Paper 132-30

SAS® with Style: Creating your own ODS Style Template for PDF Output

Lauren Haworth, Genentech, Inc., South San Francisco, CA

➢ ABSTRACT

Once you have started using the Output Delivery System, you will quickly discover that your taste in output design probably does not coincide with the built in ODS styles shipped with SAS software. While you can edit your PDF output in Acrobat to improve its appearance, a better approach is to create your own style template. This workshop will take you step-by-step through the process of creating a custom style for your PDF output.

You will learn how to make minor modifications, and how to give your output a complete makeover. If you would like all of your SAS output to be in hot pink with a gigantic script font, this workshop will show you how! Alternatively, if you would just like to use fonts and borders that coordinate with your corporate style guidelines, you can do that too. The workshop will also provide tips and tricks for taking advantage of the PDF destination.

The workshop will walk through the TEMPLATE procedure, showing how you can redefine the default style elements and attributes to customize fonts, colors, and borders to suit your own personal or corporate style. You will be given a basic style template that you can customize during the workshop and then take home to try out on your ODS output. While many of the techniques in this workshop apply to other ODS destinations, the focus will be on PDF. The workshop is aimed at beginning to intermediate ODS users, and is based on SAS versions 8.2 and 9.1.

➢ INTRODUCTION

ODS styles are a huge topic, and there is no way to cover them in depth in this format. This workshop will take a fast-track approach. We will cover just enough of the syntax to get you started. Then, we will use a sample style template that can be edited to modify color, fonts, spacing, rules, and borders. This will allow you customize most aspects of your output. However, gain complete control over the look of your output, plan to take a much more in-depth course.

➢ USING THE STYLE= OPTION

You may not have realized it, but whenever you issue an ODS command you are using a style definition. By default, ODS uses a standard style for each output destination. When you issue an ODS statement like:

```
ods pdf file='sample.pdf';
```

You are really issuing the following:

```
ods pdf file='sample.pdf' style=Printer;
```

So if you wish to switch to another style, all you have to do is add a STYLE= option and specify the name of a different style. However, the only choices you have are the standard styles shipped with your SAS software.

➢ PROC TEMPLATE

To have more control over the look of your output, you need to create your own style. This is done by using the TEMPLATE procedure. This procedure has statements that allow you to define every aspect of a style. However, if we had to specify every aspect of every new style, we would spend all of our time typing PROC TEMPLATE code. A complete style definition could run to hundreds of lines of code. To make our life easier, we have the PARENT statement. It allows a new style to be based on an existing style. Then you can add lines of code for only those things you want to change.

➢ THE EXAMPLE PROGRAM

Rather than try to explain all of the statements and syntax available for PROC TEMPLATE, let us just look at the example program (Appendix A). This program creates a new custom style. The first section of code sets up the name of the style (Custom) and indicates that it will be based on the Printer style.

```
proc template;
  define style Styles.Custom;
  parent = Styles.Printer;
```

The next section of code sets up a list of font names and assigns them characteristics. This list is used later in the program as a shorthand way to specify fonts.

```
replace fonts /
  'TitleFont' = ("Times Roman",13pt,Bold Italic)  /* Titles from TITLE statements */
  'TitleFont2' = ("Times Roman",12pt,Bold Italic) /* Proc titles ("The XX Procedure")*/
  'StrongFont' = ("Times Roman",10pt,Bold)
  'EmphasisFont' = ("Times Roman",10pt,Italic)
  'headingEmphasisFont' = ("Times Roman",11pt,Bold Italic)
  'headingFont' = ("Times Roman",11pt,Bold)       /* Table column and row headings */
  'docFont' = ("Times Roman",10pt)                /* Data in table cells */
  'footFont' = ("Times Roman",13pt)               /* Footnotes from FOOTNOTE statements */
  'FixedEmphasisFont' = ("Courier",9pt,Italic)
  'FixedStrongFont' = ("Courier",9pt,Bold)
  'FixedHeadingFont' = ("Courier",9pt,Bold)
  'BatchFixedFont' = ("Courier",6.7pt)
  'FixedFont' = ("Courier",9pt);
```

This style statement is used to supply attributes to the style element called “fonts”. By using the “replace” syntax, this code will overwrite the existing fonts style element. In this case, we are setting up 13 font names and their characteristics. See Appendix B for a reference on how and where each font name is used. Each attribute includes three characteristics in parentheses. Commas separate each characteristic. The first characteristic specified is the typeface. The next characteristic is the font size. The final characteristic is the font weight.

