

ODS Tips and Tricks

Lauren Haworth, Genentech, Inc., San Francisco

➤ ABSTRACT

Haven't gotten around to reading the entire ODS section in Online Doc? Don't have time to read a whole book on the topic? This presentation will show some simple tips and tricks from my book "Output Delivery System: The Basics". In no time you'll be impressing your boss and coworkers with beautiful customized ODS output. The presentation will include easy-to-use techniques for enhancing HTML, RTF, and printer results.

➤ HIDING PROCEDURE TITLES

"The FREQ Procedure" is meaningful to a SAS programmer, but not exactly common terminology. It would be nice to be able to get rid of these labels in our output. Luckily for us, there's a simple way to do this in ODS. A single option setting gets rid of all of these titles. And you don't even have to use PROC TEMPLATE. The syntax is:

ODS NOPTITLE;

To see how it works, we'll create a simple PDF file. Here is the output before using the new option. The standard title "The FREQ Procedure" appears above the output table.

<i>Construction Materials</i>						
<i>The FREQ Procedure</i>						
Frequency	Table of Material by Type					
Material(Primary construction material)	Type(Building type)					Total
	Retail	Residential	Office	Warehouse		
Wood	4	48	13	11		76
Concrete	15	34	42	24		115
Brick	7	36	14	12		69
Steel	3	0	66	13		82
Total	29	118	135	60		342

To get rid of this title, just add the NOPTITLE option, as in the following code.

```
ODS PDF FILE='NoProcedure.pdf';
ODS NOPTITLE;
title 'Construction Materials';
proc freq data=Buildings;
    tables material*type / norow
nocol nopercnt;
run;
ODS PDF CLOSE;
```

Here is the same table with the new option in place. The procedure title has disappeared. Notice that the title "Construction Materials" that was created by the TITLE statement in the code has not been affected.

<i>Construction Materials</i>						
Frequency	Table of Material by Type					
Material(Primary construction material)	Type(Building type)					Total
	Retail	Residential	Office	Warehouse		
Wood	4	48	13	11		76
Concrete	15	34	42	24		115
Brick	7	36	14	12		69
Steel	3	0	66	13		82
Total	29	118	135	60		342

The ODS NOPTITLE option persists until you change it back with an ODS PTITLE statement.

➤ CHANGING PROCEDURE LABELS IN THE TABLE OF CONTENTS

The PTITLE option only affects the procedure titles in the body of the output. If you use this technique with the HTML destination, you will still get the title "The FREQ Procedure" in the table of contents. Not only is this title meaningless to many users, but it can also get confusing if your SAS program generates 50 results from the same procedures. For example, 50 PROC MEANS steps would create 50 labels for "The MEANS Procedure" in your table of contents. To help with this, ODS provides a statement to specify a custom procedure label.

The ODS PROCLABEL statement is shown in the code below. In this example, the TITLE2 statement calls for the output to be headed by the title "Hours Billed by Customer". The ODS PROCLABEL statement asks for the same thing, only it applies to the table of contents and the Results window.

```

ODS HTML BODY='body.html'
      CONTENTS='contents.html'
      FRAME='main.html';
ODS PROCLABEL 'Hours Billed by
Customer';
title2 'Hours Billed by Customer';
proc means data=Billings nonobs mean
sum;
      class CustomerName;
      var Hours;
run;
ODS HTML CLOSE;

```

The resulting table of contents is shown below (the rest of the web page with the actual results is hidden to save space).

Table of Contents 1. Hours Billed by Customer Summary statistics	Quarter Hours E The MEAN Analysis
--	---

Notice the revised procedure label in the table of contents. It now reads “Hours Billed by Customer”. The subheading “Summary Statistics” remains the same.

