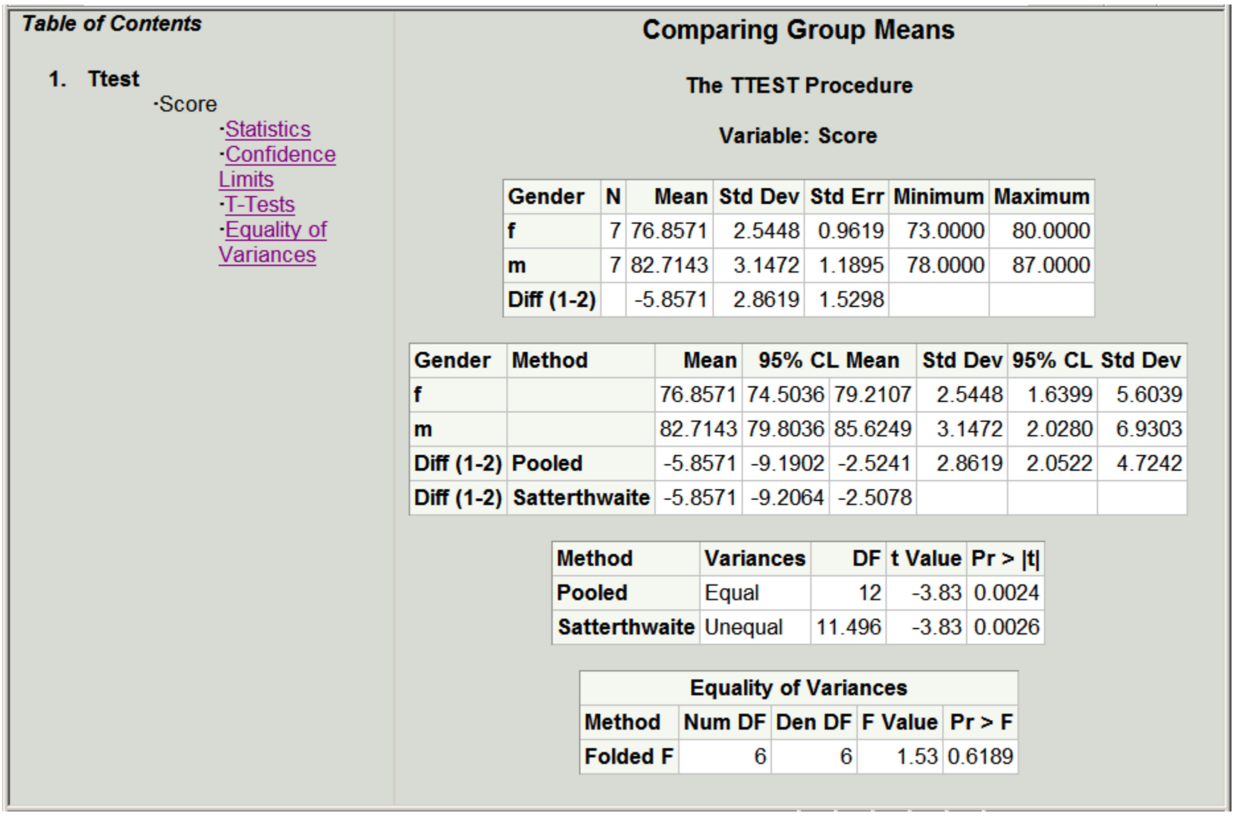
The following statements use ODS to display the output in HTML with a table of contents by using the STATISTICAL style of output, but this time closing the LISTING destination:

```
ods listing close;
ods html body='ttest.htm' contents='ttestc.htm' frame='ttestf.htm'
         style=statistical;

proc ttest;
   class Gender;
   var Score;
run;

ods html close;
ods listing;
```

The ODS HTML statement specifies three files. 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 paths, such as `frame='html/ttestf.htm'`, to put a file in a subdirectory.

Note that 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 table produced in the PROC TTEST step. You can select any label in the table of contents and the corresponding output will be displayed on the right side of the browser window.

Output 20.1.1 HTML Output with a Table of Contents, the FRAME='ttestf.htm' File



Comparing Group Means

The TTEST Procedure

Variable: Score

Gender	N	Mean	Std Dev	Std Err	Minimum	Maximum
f	7	76.8571	2.5448	0.9619	73.0000	80.0000
m	7	82.7143	3.1472	1.1895	78.0000	87.0000
Diff (1-2)		-5.8571	2.8619	1.5298		

Gender	Method	Mean	95% CL Mean		Std Dev	95% CL Std Dev	
f		76.8571	74.5036	79.2107	2.5448	1.6399	5.6039
m		82.7143	79.8036	85.6249	3.1472	2.0280	6.9303
Diff (1-2)	Pooled	-5.8571	-9.1902	-2.5241	2.8619	2.0522	4.7242
Diff (1-2)	Satterthwaite	-5.8571	-9.2064	-2.5078			

Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	12	-3.83	0.0024
Satterthwaite	Unequal	11.496	-3.83	0.0026

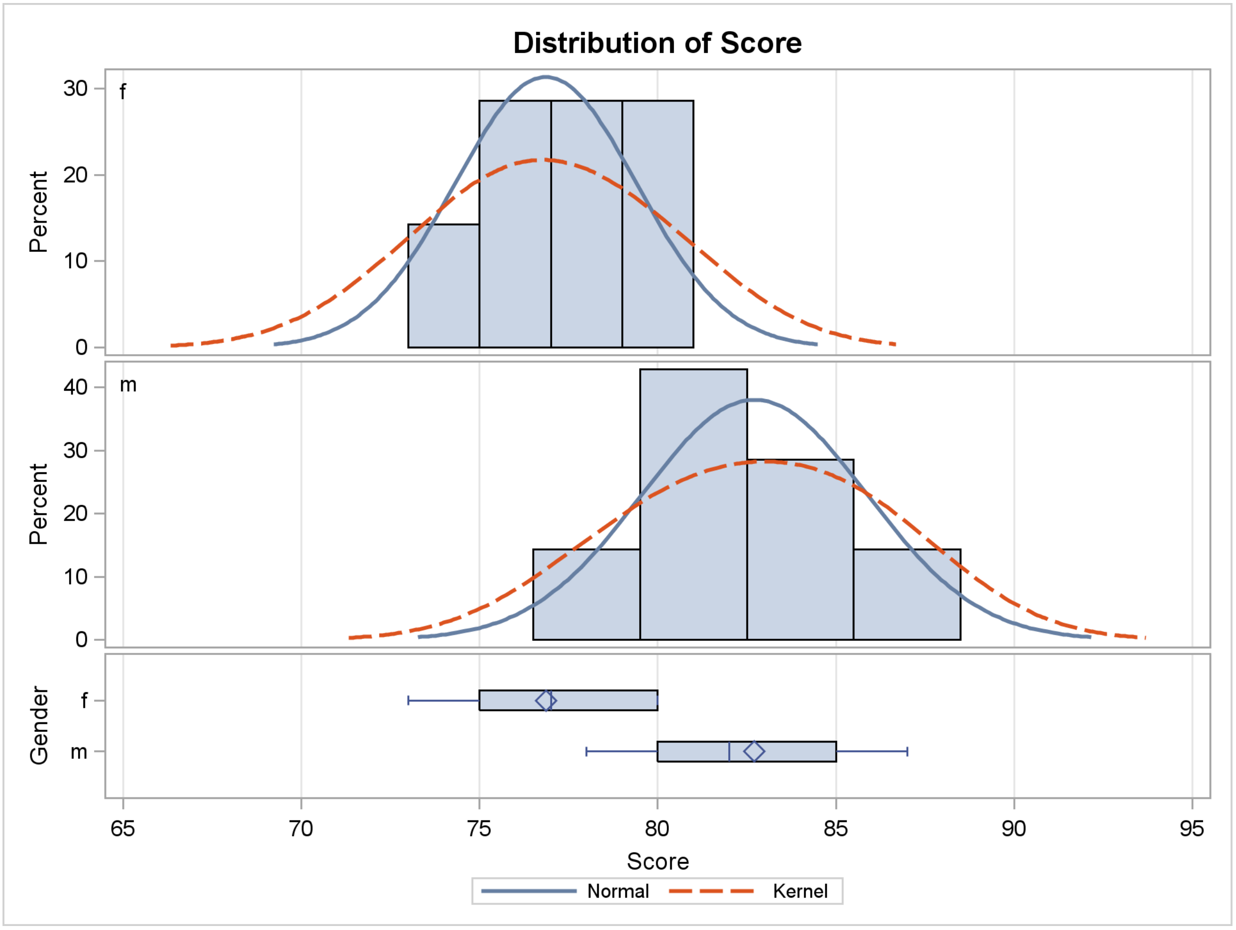
Equality of Variances				
Method	Num DF	Den DF	F Value	Pr > F
Folded F	6	6	1.53	0.6189

