Paper 039-2013

Renovating Your SAS® 9.3 ODS Output: Tools for Everything from Minor Remodeling to Extreme Makeovers

Bari Lawhorn, SAS Institute Inc.

ABSTRACT

The SAS® 9.3 HTML output that is generated with the ODS HTML statement and the new HTMLBLUE style looks great with the default settings. But realistically, there are always pieces of your output that you want to change. Sometimes you just need a little remodeling to get the results you want; other times, the desired changes call for an extreme makeover.

This paper shows how you can use certain ODS statements (for example, ODS SELECT and ODS TEXT) and ODS statement options (for example, STYLE=) for minor remodeling. The paper also illustrates how to use the REGISTRY, TEMPLATE, and DOCUMENT procedures for a more extreme makeover. These makeovers apply to HTML as well as other ODS destinations.

INTRODUCTION

Your standard SAS output results should contain informative tables and sleek graphs that are created by many of your procedures. But, realistically, just like with your home decor, there are always pieces of your output that you want to change. At home, for example, you might want to make minor changes such as changing paint color or modifying fixtures. In the same way, you might want to make comparable minor changes to your SAS output.

At other times, your home needs more drastic renovations such as tearing down walls and reconfiguring rooms. The SAS® 9.3 Output Delivery System (ODS) enables you to perform more drastic renovations on your output as well. With ODS tools, you can restrict, rearrange, or format columns, tables, and graphs. You can also make significant changes in the way information from SAS data sets is presented in all of your ODS destinations.

This paper is presented in two parts:

- Part 1: Minor Remodeling
- Part 2: Major Renovations

Part 1 of this paper first looks at simple statements that help you improve the look of your output. These statements require no previous knowledge of ODS, so the examples are useful to beginner, intermediate, and advanced programmers. The examples explained in Part 1 require the following remodeling tools:

- ODS NOPROCTITLE statement
- ODS SELECT statement
- STYLE= option
- ODS TEXT= statement
- ODS PROCLABEL= statement

Part 2 of the paper explains strategies that provide ways to create more lasting or in-depth changes in your output. The procedures that are discussed require some knowledge of ODS, so the examples in this section are most useful to intermediate and advanced programmers. The examples in this section require the following renovation tools: parameters:

- REGISTRY procedure
- TEMPLATE procedure
- the ODS DOCUMENT destination and PROC DOCUMENT (hereinafter, referred to as the DOCUMENT facility)

PART 1: MINOR REMODELING

A CLEAN SLATE: DECLUTTERING YOUR OUTPUT WITH THE ODS NOPROCTITLE AND ODS SELECT STATEMENTS

Before starting work on any home remodeling project, regardless of the scope, professional designers recommend that you start with a clean slate. In the case of a home, starting with a *clean slate* might mean, for example, that you must remove any extraneous furniture from rooms that are undergoing an improvement.

The same clean-slate concept also applies when you are renovating your SAS output. The following sections illustrate how to declutter your output by removing extraneous titles and generating only certain aspects of the output. For these tasks, you can use the ODS NOPROCTITLE statement and the ODS SELECT statement.

Remove Title Text with the ODS NOPROCTITLE Statement

Many SAS procedures add a procedure title to output tables, but does your audience really need to know that the (soon-to-be) brilliantly styled statistical tables and graphs that you plan to present are created with the CORR or UNIVARIATE procedures? Not likely. So you can declutter your output by eliminating the procedure title text (**`The procedure-name Procedure`**). To remove the title text, use a simple ODS NOPROCTITLE statement:

```
ODS NOPROCTITLE;
```

This statement is a *global SAS statement*, so until you reset it, the statement remains in effect. You can also use an abbreviated form of the statement:

```
ODS NOPTITLE;
```

Consider the following UNIVARIATE procedure:

```
proc univariate data=sashelp.heart;
   var weight;
run;
```

The procedure generates the following table, which contains the unnecessary title **`The UNIVARIATE Procedure`**:

The UNIVARIATE Procedure
Variable: Weight

Moments			
N	5203	Sum Weights	5203
Mean	153.086681	Sum Observations	796510
Std Deviation	28.9154261	Variance	836.101866
Skewness	0.55594115	Kurtosis	0.52275608
Uncorrected SS	126284474	Corrected SS	4349401.91
Coeff Variation	18.8882703	Std Error Mean	0.40086919

To eliminate the extraneous title, add the ODS NOPROCTITLE statement, as shown below:

```
ods noproctitle;

proc univariate data=sashelp.heart;
   var weight;
run;
```

The ODS NOPROCTITLE statement suppresses the extraneous title, as shown below:

Variable: Weight

Moments			
N	5203	Sum Weights	5203
Mean	153.086681	Sum Observations	796510
Std Deviation	28.9154261	Variance	836.101866
Skewness	0.55594115	Kurtosis	0.52275608
Uncorrected SS	126284474	Corrected SS	4349401.91
Coeff Variation	18.8882703	Std Error Mean	0.40086919

Focus the Scope of Your Output with the ODS SELECT Statement

Statistical procedures such as PROC CORR and PROC UNIVARIATE generate numerous tables. Depending on the syntax that you use, these procedures have the ability to generate one or more graphs as well. But for your purposes, you might want only a subset of these tables or graphs.

One way to specify which tables or graphs are produced is by using the ODS SELECT statement.

Consider the following PROC UNIVARIATE request for the P-P plot:

```
ods trace on;

proc univariate data=sashelp.heart;
   var weight;
   ppplot / normal square;
run;
```

With tracing in effect through the ODS TRACE ON statement, the following information is displayed in the SAS log.

Note: The sample program above generates seven total objects. For brevity, only three of those objects are shown below.

```
Output Added:
-------------
Name:        ExtremeObs
Label:       Extreme Observations
Template:    base.univariate.ExtObs
Path:        Univariate.Weight.ExtremeObs
-------------

Output Added:
-------------
Name:        MissingValues
Label:       Missing Values
Template:    base.univariate.Missings
Path:        Univariate.Weight.MissingValues
-------------

Output Added:
-------------
Name:        PPPlot
Label:       Panel 1
Template:    base.univariate.Graphics.PPPlot
Path:        Univariate.Weight.PPPlot.PPPlot
-------------
```

Suppose that you are interested only in the **Extreme Observations** table and the **Panel 1** (P-P) plot. In this case, the best way to subset your output is to use the ODS SELECT statement, as shown here:

```
ods select extremeobs ppplot;

proc univariate data=sashelp.heart;
   var weight;
   ppplot / normal square;
run;
```

The ODS TRACE statement is still in effect, so the log window shows only these two objects:

```
Output Added:
-------------
Name:         ExtremeObs
Label:        Extreme Observations
Template:     base.univariate.ExtObs
Path:         Univariate.Weight.ExtremeObs
-------------

Output Added:
-------------
Name:         PPPlot
Label:        Panel 1
Template:     base.univariate.Graphics.PPPlot
Path:         Univariate.Weight.PPPlot.PPPlot
-------------
```

Note: In some cases, it might be more efficient to exclude certain objects rather than select the ones you want. Although this paper does not detail this method, you can use the ODS EXCLUDE statement to exclude certain objects in your output. Using this method might be more efficient (for example, it might require less typing) when a procedure generates many tables or graphs and you only want to exclude a small number of those objects.