You can also use this technique to get rid of the procedure labels in the table of contents. This can save space on your web page, though you may lose some clarity. To remove the procedure labels, assign a blank label to the procedure, as in the example below.

```

ODS HTML BODY='body.html'
      CONTENTS='contents.html'
      FRAME='main.html';
ODS PROCLABEL ' ';
title2 'Hours Billed by Customer';
proc means data=Billings nonobs mean
sum;
      class CustomerName;
      var Hours;
run;
ODS HTML CLOSE;

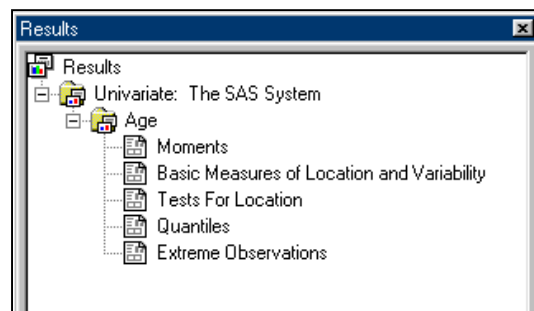
```

The output is shown in Output 3.14. Now the label has gone away. Unfortunately, a number and a blank line are still left in place. See the example in Chapter 11, “Modifying Output Structure”, for how to use the TEMPLATE procedure to get rid of the number and line.

➤ IDENTIFYING OUTPUT OBJECT NAMES

When you start using ODS OUTPUT to create output data sets, one of the first things you will discover is that SAS procedures create a lot of output objects, and sometimes it’s trick to figure out which one you want.

You can run an ODS TRACE statement to output lists of the output object names to the log window, or for SAS/STAT procedures, you can look up output object names in the manual, but there’s an easier way to identify output objects. The trick is to take advantage of the Results window. When you run a procedure in interactive SAS, you get a list of the output objects it creates in the Results window. For example, if you run a PROC UNIVARIATE, you will see the following output objects in the Results window.



In this example, you can see that the UNIVARIATE procedure has created five output objects: Moments, Basic Measures of Location and Variability, Tests for Location, Quantiles, and Extreme Observations.

If you wanted to create an output data set with the mean, standard deviation, variance, Skewness, and kurtosis statistics, all you have to do is identify which of the five output objects contain these statistics. In this example, those items are in part of the output labeled “Moments” so we can guess that it’s the first output object “Moments” that we need. If we weren’t sure, we could click on each object in the Results window and see which part of the UNIVARIATE output it brought up in the Results window.

To create a data set with the results we’ve selected, the syntax for our ODS statement will be:

```

ODS OUTPUT 'Moments'=MyOutputData;
proc univariate data=sashelp.class;
      var age;
run;
ODS OUTPUT CLOSE;

```

Notice the quotes around the output object “Moments”. The reason for the quotes is that

we're using the output object's label to identify it. Without the quotes, SAS assumes you are using the output object name. When you use the Results window to identify an output object, what you're viewing is the object label. If you want to know the name (often they're shorter, which saves typing), just right click on the object label in the Results window and select "Properties" from the menu.

This sample program creates a data set with six observations and seven variables. There's one observation for each row of information in the procedure results. Each row of data has two statistics and their values.

➤ USING DATA SET OPTIONS WITH OUTPUT DATA SETS

The ODS OUTPUT statement makes it easy to create data sets with all of the results that you can generate in printed output. But sometimes you don't want all of those results.

In the previous example, we've selected specific output data set. The contents of this data set were determined by ODS. Each time you request an output data set; you get the same standard results.

For example, if you run the RSREG procedure and request a data set with the parameter estimates using this code ...

```
ODS OUTPUT "Parameter Estimates"=Params;
proc rsreg data=traffic;
    model fatal=mph;
run;
ODS OUTPUT CLOSE;
```

... you automatically get a data set with three records and eight variables. These variables are: Dependent, Parameter, DF, Estimate, StdErr, tValue, .Probt, and CodedEstimate.

However, you may not always want all eight variables, and you may not like their default names. For example, let's say we want only three of the variables, and you would like to rename "Probt" to "Pvalue". One way to achieve this would be to write a DATA step like the following:

```
data Params2;
    set Params
    (keep=Parameter Estimate Probt
    rename=(Probt=Pvalue));
run;
```

This code will do the trick, and could be used to create the data set shown below.

Obs	Parameter	Estimate	Pvalue
1	Intercept	0.427698	<.0001
2	MPH	-0.009171	0.1418
3	MPH*MPH	0.000111	0.2457

However, this is an extra processing step. A better way to do this is to specify these changes in the ODS OUTPUT statement. When you ask for a data set in an ODS OUTPUT statement, all of the usual data set options apply. By adding parentheses after the SAS data set name, you can use all of the usual data set options like KEEP, RENAME, and DROP.