You could enable ODS Graphics by adding an ODS GRAPHICS ON statement as follows:

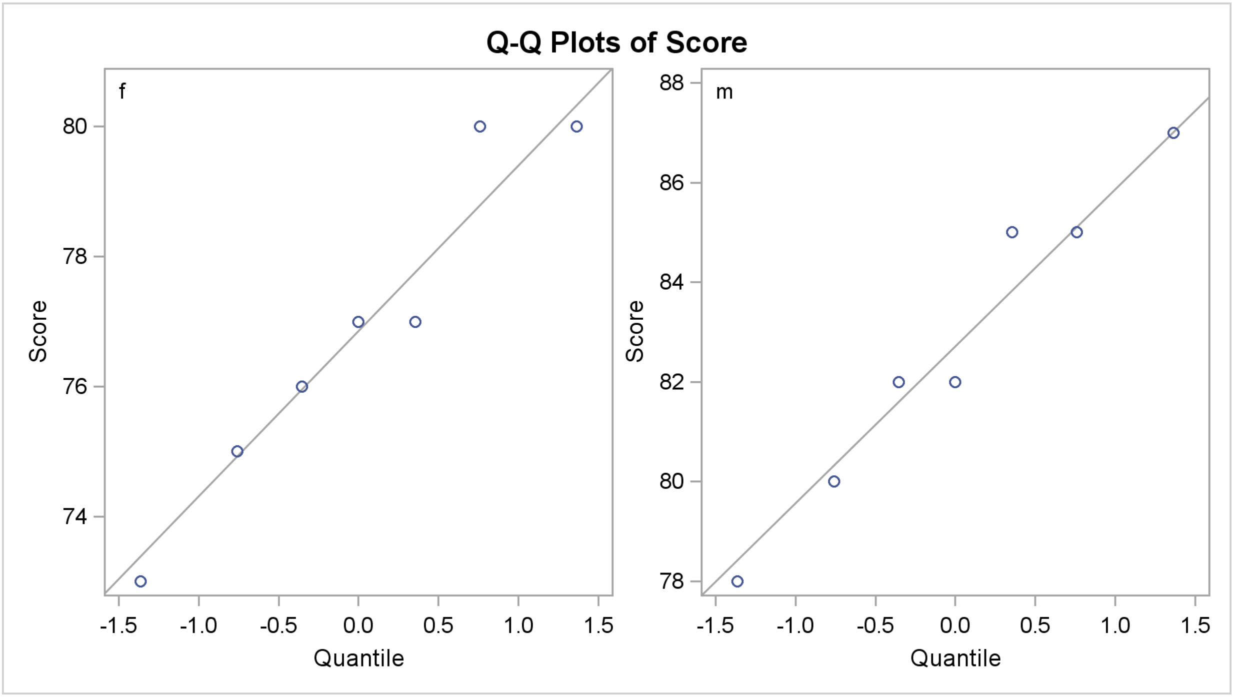
```
ods graphics on;
ods listing close;
ods html body='ttest.htm' contents='ttestc.htm' frame='ttestf.htm'
         style=statistical;

proc ttest;
   class Gender;
   var Score;
run;

ods html close;
ods listing;
ods graphics off;
```

Then in addition to the tables, you would also get the graphs shown in Figure 20.1.2. For general information about ODS Graphics, see Chapter 21, "Statistical Graphics Using ODS."