The next section of code is very similar to the fonts style element. Instead of a list of font names, this one is a list of font colors. In this case, a replace statement is again used since we are going to replace the entire list. The cryptic color names like ‘fg’ and ‘bg’ are used by the style definition to apply these colors to various parts of the output.

```
replace color_list /
  'link' = blue             /* links */
  'bgH' = grayBB            /* row and column header background */
  'bgT' = white             /* table background */
  'bgD' = white             /* data cell background */
  'fg' = black              /* text color */
  'bg' = white;             /* page background color */
```

The next section of code sets up the style element that controls rules, borders, background color and spacing for all tables. Since virtually all ODS output is in the form of tables, this is an important style element.

```
replace Table from Output /
  frame = box          /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = all          /* internal borders: none, all, cols, rows, groups */
  cellpadding = 4pt    /* the space between table cell contents and the cell border */
  cellspacing = 0.25pt    /* the space between table cells, allows background to show */
  borderwidth = 0.75pt    /* the width of the borders and rules */
  background = color_list('bgT') /* table background color */;
```

The next section of code will not be covered in this workshop. It sets up some additional font and color characteristics that we will not be modifying. In addition to the style elements included in the sample, there are dozens of other style elements that are “included” in our style. Those elements are part of the Printer style, and are included by way of the PARENT statement at the beginning of our PROC TEMPLATE. (If you would like to see the full Printer style, issue a PROC TEMPLATE with a single statement: “source styles.Printer;” and a RUN. This will dump the full definition to the log. For the purposes of this workshop, you do not need to understand the last section of code, or the code in the Printer style. We are just going to work with the fonts, color_list, and Table style elements.

At the end of the example PROC TEMPLATE are two more lines of code. These end the style definition that began with the DEFINE STYLE statement, and run the procedure.

```
  end;
run;
```

After the PROC TEMPLATE, the example program includes some code to run a sample procedure so we can see what our style looks like. This code starts with some options settings.

```
options nodate nonumber papersize="4x6 Card" orientation=landscape;
ods noptitle;
```

The OPTIONS statement gets rid of dates and page numbers, and reduces and rotates the page to make it easier to view the whole page in the results viewer. The ODS NOPTITLE statement turns off the standard procedure titles ("The FREQ Procedure") so they do not clutter up our output.

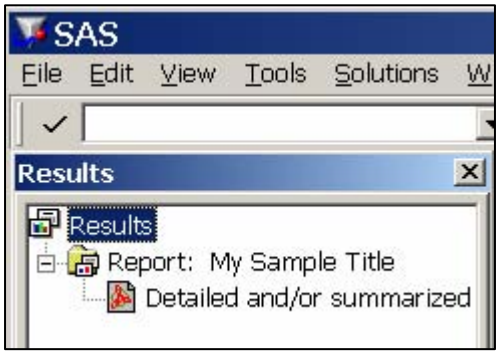
The remaining lines of example code are a simple PROC REPORT, and the ODS statements needed to create PDF output. This same code will work for HTML or RTF output as well, with a simple change to the ODS calls before and after the PROC REPORT.

```
ods pdf file='c:\sample.pdf' style=Custom;
  title 'My Sample Title';
  footnote 'My Sample Footnote';
  proc report data=sashelp.class nowd;
     column age height weight;
     define age / group;
     define height / mean f=8.;
     define weight / mean f=8.;
  run;
ods pdf close;
```

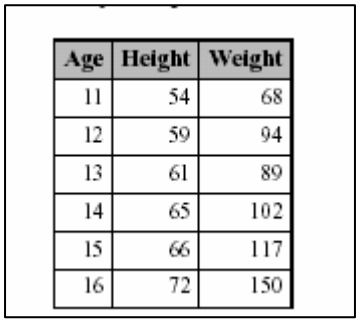
The only important thing to note here is the style=Custom option on the ODS statement. This calls our newly created style and applies it to the results. That's it for the sample program. It is a very simple example of customizing a style, but it can be very powerful, as you will see later.

➢ RUNNING THE EXAMPLE PROGRAM

Before going any further, try running this sample program. Use the Results Explorer to view the output.



The PDF file opens in the Results Viewer from within your SAS session. SAS opens an Adobe Acrobat Reader session from within the Display Manager, allowing you to view the results without switching programs.