A LITTLE COAT OF PAINT: CHANGING THE LOOK OF YOUR OUTPUT WITH THE STYLE= OPTION

No home renovation is complete without attention to the walls. In even the smallest home-improvement project, changing the color or texture of a wall or an entire room can transform the whole look and feel of a space.

Just as a can of paint provides a simple transformation in your home, the ODS STYLE= option enables you to change the entire look of your table and graphics output. In SAS 9.3, the GSTYLE option is in effect by default. Therefore, when the GSTYLE option is in effect, GRSEG output that is generated by SAS/GRAPH procedures (such as GPLOT, GCHART, and GMAP output) displays results differently depending on the STYLE= value that you set in the ODS destination statement. Graph output that is generated either by ODS Graphics or the Statistical Graphics procedures (for example, the SGPLOT, SGSCATTER, and SGRENDER procedures) always complies with this rule. These procedures generate only graphs.

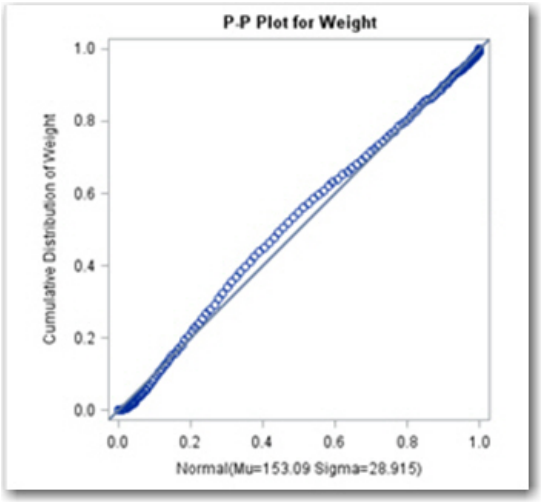
Styles are always in effect in the ODS markup destinations. In SAS® 9.3, HTMLBLUE is the default style for the HTML destination. The following code is derived from the original PROC UNIVARIATE example shown previously in the section "Remove Title Text with the ODS NOPROCTITLE Statement." In this sample, code, the ODS HTML statement includes the HTMLBLUE style.

```
ods html style=htmlblue;
ods select extremeobs ppplot;

proc univariate data=sashelp.heart;
   var weight;
   ppplot / normal square;
run;

ods html close;
```

The following table and plot are generated by this PROC UNIVARIATE step. PROC UNIVARIATE graphs are controlled by ODS Graphics.

Variable: Weight

Extreme Observations			
Lowest		Highest	
Value	Obs	Value	Obs
67	5098	276	2120
71	766	281	4858
72	2250	293	3252
82	5106	300	3360
83	4462	300	4704



Note: The STYLE= option is not needed in the SAS windowing environment because HTMLBLUE is the default style for the HTML destination. But if a style without color is needed—for a submission to a particular journal perhaps—you can use one of the several SAS styles that create gray-scale output.

The following PROC UNIVARIATE code applies the JOURNAL2 style to the output:

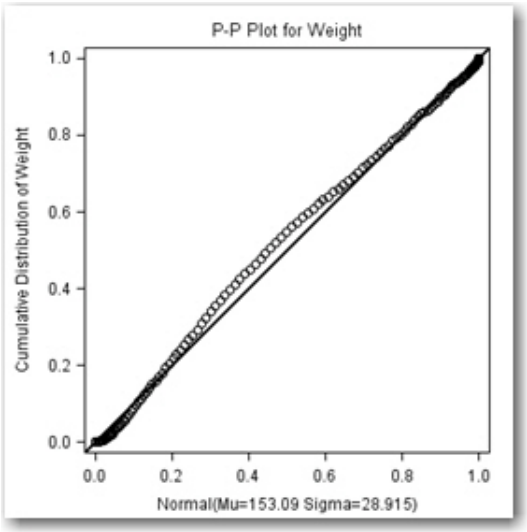
```
ods html style=journal2;
ods select extremeobs ppplot;

proc univariate data=sashelp.heart;
   var weight;
   ppplot / normal square;
run;

ods html close;
```

The resulting table and plot are basic gray-scale images.

Variable: Weight

Extreme Observations			
Lowest		*Highest*	
Value	*Obs*	*Value*	*Obs*
67	5098	276	2120
71	766	281	4858
72	2250	293	3252
82	5106	300	3360
83	4462	300	4704

P-P Plot for Weight



Many styles are shipped with SAS. You can also create your own custom styles using either the TEMPLATE procedure or a cascading style sheet (CSS) file that you can use in the STYLESHEET= option. A full discussion of the STYLE= and the STYLESHEET= options is outside the scope of this paper, but you can find many online references that offer sample code and tutorials. Being aware of these options enables you to explore your choices in depth. Customizing the look of your tables and images is easy to accomplish using either of these options in ODS destination statements

GENERAL ACCENT PIECES: USING THE ODS TEXT= STATEMENT TO ACCENTUATE OUTPUT ELEMENTS

Once the main remodeling is done in a room, designers you can add accent pieces (throw rugs, pillows, wall hangings) to unify the elements of the room and give it some flair. In a similar way, you can add a little pizzazz to your output after you make all of your primary changes. You can jazz up your output by using titles, footnotes, and descriptive text as your unifying accent pieces.

In the SAS® System, titles and footnotes occur once per page. However, in printed destinations, you might want to insert descriptive text between your tables or graphs, or you might choose to move the text closer to the tables or graphs than the title and footnote text. You can accomplish these changes easily with the ODS TEXT= statement.

The following sample code uses TITLE and FOOTNOTE statements as well as a number of ODS statements, including ODS TEXT=, to give a designer look to the output.