Output 20.1.2 PROC TTEST Graphs



Output 20.1.2 *continued*



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 specifies that only the two tables `ModelANOVA` and `Means` are to be delivered to the ODS destinations. In this example, no ODS destinations are explicitly opened. Therefore, only the default LISTING destination receives the procedure output. 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
-------------
```

Note that 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 wanted to select only one of them, you could specify either of the labels in the ODS SELECT statement instead of the name. You would 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). Note that 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 The 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

              Level of             --------------y--------------
              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 "Paths and Selection" on page 435.

Example 20.3: Excluding ODS Tables from Display

The following example demonstrates how you can use the ODS EXCLUDE statement to exclude particular tables 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 listing close;
ods html body='mixed.htm' contents='mixedc.htm' frame='mixedf.htm'
         style=statistical;

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 table 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 and LISTING destinations. 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:

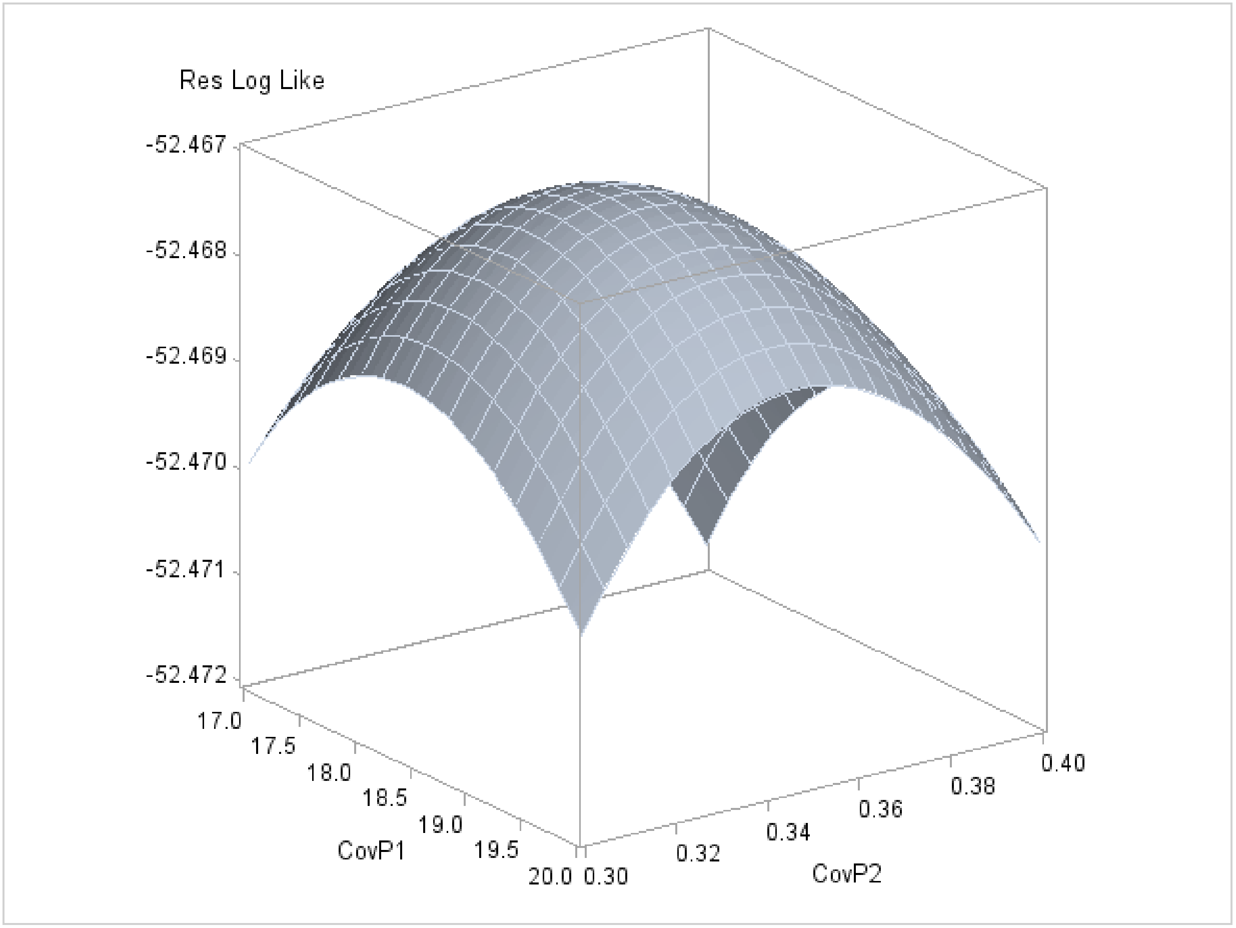
```
Current OVERALL exclude list is:
1. ParmSearch(PERSIST)
```

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 tables 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, the ODS OUTPUT statement writes the ODS table **ObStats** to a SAS data set named myObStats. The LISTING destination is closed so that no output is displayed. 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 listing close;
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= option in the ODS OUTPUT statement specifies that only the variables Car, Age, and Pred are written to the data set, and the Pred variable is renamed PredictedValue. The following statements sort the output data set myObStats, reopen the LISTING destination for output, and produce Output 20.4.1:

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

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

When a destination is closed, it remains closed until it is explicitly reopened.

Output 20.4.1 The ObStats Table Created as a SAS Data Set

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

                                      Predicted
                   Car        Age       Value

                  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 exhibiting 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 close the LISTING destination and sort the observations in the Color data set by the Region variable:

```
ods listing close;

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'));
```

Note that you can obtain the names of the tables created by any procedure in the individual procedure 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 tables with the ODS TRACE statement (see Example 20.4 and Example 20.2). 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 listing;
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. No ODS destinations are open until the ODS LISTING statement is encountered just prior to running the PRINT procedure.

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
```

Example 20.6: RUN-Group Processing

Some SAS procedures, such as PROC REG and PROC GLM, permit you to submit statements, followed by a RUN statement, followed by more statements and more RUN statements. Each group of statements, followed by a RUN statement, is called a RUN group. These procedures can produce several blocks of output for each of several RUN groups. The procedure stays active until a QUIT statement, a DATA statement, another PROC statement, or the end of the SAS session is encountered. However, ODS settings are by default cleared at RUN-group boundaries. In the following analysis, 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= option in the ODS OUTPUT statement is required to make this happen. The PERSIST= option maintains ODS settings across RUN statements for procedures that support RUN-group processing.

Consider the following population growth trends. The population of the United States from 1790 to 1970 is fit to linear and quadratic functions of time. Note that the quadratic term, YearSq, is created in the DATA step; this is necessary since polynomial effects such as Year*Year cannot be specified in the MODEL statement in PROC REG. The data are as follows:

```
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 statements, PROC REG is used, and the ODS OUTPUT statement with the PERSIST= option creates a data set with 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 the COVB matrix is requested. The RUN statement executes PROC REG and the model is fit, producing a covariance matrix of the estimates with two rows and two columns. The results are displayed in Output 20.6.1 and Output 20.6.2.

Output 20.6.1 Regression Results for the Model Population

```
                          US Population Study
               Concatenating Two Tables into One Data Set

                          The REG Procedure
                            Model: MODEL1
                    Dependent Variable: Population

               Number of Observations Read          19
               Number of Observations Used          19

                          Analysis of Variance

                                    Sum of          Mean
 Source                 DF         Squares        Square    F Value    Pr > F

 Model                   1           66336         66336     201.87    <.0001
 Error                  17      5586.29253     328.60544
 Corrected Total        18           71923

             Root MSE             18.12748    R-Square     0.9223
             Dependent Mean       69.76747    Adj R-Sq     0.9178
             Coeff Var            25.98271
```

Output 20.6.1 *continued*

```
                       Parameter Estimates

                         Parameter       Standard
 Variable      DF         Estimate          Error    t Value    Pr > |t|

 Intercept      1      -1958.36630      142.80455     -13.71      <.0001
 Year           1          1.07879        0.07593      14.21      <.0001
```

Output 20.6.2 CovB Matrix for the Model Population

```
                     Covariance of Estimates

            Variable          Intercept               Year

            Intercept      20393.138485       -10.83821461
            Year           -10.83821461       0.0057650078
```

In the next step, the YearSq variable is added to the model and the model is again fit, producing a covariance matrix of the estimates with three rows and three columns:

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

The new COVB matrix is displayed in Output 20.6.3.

Output 20.6.3 CovB Matrix for the Model Population

```
                       US Population Study
              Concatenating Two Tables into One Data Set

                        The REG Procedure
                         Model: MODEL1.1
                  Dependent Variable: Population

                     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
```

The PERSIST=RUN option maintains the ODS selection list across RUN statements for procedures that support RUN-group processing. If the PERSIST=RUN option is omitted, the selection list is cleared when the RUN statement is encountered and only the first COVB matrix is selected. 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 matrices is selected and output. The following statements display the ODS OUTPUT SAS data set and create Output 20.6.4:

```
proc print;
run;
```

Output 20.6.4 Results of the ODS OUTPUT Statement: Specifying the PERSIST Option

```
                              US Population Study
                    Concatenating Two Tables into One Data Set

 Obs _Run_  Model    Dependent  Variable     Intercept          Year        YearSq

