SCSUG2011-S06

The Greatest Hits: ODS Essentials Every User Should Know

Cynthia Zender, SAS Institute Inc., Cary, NC, USA

ABSTRACT

Just when you think you know every song (feature) in the ODS hit parade, you discover that there's an option or destination or feature that has you singing its praises because the feature boosted your reports to the next level.

This paper covers some of the essential features and options of ODS that every user needs to know to be productive. This paper shows concrete code examples of the ODS "Greatest Hits". Come to this session and learn some of the essential reasons why ODS and Base SAS® rock!

INTRODUCTION

The Output Delivery System (ODS) has many features that are well known by now. However, beyond the basic sandwich technique, there are some essential features that I consider to be the ODS Greatest Hits. The purpose of this paper is to describe these hits and to share some code and my reasons why these are all ODS essentials that I could not work without. (Along the way, see whether you can figure out the names of the musical hits that make up my major headings.)

THE NAME GAME: #1 USING ODS OUTPUT OBJECTS

The output object is the foundation of ODS. Almost every SAS procedure uses output objects. There are two types of output objects created by ODS: tabular output objects and graphical output objects. No matter which type of output object your procedure creates, you can discover useful information about the object with the ODS TRACE statement.

It might be true, as Shakespeare said, that a rose by any other name would smell as sweet. However, when it comes to output objects created by SAS procedures, an output object has a specific name that cannot be changed, and the object cannot be called by any other name. If you use the ODS TRACE statement as shown below, you can see in the SAS log that each output object has other attributes in addition to the Name attribute. The five most common attributes are Name, Label, Template, Path, and Label Path (although not every output object has or uses every attribute).

```
ods trace on / label;
. . . SAS code that produces output . . .
ods trace off;
```

Output from the ODS TRACE statement is written to the SAS Log window when only the LABEL option is used as shown above. (Output sent to the LISTING destination from the procedures used within the ODS TRACE sandwich are not the focus of this topic. Instead, we want to look at the output object information from the SAS log.)

EXAMINE OUTPUT OBJECT NAME AND OTHER ATTRIBUTES

The program below uses familiar procedures to illustrate the attributes of ODS output objects.

```
ods trace on / label;
proc sort data=sashelp.class out=class; by age name; run;

ods graphics on;
proc reg data=class;
  model age=height;
run;
ods graphics off;

proc means data=class min mean max;
  class age;
  var height weight;
run;

proc contents data=class; run;

proc print data=class;  run;
```

```
ods trace off;
```

The SORT procedure creates an output data set. However, it does not create any output that can be routed to an ODS destination, so no ODS TRACE information is written to the SAS log for the SORT procedure. Output object information for the other procedures is shown in Output 1 through Output 4 below. **Error! Reference source not found.** shows the partial tabular and graphical output object information that is created by the REG procedure.

```
Output Added:
-------------
Name:       NObs
(LOG lines deleted to fit PROC REG output on one page)

Output Added:
-------------
Name:       ANOVA
Label:      Analysis of Variance
Template:   Stat.REG.ANOVA
Path:       Reg.MODEL1.Fit.Age.ANOVA
Label Path: 'The Reg Procedure'.'MODEL1'.'Fit'.Age.'Analysis of Variance'

Output Added:
-------------
Name:       FitStatistics
Label:      Fit Statistics
Template:   Stat.REG.FitStatistics
Path:       Reg.MODEL1.Fit.Age.FitStatistics
Label Path: 'The Reg Procedure'.'MODEL1'.'Fit'.Age.'Fit Statistics'

Output Added:
-------------
Name:       ParameterEstimates
Label:      Parameter Estimates
Template:   Stat.REG.ParameterEstimates
Path:       Reg.MODEL1.Fit.Age.ParameterEstimates
Label Path: 'The Reg Procedure'.'MODEL1'.'Fit'.Age.'Parameter Estimates'

Output Added:
-------------
Name:       DiagnosticsPanel
Label:      Fit Diagnostics
Template:   Stat.REG.Graphics.DiagnosticsPanel
Path:       Reg.MODEL1.ObswiseStats.Age.DiagnosticPlots.DiagnosticsPanel
Label Path: 'The Reg Procedure'.'MODEL1'.'Observation-wise
Statistics'.Age.'Diagnostic Plots'.'Fit
Diagnostics'

Output Added:
-------------
Name:       ResidualPlot
Label:      Height
Template:   Stat.REG.Graphics.ResidualPlot
Path:       Reg.MODEL1.ObswiseStats.Age.ResidualPlots.ResidualPlot
Label Path: 'The Reg Procedure'.'MODEL1'.'Observation-wise
Statistics'.Age.'Residual Plots'.'Height'

Output Added:
-------------
Name:       FitPlot
Label:      Fit Plot
Template:   Stat.REG.Graphics.Fit
Path:       Reg.MODEL1.ObswiseStats.Age.FitPlot
Label Path: 'The Reg Procedure'.'MODEL1'.'Observation-wise Statistics'.Age.'Fit
Plot'
```

Output 1. Log from the REG Procedure

```
Output Added:
-------------
Name:       Summary
Label:      Summary statistics
Template:   base.summary
Path:       Means.Summary
Label Path: 'The Means Procedure'.'Summary statistics'
```

Output 2. Log from the MEANS Procedure

```
Output Added:
-------------
Name:       Attributes
Label:      Attributes
Template:   Base.Contents.Attributes
Path:       Contents.DataSet.Attributes
Label Path: 'The Contents Procedure'.'WORK.CLASS'.'Attributes'

Output Added:
-------------
Name:       EngineHost
Label:      Engine/Host Information
Template:   Base.Contents.EngineHost
Path:       Contents.DataSet.EngineHost
Label Path: 'The Contents Procedure'.'WORK.CLASS'.'Engine/Host Information'

Output Added:
-------------
Name:       Variables
Label:      Variables
Template:   Base.Contents.Variables
Path:       Contents.DataSet.Variables
Label Path: 'The Contents Procedure'.'WORK.CLASS'.'Variables'

Output Added:
-------------
Name:       Sortedby
Label:      Sortedby
Template:   Base.Contents.Sortedby
Path:       Contents.DataSet.Sortedby
Label Path: 'The Contents Procedure'.'WORK.CLASS'.'Sortedby'
```

Output 3. Log from the CONTENTS Procedure

```
Output Added:
-------------
Name:       Print
Label:      Data Set WORK.CLASS
Data Name:
Path:       Print.Print
Label Path: 'The Print Procedure'.'Data Set WORK.CLASS'
```

Output 4. Log from the PRINT Procedure

Notice that some of the output objects are different from the others. Most notably, the PRINT procedure does not use a table or a graph template, so there is no template attribute. Instead, the PRINT procedure has a Data Name attribute (which is blank in the above output.) Notice also that in the REG procedure output, the output objects use variable information for the Label and Label Path information. However, output objects from the CONTENTS procedure do not use the variable names for the Label or Label Path.

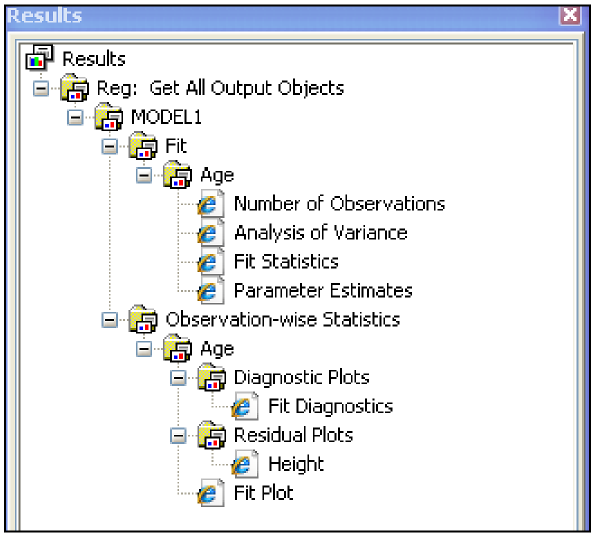
Why go to all this trouble to find out the attributes of the output objects? It is one of those “chicken and egg” situations. To do anything more interesting than send your output to an ODS destination, you first need to know the names of the output objects. This means that you have to learn about the ODS TRACE statement. When you know the names of the output objects, you can select and exclude output objects from ODS destinations. Because one of the ODS destinations is the ODS OUTPUT destination, you can use the object’s name to create a SAS data set.

USE OUTPUT OBJECT NAME TO SELECT IT

The program below runs the REG procedure and directs the output to the HTML destination. By default, all output objects are sent to the ODS destination, as you can see from Display 1. Display 1 shows the SAS Results window and all the output objects. Compare Display 1 to Display 2. Display 2 shows the output from the same program, but an ODS SELECT statement has been added that selects only the ParameterEstimates output object.

```
ods listing close;

ods html path='.' (url=none)
         file='get_everything.html' style=sasweb;
  title 'Get All Output Objects';
  ods graphics on;
  proc reg data=class;
    model age=height;
  run;
  quit;
  ods graphics off;
ods html close;
```

Display 1 shows the Results window from the above program. It shows the tabular and graphic output objects that have been routed to the HTML destination.



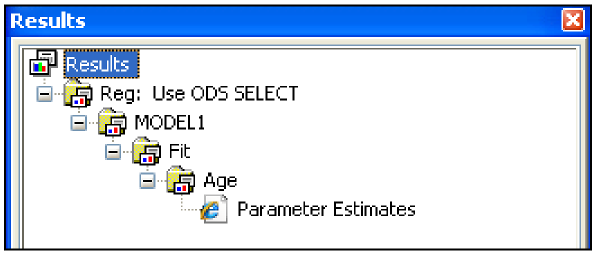
```
Results
  Reg:  Get All Output Objects
    MODEL1
      Fit
        Age
          Number of Observations
          Analysis of Variance
          Fit Statistics
          Parameter Estimates
      Observation-wise Statistics
        Age
          Diagnostic Plots
            Fit Diagnostics
          Residual Plots
            Height
          Fit Plot
```

Display 1. Expanded Results Window Showing All Output Objects

The following example is the same as the preceding example, except that I've added a SELECT statement. (The LISTING destination is still closed when this program runs.)

```
ods html path='.' (url=none)
         file='sel_parm.html' style=sasweb;
  title 'Use ODS SELECT';
  ods html select ParameterEstimates;
  ods graphics on;
  proc reg data=class;
    model age=height;
  run;
  quit;
  ods graphics off;
ods html close;
```

Display 2 shows the output of this code. Only the tabular output object for the ParameterEstimates results from the REG procedure is shown in the SAS Results window.



Display 2. Expanded Results Window Showing the ParameterEstimates Output Object

In addition to the ODS SELECT statement, there is also an ODS EXCLUDE statement. This means you can select output objects for some destinations while excluding them from other destinations. A very powerful feature, indeed. But wait, the hits just keep on coming! Now that you know the output object name, you can create an output data set from the output object.

USE OUTPUT OBJECT NAME TO CREATE AN OUTPUT DATASET

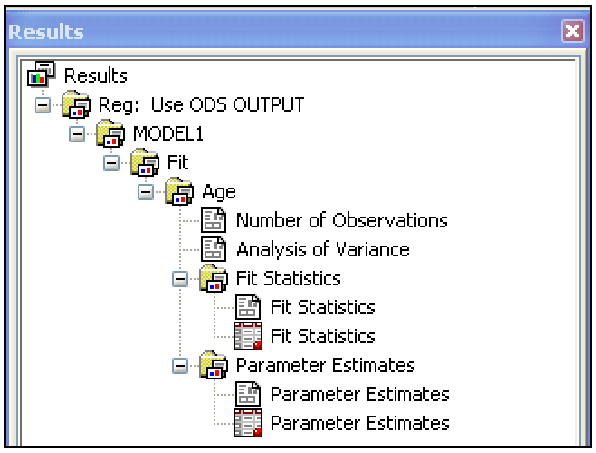
Most people know about the report destinations ODS HTML, ODS RTF, and ODS PDF. There are many other report destinations such as ODS CSV, ODS TAGSETS.EXCELXP, ODS TAGSETS.RTF, and so on. However, the ODS OUTPUT destination is a *data* or *output* destination. Many SAS procedures already have syntax for creating an output data set from the procedure output, but each procedure has its own method of creating a data set. Some procedures use an OUT= option and some use an OUTPUT statement, to name just a few possibilities. The ODS OUTPUT destination enables you to use the same syntax to create an output data set, and all you have to know (in the simplest mode) is the output object name that you want to select. The syntax model is:

```
ODS OUTPUT <obj-name> = <SAS-libref-and-dataset-name>;
```

A review of the documentation on the ODS OUTPUT statement shows that you can use the Name, Path, or Label Path attributes to select output objects in the ODS OUTPUT statement as well as ODS SELECT and ODS EXCLUDE statements. For illustration purposes, the program below uses the Name for the ParameterEstimates and FitStatistics output objects. This means that two SAS data sets are created from the following PROC REG step:

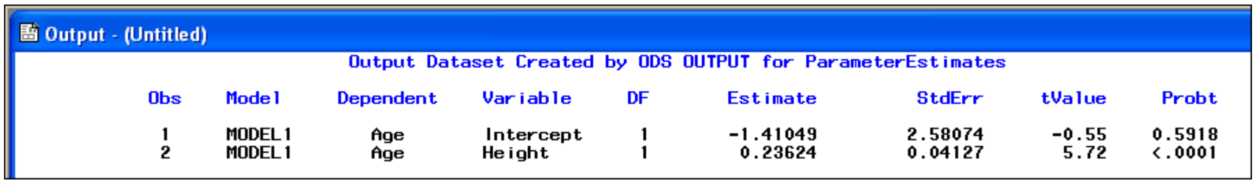
```
ods listing;
title 'Use ODS OUTPUT';
ods output ParameterEstimates=work.parmest FitStatistics=work.fitstat;

proc reg data=class;
  model age=height;
run;
quit;
```

Display 3 shows the SAS Results window after the REG procedure is finished.



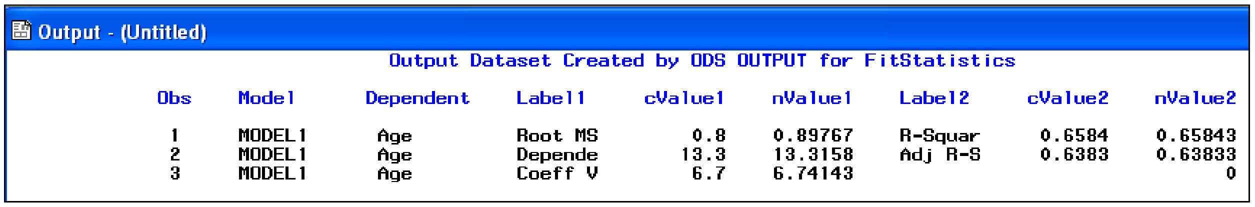
Display 3. Expanded Results Window Showing the Output Data Sets

You can see that, in addition to all of the LISTING destination output icons, the two result nodes for ParameterEstimates and FitStatistics show the icon for a SAS data set. The PRINT procedure views of these data sets are shown in Display 4 and Display 5.



```
Output - (Untitled)
                     Output Dataset Created by ODS OUTPUT for ParameterEstimates

      Obs    Model     Dependent    Variable     DF     Estimate      StdErr     tValue     Probt

       1     MODEL1       Age       Intercept     1     -1.41049     2.58074     -0.55     0.5918
       2     MODEL1       Age       Height        1      0.23624     0.04127      5.72     <.0001
```

Display 4. PRINT Procedure Output of WORK.PARMEST



```
Output - (Untitled)
                     Output Dataset Created by ODS OUTPUT for FitStatistics

   Obs    Model     Dependent    Label1     cValue1    nValue1    Label2     cValue2    nValue2

    1     MODEL1       Age       Root MS        0.8    0.89767    R-Squar    0.6584     0.65843
    2     MODEL1       Age       Depende       13.3    13.3158    Adj R-S    0.6383     0.63833
    3     MODEL1       Age       Coeff V        6.7    6.74143                                0
```

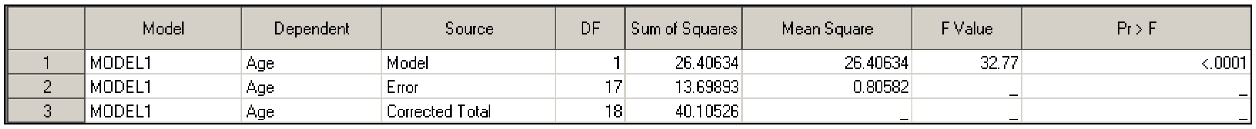
Display 5. PRINT Procedure Ouput of WORK.FITSTAT

Each data set created with the ODS OUTPUT statement has its own structure. This structure might or might not be the same as the structure created by using a procedure's own output syntax. For example, the ParameterEstimates data set has variable (column) names that correspond to statistic names (such as Estimate and StdErr). However, the FitStatistics data set can have two statistics per observation. The first observation in this output contains Root MS and R-Square statistics. Since Root MS is the value for the Label1 variable on the first observation, all the columns ending in 1 on the first observation are also Root MS-related. There is a character variable for the Root MS statistic (cValue1), if you need a less precise, character version of the value for reporting. There is also a numeric variable (nValue1), which shows more numeric precision and allows this Root MS value to be used in subsequent calculations.