The following code accomplishes the same thing as the combination of ODS and DATA step code above. However, this code does it in a single pass of the data, saving resources.

```
ODS OUTPUT "Parameter Estimates"=Params
(KEEP=Parameter Estimate Probt
RENAME=(Probt=Pvalue));
proc rsreg data=traffic;
    model fatal=mph;
run;
ODS OUTPUT CLOSE;
```

This code will produce the same data set as the previous code, and it does it in a single step. You can use any of the usual data step options in this way: DROP, KEEP, RENAME, WHERE, etc.

➤ FITTING MORE OUTPUT ON THE PAGE

Sometimes when you're building a report, there's just too much information to fit on the page. You can make yourself crazy trying to reorganize your results, change formats, reduce labels, change the page orientation, and all of the other tricks we employ.

Another approach is to build a custom ODS style to use in these situations. For example, we can build a style just for those reports where the output is too wide for the page. In the following code, we create a report that shows quantile statistics for a variable using the MEANS procedure.

```

ODS PDF FILE='ShoeSales.pdf';
proc means data=sashelp.shoes
maxdec=2
    mean std min p1 p5 p10 p25 median
    p75 p90 p95 p99 max;
    var sales;
run;
ODS PDF CLOSE;

```

The resulting report is shown below. It does not quite fit the page width. To solve this problem, we can create a style definition that uses a narrower font, and squeeze that last column into the page width.

Analysis Variable : Sales Total Sales												
Mean	Std Dev	Minimum	1st Pctl	5th Pctl	10th Pctl	25th Pctl	Median	75th Pctl	90th Pctl	95th Pctl	99th Pctl	
85700.17	129107.23	325.00	554.00	1487.00	2617.00	15312.00	38912.00	108936.00	229372.00	328474.00	700513.00	
												Analysis Variable : Sales Total Sales
												Maximum
												1298717.00

All we need to do is change the style attributes in the Fonts style element. The following TEMPLATE procedure does the trick. It changes the point size for each font style name in the Printer style definition to a font that is a single point smaller. The fixed-width font settings for batch output were left unchanged, as they were already smaller.

```

proc template;
define style PrinterSmaller;
parent=styles.Printer;
replace fonts /
'TitleFont2' = ("Times Roman",11pt,Bold Italic)
'TitleFont' = ("Times Roman",12pt,Bold Italic)
'StrongFont' = ("Times Roman",11pt,Bold)
'EmphasisFont' = ("Times Roman",9pt,Italic)
'FixedEmphasisFont' = ("Courier",9pt,Italic)
'FixedStrongFont' = ("Courier",9pt,Bold)
'FixedHeadingFont' = ("Courier",9pt,Bold)
'BatchFixedFont' = ("SAS Monospace, Courier",6.7pt)
'FixedFont' = ("Courier",9pt)
'headingEmphasisFont' = ("Times Roman",10pt,Bold Italic)
'headingFont' = ("Times Roman",10pt,Bold)
'docFont' = ("Times Roman",9pt);
end;
run;

```

When we re-run our MEANS procedure, all we need to do is specify STYLE=PrinterSmaller in our ODS PRINTER statement. Then we get the improved output shown below.

Analysis Variable : Sales Total Sales												
Mean	Std Dev	Minimum	1st Pctl	5th Pctl	10th Pctl	25th Pctl	Median	75th Pctl	90th Pctl	95th Pctl	99th Pctl	Maximum
85700.17	129107.23	325.00	554.00	1487.00	2617.00	15312.00	38912.00	108936.00	229372.00	328474.00	700513.00	1298717.00

Now all of the output fits within the page width. Once you've set up this style, you can easily call it whenever have a set of results that are slightly too large or too small for the page.

➤ CHANGING THE SPACING INSIDE TABLES

There's another way to create a style definition that fits more output on each page. ODS output is composed of tables, text, and the space between them. By default, a certain amount of space is left between the results and the table borders that surround them. If you'd like to create a more compressed table with less space, or a more open table with more space, you can create a style definition to do that.

The attribute setting you need to modify is called cellpadding. It's part of the Output style element of the Default style definition. By default, the setting is 7 pixels. This means that to the left and right of each number or label in the table, there are at least 7 pixels of space. There are also 7 pixels of space above and below each number or label.