  1      1 MODEL1    Population Intercept 20393.138485 -10.83821461              .
  2      1 MODEL1    Population Year      -10.83821461 0.0057650078              .
  3      2 MODEL1.1 Population Intercept 711450.62602 -757.2493826 0.2013282694
  4      2 MODEL1.1 Population Year      -757.2493826 0.8061328943 -0.000214361
  5      2 MODEL1.1 Population YearSq    0.2013282694 -0.000214361 5.7010894E-8
```

Note that even though the two COVB matrices do not have the same rows or columns, ODS automatically combines the two tables into one data set.

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 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 directly done 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 the levels of histamine in the animals' blood is measured 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 39.7 in Chapter 39, "The GLM Procedure." Here are the PROC GLM statements that you would run to perform the analysis:

```
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 containing the fully qualified 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 listing close;

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 listing;
```

The LISTING destination is closed so that no output is displayed. 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. This is 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 path (**`BetweenSubjects.ModelANOVA`**) in the second specification. For more information about names and qualified path names, see the discussion in the section "Paths and Selection" on page 435.

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 p;
   ods select position;
run;
```

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

Output 20.7.1 The 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:

```
ods listing;
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

                               Hypothesis
      Obs       Dependent          Type       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

               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
               6           .                .            24.03     0.0001
               7           .                .            24.03     0.0001
               8           .                .            24.03     0.0001
               9           .                .             5.78     0.0175
              10           .                .             5.78     0.0175
              11           .                .             5.78     0.0175
              12           .                .             5.78     0.0175
              13           .                .            21.31     0.0002
              14           .                .            21.31     0.0002
              15           .                .            21.31     0.0002
              16           .                .            21.31     0.0002
              17           .                .            12.48     0.0015
              18           .                .            12.48     0.0015
              19           .                .            12.48     0.0015
              20           .                .            12.48     0.0015

             Obs     Hypothesis                    Error

               1
               2
               3
               4
               5     Time                    Error SSCP Matrix
               6     Time                    Error SSCP Matrix
               7     Time                    Error SSCP Matrix
               8     Time                    Error SSCP Matrix
               9     Time_Drug               Error SSCP Matrix
              10     Time_Drug               Error SSCP Matrix
              11     Time_Drug               Error SSCP Matrix
              12     Time_Drug               Error SSCP Matrix
              13     Time_Depleted           Error SSCP Matrix
              14     Time_Depleted           Error SSCP Matrix
              15     Time_Depleted           Error SSCP Matrix
              16     Time_Depleted           Error SSCP Matrix
              17     Time_Drug_Depleted      Error SSCP Matrix
              18     Time_Drug_Depleted      Error SSCP Matrix
              19     Time_Drug_Depleted      Error SSCP Matrix
              20     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 will be 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

                                              Hypothesis
Obs   Source                 Dependent          Type      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

  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
  5      .             5.78  0.0175  Error SSCP Matrix  Hotelling-Lawley Trace
  6      .            21.31  0.0002  Error SSCP Matrix  Hotelling-Lawley Trace
  7      .            12.48  0.0015  Error SSCP Matrix  Hotelling-Lawley Trace


Obs          Value    Den DF    P-Value

  1      .                 .     .
  2      .                 .     .
  3      .                 .     .
  4     8.01087137         9     .
  5     1.92774470         9     .
  6     7.10373567         9     .
  7     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. (Note that 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 and equivalently, 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 will 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 corresponding to multivariate tests to asterisks (to see the footnote)
- add a column depicting 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. Note that 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 Template Procedure” in the *SAS Output Delivery System: User’s Guide*. The following statements display the HistTests data set by using the customized template:

```
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.

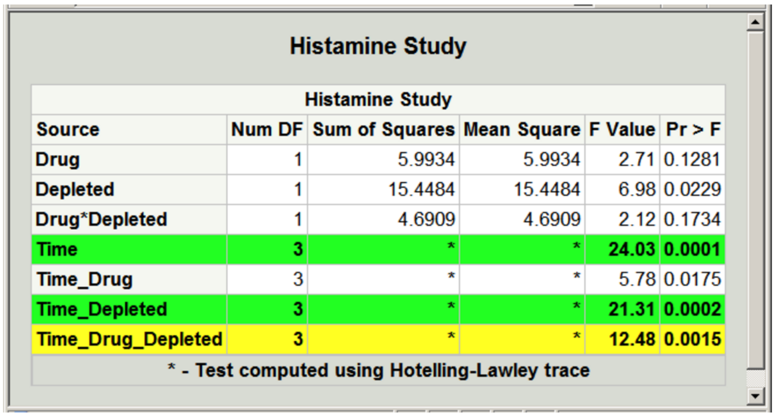
Output 20.7.5 Display of the Data Sets Using a Customized Template: Histamine Study

```
                          Histamine Study
           Listing of the Selections, Using a Customized Template

                          Histamine Study

                         Num     Sum of       Mean
                          DF    Squares     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 < 0.001 and also those with *p*-values < 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 much 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. There are many things you can do with the CELLSTYLE statement to enhance your output. Several more will be 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;

ods html style=statistical;

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

ods html close;
```

Output 20.7.6 Rows Boldly Highlighted: Histamine Study

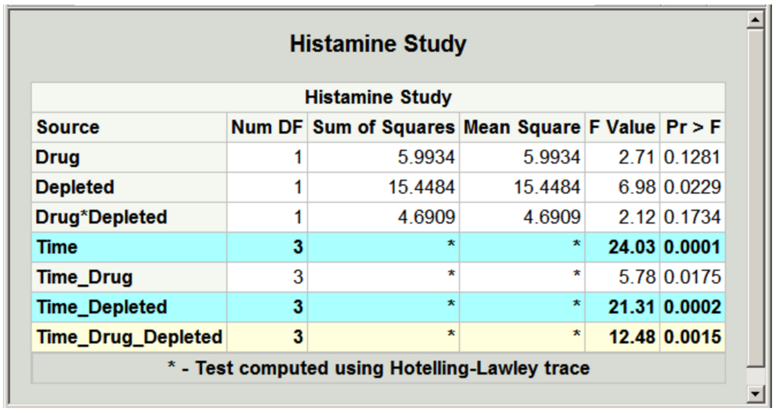


Histamine Study

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

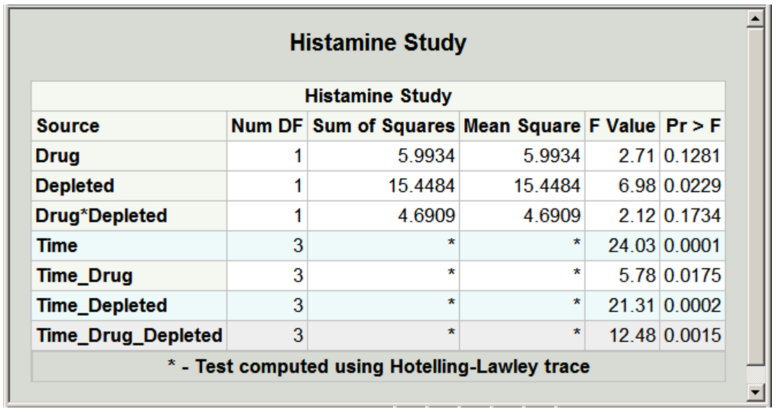


Histamine Study

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**					

SAS color values can be expressed as RGB (red, green, blue) values of the form CXRRGGBB. RR, GG, and BB are all integers in the range 0 to 255 but expressed in a HEX2. format (00 to FF) that specify the red, green, and blue values, respectively. You can run the following steps 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. In this example, a table is created 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 manufactured by different machines, using 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 like 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 listing close;
ods html body='anovab.htm' style=statistical 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 listing;
```

The LISTING destination is closed to avoid generating the output twice. 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

                          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
```

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    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 alters the values in the **`Source`** column ('Supplier' and 'Machine') of the ANOVA tests table by using the CELLSTYLE statement. 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.