In addition to using the Name of the output object, you can also use different combinations of the object Name, Path, Label, or Label Path to select output objects when you are creating a data set. I usually use the unquoted strings because the unquoted strings are shorter and require less typing. For example, the ODS OUTPUT syntax below creates three different versions of the ANOVA output object by using the Name, the full Path, and a partial Path.

```
ODS OUTPUT ANOVA=work.usename
           Reg.MODEL1.Fit.Age.ANOVA=work.usepath
           Fit.Age.Anova=work.partial;

proc reg data=class;
  model age=height;
run;
quit;
```

The three data sets created by the example above all contain the same ANOVA information for the PROC REG step. The ViewTable display of the first data set (WORK.USENAME) is shown in Display 6. Note that the ability to use different attributes to select output objects also works with the ODS SELECT and ODS EXCLUDE statements.



	Model	Dependent	Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
1	MODEL1	Age	Model	1	26.40634	26.40634	32.77	<.0001
2	MODEL1	Age	Error	17	13.69893	0.80582	_	_
3	MODEL1	Age	Corrected Total	18	40.10526	_	_	_

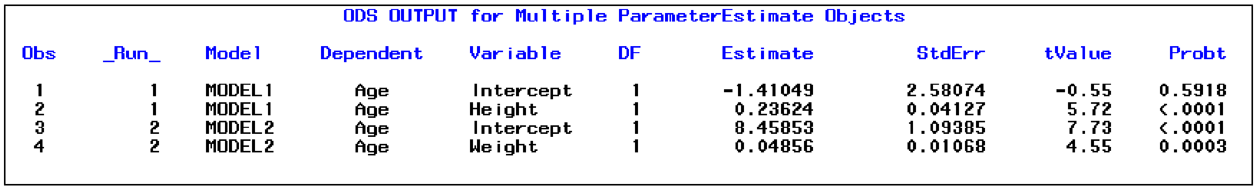
Display 6. PRINT Procedure Output of WORK.USENAME

Another feature of the ODS OUTPUT destination is the ability to get output from multiple RUN groups or output from multiple runs of the same procedure into one data set by using the PERSIST= option. The example below shows the use of the PERSIST=RUN statement with a PROC REG statement and two MODEL statements.

```
ods output ParameterEstimates(persist=run)=work.multPE;
```

```
proc reg data=class;
  var height weight;
  model age=height;
run;

  model age=weight;
run;
quit;
ods output clear;
```

Display 7 shows the results from the above code. The first two observations in the data set represent the parameter estimate information for the AGE=HEIGHT model, and the last two observations in the data set represent the parameter estimate information for the AGE=WEIGHT model.



```
                         ODS OUTPUT for Multiple ParameterEstimate Objects

Obs    _Run_    Model     Dependent    Variable     DF      Estimate       StdErr     tValue    Probt

 1       1      MODEL1       Age        Intercept    1      -1.41049      2.58074      -0.55    0.5918
 2       1      MODEL1       Age       Height        1       0.23624      0.04127       5.72    <.0001
 3       2      MODEL2       Age        Intercept    1       8.45853      1.09385       7.73    <.0001
 4       2      MODEL2       Age       Weight        1       0.04856      0.01068       4.55    0.0003
```

Display 7. PRINT Procedure Output of WORK.MULTPE

Knowing about ODS output objects and how to use their names (and other attributes) enables you to perform tasks such as selecting the PROBT statistic for a particular model and using it in other ODS reports. So learn how to play the name game with ODS output objects the next time you want only a few objects or you need to create SAS data sets from a procedure.

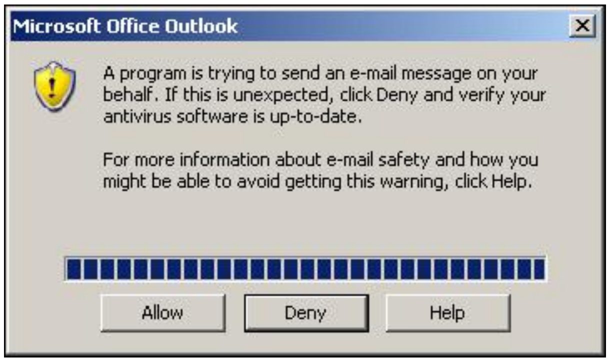
PLEASE MR. POSTMAN: #2 SENDING ODS RESULTS BY EMAIL

The ability to send ODS result files by e-mail is possible with the FILENAME EMAIL engine. It has been possible to send e-mail with SAS since the early mainframe days. With the advent of ODS, one way to use the EMAIL engine is to follow a two-step process. First, create your ODS result file. Second, generate an e-mail that sends the ODS result file as an attachment. The program code below follows this two-step approach.

```
** Step 1: Create ODS Result File;
ods rtf body='c:\temp\class.rtf' rs=none style=sasweb;

   proc print data=sashelp.class;
       title 'ODS RTF Report';
   run;

ods rtf close;

** Step 2: Use Email Engine to send RTF file as an attachment;
filename doemail email
    to=('one.person@sas.com' 'another.person@sas.com')
    from='ima.programmer@sas.com'
    cc=('also.interested@sas.com')
    subject='Look at this ODS RTF report'
    attach='c:\temp\class.rtf';

data _null_;
    file doemail;
    put 'This is a test email with an RTF attachment.';
run;
```

Because the e-mail addresses above are for illustration purposes only, this e-mail code will not work if you submit this code exactly as shown. To use the code, you need to change the TO, FROM, and CC options appropriately. Also, depending on the e-mail client that you use and the setup configuration for e-mail that you have, you might need to work with your SAS administrator to make sure that the proper system options are in place for sending e-mail. For example, the default access method on Windows is EMAILSYS=MAPI, but you might want or need to use SMTP as the access method (which requires other configuration settings).

When I use the above program to send an e-mail to myself from a backup machine (which uses Microsoft Outlook 2007 on Windows XP) to my work machine (which uses Microsoft Outlook 2010 on Windows 7), Outlook 2007 displays the message shown in Display 8.

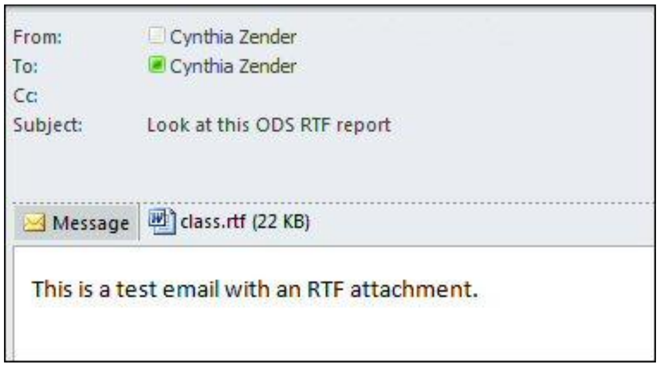


A program is trying to send an e-mail message on your behalf. If this is unexpected, click Deny and verify your antivirus software is up-to-date.

For more information about e-mail safety and how you might be able to avoid getting this warning, click Help.

Display 8. Popup Window from Outlook 2007

When I respond by clicking the Allow button, the e-mail is sent. Then, when I open the mail on my work machine, Outlook 2010 displays the message shown in Display 9.



From: Cynthia Zender
To: Cynthia Zender
Cc:
Subject: Look at this ODS RTF report

Message | class.rtf (22 KB)

This is a test email with an RTF attachment.

Display 9. View of E-mail With RTF Attachement

Keep in mind that sending an attachment is not just subject to your e-mail system, but the recipient's e-mail system as well. For example, even though ODS can create RTF, PDF, HTML, and even XML output files, it is possible that some recipients might not be able to receive XML or HTML attachments for security reasons. So you might want to investigate a bit more about this feature of SAS and ODS.

Other papers have been written about sending e-mail with SAS, so this hit is not meant to be an exhaustive treatment of how to e-mail ODS output with SAS. A search of Technical Support notes and user group papers reveals other e-mail methods, including SMTP methods. However, with the ability to e-mail ODS results under your belt, you can bring your company closer to the "paperless office".

STEP BY STEP: #3 CONTROLLING ODS RESULT FILES

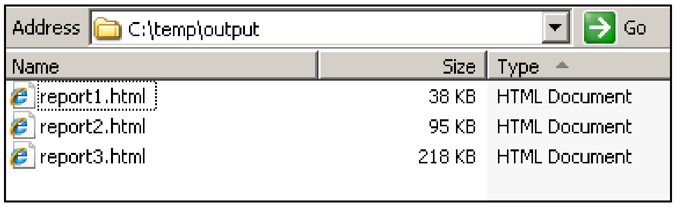
The ways to control the style of your ODS result files is an upcoming hit. This hit, or this group of related hits, show how to use options to affect your ODS result files. You can create multiple result files using the NEWFILE= option, or you can create multiple instances of output for the same destination using the ID= suboption. If you want to generate a Table of Contents, all three primary destinations support the use of the CONTENTS= option. With the ODS RTF and ODS PDF destinations, a separate Table of Contents page is inserted into the result output file. For the ODS HTML destination, the appropriate HTML frameset navigation structure is created or a separate CONTENTS file is created, depending on which syntax you use. Once you learn score for these options, I'm sure you'll find the one most useful for you.

USING THE NEWFILE= OPTION

The NEWFILE= option defines a breakpoint or starting point at which a new result file is generated and automatically named. Normally, with one ODS sandwich, all your output from one SAS procedure or multiple SAS procedures goes into the single result file that is named in the FILE= or BODY= option. You can change this default behavior by specifying the starting point for each new file.

```
ods html file='report1.html' style=sasweb newfile=proc;

  title '1) Proc Report Summary';
  ** proc report step;

  title '2) Proc Tabulate Summary';
  ** proc tabulate step;

  title '3) Proc Freq Report';
  ** proc freq step;

ods _all_ close;
```

To see the effect of using the NEWFILE= option, instead of looking at the output from each procedure, Display 10 shows the three output files created by ODS.



Name	Size	Type
report1.html	38 KB	HTML Document
report2.html	95 KB	HTML Document
report3.html	218 KB	HTML Document

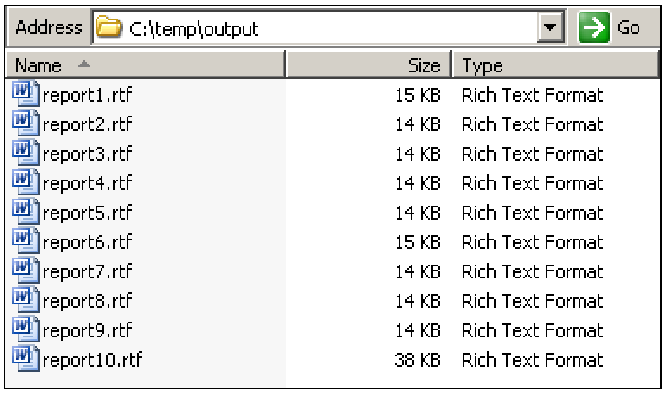
Display 10. Windows Explorer View of ODS HTML Results Files

Note that the output files are named REPORT1.HTML, REPORT2.HTML, and REPORT3.HTML because the breakpoint for each new file occurred with the start of each procedure. ODS named the files by incrementing the right-most number in the original FILE= option. If we had originally used a name of REPORT40.HTML in the FILE= option, then with three procedures, the output files would have been named REPORT40.HTML, REPORT41.HTML, and REPORT42.HTML. If we has specified , Y2010Rep1.HTML as the value for the FILE= option, the output files would have been named Y2010Rep1.HTML, Y2010Rep2.HTML, and Y2010Rep3.HTML because it is the right-most number that gets incremented.)

On the other hand, in the code shown below, the NEWFILE= option specifies that a new file is generated for each BYGROUP (assuming that the data are sorted for use with the BY statement).

```
ods rtf file='report1.rtf' newfile=bygroup;

  ** proc report step;
    by employee_country;

ods _all_ close;
```

In Display 11, we see that one RTF file was created for each of the ten unique values of EMPLOYEE_COUNTRY.



Address C:\temp\output

Name	Size	Type
report1.rtf	15 KB	Rich Text Format
report2.rtf	14 KB	Rich Text Format
report3.rtf	14 KB	Rich Text Format
report4.rtf	14 KB	Rich Text Format
report5.rtf	14 KB	Rich Text Format
report6.rtf	15 KB	Rich Text Format
report7.rtf	14 KB	Rich Text Format
report8.rtf	14 KB	Rich Text Format
report9.rtf	14 KB	Rich Text Format
report10.rtf	38 KB	Rich Text Format

Display 11. Windows Explorer View of ODS RTF Results Files

The breakpoints that you can use with the NEWFILE= option are NONE (default), BYGROUP, OUTPUT, PAGE, and PROC. Not all ODS destinations support the use of the NEWFILE= option, but the three major destinations (HTML, RTF, and PDF) all support this option as do other destinations such as ODS CSV, ODS CSVALL, and ODS PS. The best place to find out whether your destination of interest supports the NEWFILE= option is to look in the ODS documentation.

USING THE ID= SUBOPTION

ODS allows you to have multiple different destinations open at the same time, but it does not allow you to have multiple instances of the same destination open at the same time. For example, you cannot do this in your code:

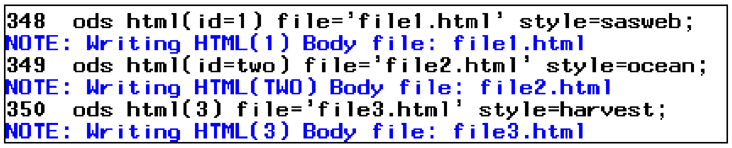
```
ods html file='use_sasweb.html' style=sasweb;
ods html file='use_analysis.html' style=analysis;
. . . more code . . .
ods _all_ close;
```

If you tried to use the above code, the second ODS HTML result file (USE_ANALYSIS.HTML) would contain all of the output, while the first file (USE_SASWEB.HTML) would appear to be empty. This happens because, behind the scenes, each open destination has a manager whose job it is to direct output to the FILE= (or BODY=) location. The fancy name for the manager is an Output Destination Agent (ODA). When you use the form of syntax above, the ODA does not know which of the two specified result files should get the procedure results. If there is no way to identify or distinguish between the two files, the ODA sends the procedure output to the second, or last, ODS HTML file referenced.

This is where the ID= suboption comes into play. Using the ID= suboption gives you (and the ODA) a way to distinguish between the multiple output files that you want to create for the same destination. When you specify the ID= suboption on your ODS invocation statement, each ID= specification causes a separate ODA to be invoked to handle the output that SAS is sending to ODS result files. The fancy description for what happens is that a new ODA is *instantiated* for every ID= suboption. For the code below, three ODAs are instantiated to manage the output.

```
ods html (id=1) file='file1.html' style=sasweb;
ods html (id=two) file='file2.html' style=ocean;
ods html (3) file='file3.html' style=harvest;
  proc report data=employees nowd split='#';
    title 'Report 1 ID= Behavior';
  run;

ods html (two) close;

  proc report data=employees nowd split='#';
    title 'Report 2 ID= Behavior';
  run;

ods html (id=1) close;
ods html (3) close;
```

In this code, three result files are created. Suboptions always go in parentheses on an ODS invocation statement. So immediately after the ODS HTML invocation, the ID= text strings are placed inside parentheses. The ID= suboption must be placed immediately after the destination specification. The text string that you specify does not need quotes. You can specify a number, such as ID=1, or you can specify a word, such as ID=TWO. Note in Display 12 how the NOTE message in the SAS log echoes the file information back to you.