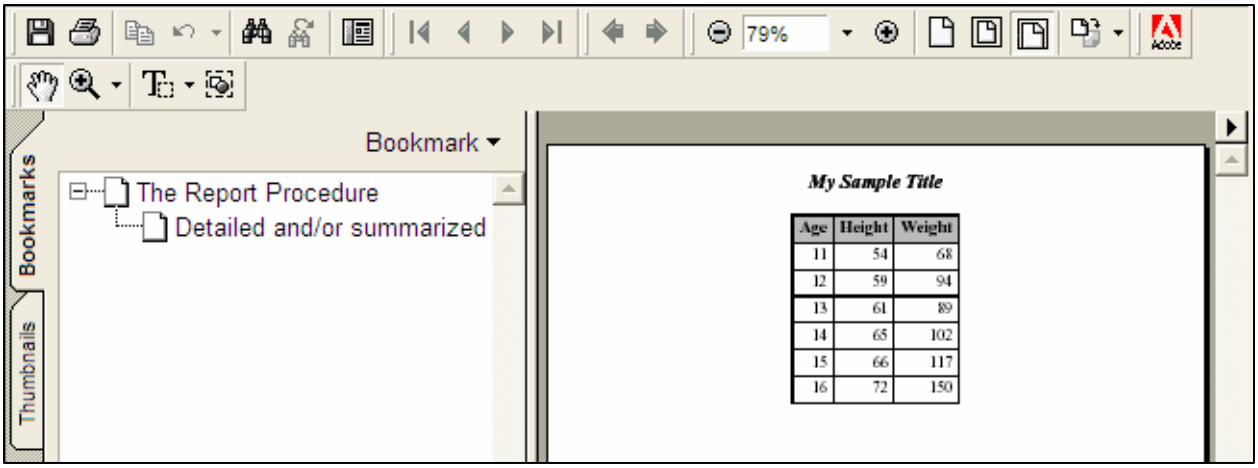


Age	Height	Weight
11	54	68
12	59	94
13	61	89
14	65	102
15	66	117
16	72	150

If you have created PDF output before, you will realize that right now the Custom style does not look very different from the default Printer style. The remainder of this workshop will be devoted to customizing the style.

➢ FIXING THE TABLE OF CONTENTS

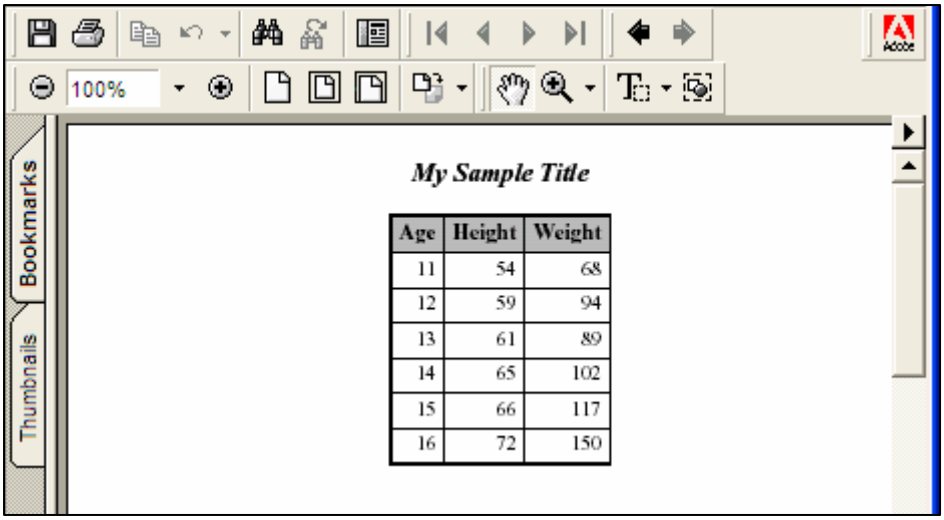
When you view your PDF output in Acrobat, you may have noticed that two panels appear. The one on the left shows a table of contents that identifies the bookmarks in the document. The one on the right shows your output. The default bookmarks are created by ODS and are not very user-friendly. They use the standard SAS procedure names and labels. It is possible to customize the table of contents using the ODS PROCLABEL statement and the CONTENTS= option on PROC REPORT, but that is a topic for another paper. For now, we will just turn it off so that the document opens with the output alone in a single window.



To get rid of the table of contents, add the option NOTOC to the ODS PDF statement as follows:

```
ods pdf file='c:\sample.pdf' style=Custom notoc;
```

Rerun the code after making this change. Now you will see that the new version comes up with just the main output window showing. Even if you were to click on the bookmarks tab, there would be no bookmarks to show.