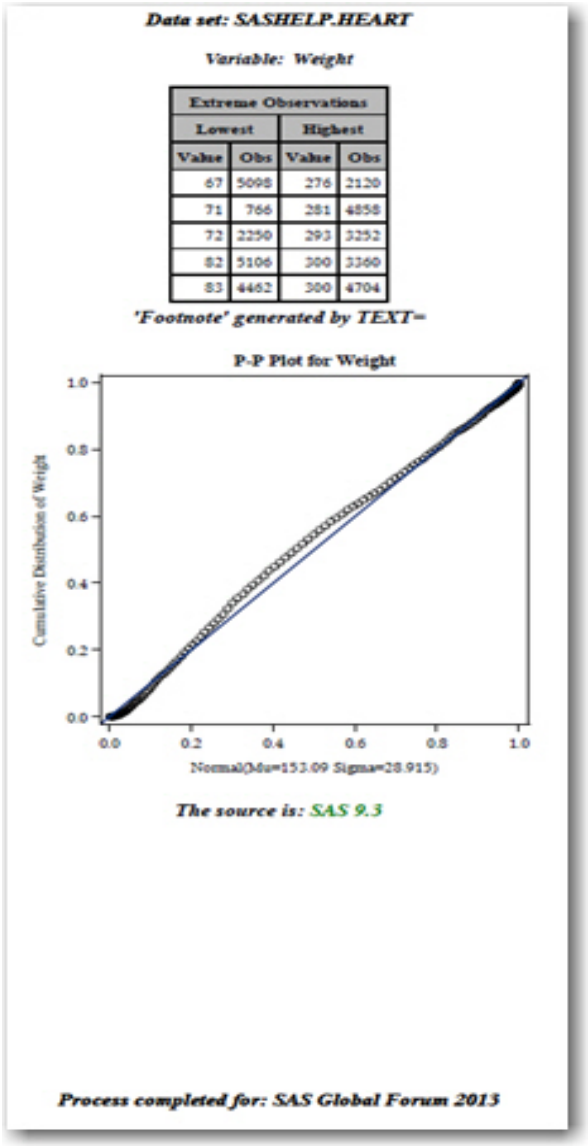
```
title "Data set: SASHELP.HEART";
footnote "Process completed for: SAS Global Forum 2013";
ods escapechar="^";

ods pdf file="file.pdf" startpage=no;
ods noproctitle;
ods select extremeobs;

proc univariate data=sashelp.heart;
   var weight;
run;

ods text="^{style systemfooter[just=c]'Footnote' generated by TEXT=}";

ods select ppplot;
ods graphics / noborder;

proc univariate data=sashelp.heart;
  var weight;
  ppplot / normal square;
run;

ods text=" ";
ods text="^{style systemfooter[just=c]The source is: ^{style [foreground=green]
SAS &sysver.}}";

ods pdf close;
```

This example uses two separate PROC UNIVARIATE steps, each containing an ODS SELECT statement: One step creates a table; the other step creates a P-P plot. Putting the procedures in two separate steps enables you to use an ODS TEXT= statement between them in order to place descriptive text between the table and the P-P plot. Another ODS TEXT= statement is used after the second PROC UNIVARIATE step. This statement places additional descriptive text directly after the P-P plot.

The ODS ESCAPECHAR= statement, included near the beginning of the code sample, enables you to use inline style commands to dress up the text even more. The escape character (^) is then used with the SYSTEMFOOTER style element in the ODS TEXT= statement. The SYSTEMFOOTER style element creates text using the same style

attributes as those used for a footnote. That is, the text is bold and italic, and it uses a larger font than the default size for text generated with the ODS TEXT= statement.

The previous code sample generates the following output:



DESIGNER ACCENT PIECES: USING THE ODS PROCLABEL= STATEMENT TO PERSONALIZE YOUR TABLE OF CONTENTS

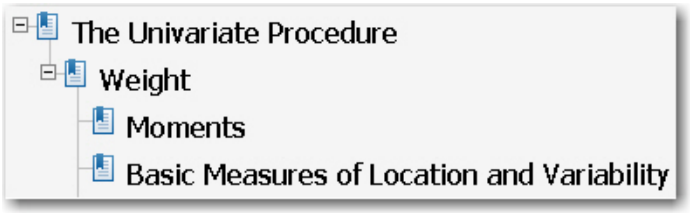
In a more sophisticated renovation, additional design and style elements are needed to unify all of the pieces involved in the renovation and to express your personal style. For such a renovation, an interior designer might create a design storyboard for you that enables you to see all of the pieces of your new room in snippets. The SAS table of contents (TOC) in ODS output offers a similar concept.

In SAS output, a navigation tool such as a TOC is used often to facilitate the readability of a file and to help the audience quickly find the relevant portions of output. The ODS destinations HTML, EXCELXP, RTF, and PDF enable you to create TOCs. ODS PDF generates a TOC by default, but the default text in the TOC is utilitarian and often does not allow the custom navigation that readers of the file might want. However, you can make the TOC more stylish and useful with the ODS PROCLABEL= statement. This statement enables you to change the text of the TOC's first-level node easily so that it more accurately describes the results in your output file.

For example, consider the following code. It creates a default TOC in a PDF file that is generated with the ODS PDF destination:

```
ods pdf file="default.pdf";

proc univariate data=sashelp.heart;
   var weight;
run;

ods pdf close;
```

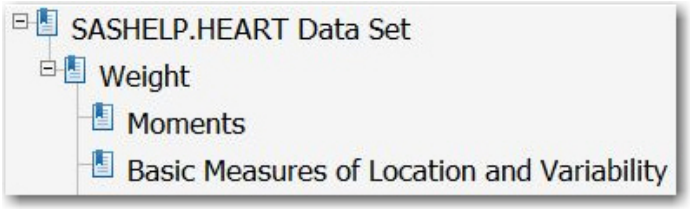
The sample code generates the following TOC and automatically names the first-level node after the procedure that is used:



The recipient of your PDF file is most likely not concerned with the SAS procedure that created the results. He or she is more likely to be interested in the specific data that is used for the analysis. For example, the first-level node might be more useful if it names the data set that is used, as shown in the following code and output:

```
ods pdf file="proclabel.pdf";
ods proclabel="SASHELP.HEART Data Set";

proc univariate data=sashelp.heart;
   var weight;
run;

ods pdf close;
```

The resulting TOC shows a first-level node that is now titled **`SASHELP.HEART Data Set`**, where SASHELP.HEART is the name of the data set that is used in the analysis.



This example is just one of the many changes that you can make to your TOC to increase the value of your results. More ideas and sample code are offered in the paper "Let's Give'Em Something to TOC about: Transforming the Table of Contents of Your PDF File." (Lawhorn, 2011)

PART 2: MAKING MAJOR RENOVATIONS

EXTREME MAKEOVER: MAKING MORE SIGNIFICANT OR PERMANENT CHANGES TO YOUR OUTPUT

If you have ever tackled even a small home-improvement project, you know that once you get started, one project leads to another. A coat of paint in one room leads changing the window treatments updating the flooring, and so on.

The same is true with designing your output. Sometimes, minor changes made in the look of your SAS output are sufficient; other times, you might decide that you want to enhance the output with a more extreme makeover. Or you might want to make your changes permanent. To do that, you need more powerful tools. The SAS REGISTRY and TEMPLATE procedures, along with the DOCUMENT facility, provide the power you need to facilitate such changes.