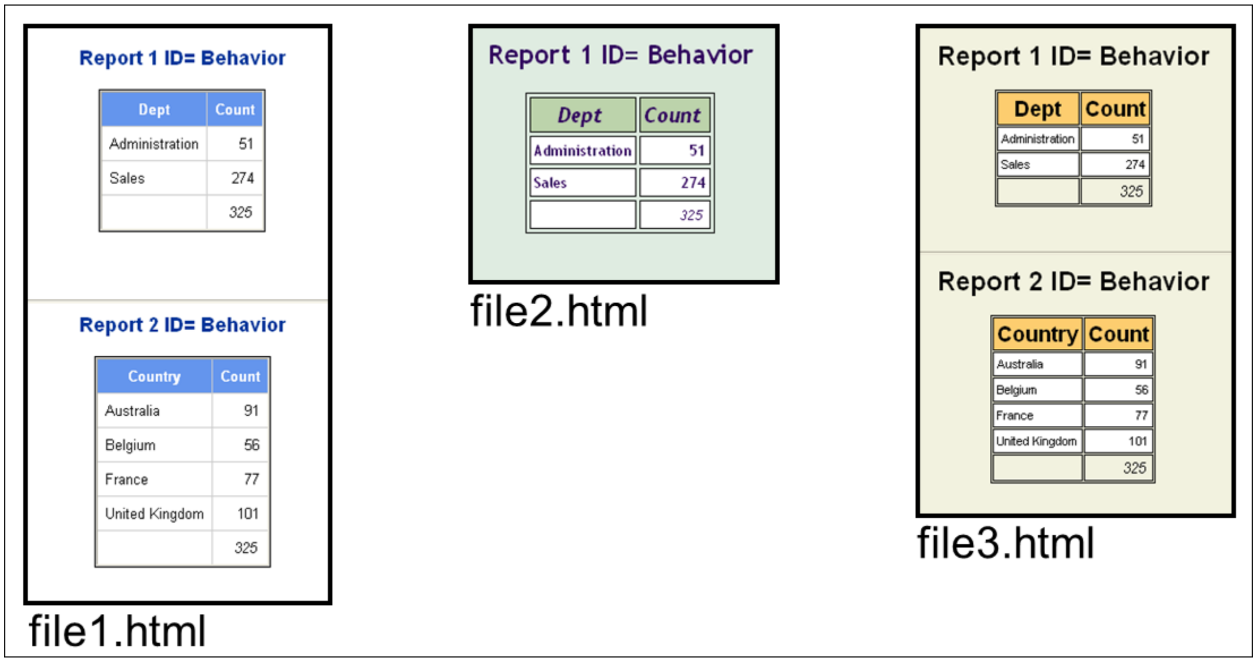


```
348  ods html(id=1) file='file1.html' style=sasweb;
NOTE: Writing HTML(1) Body file: file1.html
349  ods html(id=two) file='file2.html' style=ocean;
NOTE: Writing HTML(TWO) Body file: file2.html
350  ods html(3) file='file3.html' style=harvest;
NOTE: Writing HTML(3) Body file: file3.html
```

Display 12. SAS Log Messages With the ID= Suboption

The ID= suboption specification itself is not shown in the note. You only see the suboption value. ODS knows that the string inside the parentheses is the ID= string for the ODA. Technically, you do not need the ID= option specification at all (as shown in several places in the code above).

As shown above, selectively opening and closing instances of the same destination enables you to do things like use three different styles, send the output from only one PROC REPORT step (ID=two) to one file, and send the output from both PROC REPORT steps (ID= 1 and 3) to the other files. . Display 13 shows what the three outputs look like when each is viewed in a browser.



Report 1 ID= Behavior

Dept	Count
Administration	51
Sales	274
	325

Report 2 ID= Behavior

Country	Count
Australia	91
Belgium	56
France	77
United Kingdom	101
	325

file1.html

Report 1 ID= Behavior

Dept	Count
Administration	51
Sales	274
	325

file2.html

Report 1 ID= Behavior

Dept	Count
Administration	51
Sales	274
	325

Report 2 ID= Behavior

Country	Count
Australia	91
Belgium	56
France	77
United Kingdom	101
	325

file3.html

Display 13. ODS HTML Resuts Files for All Three Invocations

Another good example of using the ID= suboption would be using it to create output both on a local drive and on a network drive as a backup copy. Using the ID=suboption, you can create two files on different drives with only one run of your procedure step(s).

USING THE CONTENTS= OPTION

Even in the paperless office, there might be a need for a Table of Contents (TOC) for your reports. TOC generation happens differently for destinations that create paginated output like RTF and PDF versus destinations like HTML that do not create paginated output. Compare Display 14, which shows how the TOC would look for RTF and PDF output, with Display 15, which shows the CONTENTS= file for ODS HTML.

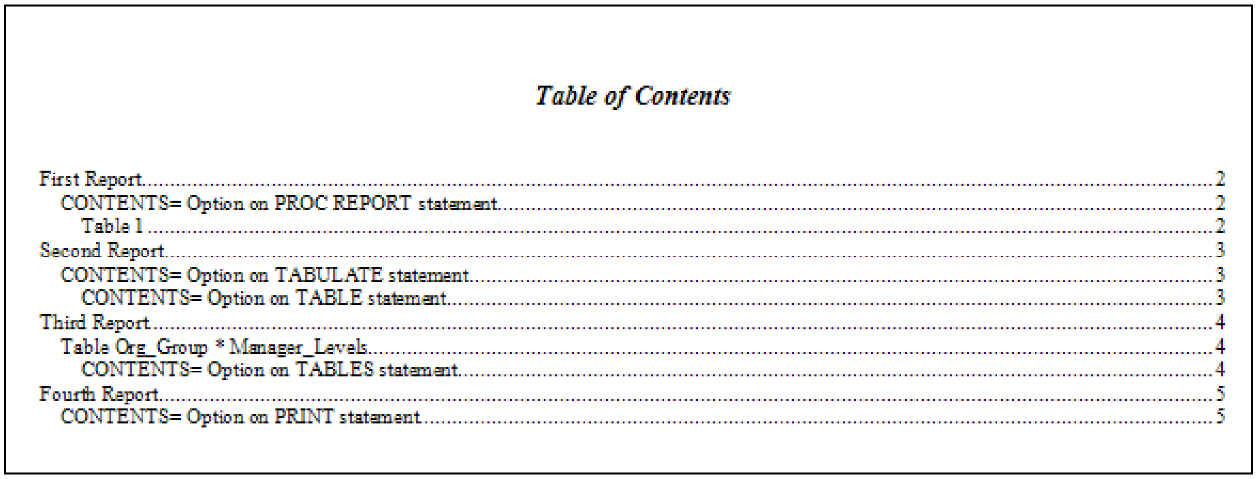


Table of Contents

```
First Report.....................................................2
   CONTENTS= Option on PROC REPORT statement.....................2
      Table 1....................................................2
Second Report....................................................3
   CONTENTS= Option on TABULATE statement........................3
      CONTENTS= Option on TABLE statement........................3
Third Report.....................................................4
   Table Org_Group * Manager_Levels..............................4
      CONTENTS= Option on TABLES statement.......................4
Fourth Report....................................................5
   CONTENTS= Option on PRINT statement...........................5
```

Display 14. RTF Table of Contents Page

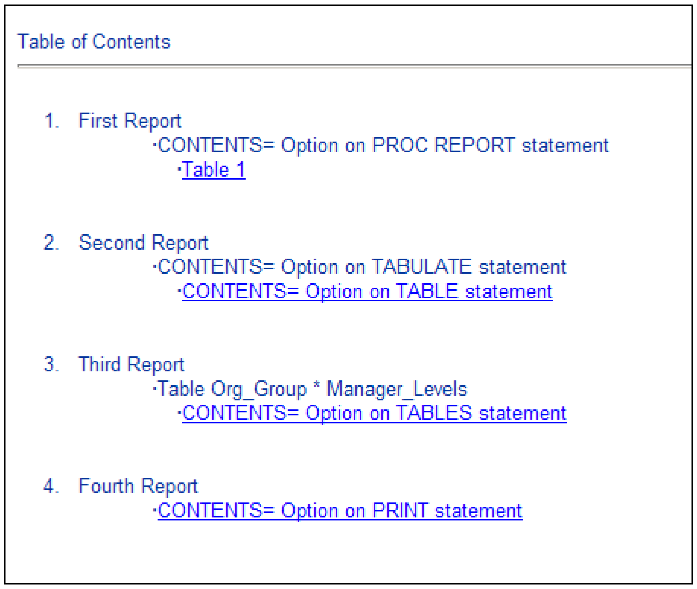


Table of Contents

```
1.  First Report
         ·CONTENTS= Option on PROC REPORT statement
            ·Table 1

2.  Second Report
         ·CONTENTS= Option on TABULATE statement
            ·CONTENTS= Option on TABLE statement

3.  Third Report
         ·Table Org_Group * Manager_Levels
            ·CONTENTS= Option on TABLES statement

4.  Fourth Report
         ·CONTENTS= Option on PRINT statement
```

Display 15. HTML Table of Contents Web Page

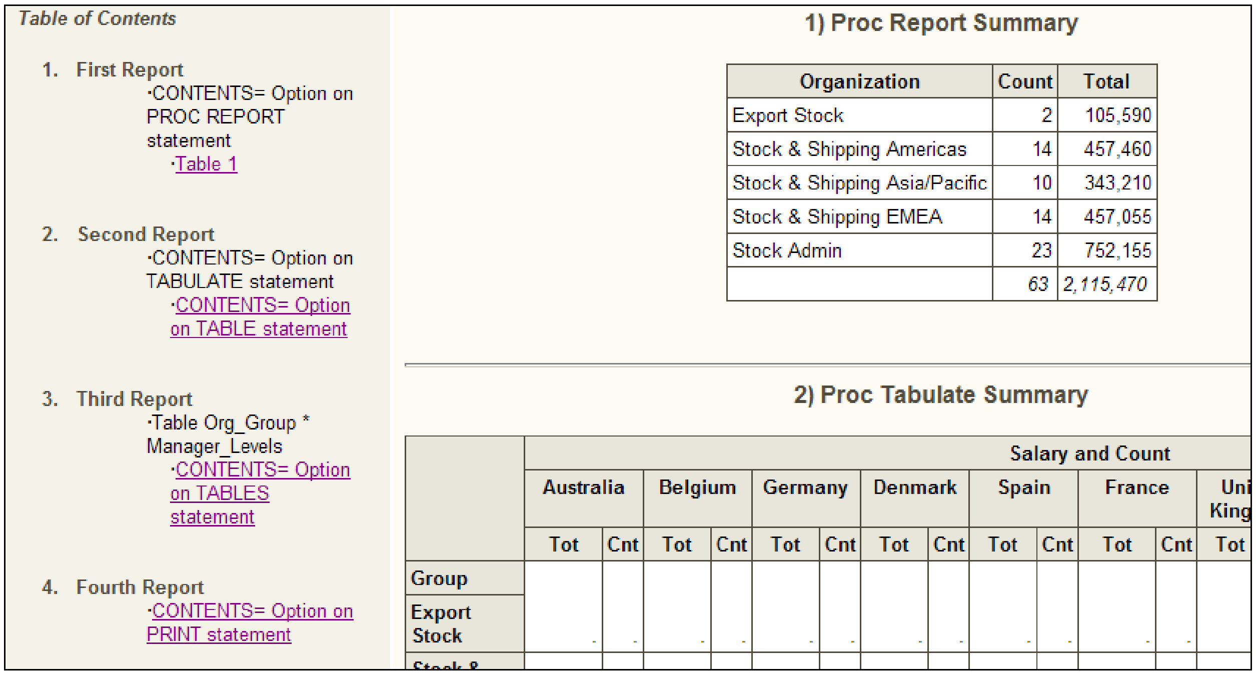


Table of Contents

1. **First Report**
   ·CONTENTS= Option on PROC REPORT statement
   ·Table 1
2. **Second Report**
   ·CONTENTS= Option on TABULATE statement
   ·CONTENTS= Option on TABLE statement
3. **Third Report**
   ·Table Org_Group * Manager_Levels
   ·CONTENTS= Option on TABLES statement
4. **Fourth Report**
   ·CONTENTS= Option on PRINT statement

1) Proc Report Summary

Organization	Count	Total
Export Stock	2	105,590
Stock & Shipping Americas	14	457,460
Stock & Shipping Asia/Pacific	10	343,210
Stock & Shipping EMEA	14	457,055
Stock Admin	23	752,155
	63	*2,115,470*

2) Proc Tabulate Summary

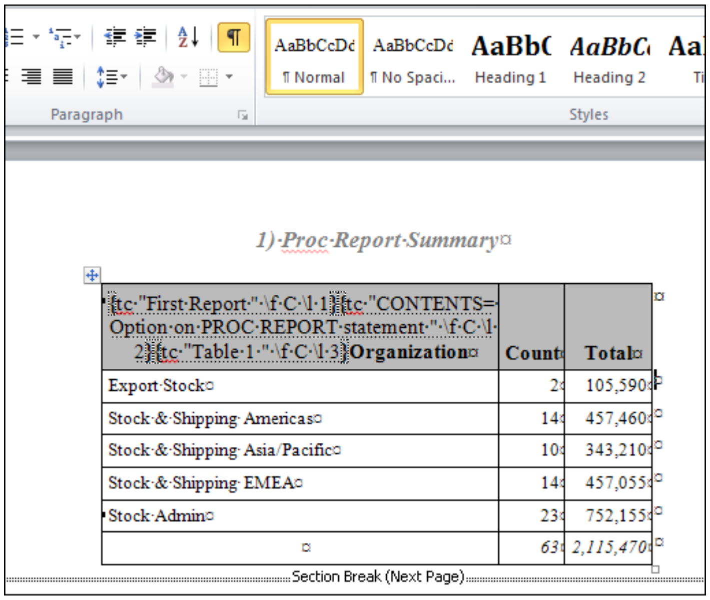
	Salary and Count												
	Australia		Belgium		Germany		Denmark		Spain		France		Uni King
	Tot	Cnt	Tot	Cnt	Tot	Cnt	Tot	Cnt	Tot	Cnt	Tot	Cnt	Tot
Group													
Export Stock	.	.	.	.	.	.	.	.	.	.	.	.	
Stock &													

Display 16. Table of Contents Shown in HTML Frame Structure

The TOC for PDF output would look essentially like the RTF file, and the invocation for PDF and RTF destinations is essentially the same—the CONTENTS=YES option. The HTML destination, on the other hand, requires that you provide a filename for the TOC file. Then, you have to decide whether you want to show the TOC in an HTML frameset structure. If you only want a separate TOC file to incorporate into a Web site, then specify only the CONTENTS= option together with the BODY= or FILE= option. If you want an HTML frameset structure, then you have to specify the BODY= or FILE= option together with both the CONTENTS= and FRAME= options. The ODS statements that created Displays 14, 15, and 16 are shown below.

```
ods rtf file='toc.rtf' contents=yes toc_data;
ods pdf file='toc.pdf' contents=yes;
ods html(id=one) file='body1.html' contents='toc1.html' style=sasweb;
ods html(id=two) file='body2.html' contents='toc2.html'
                 frame='frame2.html' style=analysis;

ods proclabel='First Report';
  title '1) Proc Report Summary';
  proc report data=employees2 nowd split='#'
       contents="CONTENTS= Option on PROC REPORT statement";
  run;

ods proclabel = 'Second Report';
  title '2) Proc Tabulate Summary';
  proc tabulate data=employees2 f=comma8.
       contents="CONTENTS= Option on TABULATE statement";
  run;

ods proclabel = 'Third Report';
  title '3) Summary Report';
  proc freq data=employees2;
  run;

ods proclabel = 'Fourth Report';
  title '4) Detail Report';
  proc print data=employees2 noobs split='_'
  run;

ods _all_ close;
```

The main difference between the PDF and RTF invocation is that the RTF file needs to have the appropriate field codes created for the major contents sections. The option that turns on the insertion of field codes for RTF is the TOC_DATA option. With this option turned on, you can actually see the field codes when the RTF file is opened in Microsoft Word if you have the Show/Hide button selected, as shown in Display 17.