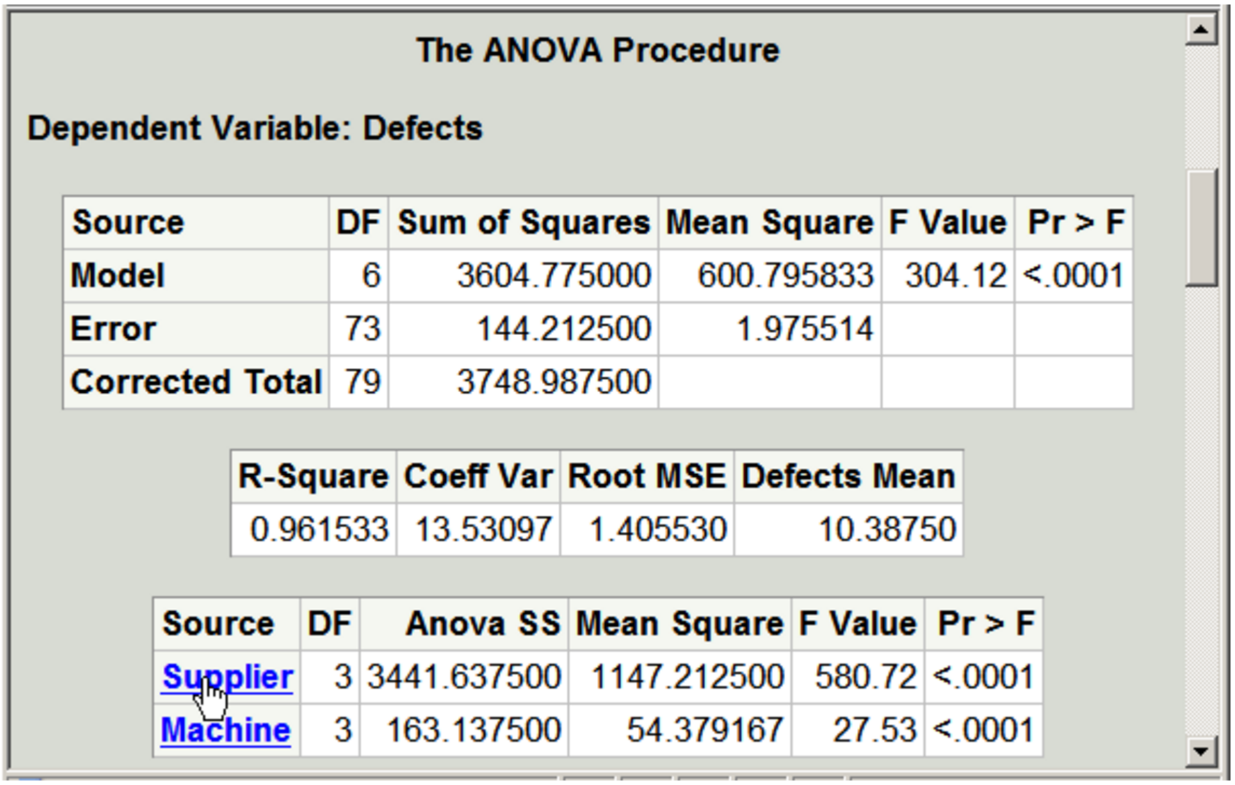
You can see the value to use in the URL by viewing the HTML source file, *anovab.htm*. You can either open the HTML file in a text editor or view it in a browser window and select **View ▶ Source**. Search for '<a name=' to find the URL names. The first table is *anova1*, the second is *anova2*, the 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. Note that if you do not use the ODS SELECT statement, or if you do anything to change the tables that come out, 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 by using the modified template. The following statements produce the results:

```
ods listing close;
ods html body='anovab.htm' style=statistical 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 listing;
```

The ANOVA table is displayed in Output 20.8.2.

Output 20.8.2 HTML Output from PROC ANOVA: Linked Output



The ANOVA Procedure

Dependent Variable: Defects

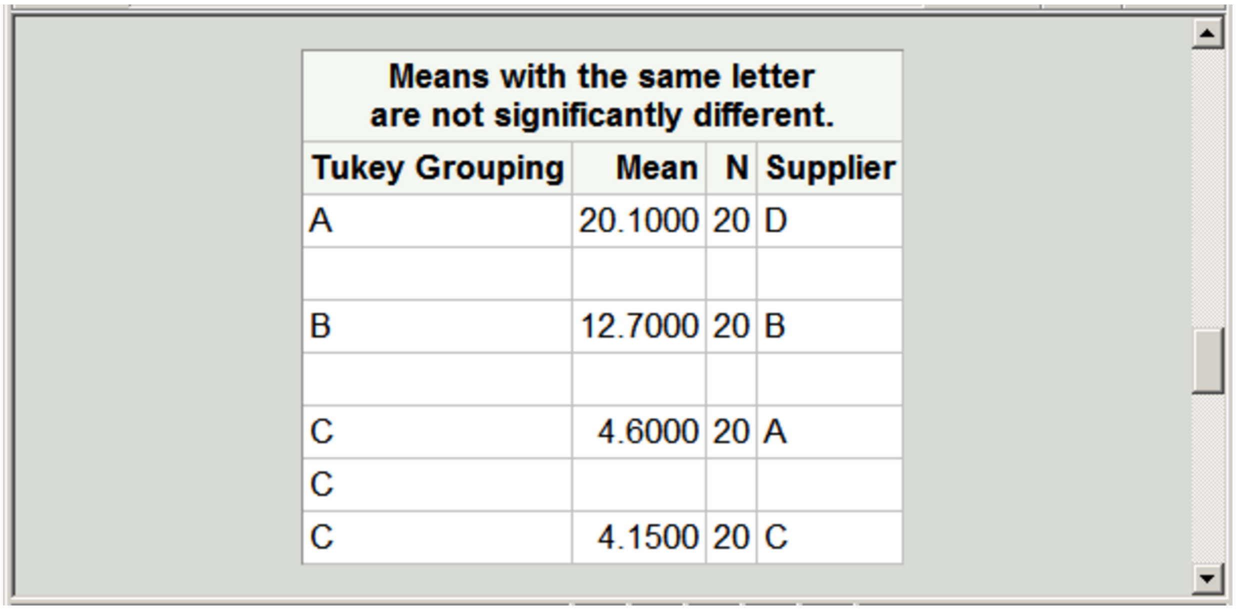
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	3604.775000	600.795833	304.12	<.0001
Error	73	144.212500	1.975514		
Corrected Total	79	3748.987500			

R-Square	Coeff Var	Root MSE	Defects Mean
0.961533	13.53097	1.405530	10.38750

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

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



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 are now two versions of the template by running the following statements:

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

These statements produce Output 20.8.4.