A STYLE THAT STICKS: USING THE REGISTRY PROCEDURE TO MAKE PERMANENT CHANGES TO A DESTINATION

Good interior designers help you express your own personal style and identity in every room of your home. In the same way, corporations and universities often have their own design styles that project their identities across web pages and printed materials. To maintain identity, a corporate style relies on certain elements that always appear in print and media pieces.

For example, you might have output for which it is important to maintain a particular corporate style for a particular ODS destination each time that destination is used. If you want to avoid having to add the style each time by using the STYLE= option in the ODS destination statement, then PROC REGISTRY is the best tool for the job. The SAS registry controls much of the ODS output. PROC REGISTRY enables you to adjust the keys in the SAS registry to make changes permanent by storing them in the SASUSER or SASHELP library. PROC REGISTRY input is a text file with the registry key name-and-value pairs that specify the changes that you want.

The following sample code creates a file named printer.sasxreg that sets the default style for the ODS PRINTER destination to SASDOCPRINTER. The PROC REGISTRY step imports this file and subsequently updates the ODS\DESTINATIONS\PRINTER key in the SAS registry:

```
data _null_;
   file 'printer.sasxreg';
   put '[ods\destinations\printer]';
   put' ''Selected Style''=''sasdocprinter'' ';
run;

proc registry import='printer.sasxreg';
run;
```

The next sample code uses the change made to the SAS registry in the previous code:

```
ods pdf file="file.pdf";
title "Using SASDOCPRINTER style via the SAS Registry";

proc report nowd data=sashelp.cars(obs=5);
run;

ods pdf close;
```

Using an ODS PDF FILE= statement such as the one above generates PDF output, as shown here:

Using SASDOCPRINTER style via the SAS Registry

Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	Engine Size (L)
Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5
Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2
Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4
Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2
Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5

By default, the IMPORT= option imports the printer.sasxreg file to the SASUSER portion of the SAS registry. It is possible to import the file also to the SASHELP library instead by using the USESASHELP option in the PROC REGISTRY statement. This step might require the assistance of a SAS administrator because it requires Write permission to the SASHELP library. (**Note:** Some people already have Write permission. If you do not have permission, contact your SAS administrator.)

After you submit the previous code, all subsequent ODS PDF, ODS PCL, and ODS PRINTER statements will use the SASDOCPRINTER style instead of STYLE.PRINTER, which is the default style in SAS 9.3. By using PROC

REGISTRY, you eliminate the need to use a statement similar to the following each time you want to use the SASDOCPRINTER style:

```
ods pdf file="file.pdf" style=sasdocprinter;
```

Note: Specifying the STYLE= option in an ODS destination statement temporarily overrides any permanent change that you make.

A TABLE TRANSFORMATION: USING THE TEMPLATE PROCEDURE TO CHANGE THE LOOK OF A STANDARD TABLE

For homes that are going on the market, realtors typically advise making renovations that have universal appeal to attract a variety of home buyers. However, when you buy a new home, you want to make it your own. So you might decide to update the universal style with more personalized elements, such as a custom kitchen and light fixtures.

In a similar manner, SAS development teams have worked hard to ensure that the tables generated by SAS procedures offer the most relevant information and appeal to the widest audience. However, sometimes you need to change the default statistics in output tables. That is, you either need to modify the header information or adjust the format of a particular column.

The next example focuses on adjusting column format and making physical changes to the statistic, but these changes are not permanent. The example uses a FORMAT statement and PROC TEMPLATE to create the changes.

```
ods trace on;

proc univariate data=sashelp.heart normal;
   var cholesterol;
   class status;
run;

ods trace off;
```

The SAS log shows all of the tables that this step generates, but for brevity, only two of the seven tables are shown here.

```
Output Added:
-------------
Name:        TestsForLocation
Label:       Tests For Location
Template:    base.univariate.Location
Path:        Univariate.Cholesterol.Dead.TestsForLocation
-------------

Output Added:
-------------
Name:        TestsForNormality
Label:       Tests For Normality
Template:    base.univariate.Normal
Path:        Univariate.Cholesterol.Dead.TestsForNormality
-------------
```

The code generates the following Tests for Normality table:

Tests for Normality				
Test	Statistic		p Value	
Kolmogorov-Smirnov	D	0.054671	Pr > D	< 0.0100
Cramer-von Mises	W-Sq	2.52857	Pr > W-Sq	< 0.0050
Anderson-Darling	A-Sq	16.36646	Pr > A-Sq	< 0.0050

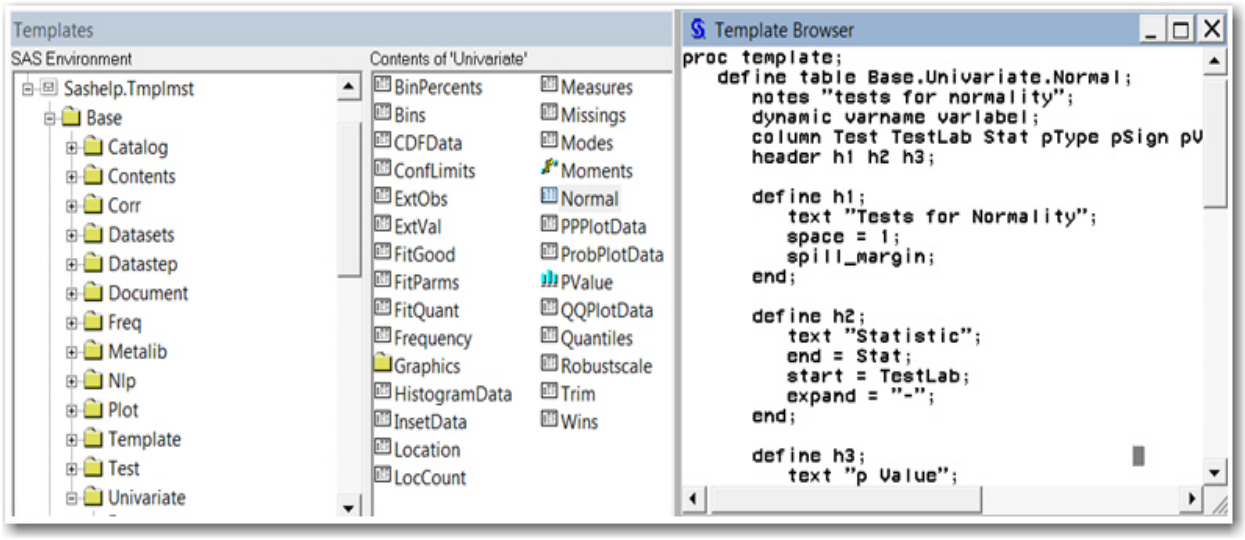
Perhaps you need more from this table. In this case, the *p*-value that appears in the table does not show as many decimal places as you would like. So you need to change the format to show the number of decimals places that you want.