1)·Proc·Report·Summary¤

{tc·"First·Report"·\f·C·\l·1}{tc·"CONTENTS=·Option·on·PROC·REPORT·statement"·\f·C·\l·2}{tc·"Table·1·"·\f·C·\l·3}**Organization**¤	**Count**¤	**Total**¤
Export·Stock¤	2¤	105,590¤
Stock·&·Shipping·Americas¤	14¤	457,460¤
Stock·&·Shipping·Asia/Pacific¤	10¤	343,210¤
Stock·&·Shipping·EMEA¤	14¤	457,055¤
Stock·Admin¤	23¤	752,155¤
¤	*63*¤	*2,115,470*¤

Section Break (Next Page)

Display 17. TOC Field Codes Shown in Microsoft Word

The PDF options NOTOC or BOOKMARKGEN=NO (or NOBOOKMARKGEN) prevent a TOC from being created even if you specified CONTENTS=YES. In fact, if you try to use BOOKMARKGEN=NO with the CONTENTS=YES option, the SAS log shows the following message:

```
WARNING: CONTENTS=ON and NOBOOKMARKGEN are incompatible options
```

The bottom line on this group of greatest hits is that they give you several methods to enhance your ODS result files, including whether additional output such as contents information or a separate TOC page is created.

THE WRITING ON THE WALL: #4 USING ODS ESCAPECHAR

You can enhance your ODS output by using ODS ESCAPECHAR functions. The ODS ESCAPECHAR functions were first introduced in SAS 8.2. The original syntax is described in my paper "Funny Stuff in My Code" (http://www2.sas.com/proceedings/forum2007/099-2007.pdf).

Although ESCAPECHAR syntax changed in SAS 9, the page numbering capability has remained the same. Several popular functions generate "Page X of Y" page numbers for RTF and PDF output by using an ODS escape character with the {THISPAGE} and {LASTPAGE} ESCAPECHAR functions. If you do not know what I mean by page numbering capability, you might know it by a different name: "Page X of Y" page numbering. "Page X of Y" describes a page numbering method in which the page numbers can be controlled and customized in the following manner:

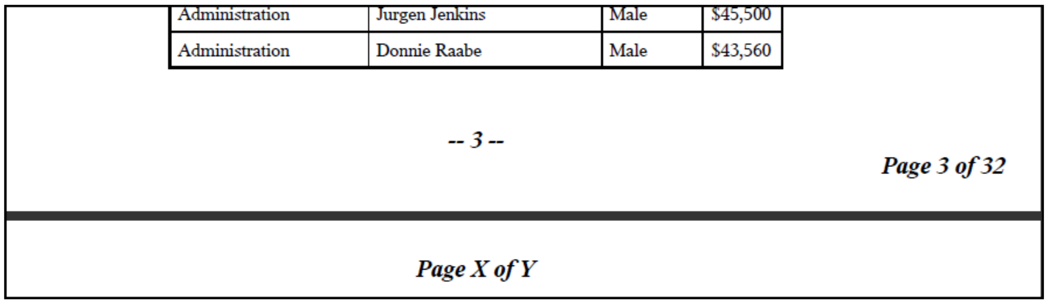
- Page numbers can appear in the TITLE or FOOTNOTE statement (top of page or bottom of page)
- Page numbers can be left-justified, right-justified, or centered
- Page numbers can be just the current number, or can be in the form Page 1 of 10, Page 2 of 10, and so on.

The ODS ESCAPECHAR= statement declares an escape character that is used to introduce special control sequences into RTF, PDF, and HTML files. Since HTML is not a paginated destination, the specific page numbering controls do not have any impact on HTML-based destinations. However, for RTF and PDF files, the following code provides you a way to control page numbering. The PDF page numbering that ODS ESCAPECHAR implements works well for output that does not contain graphic images or contains images but does not use the {LASTPAGE} function. For more information, refer to the Technical Support note 34573 entitled "Use of LASTPAGE inline function causes images to disappear." The example below works as it does because the code does not produce or use any graphic images in the PDF results.

```
options nodate nonumber center missing=' ' pageno=1;
ods escapechar='#';

ods pdf file='pg_xofy.pdf';
ods rtf file='pg_xofy.rtf';

  footnote1 '-- #{thispage} --';
  footnote2 j=r 'Page #{thispage} of #{lastpage}';

  proc report data=employees nowd;
    title 'Page X of Y';
    more code
  run;

ods _all_ close;
```

Notice that the ODS ESCAPECHAR statement declares the escape character as the number sign(#). This means that every time the # sign is encountered, whatever comes after this sign is treated as a special control function, or string, which affects the output. Without describing all of the possible ESCAPECHAR control functions, this example shows how to put a single page number in FOOTNOTE1 and a “Page X of Y” set of page numbers in FOOTNOTE2. The text inserted by FOOTNOTE1 is centered (the default). The text inserted by FOOTNOTE2 is right-justified because of the justification option used before the text string.

The NODATE and NONUMBER options turn off the regular placement of the date and page number on the first title line. The output at the bottom of page 3 and the top of page 4 in the PDF file is shown in Display 18.



Administration	Jurgen Jenkins	Male	$45,500
Administration	Donnie Raabe	Male	$43,560

-- 3 --

Page 3 of 32

Page X of Y

Display 18. Page Numbers in PDF Output

Because the RTF output is essentially the same as the PDF output, it is not shown here. There is some overhead associated with using the {LASTPAGE} ESCAPECHAR function in PDF files, because the PDF file number of total pages must be known before the actual number can be inserted into result file. The two separate functions, one for the current page and one for the number of total pages, work in both RTF and PDF destinations.

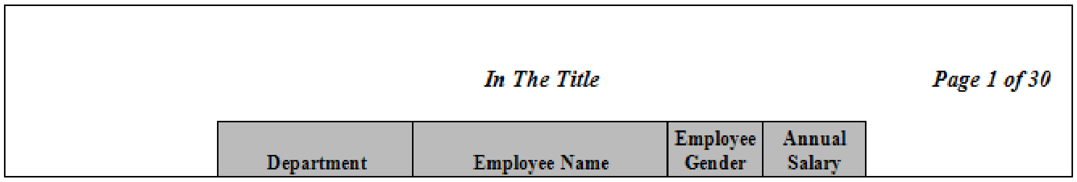
For the RTF destination, there is a separate ESCAPECHAR function, the {PAGEOF} function, that inserts RTF-compliant field codes for page numbers into the result file. For example, when you use the {PAGEOF} ESCAPECHAR function for RTF output, as shown in this TITLE statement

```
title 'In The Title'  j=r  'Page #{pageof}';
```

then the following RTF control stringst are inserted into the RTF result file:

```
{Page {\field{\*\fldinst { PAGE }}}{ of }{\field{\*\fldinst { NUMPAGES }}}\cell}
```

RTF field codes would not be recognized with the PDF destination, so the string would be passed unchanged to the PDF (or HTML) destinations into the page number strings. The RTF output from the TITLE statement is shown in Display 19.



In The Title *Page 1 of 30*

Department	Employee Name	Employee Gender	Annual Salary

Display 19. Page Numbers in RTF Output

As you can see from this small preview, ODS ESCAPECHAR bears more study. This functionality is especially useful if you need to insert Unicode, superscript, or subscript characters into your output or if you want to perform in-line formatting by changing the style for a text string or character variable.

YOU DON'T KNOW WHAT YOU'VE GOT: #5 SENDING ODS RESULTS TO EXCEL

One way to export SAS data sets to Excel (or other file formats) is to use the EXPORT procedure or the SAS Excel LIBNAME engine. But what if you want the output from the GLM or TABULATE procedures in Excel without creating an output data set? Using ODS, you can create three types of files that can be opened and rendered with Excel. Several ODS destinations create ASCII text files that can be opened and rendered with Excel:

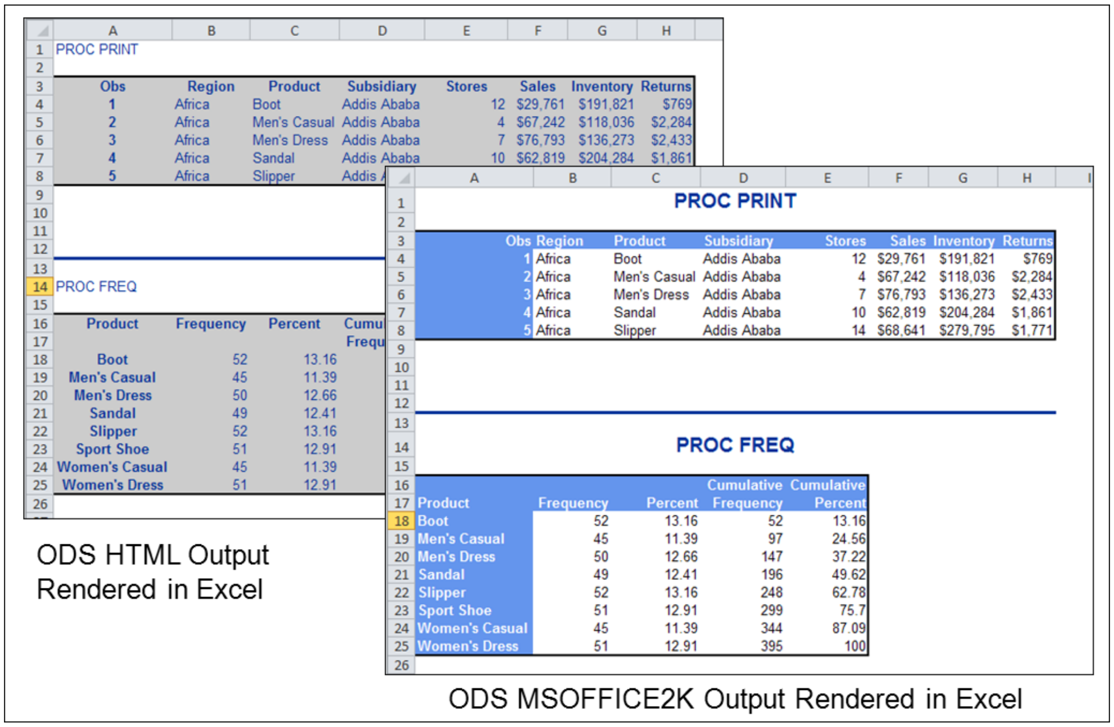
- Plain text files: The ODS CSV and ODS CSVALL destinations create ASCII text files with comma-delimited values. (In SAS 9, these destinations have a suboption that allows you to choose a different delimiter.)
- HTML files: The ODS HTML, ODS MSOFFICE2K, ODS TAGSETS.MSOFFICE2K_X, and ODS TAGSETS.TABLEEDITOR destinations create HTML files that can be opened and rendered with Excel (starting with Microsoft Office 97).
- XML Spreadsheet Markup Language Files: The ODS TAGSETS.EXCELXP destination creates files that can be opened and rendered with Excel (starting with Microsoft Office 2002/2003).

None of these destinations create true, binary Excel files. Only the EXPORT procedure or the SAS Excel LIBNAME engine can do that. When you use ODS, just giving the file an extension of .XLS does not alter what is inside the file. In this case, an HTML or an XML file with the file extension of .XLS tricks the Windows registry into launching Excel when the file is double-clicked. Otherwise, the HTML file would open in a browser, and the XML file would probably open in a browser or XML editor.

So, now that I've told you, you can't say that you didn't know what type of files ODS creates for Excel. The fact is that each of the ODS destinations above creates output for Excel. Each destination has pros and cons, and you will have to experiment to see which method suits your needs. For example, if you need to create multi-sheet workbooks from your procedure output, then that is an argument for using TAGSETS.MSOFFICE2K_X or TAGSETS.TABLEEDITOR (both of which create HTML result files) or TAGSETS.EXCELXP (which creates a Spreadsheet Markup Language result file). If you need procedure output with no formatting, then that is an argument for using the ODS CSV or ODS CSVALL destinations.

The other thing to realize is that Excel has its own way of doing things. For example, you might have a perfectly good product number with leading zeros displaying in SAS output. However, when you send a report with the product number to Excel, the default format for a numeric column is the General format, which does not respect the SAS leading zeros. Luckily, if you use the REPORT, PRINT, or TABULATE procedures, you have direct ways, using either the HTMLSTYLE or TAGATTR style attributes, to send a Microsoft format file from SAS to Excel. (These methods are the topic of my SAS Global Forum 2011 paper "Don't Gamble with Your Output: How to Use Microsoft Formats with ODS".)

Here's another example of how Excel has its own way of doing things. The ODS HTML destination creates HTML 4.0-compliant HTML tags, which Microsoft Excel really doesn't like. For example, compare the ODS HTML results (opened in Excel) with the ODS MSOFFICE2K results opened in Excel, as shown in Display 20. Both of these steps used the SASWEB style. (Excel also "likes output from the ODS HTML3 destination, which creates HTML 3.2-compliant tags.)


ODS HTML Output Rendered in Excel
ODS MSOFFICE2K Output Rendered in Excel

PROC PRINT

Obs	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
1	Africa	Boot	Addis Ababa	12	$29,761	$191,821	$769
2	Africa	Men's Casual	Addis Ababa	4	$67,242	$118,036	$2,284
3	Africa	Men's Dress	Addis Ababa	7	$76,793	$136,273	$2,433
4	Africa	Sandal	Addis Ababa	10	$62,819	$204,284	$1,861
5	Africa	Slipper	Addis A				

PROC FREQ

Product	Frequency	Percent	Cumu Frequ
Boot	52	13.16	
Men's Casual	45	11.39	
Men's Dress	50	12.66	
Sandal	49	12.41	
Slipper	52	13.16	
Sport Shoe	51	12.91	
Women's Casual	45	11.39	
Women's Dress	51	12.91	

PROC PRINT

Obs	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
1	Africa	Boot	Addis Ababa	12	$29,761	$191,821	$769
2	Africa	Men's Casual	Addis Ababa	4	$67,242	$118,036	$2,284
3	Africa	Men's Dress	Addis Ababa	7	$76,793	$136,273	$2,433
4	Africa	Sandal	Addis Ababa	10	$62,819	$204,284	$1,861
5	Africa	Slipper	Addis Ababa	14	$68,641	$279,795	$1,771

PROC FREQ

			Cumulative	Cumulative
Product	Frequency	Percent	Frequency	Percent
Boot	52	13.16	52	13.16
Men's Casual	45	11.39	97	24.56
Men's Dress	50	12.66	147	37.22
Sandal	49	12.41	196	49.62
Slipper	52	13.16	248	62.78
Sport Shoe	51	12.91	299	75.7
Women's Casual	45	11.39	344	87.09
Women's Dress	51	12.91	395	100

Display 20. ODS HTML Results Compared to ODS MSOFFICE2K Results

As you can see, Excel interpreted the style used in the ODS HTML tags differently than the style specification for the Microsoft HTML tags. This means you get the best results (where the style is concerned) with destinations that are Microsoft "friendly". There are a lot of user group papers describing the use of ODS to create output files that can be opened and rendered with Excel. Once you understand what you really get from SAS and ODS, you can make Excel and SAS output work well together.

ANY WAY YOU WANT IT: #6 REARRANGING OUTPUT OBJECTS WITH ODS DOCUMENT

There is too much ODS DOCUMENT (and DOCUMENT procedure) code involved in this hit to include in this paper. If you download the zip file of programs that goes with this paper, you can get all of the code. ODS DOCUMENT is not, technically, an ODS report destination. An ODS DOCUMENT store is an item store like the SAS registry and the ODS template stores.

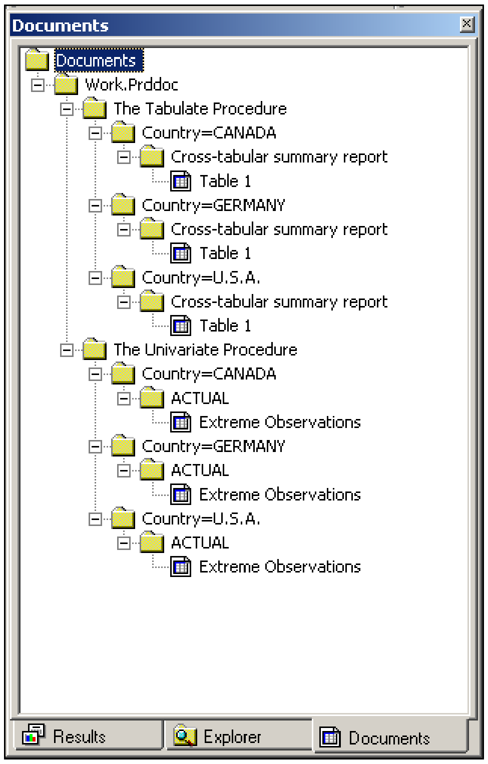
The ODS DOCUMENT destination (and the DOCUMENT procedure in batch syntax mode) enable you to create document stores, which act as a conceptual freezer where you can store your ODS output objects between the time a procedure finishes with the output object and the time the output object is sent to an ODS report destination. The ODS DOCUMENT destination has a document store that holds output objects in their original hierarchy and creation structures, but you can rearrange the hierarchy and structure of these ouput objects. In this document store, ODS documents are saved in a proprietary format and can be viewed only with SAS software. The ODS DOCUMENT destination holds the output objects in such a way that they are able to be replayed or rerun to one of the report destinations. The ODS DOCUMENT destination is a SAS proprietary destination. To view or modify what is in the document store, you have to use either the ODS DOCUMENT window or the DOCUMENT procedure.