Output 20.8.4 The 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.TMPLMST
                 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;
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 html body='glmb.htm' contents='glmc.htm' frame='glmf.htm'
         style=statistical;
```

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 corresponding to the bar. Note that 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 will work 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.

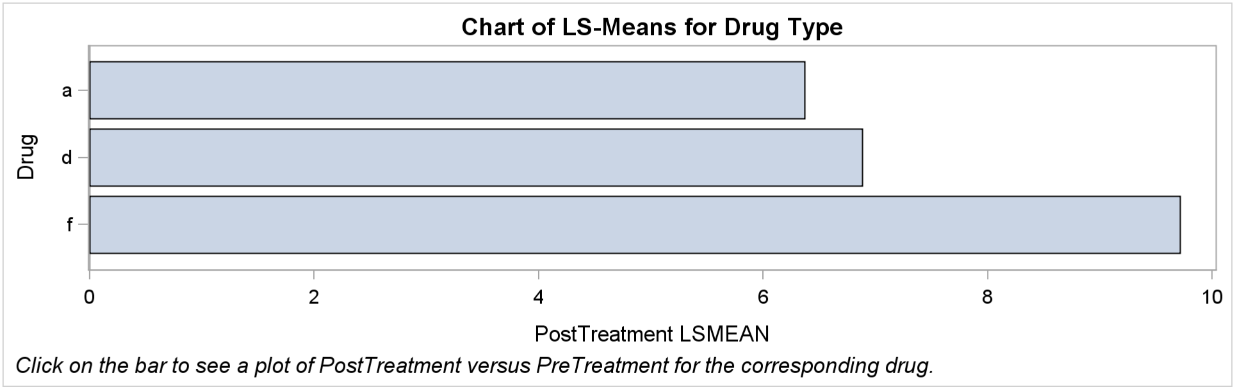
The following statements provide that second analysis. The three files referred to by the DrugClick variable are created as follows:

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

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

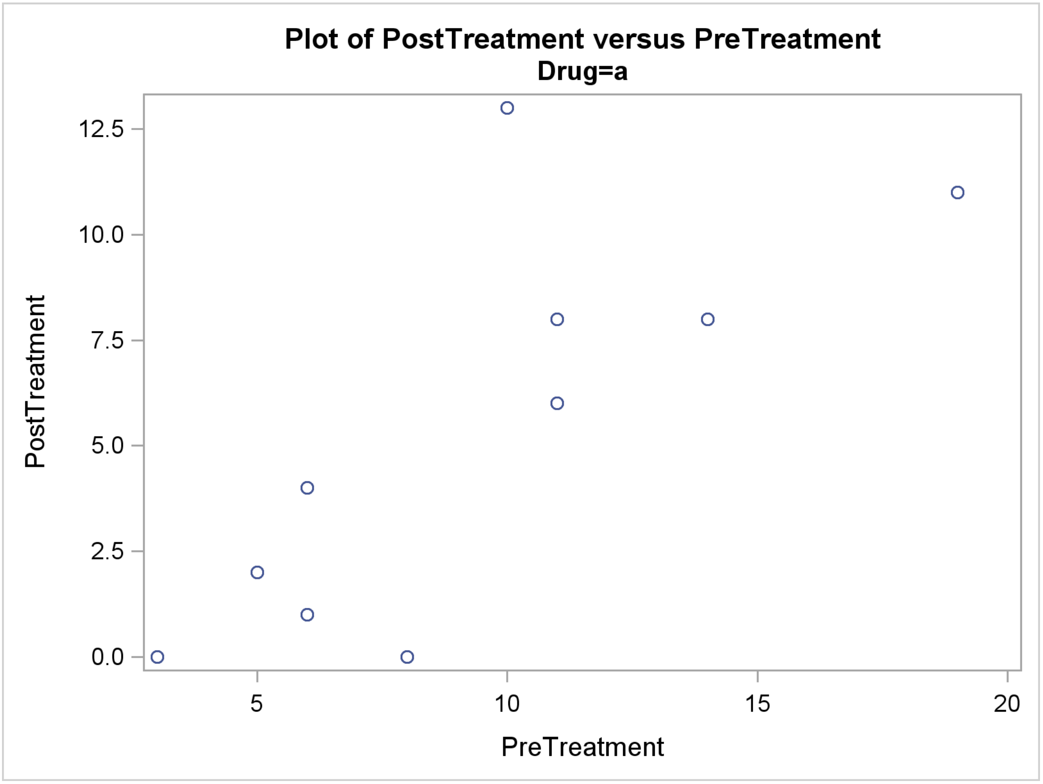
The NEWFILE= option in the ODS HTML statement creates a new HTML file for each page of output. Note that page breaks occur only when a procedure explicitly starts a new page. The NEWFILE= option also increments the filename numeric suffix for each new HTML file created, with the first filename corresponding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

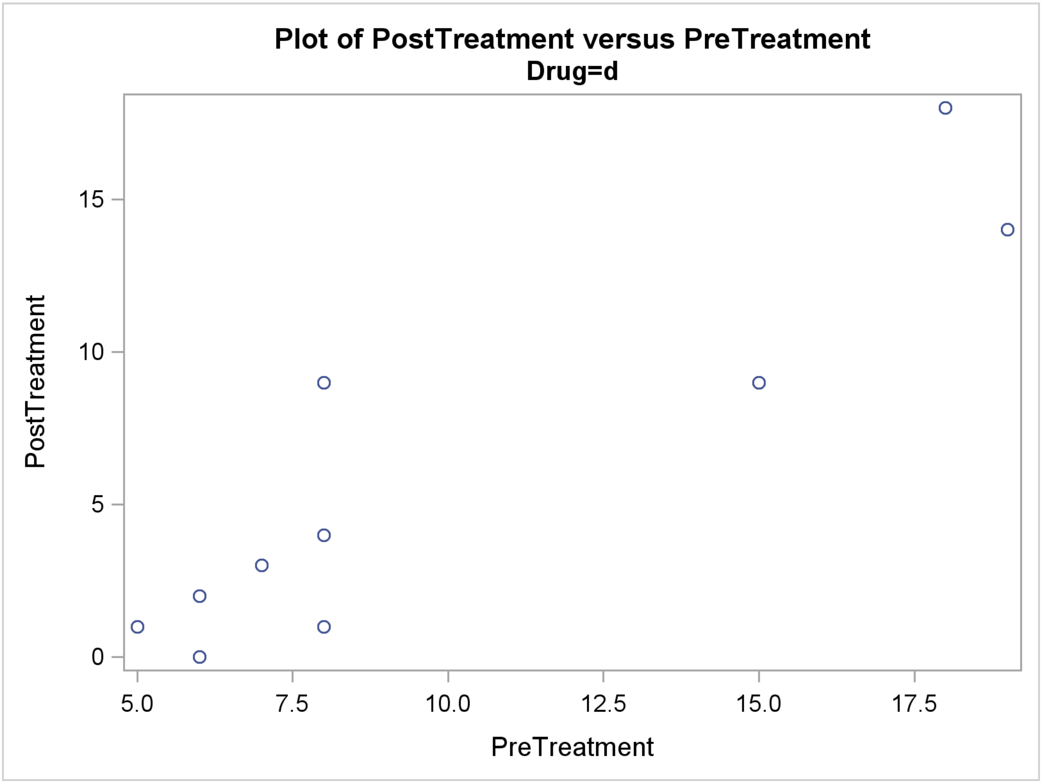
PROC SGPLOT is used, producing a plot of the variable PostTreatment versus the variable PreTreatment for each value of the Drug variable. Three plots are created, and each plot is contained in a separate HTML file. The files are named *drug1.htm*, *drug2.htm*, and *drug3.htm*. The filenames match those 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

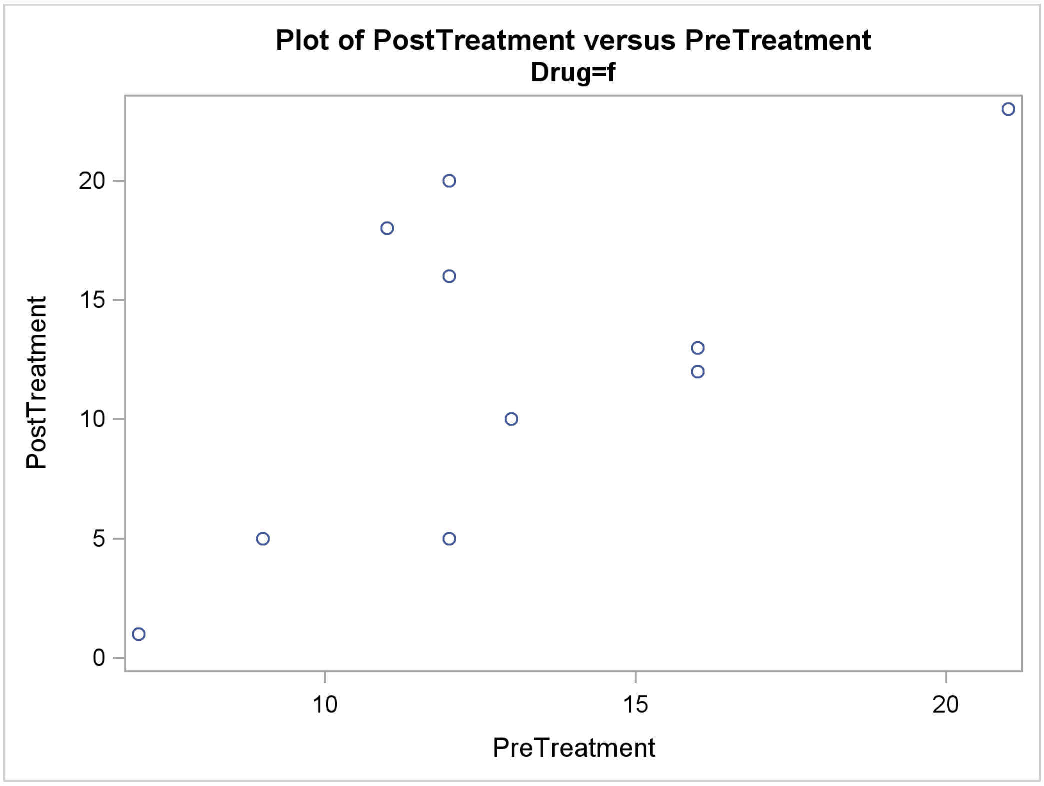


Output 20.9.2 Plots by Drug Type





Output 20.9.2 *continued*



Example 20.10: Correlation and Covariance Matrices

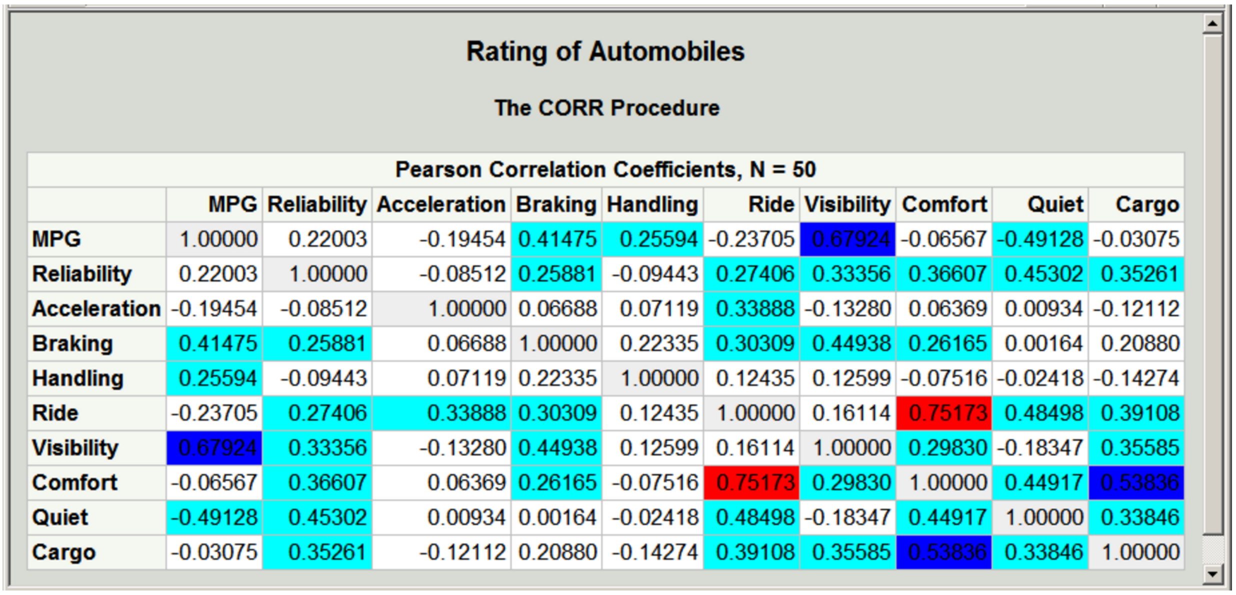
This example demonstrates how you can use ODS to set the background color of individual cells in a table. The color is set to reflect the magnitude of the value in the cell. You can use color to call attention to larger values and to see the pattern in the data in a way that is hard to visualize just 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

... 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 html body='corr.html' style=statistical;
ods listing close;
proc corr data=cars noprob;
   ods select PearsonCorr;
run;
ods listing;
ods html close;

proc template;
   delete Base.Corr.StackedMatrix;
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 SAS and uses the standard configuration of software supplied by SAS, you need to ensure that the SAS system option MAUTOSOURCE is in effect to 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 ranging 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. Note, however, that 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 html body='corr.html' style=statistical;
ods listing close;
proc corr data=cars noprob;
   ods select PearsonCorr;
run;
```

```
ods listing;
ods html close;

proc template;
   delete Base.Corr.StackedMatrix;
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:

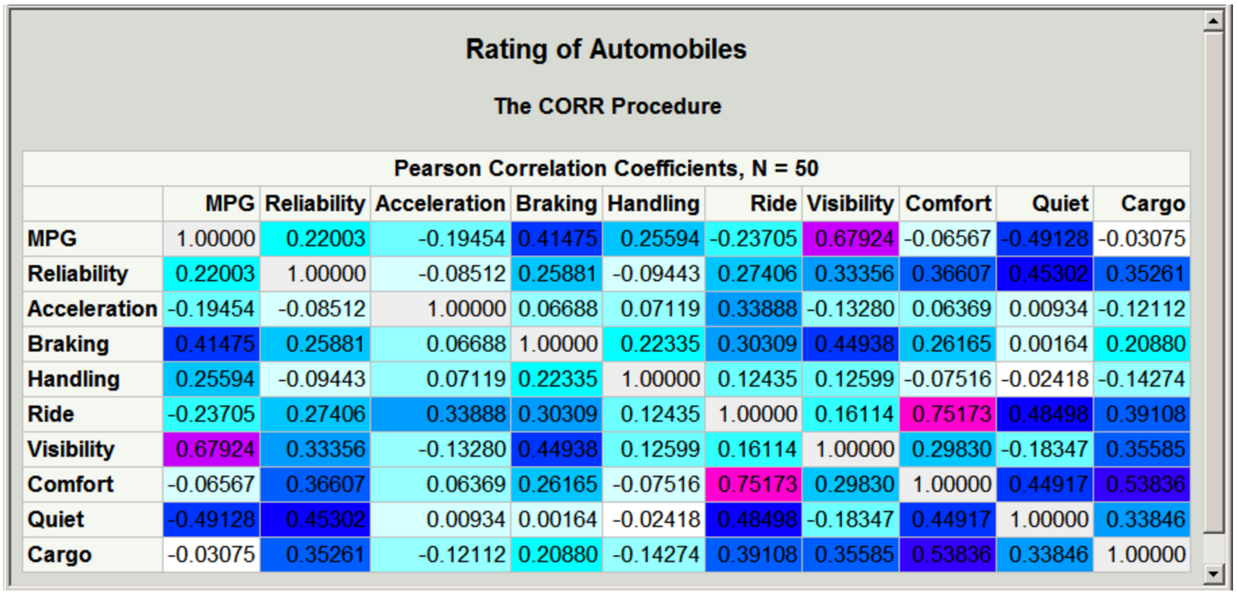
```
cellstyle _val_<=-1 as {backgroundcolor=CXEFEEEE},
          _val_<=-0.95 as {backgroundcolor=CXFF0020},
          _val_<=-0.9 as {backgroundcolor=CXFF0062},
          _val_<=-0.85 as {backgroundcolor=CXFF008D},
          _val_<=-0.8 as {backgroundcolor=CXFF00CF},
```

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 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, and that is what is done 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 will 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 original data are used, for example, in the second example of the PROC MIXED documentation. 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 8 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
```