If you wish to create a more compressed table, you simply reduce the value of the cellpadding setting. The following code sets the cellpadding to 3 pixels.

```

PROC TEMPLATE;
    DEFINE STYLE DefaultSqueeze;
        PARENT=Styles.Default;
        REPLACE Output from Container /
            background = colors('tablebg')
            rules = GROUPS
            frame = BOX
            cellpadding = 3
            cellspacing = 1
            bordercolor = colors('tableborder')
            borderwidth = 1;
    END;
RUN;

```

The results of this change can be seen by comparing the first version of our HTML file:

Analysis Variable : CostSqFt Cost per square foot		
Building height	N	Mean
1 story	117	170.75
2 stories	122	180.83
mid-rise	54	184.65
high-rise	49	175.59

With this revised version:

Analysis Variable : CostSqFt Cost per square foot		
Building height	N	Mean
1 story	117	170.75
2 stories	122	180.83
mid-rise	54	184.65
high-rise	49	175.59

In general, you will find that the default setting is just fine. However, if you need to squeeze a bit more output on a page, reducing the cell padding is one way to go. Similarly, if you wish to leave a bit more room to make a crowded table easier to read, increase the cell padding.

This example is for HTML output. If you are creating RTF or Printer output, the cellpadding attribute uses settings in points. The default value is 4pt for Printer and 3pt for RTF.

➤ ADDING PAGE NUMBERS TO RTF OUTPUT

When you generate a lengthy set of output, it's nice to provide page numbers in the format "Page XX of YY" so the user knows that they have received all of the pages.

Before ODS, the only way to accomplish this was to write out the report with dummy footnotes, and then read the file back in, count the pages, and then write it out again with the correct page numbers.

Now that SAS can create RTF output using ODS, we have the opportunity to take advantage

of our word processor's ability to create this kind of page numbers automatically. All you have to do is pass the correct RTF tags to your ODS RTF output.

The first step is to run PROC TEMPLATE to change one setting that affects how these added tags are handled:

```
proc template;
  define style styles.TempRTF;
    parent=styles.RTF;
    style SystemFooter from SystemFooter /
      protectspecialchars=off;
  end;
run;
```

Then we will use two tricks to add the footnote. First, we're going to use in-line formatting commands so that the page numbers are passed only to the RTF destination. To do that we need to set up a character to identify in-line formatting commands:

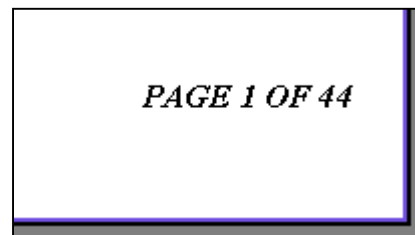
```
ODS ESCAPECHAR="^";
```

Then, we create a special footnote. It uses an in-line formatting command "^{^R/RTF}" to request that the following text be sent only to the RTF destination. Then it contains the RTF commands used to specify page numbers in the "Page XX of YY" format.

```
footnote J=R "^R/RTFPAGE
}{\field{\*\fldinst {\cs17 PAGE
}}{\fldrslt {\cs17\lang1024
1}}}{\cs17 OF }{\field{\*\fldinst
{\cs17 NUMPAGES}}{\fldrslt
{\cs17\lang1024 1}}}{\cs17 '";
```

This tag is cryptic and confusing, but if you type it out correctly, it tells word processors how to set up the page numbering.

Now, if we create both an HTML and an RTF file from our output, the RTF file will use this footnote to assign page numbers, and the HTML file will have a blank footnote. The following snippet of the RTF output shows how the page numbers appear.



➤ REMOVING THE TABLE BORDERS AND RULES

One of the more common complaint about ODS output is that it has too many borders and rules. Many users prefer more simple output that just shows the results without all of the extra formatting. You can also make the borders and rules go away altogether. All you have to do is create a new style definition with a few minor changes. We need to make a few changes to the Output style element of the Default style definition. As a reminder, the original settings of the Output style element are shown below.

```
Style Output from Container /  
background = colors('tablebg')  
rules = GROUPS  
frame = BOX  
cellpadding = 7  
cellspacing = 1  
bordercolor = colors('tableborder')  
borderwidth = 1;
```

There are three of the settings that need to be changed if we wish to create a table without borders or rules. The first one is the rules attribute setting. The Default style definition uses the setting GROUPS. This calls for rules between the table header and the table body, and between the table body and the table footer. If we wish to get rid of these rules, the setting must be changed to NONE.