Consider this scenario: You have a job that runs for six hours and performs some complicated analysis. The only copy of the report is in HTML form on the company Web server. You can only run the job overnight. But, your boss comes in at 9:30 a.m. and says she needs a copy of the report in a PDF version, with page numbers, in landscape mode, for a noon meeting. You don't have six hours to rerun the analysis to the ODS PDF destination. What do you do?

If you froze (saved) copies of the analysis output objects in an ODS DOCUMENT store (because the job takes so long to run), then you have another option. You can submit a DOCUMENT procedure replay program to replay the ODS DOCUMENT output objects to the ODS PDF destination without spending the six hours to rerun the original procedure. Pretty neat trick.

Here's a different scenario: You have a PROC TABULATE step with BY-group processing, followed by a PROC UNIVARIATE step with BY-group processing. In the default folder structure created for the job, all of the BY groups for the PROC TABULATE step appear before the BY groups for the PROC UNIVARIATE step. Your boss is tired of flipping back and forth from one section to another to compare the TABULATE procedure report with the EXTREMEOBS portion of the UNIVARIATE procedure report. She requests that you organize the output report so that you have the TABULATE procedure report immediately followed by only the EXTREMEOBS report for the same BY group.

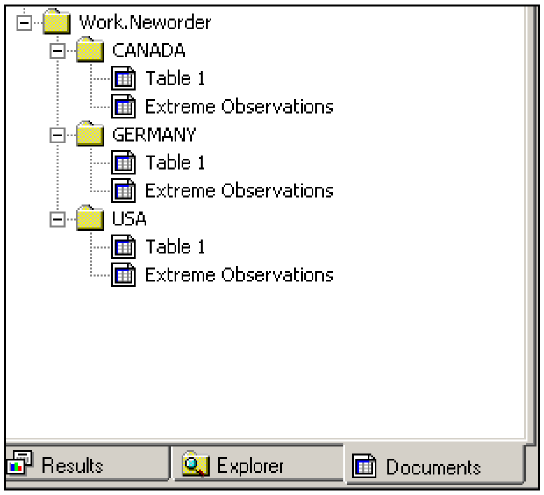
Again, if you save copies of your original output objects from the TABULATE and UNIVARIATE procedures in an ODS DOCUMENT store, the document store would look as shown in Display 21.



```
Documents
Documents
  Work.Prddoc
    The Tabulate Procedure
      Country=CANADA
        Cross-tabular summary report
          Table 1
      Country=GERMANY
        Cross-tabular summary report
          Table 1
      Country=U.S.A.
        Cross-tabular summary report
          Table 1
    The Univariate Procedure
      Country=CANADA
        ACTUAL
          Extreme Observations
      Country=GERMANY
        ACTUAL
          Extreme Observations
      Country=U.S.A.
        ACTUAL
          Extreme Observations
```

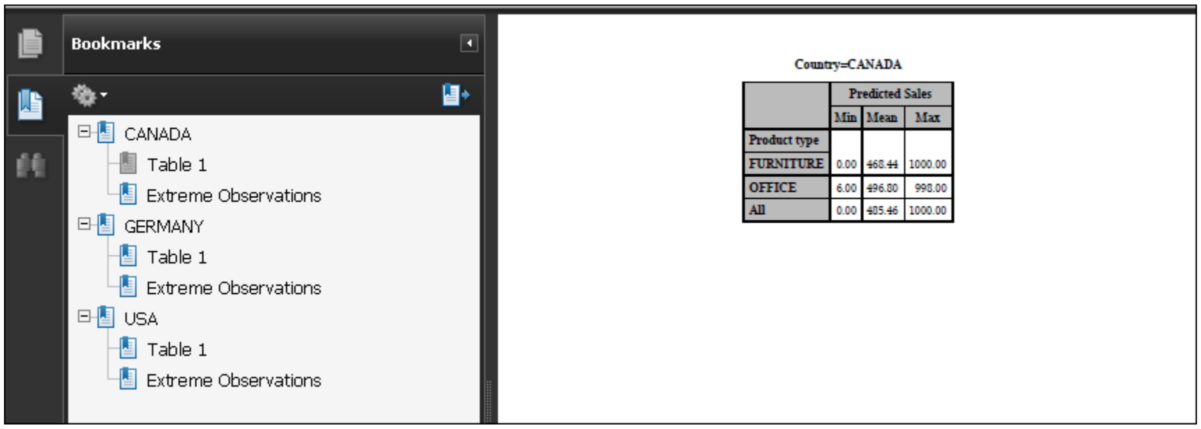
Display 21. ODS DOCUMENTS Window Showing Original Document Structure

Given that you have saved the original output objects in the original structure, you can now rearrange the output objects in a hierarchical structure that is different from the original procedure-centric structure. Then, you can rerun the new ODS document to the destination of your choice. Display 22 shows the rearranged, new ODS DOCUMENT structure, and Display 23 shows the PDF bookmarks that reflect the new structure (showing only the TABULATE procedure output and the ExtremeObs output object from the UNIVARIATE procedure).



```
Work.Neworder
  CANADA
    Table 1
    Extreme Observations
  GERMANY
    Table 1
    Extreme Observations
  USA
    Table 1
    Extreme Observations
```

Display 22. ODS DOCUMENTS Window Showing Restructured Output Objects



Country=CANADA

	Predicted Sales		
	Min	Mean	Max
Product type			
FURNITURE	0.00	468.44	1000.00
OFFICE	6.00	496.80	998.00
All	0.00	485.46	1000.00

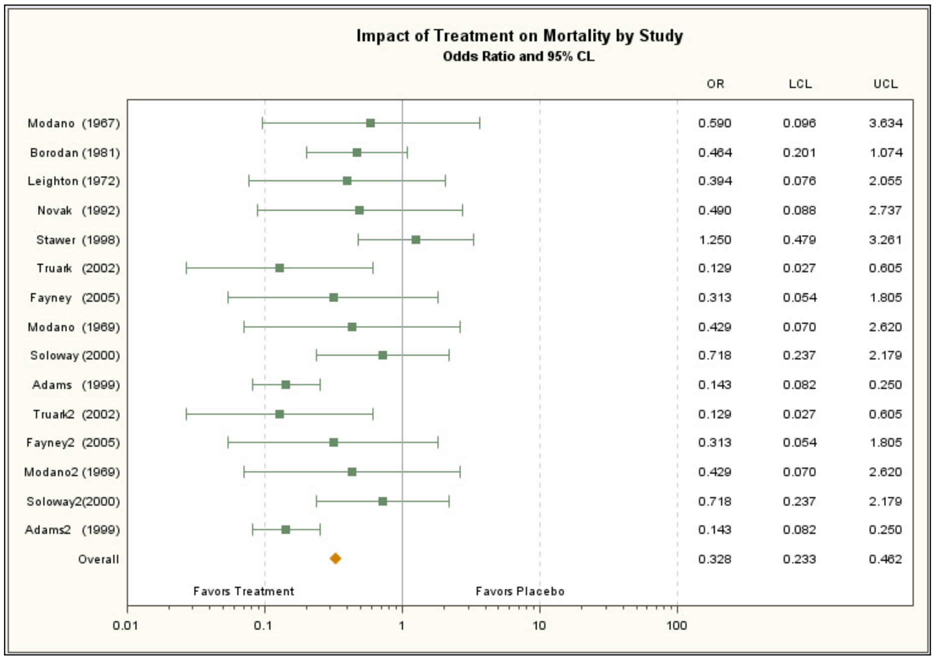
Display 23. ODS PDF Bookmarks Window For Replayed Document

You might not have an immediate use for the ODS DOCUMENT destination (and the DOCUMENT procedure) based on these few scenarios. However, I'm convinced that the ODS DOCUMENT destination is one of those sleeper hits that will prove to be very useful sometime in your future as a SAS programmer.

PICTURES AT AN EXHIBITION: #7 CREATING IMAGES WITH ODS

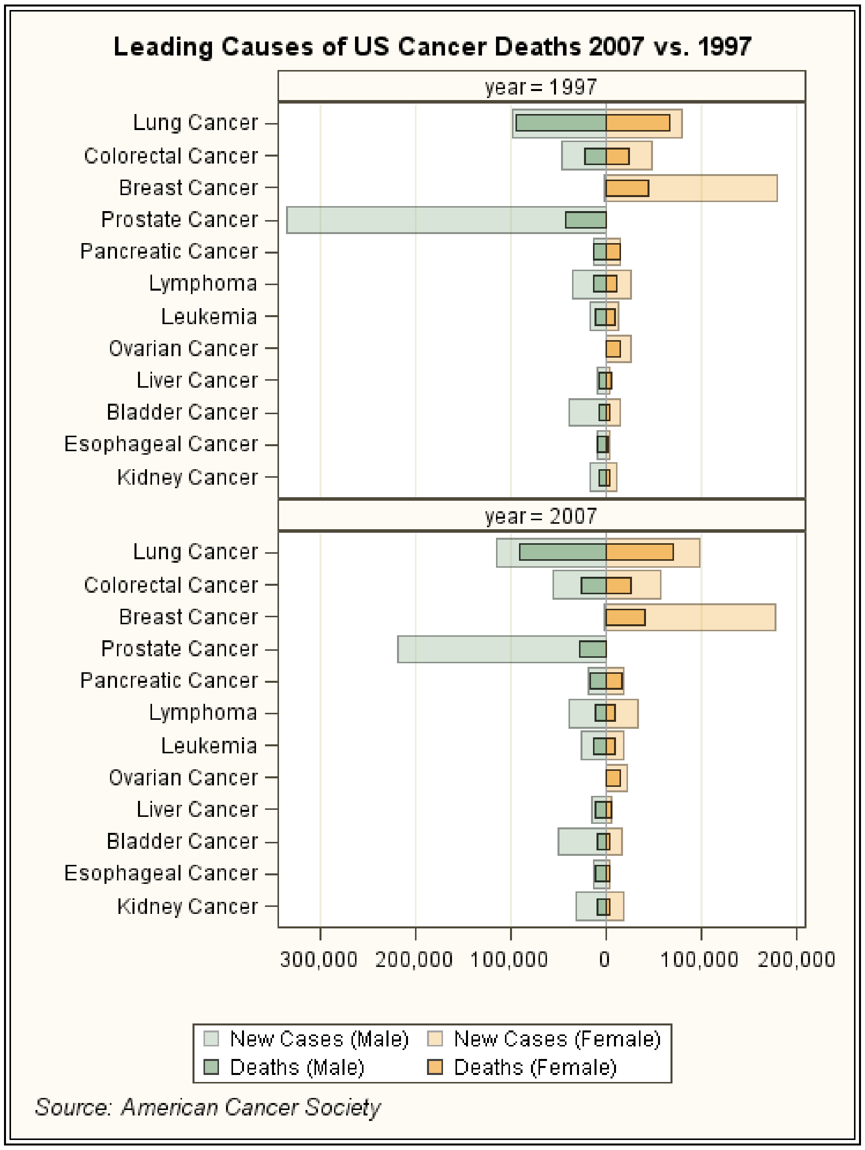
Supposedly, a picture is worth a thousand words. I probably can't say enough good things about the next two pictures even if I had 10,000 words. And, although the code that produced these pictures is much less than 1000 lines long, I'm not going to show all the code for this hit, partly because the zip file of programs has all the code that produced these pictures, and also because the pictures themselves prove my point that ODS GRAPHICS can help you rock out your reporting.

Display 24 shows an example of a forest plot that Sanjay Matange is going to cover in his SAS Global Forum 2011 paper entitled "Tips and Tricks for SG Procedures and GTL for Clinical Graphs". Display 25 shows a paneled butterfly plot, modified from the example that Susan Schwartz demonstrated in her paper entitled "Butterflies, Heat Maps, and More: Explore the New Power of SAS/GRAPH".



	OR	LCL	UCL
Modano (1967)	0.590	0.096	3.634
Borodan (1981)	0.464	0.201	1.074
Leighton (1972)	0.394	0.076	2.055
Novak (1992)	0.490	0.088	2.737
Stawer (1998)	1.250	0.479	3.261
Truark (2002)	0.129	0.027	0.605
Fayney (2005)	0.313	0.054	1.805
Modano (1969)	0.429	0.070	2.620
Soloway (2000)	0.718	0.237	2.179
Adams (1999)	0.143	0.082	0.250
Truark2 (2002)	0.129	0.027	0.605
Fayney2 (2005)	0.313	0.054	1.805
Modano2 (1969)	0.429	0.070	2.620
Soloway2(2000)	0.718	0.237	2.179
Adams2 (1999)	0.143	0.082	0.250
Overall	0.328	0.233	0.462

Display 24. Forest Plot With Overlaid Scatter Plots Using the SGPLOT Procedure



Display 25. Paneled Butterfly Plot Using the SGPANEL Procedure

To entice you to download the zip file of programs, here's the bottom line on the level of coding needed to produce these two images. Display 24 was created by a PROC SGPLOT step with four SCATTER statements, two REFLINE statements, two INSET statements, and a statement for each of the main axes: XAXIS, X2AXIS, and YAXIS. Display 24 used a custom style template to standardize all the fonts and colors as desired. Display 25 was created by a PROC SGPANEL step with four HBAR statements and a PANELBY statement. Display 25 used the default settings for the ANALYSIS style. And of course, each program had the usual TITLE and FORMAT statements.

No Annotate statements, no GOPTIONS statements, and no AXIS, SYMBOL, or PATTERN statements were used to create two images! Now, don't get me wrong. I am a control person. I love Annotate, I love the AXIS and SYMBOL statements, and I love traditional, device-based SAS/GRAPH. However, the new ODS GRAPHICS capabilities are too good to ignore. When I say ODS GRAPHICS, I mean the new SG procedures (SGPLOT, SGPANEL, SGSCATTER), the Graph Template Language (GTL) with the SGRENDER procedure, and the new ODS Graphics Designer.

And, as if the pictures and the new capabilities weren't reason enough to check out ODS GRAPHICS, there's the way that style template information now affects all graphics output – both traditional, device-based output and the new template-based output. If I start gushing about all the wonderful things about fonts and font rendering and the 16.7 million colors…well, I'll never get to end of the greatest hits.

Don't let this hit fall on deaf ears.

THE LONG AND WINDING ROAD: #8 UNDERSTANDING STYLE AND ODS RESULTS

There are many different ways to affect the style of ODS output:

- STYLE= option in the ODS statement
- STYLESHEET= option in the ODS HTML statement
- CSSSTYLE= option in the ODS HTML, ODS RTF, and ODS PDF statements (SAS 9.2)
- STYLE= statement-level options for the PRINT, REPORT, and TABULATE procedures
- ODS ESCAPECHAR statement and the {STYLE} function (SAS 9.2)
- SAS style templates.

Of course, first and foremost, you have to understand that for tabular output, the style information that you want to use must be supported by the ODS destination to which you route your procedure output. For example, the ODS LISTING destination does not use style information for tabular output. However, for graphical output, the ODS LISTING destination does support style information.

USING THE STYLE= OPTION

If you don't know by now, the simple and quickest way to affect style is to use the STYLE= option on your ODS invocation statement:

```
ods <dest> file='filename.ext' style=<style>;
```

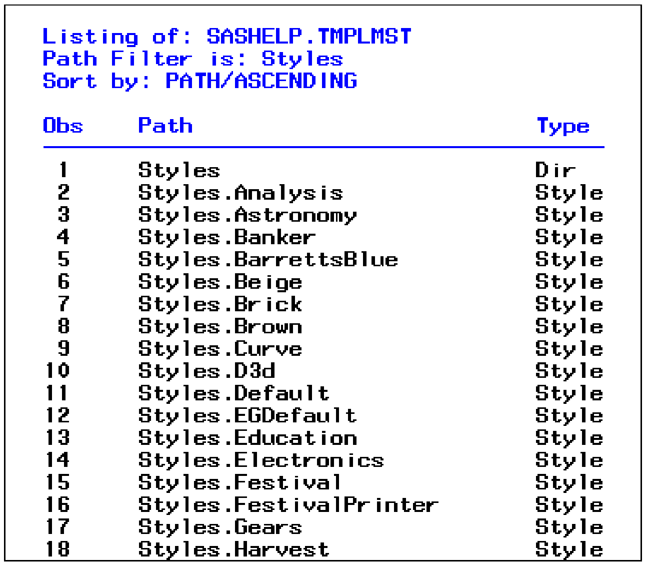
For graphical output, in addition to the above method, you can affect graphical output from SAS/GRAPH procedures for the LISTING destination by specifying:

```
ods listing style=<style>;
```

The list of available style templates is in the SASHELP.TMPLMST item store. In SAS Enterprise Guide, the list of style sheets is shown by the Style Wizard. In batch mode, you can display the list of available style templates by submitting this code:

```
proc template;
  list styles / store=sashelp.tmplmst;
run;
```

Display 26 shows a partial report of the available style templates from the PROC TEMPLATE step as shown in the Output window.