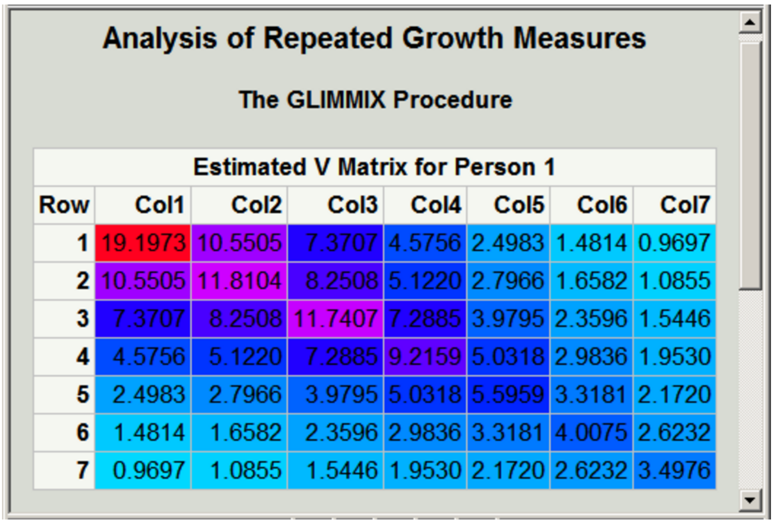
```
12   M   21.5  23.5  24.0  28.0  17.0  24.5  26.0  29.5
13   M   22.5  25.5  25.5  26.0  23.0  24.5  26.0  30.0
;
```