The web page below shows how your output appears with this setting in place.

Analysis Variable : CostSqFt Cost per square foot		
Building type	N	Mean
Retail	29	184.62
Residential	118	171.75
Office	135	180.12
Warehouse	60	177.97

The second setting that is used to control borders is the frame attribute setting. The Default style definition uses BOX. This setting calls for borders around the outside of the table. To get rid of these borders, the setting must be changed to VOID. Here's what the table borders look like with this second change in place.

Analysis Variable : CostSqFt Cost per square foot		
Building type	N	Mean
Retail	29	184.62
Residential	118	171.75
Office	135	180.12
Warehouse	60	177.97

At this point, you may think that the task is complete. However, if you look closely the results above, you will see thin white rules between the table cells. These are not actually rules, but are spaces between the table cells. However, they look like rules, so we'd like to remove them. To do this we need to change the cellspacing attribute setting from 1 to 0. Our final code is shown below.

```
PROC TEMPLATE;  
DEFINE STYLE DefaultNoBorderRule;  
PARENT=Styles.Default;  
REPLACE Output from Container /  
background = colors('tablebg')  
rules = NONE  
frame = VOID  
cellpadding = 7  
cellspacing = 0  
bordercolor = colors('tableborder')  
borderwidth = 1;  
END;  
RUN;
```

The final version of our output is shown below.

Analysis Variable : CostSqFt Cost per square foot		
Building type	N	Mean
Retail	29	184.62
Residential	118	171.75
Office	135	180.12
Warehouse	60	177.97

This example used the HTML destination and the Default style template, but you can make similar modifications for the style templates for the RTF and Printer destinations.

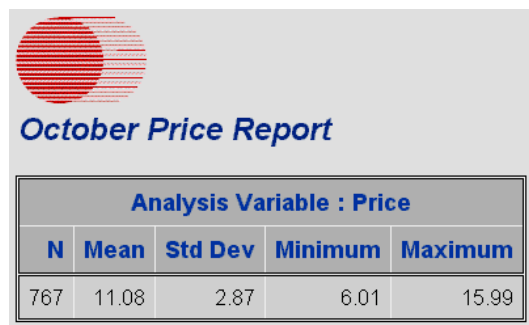
➤ ADDING A LOGO TO YOUR OUTPUT

Once you start customizing your style definitions, one of the first things you will want to do is add a company logo or other graphic to your page headings. This is a great way to add your corporate identity to any results you send out. This example will show how to add a logo, and then demonstrate the technique for HTML, RTF, and printer output.

First, we need to build a new style definition. You do this by using an attribute of the Body style element. The attribute is called preimage, and you use it to name the image file to be used. For HTML output, these images must be in a browser-compatible format like GIF or JPEG. For RTF output, the images need to be JPEG files. For printer output (PostScript, PDF), you can use virtually any image format.

```
PROC TEMPLATE;  
  DEFINE STYLE DefaultLogo;  
    PARENT=Styles.Default;  
    REPLACE Body from Document /  
      preimage="logo.gif";  
  END;  
RUN;
```

The web page below shows the results from the new style definition in HTML. The logo is displayed at the top of the page.



Analysis Variable : Price				
N	Mean	Std Dev	Minimum	Maximum
767	11.08	2.87	6.01	15.99

Okay, you're probably thinking, "this is the ugliest design I have ever seen". The red logo looks awful with the blue and gray Default style definition. This example is just the beginning. If you are going to develop a corporate style definition, you will want to work with the color settings as well as the prehtml code in order to create an attractive look. But once created, you can use it repeatedly to ensure a consistent corporate look for all of your output.

➤ CONCLUSION

This paper is an extremely brief introduction to some of the more useful ODS tips and tricks. To really understand ODS and to take advantage of

its power, you'll need to get to know ODS a lot better. See the examples in the ODS sections of Online Doc, read up on the many SUGI papers on ODS, and of course, there's always my book: *Output Delivery System: The Basics*.

➤ ACKNOWLEDGEMENTS

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

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

➤ CONTACTING THE AUTHOR

Please direct any questions or feedback to the author at: info@laurenhaworth.com