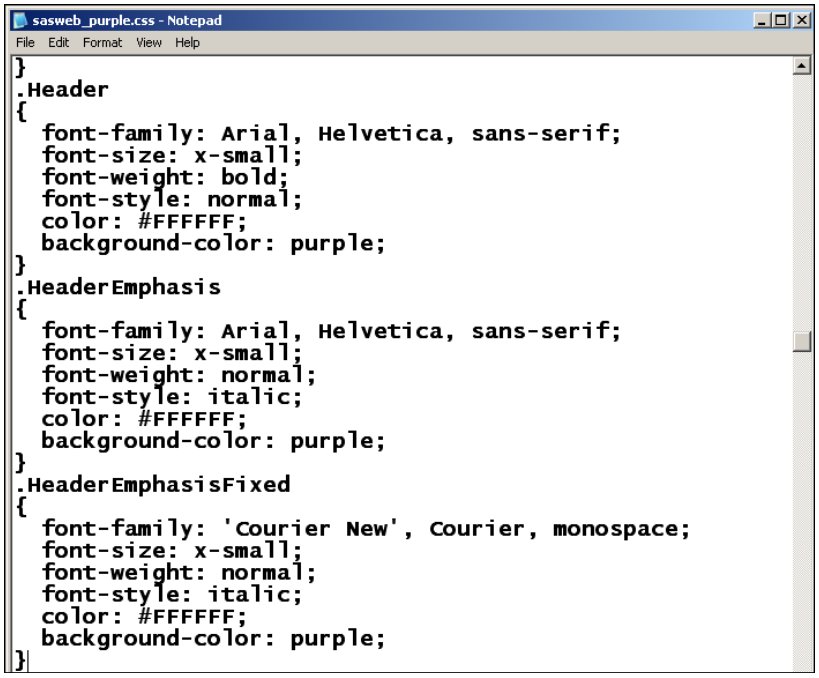
```
Listing of: SASHELP.TMPLMST
Path Filter is: Styles
Sort by: PATH/ASCENDING

Obs    Path                          Type
-----------------------------------------
  1    Styles                        Dir
  2    Styles.Analysis               Style
  3    Styles.Astronomy              Style
  4    Styles.Banker                 Style
  5    Styles.BarrettsBlue           Style
  6    Styles.Beige                  Style
  7    Styles.Brick                  Style
  8    Styles.Brown                  Style
  9    Styles.Curve                  Style
 10    Styles.D3d                    Style
 11    Styles.Default                Style
 12    Styles.EGDefault              Style
 13    Styles.Education              Style
 14    Styles.Electronics            Style
 15    Styles.Festival               Style
 16    Styles.FestivalPrinter        Style
 17    Styles.Gears                  Style
 18    Styles.Harvest                Style
```

Display 26. Partial List of Style Definitions Available For Use with ODS

USING CASCADING STYLESHEETS WITH ODS

The use of the new CSSSTYLE= option with ODS was the topic of my paper entitled: “CSSSTYLE: Stylish Output with ODS and SAS 9.2” (available at http://support.sas.com/resources/papers/proceedings09/014-2009.pdf). In addition, that paper also discussed the use of the STYLESHEET= option, which has always been available for ODS HTML and other HTML-based destinations. Cascading style sheets and style sheet technology is not owned by SAS. It is an industry standard for how web pages should use style specifications. For example, let’s say that you work for “The Purple Palace,” and the corporate style is that all posted tables and reports use purple for the table headings.

The SASWEB style might be perfect for you to start with, since it only needs a few style properties changed to purple in order for your reports to comply with the standards. You do not want to make a style template, since your web reports need to go on the corporate Web site. But, you also use RTF and PDF result files for some of your reports. So you want to use the new CSSSTYLE= option to create a cascading style sheet that follows the corporate guidelines and affects RTF and PDF results in addition to the HTML results. With the STYLESHEET= option, you can make a copy of the SASWEB style template in CSS format. Then, you can edit the CSS version of SASWEB to change all instances of SASWEB blue (#6495ED) to the Purple Palace shade of purple. A changed version of such a CSS file, called SASWEB_PURPLE.CSS, is shown in Display 27.

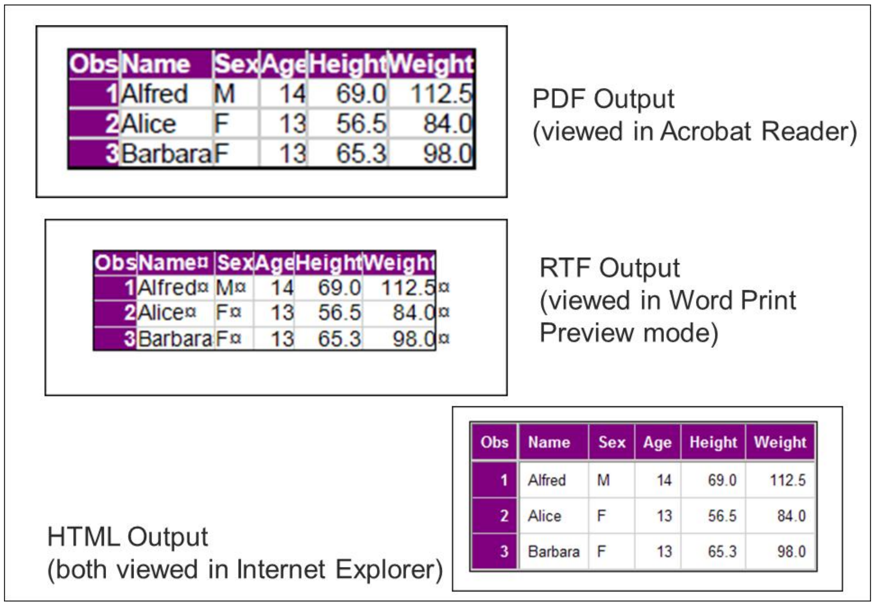


```
}
.Header
{
  font-family: Arial, Helvetica, sans-serif;
  font-size: x-small;
  font-weight: bold;
  font-style: normal;
  color: #FFFFFF;
  background-color: purple;
}
.HeaderEmphasis
{
  font-family: Arial, Helvetica, sans-serif;
  font-size: x-small;
  font-weight: normal;
  font-style: italic;
  color: #FFFFFF;
  background-color: purple;
}
.HeaderEmphasisFixed
{
  font-family: 'Courier New', Courier, monospace;
  font-size: x-small;
  font-weight: normal;
  font-style: italic;
  color: #FFFFFF;
  background-color: purple;
}
```

Display 27. Modified Cascading Style Sheet

Then, if you use the changed CSS file, as shown in the code below, you get the results shown in Display 28.

```
ods pdf file='.\use_cssstyle.pdf' cssstyle='.\sasweb_purple.css';
ods rtf file='.\use_cssstyle.rtf' cssstyle='.\sasweb_purple.css';
ods html(id=2) path='.' (url=none) file='use_cssstyle.html'
               cssstyle='sasweb_purple.css';
  proc print data=sashelp.class(obs=3);
  run;
ods _all_ close;
```


PDF Output
(viewed in Acrobat Reader)
RTF Output
(viewed in Word Print Preview mode)
HTML Output
(both viewed in Internet Explorer)

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0

Display 28. Results Files Using SASWEB_PURPLE.CSS STYLE PROPERTIES

Companies use cascading style sheet (CSS) technology because it gives them a way to specify the same look and feel for their corporate Web site. For example, one or two CSS files could control the look and feel for a Web site composed of hundreds of linked web pages. This is not meant to be a comprehensive explanation of how CSS technology works. For that, the best resource is the W3C schools site, http://www.w3schools.com/css/, where you can learn about CSS technology from the folks who invented it. Not every browser supports the most current CSS standard, so you will need to design your CSS for HTML files keeping that fact in mind. ODS and SAS, however, use a limited subset of all of the possible CSS style properties, as described in the ODS documentation. It's even possible that your company already has a CSS expert who can help you get started understanding your corporation's CSS standards.

USING THE STYLE= STATEMENT-LEVEL OPTIONS

The PRINT, REPORT, and TABULATE procedures all support the STYLE= option both in the procedure statement and in the procedure's action statements. This option overrides the style attribute information in SAS style templates. These three procedures—PRINT, REPORT and TABULATE—will handle the bulk of your reporting needs (if you are not using SAS/STAT procedures). Because these procedures provide a way to directly affect the style of your results, including the ability to perform traffic-lighting, it is worthwhile to talk about STYLE= options for these three procedures. Note that table templates (for procedures that use table templates) also support the STYLE= option, but that is a more advanced topic than what I want to discuss in this hit.

The syntax for the STYLE= option is fairly similar for the PRINT and REPORT procedures, but the syntax for the TABULATE procedure is slightly different. In most statements for the PRINT and REPORT procedures, you specify (inside parentheses) the report area or location that you want to change. The general syntax model is:

```
   STATEMENT / style(<area>)={<attr1>=<value1> <attrn>=<valuen>};
```

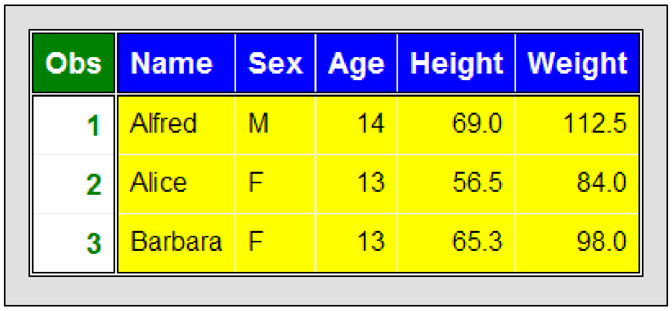
The areas or report locations that you can specify for the REPORT and PRINT procedures varies a bit for the PRINT procedure versus the REPORT procedure. But there are some report areas, such as the HEADER area, that are the same. Some style options can appear on multiple statements. For example, for the REPORT procedure, the STYLE(HEADER) option can appear in the PROC REPORT statement or in the DEFINE statement. For the PRINT procedure, the option for the HEADER area can appear in the PROC PRINT statement or in the VAR statement.

However, for the TABULATE procedure, the area that is affected by the STYLE= option is based on the statement where the option appears, so you do not need to specify an area or report location in parentheses. The general syntax model for the TABULATE procedure is:

```
   STATEMENT / style={<attr1>=<value1> <attrn>=<valuen>};
```

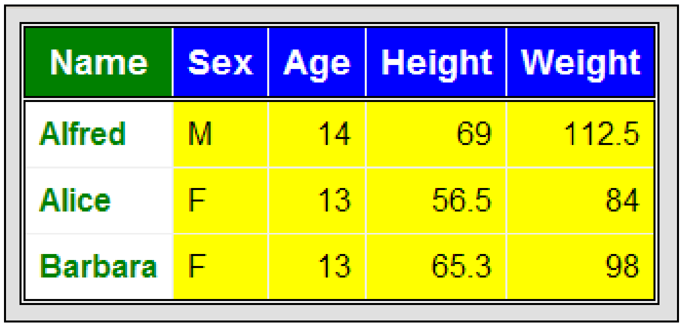
For all three procedures, you can also specify the STYLE= option on the procedure statement itself (without using a slash as the delimiter). Consider the code below and the corresponding output shown in Displays 29, 30, and 31.

```
ods html file='style08a.html';
ods rtf file='style08a.rtf';
ods pdf file='style08a.pdf';

PROC PRINT data=sashelp.class(obs=3)
    style(header)={background=cx0000ff foreground=cxffffff font_face=Arial}
    style(data)={background=yellow}
    style(obsheader)={background=green foreground=white}
    style(obs)={background=white foreground=green};
RUN;

PROC REPORT data=sashelp.class(obs=3) nowd
     style(header)={background=cx0000ff foreground=cxffffff font_face=Arial}
     style(column)={background=yellow};
  define name / 'Name'
    style(header)={background=green foreground=white}
    style(column)={background=white foreground=green font_weight=bold};
RUN;

PROC TABULATE data=sashelp.class
     style={background=yellow};
  class age sex / style={background=cx0000ff foreground=cxffffff font_face=Arial};
  classlev age / style={background=cyan};
  classlev sex / style={background=white foreground=blue};
  table
    sex all,
    age all={label='Count' style={background=white vjust=b}}
         *{style={background=purple foreground=white font_weight=bold}}
     / box={label='Box Area' style={vjust=b background=green
                                     foreground=white}};
  keylabel n=' '
           all='Total';
  keyword all / style=Header{background=pink};
RUN;

ODS _all_ CLOSE;
```



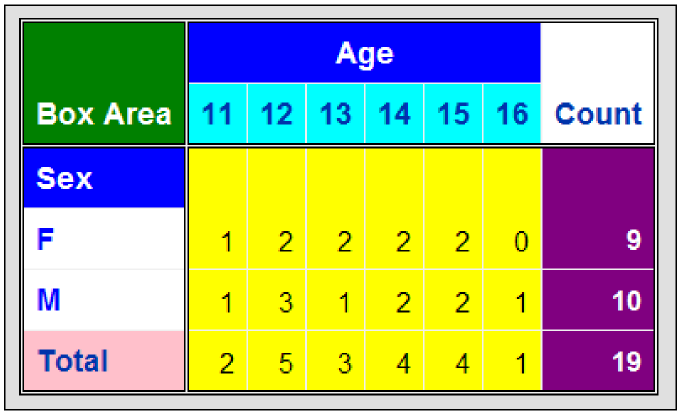
Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0

Display 29. HTML Output for the PRINT Procedure



Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98

Display 30. HTML Output for the REPORT Procedure



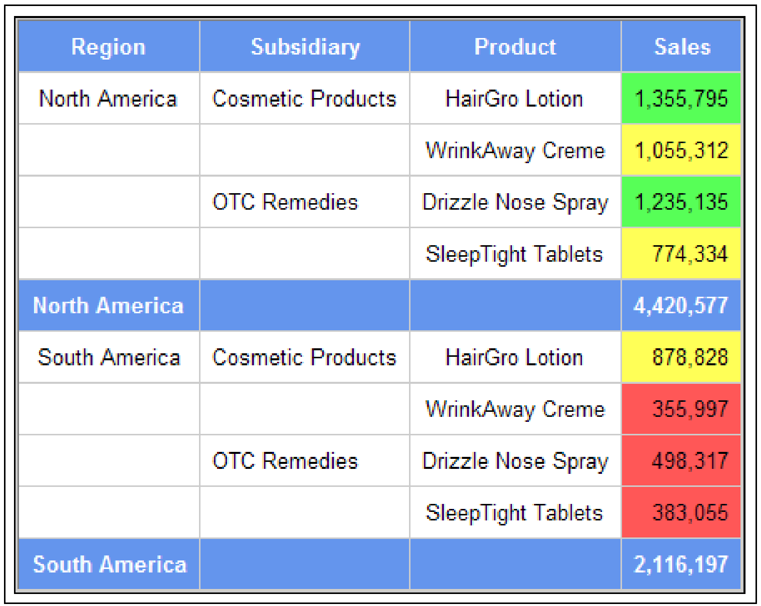
	Age						
Box Area	11	12	13	14	15	16	Count
Sex							
F	1	2	2	2	2	0	9
M	1	3	1	2	2	1	10
Total	2	5	3	4	4	1	19

Display 31. HTML Output for the TABULATE Procedure

Although the colors are somewhat garish, you should be able to trace back from the output to the statement option that changed the output. For example, the PRINT procedure program shows that `STYLE(OBSHEADER)` changed the foreground color of the heading “Obs” to white and the background color of that cell (and only that cell) to green. The `STYLE(OBS)` option changed the foreground color of the observation numbers to green and the background color to white. Similarly, if you look at the REPORT procedure output, you can see that all of the heading cells are white on blue as defined in the `STYLE(HEADER)` option in the PROC REPORT statement. However, the NAME column has white on green for the heading (set by the `STYLE(HEADER)` option in the DEFINE statement) and green on white for the name values (as set by the `STYLE(COLUMN)` option in the DEFINE statement).