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 html body='ar1.html' style=statistical;
ods listing close;
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 listing;
ods html close;

proc template;
   delete Stat.Glimmix.V;
   delete Stat.Glimmix.VCorr;
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

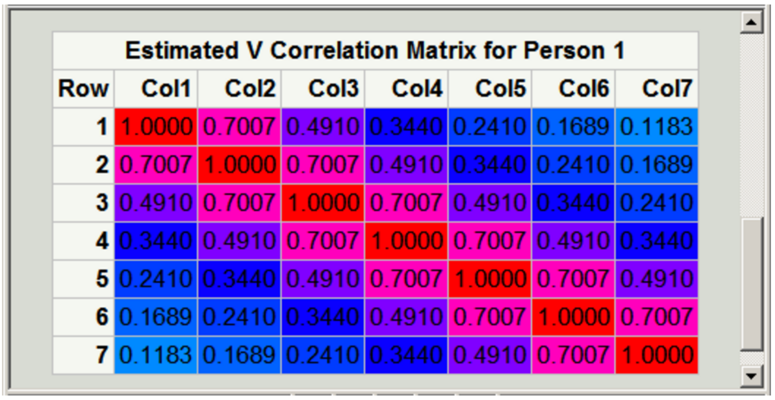


Analysis of Repeated Growth Measures

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

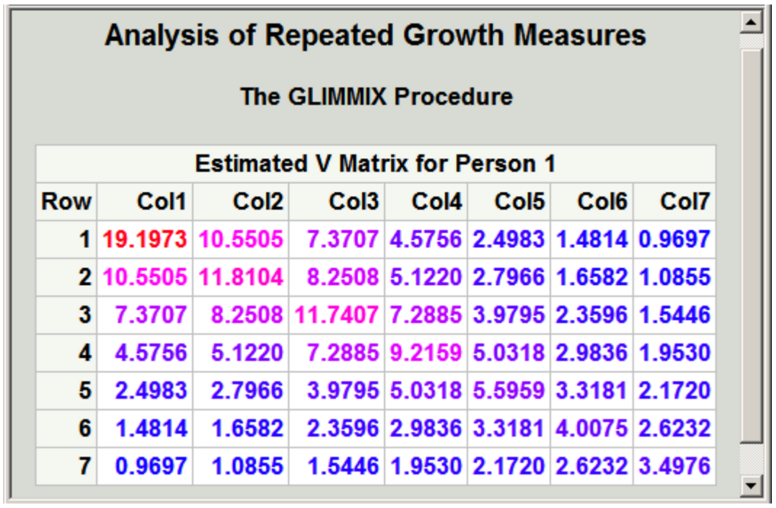
Alternatively, you could just use the `Paint` macro to do the color interpolation, and use its output data set to create other types of style effects. The following statements show one way to set the font to bold and set the foreground color based on the values of the covariances:

```
%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 html body='ar1.html' style=statistical;
ods listing close;
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 listing;
ods html close;

proc template;
   delete Stat.Glimmix.V;
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*, Cary, NC: SAS Institute Inc.

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*, Sixth Edition, Ames: Iowa State University Press.

Subject Index

Syntax Index

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **`yourturn@sas.com`**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **`suggest@sas.com`**.