To make any changes to this table, you must identify the table template from which this table is built. You can find that information in the SAS log when the ODS TRACE statement is in effect. In this case, the log (shown in the log output above) shows that the template is named **`Base.Univariate.Normal`**.

You can see the contents of the template by submitting the ODST (ODSTEMPLATES) command from the SAS command line. This command opens the Templates window. You can also access the source code by submitting the following code from the SAS Enhanced Editor:

```
proc template;
   source Base.Univariate.Normal;
run;
```

In the window, navigate to the template file, as shown below, to see the source code for the table template.



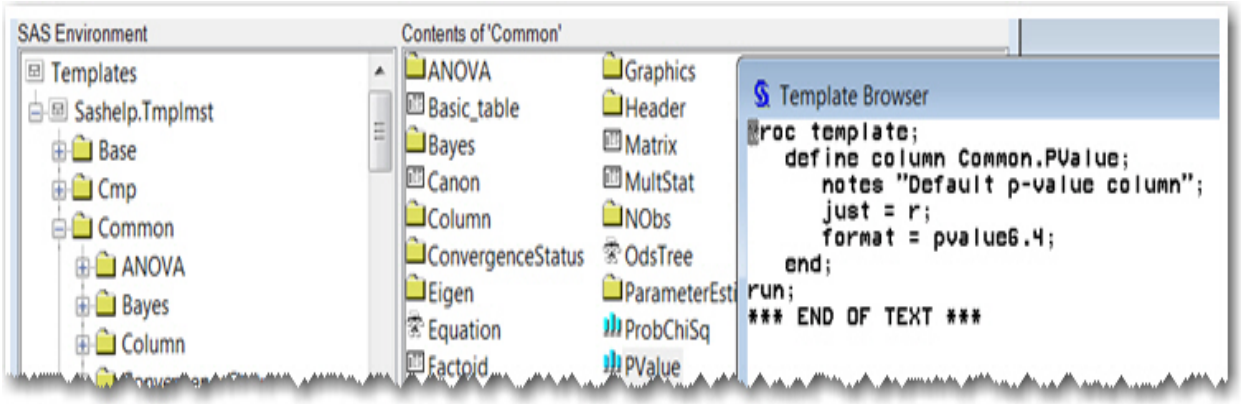
The table definition in the template includes the following DEFINE statement, which is not visible in the code in the template window shown previously. This is the *p*-value's column definition, but it does not give a specific SAS format. Instead, it has a PARENT= specification, which points to the **`Base.Univariate.PValue`** template.

```
define pValue;
   parent = base.univariate.PValue;
   print_headers = OFF;
end;
```

Your goal is to locate a SAS format. So, next, go back to the Templates window and navigate to the **Base.Univariate.PValue** template. In that template file, you see the following PROC TEMPLATE code:

```
proc template;
   define column Base.Univariate.PValue;
      notes "Default UNIVARIATE p-value column";
      parent = Common.PValue;
   end;
run;
```

However, this code shows you yet another PARENT= specification as well. So, from the TEMPLATES window, go up a level (from the **Base** node) and navigate to the **Common.PValue** template, as shown in the following display. Within this template file, you should find the following column definition that includes a FORMAT statement:



At this point in your renovation project, you need to ask yourself what you really want. Do you want to change the *p*-value formatting for **only** the Tests for Normality table? Or do you want to change the format for all *p*-values that are created by PROC UNIVARIATE (for example, those values that are displayed in the Tests for Location table)? Or maybe you prefer to change the *p*-value formatting for **all** procedures in this SAS session that display *p*-values.

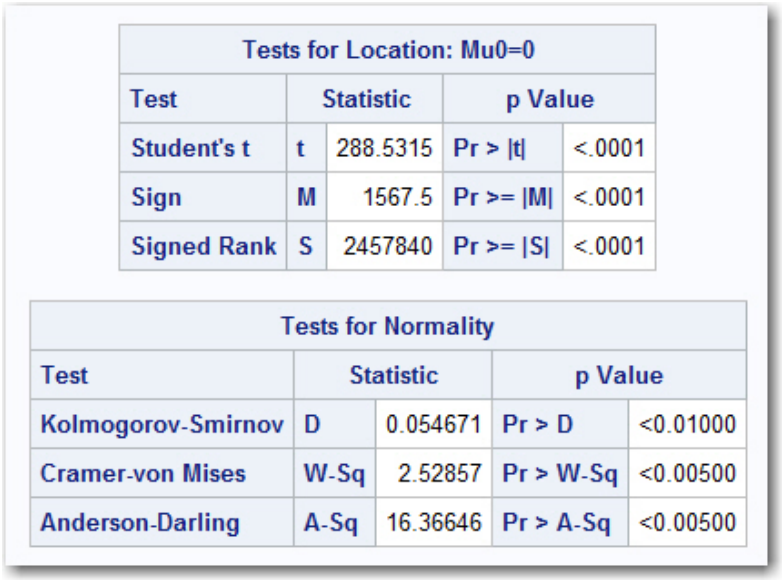
Suppose that you want to change the *p*-value format for only the Tests for Normality table. You can make this change with the following PROC TEMPLATE code:

```
proc template;
   edit Base.Univariate.Normal;
      define pValue;
         parent=Base.Univariate.PValue;
         print_headers=off;
         format=pvalue7.5;
      end;
   end;
run;
```

Now, you can display the Tests for Location table and the Tests for Normality table by using the following PROC UNIVARIATE code:

```
ods select testsfornormality testsforlocation;
proc univariate data=sashelp.heart normal;
     var cholesterol;
     class status;
 run;
```

When the Tests for Normality and the Tests for Location tables are displayed, only the Tests for Normality table shows a change in the *p*-value format.