The TABULATE procedure output, on the other hand, shows many different changes, and you have to trace the changes back to the TABULATE statement on which the option appeared. For example, the header cells for the variables SEX and AGE have a background color of blue and a foreground color of white because of the STYLE options in the CLASS statement. The values of SEX and of AGE are each styled differently by the two CLASSLEV statements in the PROC TABULATE step.

As exciting as this ability is, even more exciting is the ability to perform traffic-lighting as shown in Display 32 in ourput from the REPORT procedure.



Region	Subsidiary	Product	Sales
North America	Cosmetic Products	HairGro Lotion	1,355,795
		WrinkAway Creme	1,055,312
	OTC Remedies	Drizzle Nose Spray	1,235,135
		SleepTight Tablets	774,334
North America			4,420,577
South America	Cosmetic Products	HairGro Lotion	878,828
		WrinkAway Creme	355,997
	OTC Remedies	Drizzle Nose Spray	498,317
		SleepTight Tablets	383,055
South America			2,116,197

Display 32. HTML Output With Traffic-Lighting from the REPORT Procedure

Notice how each SALES cell uses a different color background. A user-defined format was created to specify the background colors based on the SALES summary values. After the user-defined format is created, it must actually be used in the REPORT procedure code, as shown below.

```
proc format;
   value tlite low -<750000 = 'light red'
               750000 - 1200000 = 'light yellow'
               1200000<- 2000000  = 'light green'
               other = 'cx6495ED';
run;

ods html file='Demo08_tlite.html' style=sasweb;
ods rtf file='Demo08_tlite.rtf';
ods pdf file='Demo08_tlite.pdf';
proc report data=salesdata nowd style(summary)=Header;
  column Region Subsidiary Product Sales;
  define Region/ group center;
  define Subsidiary / group;
  define Product/ group 'Product' center;
  define Sales/ sum 'Sales' f=comma16. style(column)={background=tlite.};
  break after Region / skip summarize dol dul;
run;
ods _all_ close;
```

All three procedures support this method–a user-defined format–of performing traffic-lighting. The REPORT procedure supports a second method, the CALL DEFINE statement, which is described in the REPORT procedure documentation and in many of the user group papers about the REPORT procedure. My PharmaSUG tutorial entitled “Practically Perfect Presentations” (available at http://www.lexjansen.com/pharmasug/2007/hw/hw03.pdf) shows the use of the STYLE= option in a CALL DEFINE statement.

You do not have to sing the blues (or do your traffic-lighting manually) if you learn to sing along with the the STYLE= option hit.

Understanding How RGB Colors Are Specified

Now it’s time for a side trip into the world of color. There are many ways to specify color for ODS style attributes. Not surprisingly, most of the methods that you can use with SAS/GRAPH options are also supported by ODS. This means that you have a wealth of information available about specifying colors for ODS styles. The bottom line, however, is that for the most control over your colors, you might want to learn how the RGB (Red/Green/Blue) hexadecimal specification for colors works. Almost all of the SAS style templates use RGB colors in the template, and it is very easy to understand what the RGB colors mean.

RGB color names for ODS and SAS/GRAPH are of the form CXrrggbb, where

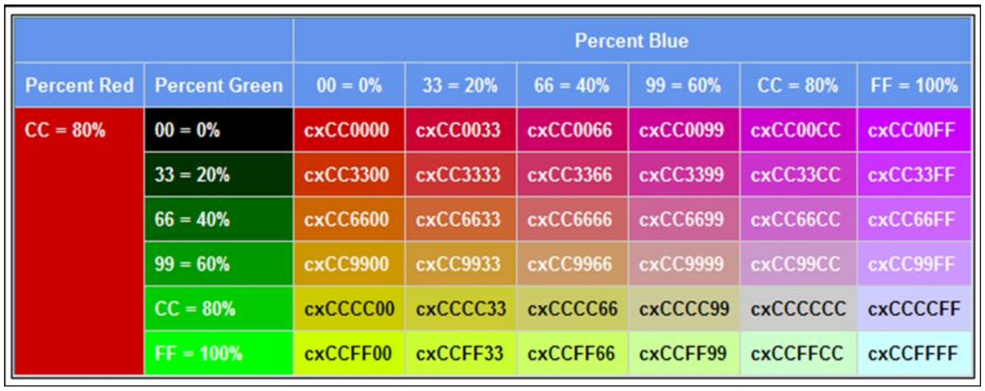
- **CX** indicates an RGB color specification
- **rr** is the red component
- **gg** is the green component
- **bb** is the blue component

The components are specified as hexadecimal numbers in the range 00 through FF. These numbers correspond to decimal (base 10) numbers from 0 to 255. The hexadecimal values are the equivalent of percentages, based on the formula shown in Table 1. Therefore, a color specification of CX000000 means 0% red, 0% green, and 0% blue, or the absence of all color, which is black. The color CXFFFFFF means 100% red, 100% green, and 100% blue, or the presence of all colors, which is white. The color CX6495ED is a mix of almost 40% red, almost 60% green, and almost 100% blue. Table 1 shows hexadecimal values, with their corresponding decimal numbers and percentages.

Hexadecimal Value	Decimal Number	Color Percent	Formula
FF	255	100%	255/255 = 1
CC	204	80%	204/255 = .8
99	153	60%	153/255 = .6
66	102	40%	102/255 = .4
33	51	20%	51/255=.2
00	0	0%	0/255=0

Table 1. Hexadecimal Numbers and Their Color Percentages

When you see a color specification such as CXCCCCFF, though, it is still hard to visualize exactly what color it corresponds to. Of course, you can use to a color picker application on the web or write a SAS program to convert hexadecimal values to colors. In that case, you can see how that particular color value (CXCCCCFF) is a lovely shade of light lavender, as shown in Display 33.



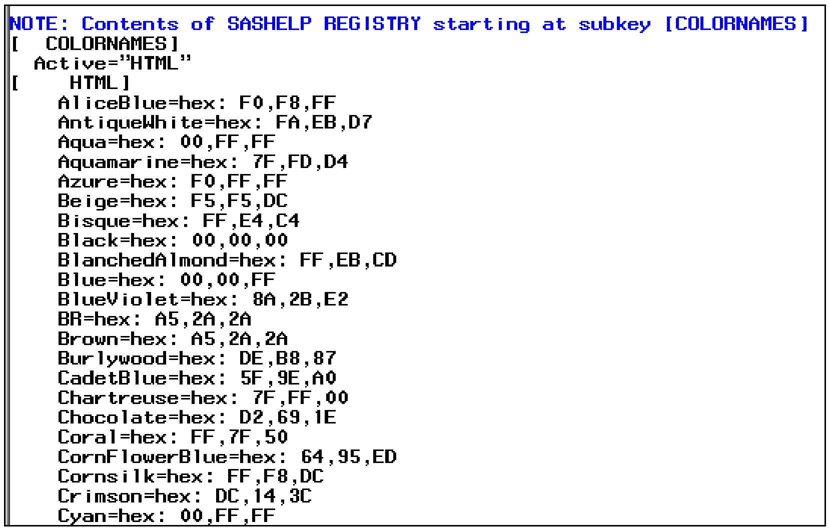
		Percent Blue					
Percent Red	Percent Green	00 = 0%	33 = 20%	66 = 40%	99 = 60%	CC = 80%	FF = 100%
CC = 80%	00 = 0%	cxCC0000	cxCC0033	cxCC0066	cxCC0099	cxCC00CC	cxCC00FF
	33 = 20%	cxCC3300	cxCC3333	cxCC3366	cxCC3399	cxCC33CC	cxCC33FF
	66 = 40%	cxCC6600	cxCC6633	cxCC6666	cxCC6699	cxCC66CC	cxCC66FF
	99 = 60%	cxCC9900	cxCC9933	cxCC9966	cxCC9999	cxCC99CC	cxCC99FF
	CC = 80%	cxCCCC00	cxCCCC33	cxCCCC66	cxCCCC99	cxCCCCCC	cxCCCCFF
	FF = 100%	cxCCFF00	cxCCFF33	cxCCFF66	cxCCFF99	cxCCFFCC	cxCCFFFF

Display 33. Partial Output Showing Color Mapping for RGB Hexadecimal Values

Display 33 shows the partial output from a program included in the zip file of programs for this paper. You can even specify color names as familiar color names, like yellow, blue, pink, and so on. However, these color names are translated to their RGB hexadecimal codes by a table that is stored in the SAS registry. To see how color names map to hexadecimal values, submit the following REGISTRY procedure code:

```
proc registry list startat="COLORNAMES"; run;
```

Partial output from the PROC REGISTRY step (as written to the SAS log) is shown in Display 34.



```
NOTE: Contents of SASHELP REGISTRY starting at subkey [COLORNAMES]
[  COLORNAMES]
  Active="HTML"
[    HTML]
    AliceBlue=hex: F0,F8,FF
    AntiqueWhite=hex: FA,EB,D7
    Aqua=hex: 00,FF,FF
    Aquamarine=hex: 7F,FD,D4
    Azure=hex: F0,FF,FF
    Beige=hex: F5,F5,DC
    Bisque=hex: FF,E4,C4
    Black=hex: 00,00,00
    BlanchedAlmond=hex: FF,EB,CD
    Blue=hex: 00,00,FF
    BlueViolet=hex: 8A,2B,E2
    BR=hex: A5,2A,2A
    Brown=hex: A5,2A,2A
    Burlywood=hex: DE,B8,87
    CadetBlue=hex: 5F,9E,A0
    Chartreuse=hex: 7F,FF,00
    Chocolate=hex: D2,69,1E
    Coral=hex: FF,7F,50
    CornFlowerBlue=hex: 64,95,ED
    Cornsilk=hex: FF,F8,DC
    Crimson=hex: DC,14,3C
    Cyan=hex: 00,FF,FF
```

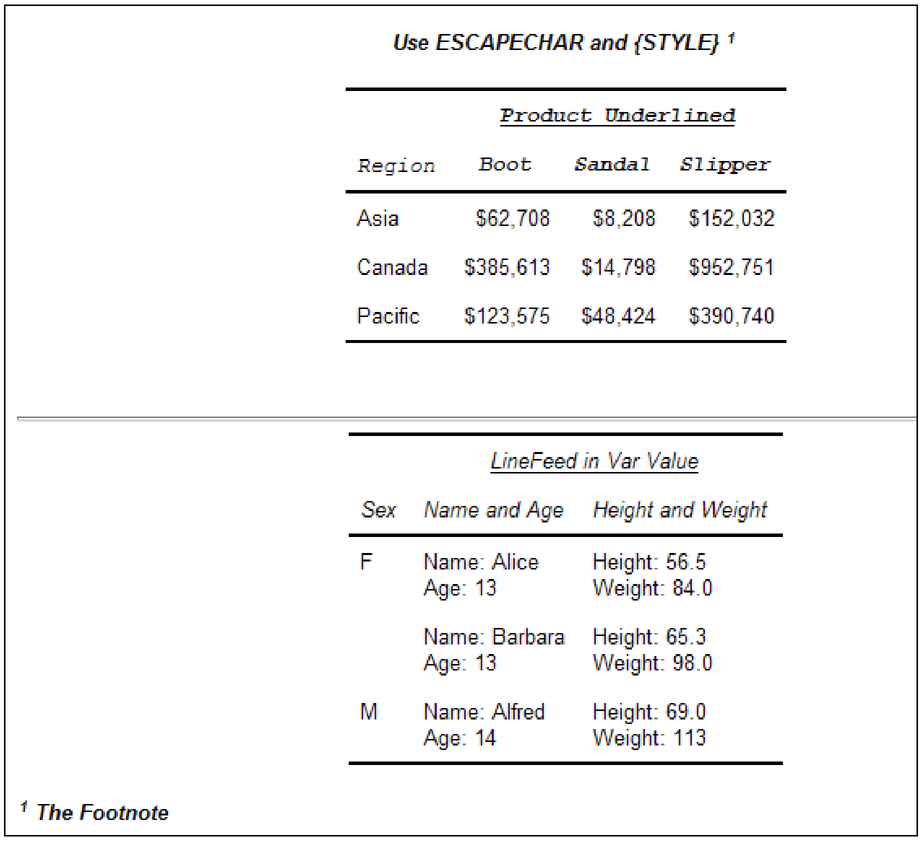
Display 34. Partial Output from the REGISTRY Procedure Showing RGB Hexadecimal Values for Color Names

So although you could use the color names Chartreuse or CornflowerBlue, know that they are going to be mapped to CX7FFF00 and CX6495ED, respectively.

USING THE ESCAPECHAR {STYLE} AND OTHER FUNCTIONS

Sometimes, you need to insert superscripts or subscripts into your report output. Other times, you need to put a line feed or carriage return into a cell's contents. Or, maybe you need to underline a particular piece of report output. All of these things are possible when you use the ODS ESCAPECHAR statement with the {STYLE} function. Consider the output shown in Display 35. It shows ODS HTML output that you do not normally get by default. For example, note the superscript in the title and the footnote. This was accomplished with the use of the ESCAPECHAR `{SUPER}` function as shown below.

```
title 'Use ESCAPECHAR and {STYLE} ^{super 1}';
footnote j=l '^{super 1} The Footnote';
```



Use ESCAPECHAR and {STYLE} [1]

	Product Underlined		
Region	Boot	Sandal	Slipper
Asia	$62,708	$8,208	$152,032
Canada	$385,613	$14,798	$952,751
Pacific	$123,575	$48,424	$390,740

LineFeed in Var Value

Sex	Name and Age	Height and Weight
F	Name: Alice Age: 13	Height: 56.5 Weight: 84.0
	Name: Barbara Age: 13	Height: 65.3 Weight: 98.0
M	Name: Alfred Age: 14	Height: 69.0 Weight: 113

[1] *The Footnote*

Display 35. ODS HTML Output Showing the Use of the ESCAPECHAR Function

The overall look and feel of the output came from the use of the JOURNAL style (no interior table lines, except under the column headings). However, the two underlined headings came from using the TEXTDECORATION style attribute with the ODS ESCAPECHAR statement and the {STYLE} function.

```
data class(keep=sex name_age ht_wt);
  length name_age $50 ht_wt $50 namestr agestr htstr wtstr $15;
  set sashelp.class (obs=3);
  namestr = 'Name: '||left(name);
  agestr = 'Age: '||put(age,2.0);
  htstr = 'Height: '||put(height,4.1);
  wtstr = 'Weight: '||put(weight,4.1);
  name_age = catx('^{newline 1}',namestr,agestr);
  ht_wt = catx('^{newline 1}',htstr,wtstr);
run;

ods html file='use_ESC_style.html' style=journal;
ods rtf file='use_ESC_style.rtf' style=journal startpage=no;
ods pdf file='use_ESC_style.pdf' style=journal startpage=no;

ods escapechar='^';

proc report data=shoes nowd;
  title 'Use ESCAPECHAR and {STYLE} ^{super 1}';
  column region sales,product;
  define region / group
         style(header)={font_face='Courier New'};
  define product / across '^{style[textdecoration=underline]Product Underlined}'
         style(header)={font_weight=bold font_face='Courier New'};
  define sales / sum ' ';
run;
```

```
footnote j=l '^{super 1} The Footnote';
proc report data=class nowd;
  title;
  column sex
    ('^{style[textdecoration=underline]LineFeed in Var Value}' name_age ht_wt);
  define sex / order;
  define name_age / display 'Name and Age';
  define ht_wt / display 'Height and Weight';
run;

ods _all_ close;
```

Besides the superscript and the underlining, another interesting feature is the fact that the {NEWLINE} ESCAPECHAR function has been inserted into the NAME_AGE variable (a concatenation of the NAME and AGE variables) and the HT_WT variable (a concatenation of the HEIGHT and WEIGHT variables). These concatenated variables were created in a DATA step program before the PROC REPORT step. The {NEWLINE} function inserts a carriage return/line feed character into a text string, so that when ODS detects the presence of the {NEWLINE} function, it inserts the appropriate control into the result file for HTML, RTF, and PDF destinations.

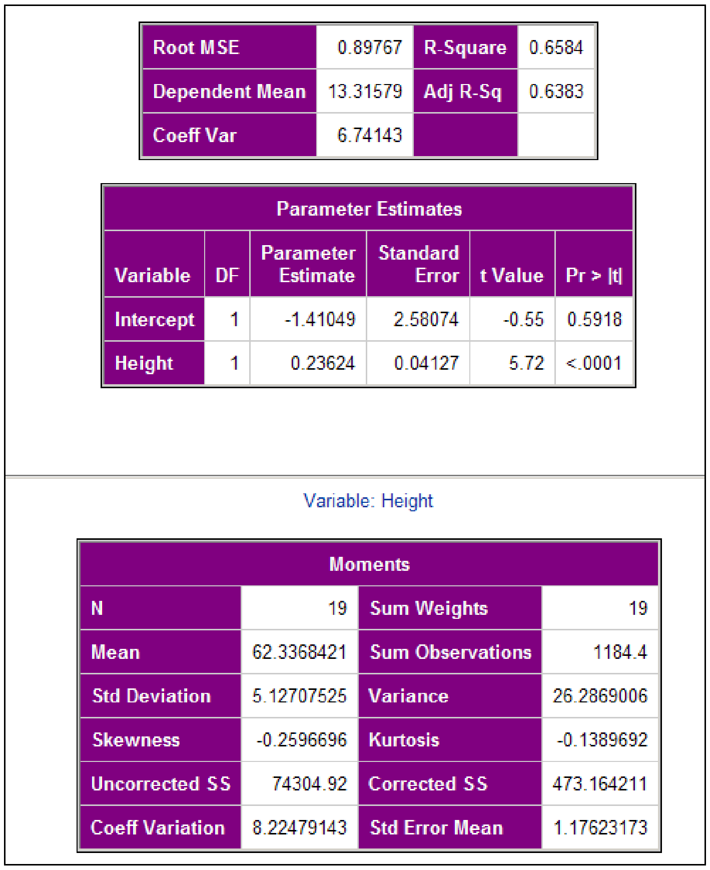
This is just the tip of the iceberg as far as what you can accomplish with the ODS ESCAPECHAR statement and the ability to affect the style of your output using these in-line formatting controls.

USING SAS STYLE TEMPLATES

SAS style templates might be listed last in this paper, but except for CSS stylesheets, they are the only way to affect your output from other procedures, not just the REPORT, PRINT, or TABULATE procedures. Since the advent of SAS 9.2, creating style templates has never been simpler. Let’s say that we are working for “The Purple Palace” and need to create a style template so that no matter which procedure is used, the output looks the same as SASWEB output, except the header cells are purple instead of blue. Using the new CLASS statement, you do not have to worry about inheritance or tracing the values for the attributes in use. You only have to know which style element you want to change (in this case, the HEADER style element) in order to write your style template.

```
ods path work.tmp(update) sasuser.templat(update) sashelp.tmplmst(read);
proc template;
  define style styles.purple;
    parent=styles.sasweb;
    class Header /
       background=purple
       foreground=white;
  end;
run;
```

Then, once you use the new style template on your ODS destination statement, you can see in Display 36, that the headings are purple in all of the output from all of the procedures. (HTML output is shown, but RTF and PDF output would be the same.) Only the REG procedure output and partial UNIVARIATE procedure output are shown..



Root MSE	0.89767	R-Square	0.6584
Dependent Mean	13.31579	Adj R-Sq	0.6383
Coeff Var	6.74143		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-1.41049	2.58074	-0.55	0.5918
Height	1	0.23624	0.04127	5.72	<.0001

Variable: Height

Moments

N	19	Sum Weights	19
Mean	62.3368421	Sum Observations	1184.4
Std Deviation	5.12707525	Variance	26.2869006
Skewness	-0.2596696	Kurtosis	-0.1389692
Uncorrected SS	74304.92	Corrected SS	473.164211
Coeff Variation	8.22479143	Std Error Mean	1.17623173

Display 36. Partial HTML Output Showing the Use of the New Style Template

The code that created the above display is shown below.

```
ods html file='demo08_purple_style.html' style=styles.purple;
ods rtf file='demo08_purple_style.rtf' style=styles.purple;
ods pdf file='demo08_purple_style.pdf' style=styles.purple;

proc freq data=sashelp.class;
  tables sex;
  label sex = 'Gender';
run;

ods select fitstatistics parameterestimates;
proc reg data=sashelp.class;
  model age=height;
run;

ods select basicmeasures;
proc univariate data=sashelp.class;
  var height;
run;

ods _all_ close;
```

With just a little bit of effort, you can design custom style templates for use with your reports. To help get you started, I recommend the ODS tip sheet on style templates that is available at http://support.sas.com/rnd/base/ods/scratch/styles-tips.pdf. (Note that the URL contains the lowercase letters “RND” and not “MD” after “support.sas.com”.)

IT'S NOW OR NEVER: #9 KNOWING WHEN TO USE THE TEMPLATE PROCEDURE

The key to knowing when to use the TEMPLATE procedure is to understand 1) what you need to change in your output, 2) whether a template change will accomplish what you want, and 3) which template is the right template to change.

ODS has four main types of templates: table templates, style templates, graph templates, and tagset templates. (These templates should not be confused with the type of SAS/GRAPH template that is used with the GREPLAY procedure.) Consider the output of the MEANS procedure. If you tell me that you want to make all of the heading cells in the MEANS procedure output purple, I would tell you that you could change that feature in a table template. However, the more appropriate place to change all heading cells to purple is in a style template. If you told me that you wanted to apply the DOLLARw.d format to the SUM and MEAN statistics from the MEANS procedure, then I would tell you that the most appropriate place to do that is in the table template, BASE.SUMMARY, which controls the formats used for the calculated statistics. It is a rare occasion that you need to change or alter a tagset template – unless you know that you need to modify the underlying markup language tags or you need to generate custom XML tags from a SAS procedure or process.

However, if you need to accomplish traffic-lighting for MEANS procedure output, then that is a job for the CELLSTYLE-AS statement inside a table template. If you need to use an ordered list in HTML output instead of the standard <TABLE> tag, or you need to insert some custom Javascript into your HTML, then that might be a job for a tagset template. If you use TAGSETS.EXCELXP and you want to insert a line feed into a data cell, that is probably a job for the ODS ESCAPECHAR statement and not a template at all. In fact, just because you use the TAGSETS.EXCELXP or TAGSETS.MSOFFICE2K_X destinations to create XML or HTML output for Excel, you rarely need to alter a tagset template to work with those two destinations. Usually, you need to either use a changed style template or use the STYLE= option with the REPORT, PRINT, or TABULATE procedures.

Graph template usage falls into two categories:

1) modifying an existing graph template that is used by one of the SAS/STAT procedures to incorporate some graphical change that cannot be accomplished with a style template. For example, you might want to change the axis tick marks or add an inset box.

2) generating your own custom graphical output using the Graph Template Language (GTL), which was introduced in SAS 9.2.

For an introduction to the four main template types and an example of modifying each type of template, I recommend my 2009 SAS Global Forum paper, "Tiptoe through the Templates" (available at http://support.sas.com/resources/papers/proceedings09/227-2009.pdf). For more information about table templates, look for papers by Kevin Smith and Phil Holland or refer to this tip sheet on table templates, http://support.sas.com/rnd/base/ods/scratch/table-tips.pdf, written by the ODS developers. If you need to modify tagset templates, then I call your attention to user group papers written by Eric Gebhart and Chevell Parker. Last, but not least, for more information about graph templates and GTL, look for papers by Sanjay Matange, Susan Schwarz, and Dan Heath, and for papers and books by Warren Kuhfeld.

Because the TEMPLATE procedure has a separate syntax for each template type, my recommendation is to start slowly, but start now to learn how to use the TEMPLATE procedure. The TEMPLATE procedure gives you a way to interact with the infrastructure of the Output Delivery System, and there's no time like the present to start learning how to make the TEMPLATE procedure work for you.

WHAT'S NEW PUSSYCAT: #10 FINDING OUT NEW ODS FEATURES

One of the best things about the Output Delivery System is innovation. New features of ODS are continually being developed, such as:

- ODS LAYOUT statement
- ODS DATA step Object syntax
- ODS GRAPHICS support for SAS/STAT procedures (more procedures added every release)
- ODS Graphics Designer

The best places to find out about new features of ODS are to look at previous SAS Global Forum proceedings and the main ODS Web site at http://support.sas.com/rnd/base/index.html. Some of the ODS GRAPHICS papers can be found at http://support.sas.com/rnd/base/topics/statgraph/, and others can be found in SAS Global Forum and other user group papers.

CONCLUSION

(See whether you can identify all the real song titles used in this conclusion. Answers are at the end of the paper.)

I'm telling you now, that I'm a believer in ODS! And, all I really want to do here at the end of our road, is to get everybody talking about the good things in ODS.

Now that you've got what it takes to sing the praises of ODS, I hope to hear you say, "I'll try something new!" I'm leaving it all up to you.

RECOMMENDED READING

Haworth, Lauren E., Zender, Cynthia L., and Burlew, Michele M. 2009.Output Delivery System: the Basics and Beyond . Cary, NC: SAS Institute Inc.

Kuhfeld, Warren F.. 2010. Statistical Graphics in SAS: An Introduction to the Graph Template Language and the Statistical Graphics Procedures. Cary, NC: SAS Institute Inc.

Wimmer, Frank, Ellis, Ken. 2001. "Visual Styles for V9 SAS® Output". Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference. Cary, NC: SAS Institute Inc. (Paper 172-26)

Cartier, Jeff. 2003. "It's All in the Presentation". Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference. Cary, NC: SAS Institute Inc. (Paper 147-28)

Cartier, Jeff. and Heath, Dan. 2007. "Using ODS Styles with SAS/GRAPH®". Proceedings of the SAS Global Forum 2007 Conference. Cary, NC. SAS Institute Inc. (Paper 088-2007)

Gebhart. Eric. 2007. "ODS Markup, Tagsets, and Styles! Taming ODS Styles and Tagsets". Proceedings of the SAS Global Forum 2007 Conference. Cary, NC. SAS Institute Inc. (Paper 225-2007)

Heath, Dan. 2007. " SAS/GRAPH® Procedures for Creating Statistical Graphics in Data Analysis". Proceedings of the SAS Global Forum 2007 Conference. Cary, NC. SAS Institute Inc. (Paper 193-2007)

Heath, Dan. 2008. "Effective Graphics Made Simple Using SAS/GRAPH® SG Procedures". Proceedings of the SAS Global Forum 2008 Conference. Cary, NC. SAS Institute Inc. (Paper 255-2008).

Heath, Dan. 2009. "Secrets of the SG Procedures". Proceedings of the SAS Global Forum 2009 Conference. Cary, NC: SAS Institute Inc. (Paper 324-2009)

Holland, Philip R. 2010. "Developing ODS Templates for Nonstandard Graphs in SAS 9.2". Proceedings of the SAS Global Forum 2010 Conference. Cary, NC: SAS Institute Inc. (Paper 226-2010)

Huntley, Scott. 2006. "Let the ODS PRINTER Statement Take Your Output into the Twenty-First Century". Proceedings of the Thirty-First Annual SAS Users Group International Conference.Cary, NC: SAS Institute Inc. (Paper 227-31)

Kuhfeld, Warren F. 2010. "The Graph Template Language and the Statistical Graphics Procedures: An Example-Driven Introduction". Proceedings of the PharmaSUG 2010 Conference. Cary, NC: SAS Institute Inc. (Paper TU-SAS01)

Matange, Sanjay. 2008. "Introduction to the Graph Template Language". Proceedings of the SAS Global Forum 2008 Conference. Cary, NC: SAS Institute Inc. (Paper 313-2008)

O'Connor, Daniel. 2003. "Next Generation Data _NULL_ Report Writing Using ODS OO Features". Proceedings of the Twenty-Eighth Annual SAS Users Group International Conference. Cary, NC: SAS Institute Inc. (Paper 22-28)

O'Connor, Daniel. 2009. "The Power to Show: Ad Hoc Reporting, Custom Invoices, and Form Letters". Proceedings of the SAS Global Forum 2009 Conference. Cary, NC: SAS Institute Inc. (Paper 313-2009)

O'Connor, Daniel. and Huntley, Scott. 2009. "Breaking New Ground with SAS® 9.2 ODS Layout Enhancements". Proceedings of the SAS Global Forum 2009 Conference. Cary, NC: SAS Institute Inc. (Paper 043-2009)

Parker, Chevell. 2003. "Generating Custom Excel Spreadsheets using ODS". Proceedings of the Twenty-Eighth

Annual SAS Users Group International Conference. Cary, NC. SAS Institute Inc. (Paper 12-28)

Rodriguez, Robert N. 2008. "Getting Started with ODS Statistical Graphics in SAS 9.2". Proceedings of the SAS Global Forum 2008 Conference. Cary, NC: SAS Institute Inc. (Paper 305-2008)

Schwartz, Susan. 2008. "Butterflies, Heat Maps, and More: Explore the New Power of SAS/GRAPH®". Proceedings of the SAS Global Forum 2008 Conference. Cary, NC. SAS Institute Inc. (Paper 243-2008).

Schwartz, Susan. 2009. "Clinical Trial Reporting Using SAS/GRAPH® SG Procedures". Proceedings of the SAS Global Forum 2009 Conference. Cary, NC: SAS Institute Inc. (Paper 174-2009)

Smith, Kevin D. 2006. "The TEMPLATE Procedure Styles: Evolution and Revolution". Proceedings of the Thirty-first Annual SAS Users Group International Conference. Cary, NC. SAS Institute Inc. (Paper 053-31)

Smith, Kevin D. 2007. "The Output Delivery System (ODS) from Scratch". Proceedings of the SAS Global Forum 2007 Conference. Cary, NC. SAS Institute Inc. (Paper 219-2007)

Truxillo, Catherine. and Zender, Cynthia. 2005. "Customizing ODS Statistical Graphs". Proceedings of the Thirtieth Annual SAS Users Group International Conference, . . (Paper 239-30)

Zender, Cynthia L. 2010. "SAS® Style Templates: Always in Fashion". Proceedings of the SAS Global Forum 2010 Conference. Cary, NC: SAS Institute Inc. (Paper 033-2010)

Zender, Cynthia L. 2009. "CSSSTYLE: Stylish Output with ODS and SAS® 9.2". Proceedings of the SAS Global Forum 2009 Conference. Cary, NC: SAS Institute Inc. (Paper 014-2009)

Zender, Cynthia L. 2009. "Tiptoe through the Templates". Proceedings of the SAS Global Forum 2009 Conference. Cary, NC: SAS Institute Inc. (Paper 227-2009)

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. (Paper 318-2009)

Zender, Cynthia L. 2008. "Creating Complex Reports". Proceedings of the SAS Global Forum 2008 Conference. Cary, NC: SAS Institute Inc. (Paper 173-2008)

Zender, Cynthia L. 2007. "Funny ^Stuff~ in My Code: Using ODS ESCAPECHAR". Proceedings of the SAS Global Forum 2007 Conference. Cary, NC: SAS Institute Inc. (Paper 099-2007)

Zender, Cynthia L. 2005. "The Power of TABLE Templates and DATA _NULL_". Proceedings of the Thirtieth Annual SAS Users Group International Conference, (Paper 088-30)

ACKNOWLEDGMENTS

This paper would not be possible without the work of the ODS and ODS GRAPHICS developers. In addition, I owe many thanks to the paper reviewers, Chevell Parker, Bari Lawhorn, Michele Ensor, Lorilyn Russell, Sanjay Matange, Dan Heath, and Eric Gebhart, whose suggestions have improved this paper. Thanks are also due to Dr. Leroy Bessler, who first requested a presentation on this topic for the 2010 Midwest SAS Users Group. That original presentation was the impetus for this paper and the SAS Global Forum presentation of the same name. And, last, but certainly not least, I'd like to thank Jeanne Ferneyhough who edited this paper and made it better than it started.

CONTACT INFORMATION

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

Cynthia Zender
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
919-531-9012
E-mail: Cynthia.Zender@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Answer: Hits Used in the Conclusion

I'm Telling You Now by Freddie & The Dreamers (1965); I'm a Believer by The Monkees (1966); All I Really Want To Do by Cher (1965); The End Of Our Road by Gladys Knight & The Pips (1968); Everybody's Talkin' by Nilsson (1969); Good Thing by Paul Revere and the Raiders (1967); You Got What It Takes by The Dave Clark Five (1967); I'll Try Something New by Diana Ross & The Supremes (1969); I'm Leaving It (All) Up To You by Donny & Marie Osmond (1974)