Tests for Location: Mu0=0

Test	Statistic		p Value	
Student's t	t	288.5315	Pr > |t|	<.0001
Sign	M	1567.5	Pr >= |M|	<.0001
Signed Rank	S	2457840	Pr >= |S|	<.0001

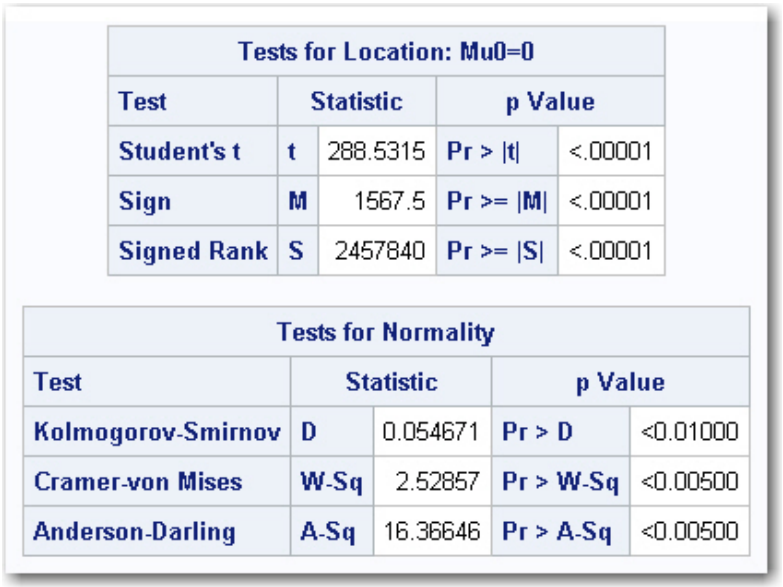
Tests for Normality

Test	Statistic		p Value	
Kolmogorov-Smirnov	D	0.054671	Pr > D	<0.01000
Cramer-von Mises	W-Sq	2.52857	Pr > W-Sq	<0.00500
Anderson-Darling	A-Sq	16.36646	Pr > A-Sq	<0.00500

To make changes to both tables, you need to edit the **`Base.Univariate.PValue`** template, as shown here:

```
proc template;
   edit Base.Univariate.PValue;
      print_headers=off;
      format=pvalue7.5;
   end;
run;
```

Running the previous PROC UNIVARIATE step results in the following tables, both of which show the format change:



Tests for Location: Mu0=0

Test	Statistic		p Value	
Student's t	t	288.5315	Pr > |t|	<.00001
Sign	M	1567.5	Pr >= |M|	<.00001
Signed Rank	S	2457840	Pr >= |S|	<.00001

Tests for Normality

Test	Statistic		p Value	
Kolmogorov-Smirnov	D	0.054671	Pr > D	<0.01000
Cramer-von Mises	W-Sq	2.52857	Pr > W-Sq	<0.00500
Anderson-Darling	A-Sq	16.36646	Pr > A-Sq	<0.00500

If the scope of this formatting change should be even broader, extending to **any** procedure that generates a *p*-value, then you need to edit the **`Common.PValue`** template.

```
proc template;
   define column Common.PValue;
      notes "Default p-value column";
      just=r;
      format=pvalue7.5;
   end;
run;
```

The PROC UNIVARIATE results will match those shown in the previous example. In addition, any other procedure (for example the FREQ or CORR procedures) that can display a *p*-value column results in output that uses the PVALUE7.5 format.

In addition to enabling you to make structural changes similar to the previous examples, PROC TEMPLATE enables you to integrate stylistic changes directly in the table and column templates. The style and structural changes that are made possible by PROC TEMPLATE are too numerous to list here. However, no mention of PROC TEMPLATE is complete without showing how you can combine it with the FORMAT procedure to provide trafficlighting and how you can use the COMPUTE statement in PROC TEMPLATE to create customized columns.

The following example uses PROC FORMAT to create a numeric format named TRAFFIC. This format is then used in the STYLE= statement in the PROC TEMPLATE step.

```
proc format;
   value traffic low - .005="red"
                 .005 <- high="green";
run;

proc template;
   edit Base.Univariate.Normal;
      define pValue;
         parent=Base.Univariate.PValue;
         print_headers=OFF;
         format=pvalue7.5;
         style={foreground=traffic. };
      end;
   end;
run;
```

The sample code produces the following tables. Notice the trafficlighting for the **`p Value`** column in the Tests for Normality table. Values less than or equal to 0.005 are shown in red; values greater than 0.005 are shown in green.

Tests for Location: Mu0=0

Test	Statistic		p Value	
Student's t	t	288.5315	Pr > |t|	<.0001
Sign	M	1567.5	Pr >= |M|	<.0001
Signed Rank	S	2457840	Pr >= |S|	<.0001

Tests for Normality

Test	Statistic		p Value	
Kolmogorov-Smirnov	D	0.054671	Pr > D	<0.01000
Cramer-von Mises	W-Sq	2.52857	Pr > W-Sq	<0.00500
Anderson-Darling	A-Sq	16.36646	Pr > A-Sq	<0.00500

The next example highlights the COMPUTE statement, with which you can further customize the **p Value** column that is shown in the previous example. In this example, the change is made only to the Tests for Normality table. A comparison is shown of the *p*-value with five decimal places versus the *p*-value with four decimal places.

The code for this example requires the COLUMN statement that is taken from the **Base.Univariate.Normal** table definition discussed earlier in this example. In addition, the calculated variable CALCP is added to the COLUMN statement.

```
proc template;
   edit Base.Univariate.Normal;
      column Test TestLab Stat pType calcp psign pValue;
      define calcp;
         header="Comparison";
         print_headers=on;
         compute as '('||psign||trim(put(pvalue,pvalue7.5))||
                    ' vs '||psign||trim(put(pvalue,pvalue6.4))||')';
      end;
      define pValue;
         parent=Base.Univariate.PValue;
         print=no;
      end;
   end;
run;

ods select testsfornormality testsforlocation;

proc univariate data=sashelp.heart normal;
   var  cholesterol;
   class status;
run;
```

In the resulting table, you can see that the PRINT=NO statement suppresses values in the **p Value** column, and a new column is added that shows the *p*-value comparison.

Tests for Normality

Test	Statistic		p Value	Comparison
Shapiro-Wilk	W	0.962036	Pr < W	(<0.00010 vs <0.0001)
Kolmogorov-Smirnov	D	0.052813	Pr > D	(<0.01000 vs <0.0100)
Cramer-von Mises	W-Sq	1.09348	Pr > W-Sq	(<0.00500 vs <0.0050)
Anderson-Darling	A-Sq	6.776013	Pr > A-Sq	(<0.00500 vs <0.0050)

As you can see from these examples, PROC TEMPLATE offers a world of customization to existing table templates, and it enables you to build your own tables. Be aware that you can use PROC TEMPLATE with the DATA _NULL_ step. However, this functionality falls in the realm of **new construction**, so it is not explored in the scope of this paper. See "Recommended Reading," later in this paper, for two excellent resources that cover this topic.

KNOW YOUR SOURCE: USING THE ODS VERIFY STATEMENT TO IDENTIFY AND VERIFY THE COLUMN, STYLE, AND TABLE TEMPLATES THAT ARE IN USE

When you involve professionals in your renovation project, you will want to check their credentials. You want to verify their list of references, and you want to see examples of projects that they have completed. The same is true of your ODS output. If there are any doubts about the current status of the tables or columns in use by your program, you can use the handy ODS VERIFY statement to determine what overrides are in effect that are not the default settings.

```
242  ods verify on;
243  proc univariate data=sashelp.heart normal;
244       var cholesterol;
245       class status;
246   run;

WARNING: Template COMMON.PVALUE was not supplied by SAS Institute!
```

The **`Common.PValue`** column template was updated in the previous section “A Table Transformation: Using the TEMPLATE Procedure to Change the Look of a Standard Table.” When you use the ODS VERIFY ON statement, the log issues a warning to inform you that this template is not supplied by SAS.

By default, PROC TEMPLATE stores results in the SASUSER library, so any changes made to the table or column template remain in effect across sessions. But suppose that you use the changes that were made by the previous code and you want those changes to appear **only** in the current SAS session. In that case, you should use PROC TEMPLATE to delete the edited table and column templates. For this example, then, the sample code should end with following PROC TEMPLATE step:

```
proc template;
   delete base.univariate.normal;
   delete base.univariate.PValue;
   delete Common.PValue;
run
```

A STRUCTURAL TRANSFORMATION: USING THE DOCUMENT FACILITY TO CHANGE OUTPUT ORDER

Some renovation projects take on the scope of extreme. When walls must be removed or added to reconfigure one or more living spaces, the required tools and expertise move to a higher level. To perform such an extensive transformation on ODS tables and graphs, you need extreme power tools: the ODS DOCUMENT statement and the DOCUMENT procedure (collectively known as the ODS DOCUMENT facility) along with SAS macro language.

The project that is discussed in this section is based on a lengthy code sample that displays the efficiency, power, and flexibility of your SAS power tools. This code is available for you to download from the following location:

`support.sas.com/rnd/papers/sasgf13/170996_SASResultsRenovation.zip`

The renovation project that is encompassed in the appendix code has been discussed previously in two other documents:

- SAS Sample 36505, "Organizing output based on BY-variable values: PROC DOCUMENT or Macro" (SAS Institute Inc., 2009)
- "Have It Your Way: Rearrange and Replay Your Output with ODS DOCUMENT" (Zender, 2009)

The project represented in the sample code illustrates how to rearrange BY-group information so that it displays across procedure steps, and it adds SAS macro logic to automate this process, which is described in the two resources listed above. This project completely restructures output that is generated by two procedure steps (PROC TABULATE and PROC SGPLOT) in the sample code.

When output is generated from multiple PROC steps, the default order for displaying the output is as follows:

- PROC 1:
  - BY-group 1
  - BY-group 2

    . . .*more BY groups***. . .**
  - BY-group *n*
- PROC 2:
  - BY-group 1
  - BY-group 2

    . . .*more BY groups***. . .**
  - BY-group *n*

While this ordering is correct, or structurally sound, it might not be the order that you or your customers need. The BY-group information is relevant to someone who needs to use the tables together as a unit.

The sample code in the appendix illustrates how to display results in a different order, with only a single run of each PROC step. The order defined in the sample is as follows:

- BY-group 1
  - PROC 1
  - PROC 2
- BY-group 2
  - PROC 1
  - PROC 2

  . . .*more BY groups***. . .**
- BY-group *n*
  - PROC 1
  - PROC 2

Power-Tool of Choice: Using the DOCUMENT Facility for Maximum Efficiency

Many renovations are undertaken because you want to improve the flow of your space and to make tasks such as cooking or entertaining more efficient. Often, a renovation is more of a desire than a necessity; a desire for improvement motivates you to consider alternatives for your space. In the same way, the sample code for this project is not absolutely necessary to change the order of your output. There are numerous ways to accomplish the same task. However, the numbers of columns and observations in SAS data sets are growing at an astronomical rate. With ever larger data sets being processed, it has become even more important to be as efficient as possible when you analyze such large data sets. The DOCUMENT facility enables you to maximize efficiency when processing data in the following way.

The item store that is generated by the ODS DOCUMENT destination stores PROC and DATA step output in binary format. The DOCUMENT procedure enables you to manipulate an item store that is created by the DOCUMENT destination, so that the output stored in the item store can be replayed to one or more ODS destinations. You can replay the output in any order, and the replay is accomplished without any of the original overhead processing on your data.

Prudent Advice: Always Read the Instruction Manual

Most major renovation projects cannot be completed over the course of a weekend and without the help of a blueprint. Likewise, the sample code that is required for this example to change output order is lengthy, and it requires significant time on the part of the programmer. Because the sample code is so lengthy, it has been made available for you to download, as mentioned previously. This code contains an instruction manual in the form of copious comments. The explanations offered in the comments encourage you to become familiar with the DOCUMENT facility and to expand your use of the SAS macro language so that you can become an expert with these power tools. While the initial coding time might take longer than running your PROC steps over and over, it is well worth the time and effort given the efficiency that you gain by having to run each procedure only once and then replaying the output in the order that you want, to any number of destinations.

The code sample first uses the ODS DOCUMENT statement to store the results of PROC TABULATE and PROC SGPLOT (which were run against a sorted version of the SASHELP.CARS data set) into one item store for each

PROC step. The MPRINT option displays the SAS code that is generated by the macro logic in the SAS log. The following snippet shows the creation of the document PERM.FIRST by the ODS DOCUMENT statement. The code stores the PROC TABULATE output with all BY-group values for the variable TYPE.

```
MPRINT(REORDER):   ods _all_ close;
MPRINT(REORDER):   ods document name=perm.first(write);
MPRINT(REORDER):   proc tabulate data=perm.cars contents='';
MPRINT(REORDER):   Title "Type is: #byval(Type)";
MPRINT(REORDER):   by Type;
MPRINT(REORDER):   class origin cylinders;
MPRINT(REORDER):   var mpg_city;
MPRINT(REORDER):   table origin*cylinders, mpg_city*mean;
MPRINT(REORDER):   run;

NOTE: There were 428 observations read from the data set PERM.CARS.
NOTE: PROCEDURE TABULATE used (Total process time):
      real time           0.04 seconds
      cpu time            0.03 seconds


MPRINT(REORDER):   ods document close;
```

Then, metadata information for the PERM.FIRST item store is placed into a data set with the ODS OUTPUT statement, as shown in the following output. That data set is used later to generate macro variables for each of the unique table and graph entries for every BY group in the data set.

```
MPRINT(REORDER):   ods output properties=firstdoc;
MPRINT(REORDER):   proc document name=perm.first;
MPRINT(REORDER):   ods listing;
MPRINT(REORDER):   list / levels=all;
MPRINT(REORDER):   run;
NOTE: The data set WORK.FIRSTDOC has 13 observations and 2 variables.
MPRINT(REORDER):   quit;

NOTE: PROCEDURE DOCUMENT used (Total process time):
      real time           0.03 seconds
      cpu time            0.03 seconds


MPRINT(REORDER):   proc print data=firstdoc;
MPRINT(REORDER):   run;
```

The PROC PRINT step is unnecessary, but it is included here to illustrate the contents of the document. This PROC PRINT step generates the following table of observations:

Obs	Path	Type
1	\Tabulate#1	Dir
2	\Tabulate#1\ByGroup1#1	Dir
3	\Tabulate#1\ByGroup1#1\Table#1	Table
4	\Tabulate#1\ByGroup2#1	Dir
5	\Tabulate#1\ByGroup2#1\Table#1	Table
6	\Tabulate#1\ByGroup3#1	Dir
7	\Tabulate#1\ByGroup3#1\Table#1	Table
8	\Tabulate#1\ByGroup4#1	Dir
9	\Tabulate#1\ByGroup4#1\Table#1	Table
10	\Tabulate#1\ByGroup5#1	Dir
11	\Tabulate#1\ByGroup5#1\Table#1	Table
12	\Tabulate#1\ByGroup6#1	Dir
13	\Tabulate#1\ByGroup6#1\Table#1	Table

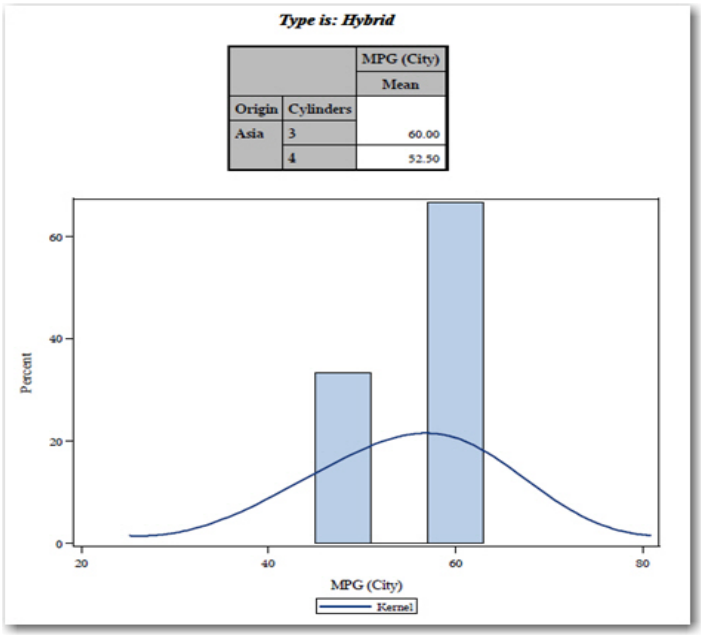
Next, the code creates macro variables to hold the BY variable, the number of BY groups, the name of the input data set, and each of the table and graph entries in each document. The following output shows the macro variables that are created from the data set FIRSTDOC:

```
REORDER FIRSTPATH1 \Tabulate#1\ByGroup1#1\Table#1
REORDER FIRSTPATH2 \Tabulate#1\ByGroup2#1\Table#1
REORDER FIRSTPATH3 \Tabulate#1\ByGroup3#1\Table#1
REORDER FIRSTPATH4 \Tabulate#1\ByGroup4#1\Table#1
REORDER FIRSTPATH5 \Tabulate#1\ByGroup5#1\Table#1
REORDER FIRSTPATH6 \Tabulate#1\ByGroup6#1\Table#1
REORDER DSN sashelp.cars
REORDER TOTAL 6
REORDER BYVAR Type
```

A macro DO loop generates a new DOCUMENT item store. From this item store, each BY group is replayed into an open PDF destination. The final PDF file (after.pdf) combines, per BY group, a TABULATE table and an SGPLOT image on each page. The following output is a snippet of the log that is created:

```
MPRINT(REORDER):   proc document name=reorder(write);
MPRINT(REORDER):   ods pdf file='after.pdf' notoc;
NOTE: Writing ODS PDF output to DISK destination "C:\Program Files\SASH
MPRINT(REORDER):   replay \perm.first\Tabulate#1\ByGroup1#1\Table#1 ;
MPRINT(REORDER):   run;

MPRINT(REORDER):   ods pdf startpage=no ;
MPRINT(REORDER):   replay \perm.second\Sgplot#1\ByGroup1#1\SGPlot#1 ;
MPRINT(REORDER):   run;

MPRINT(REORDER):   ods pdf startpage=now ;
MPRINT(REORDER):   replay \perm.first\Tabulate#1\ByGroup2#1\Table#1 ;
MPRINT(REORDER):   run;

MPRINT(REORDER):   ods pdf startpage=no ;
MPRINT(REORDER):   replay \perm.second\Sgplot#1\ByGroup2#1\SGPlot#1 ;
MPRINT(REORDER):   run;
```

One of the pages of output that is produced in the after.pdf file is shown here:



The original data set was processed twice—once by PROC TABULATE and once by PROC SGPLOT. Alternative logic to obtain the same results without using the DOCUMENT facility would require the data set to be processed twice for every unique value of the BY group.

CONCLUSION

Whether your SAS output needs a minor update or a major overhaul, SAS provides you with all the tools you need for your renovation projects. This paper introduced you to simple ODS statements and sophisticated procedures that enable you to improve the appearance of your output. Understanding these ODS tools and how to implement them will help you perfect your output so that your personal or corporate style is reflected in any output that you create with SAS.

REFERENCES

Lawhorn, Bari. 2011. "Let's Give'Em Something to TOC about: Transforming the Table of Contents of Your PDF File." *Proceedings of the SAS Global Forum 2011 Conference.* Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings11/252-2011.pdf**

SAS Institute Inc. 2009. SAS Sample 36505, "Organizing output based on BY-variable values: PROC DOCUMENT or Macro." Cary, NC: SAS Institute Inc. Available at **http://support.sas.com/kb/36/505.html**.

Zender, Cynthia L. 2009. "Have It Your Way: Rearrange and Replay Your Output with ODS DOCUMENT." *Proceedings of the SAS Global Forum 2009 Conference.* Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings09/318-2009.pdf**.

RECOMMENDED READING

Smith, Kevin D. 2012. “ODS DOCUMENT From Scratch.” *Proceedings of the SAS Global Forum 2012 Conference.* Cary, NC SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings12/273-2012.pdf**.

Smith, Kevin D. 2007. “PROC TEMPLATE Tables from Scratch.” *Proceedings of the SAS Global Forum 2007 Conference.* Cary, NC: SAS Institute Inc. Available at **www2.sas.com/proceedings/forum2007/221-2007.pdf**.

Zender, Cynthia L. 2005. "The Power of TABLE Templates and DATA _NULL_.” *Proceedings of the Thirtieth Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. Available at **www2.sas.com/proceedings/sugi30/088-30.pdf**.

ACKNOWLEDGMENTS

The author would like to thank Jane Eslinger, David Kelley, and Cynthia Zender for their contributions to and reviews of this paper.

CONTACT INFORMATION

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

Bari Lawhorn
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: support@sas.com
Web: support.sas.com