➢ CHANGING THE TYPEFACES

The next thing we will learn how to modify is the typefaces. We will be working with the fonts style element. To change the font in part of your output, all you have to do is use PROC TEMPLATE to modify the font definition that applies to that part of your output. Appendix B lists each of the font names and where they apply.

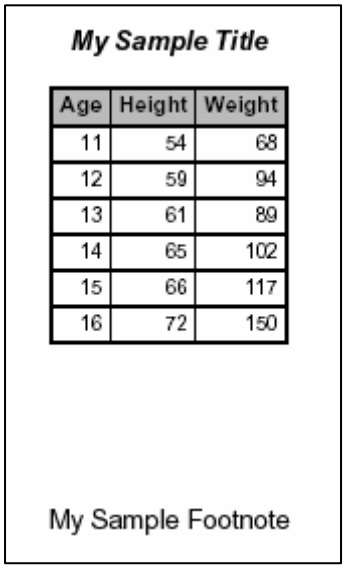
For each font name, we can modify three characteristics. The first is the typeface. To make a change, simply replace the typefaces listed between quotes with typefaces of your choice. If you are using SAS version 8.x, keep in mind that the person receiving your output will need to have the same fonts in order to view the PDF file properly. If you are

using version 9.x (which embeds the fonts) or if you are delivering only hard-copy output, then you can use any font available on your system. If you need to pick fonts that are commonly available, Appendix C lists some fonts that tend to be available on most Windows systems.

Using the sample program, try changing the typefaces used in the fonts style element. View the PDF file again to see how the change affected the output. A sample modification:

```
replace fonts /
  'TitleFont' = ("Arial",13pt,Bold Italic)   /* Titles from TITLE statements */
  'TitleFont2' = ("Arial",12pt,Bold Italic) /* Proc titles ("The XX Procedure")*/
  'StrongFont' = ("Arial",10pt,Bold)
  'EmphasisFont' = ("Arial",10pt,Italic)
  'headingEmphasisFont' = ("Arial",11pt,Bold Italic)
  'headingFont' = ("Arial",11pt,Bold)        /* Table column and row headings */
  'docFont' = ("Arial",10pt)                 /* Data in table cells */
  'footFont' = ("Arial",13pt)                /* Footnotes from FOOTNOTE statements */
  'FixedEmphasisFont' = ("Courier",9pt,Italic)
  'FixedStrongFont' = ("Courier",9pt,Bold)
  'FixedHeadingFont' = ("Courier",9pt,Bold)
  'BatchFixedFont' = ("Courier",6.7pt)
  'FixedFont' = ("Courier",9pt);
```

The resulting output:



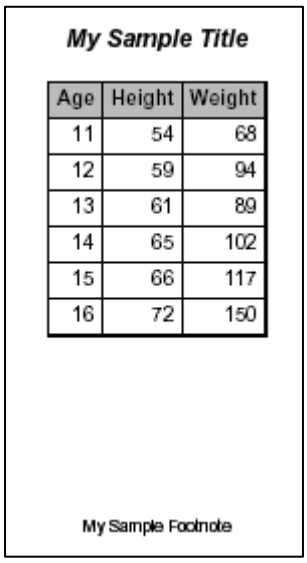
➢ CHANGING THE FONT SIZES

Now that we have the typefaces we want, we can turn to the font sizes. The default PDF output from ODS uses font sizes ranging from 10 point to 13 point. The default PDF style uses the same font specification for titles and footnotes. Our example style uses a different setting for each, so that you can have large titles and small footnotes. Otherwise, the example style has been set up with fonts similar to those in the default PDF style.

Try making some of the fonts larger or smaller and see how this affects the output. As a sample modification, the code below reduces the titles to 12 points, and the footnotes to 8 points. In addition, the emphasized heading fonts are reduced to 10 points.

```
replace fonts /
  'TitleFont' = ("Arial",12pt,Bold Italic)   /* Titles from TITLE statements */
  'TitleFont2' = ("Arial",12pt,Bold Italic) /* Proc titles ("The XX Procedure")*/
  'StrongFont' = ("Arial",10pt,Bold)
  'EmphasisFont' = ("Arial",10pt,Italic)
  'headingEmphasisFont' = ("Arial",10pt,Bold Italic)
  'headingFont' = ("Arial",10pt,Bold)        /* Table column and row headings */
  'docFont' = ("Arial",10pt)                 /* Data in table cells */
  'footFont' = ("Arial",8pt)                /* Footnotes from FOOTNOTE statements */
  'FixedEmphasisFont' = ("Courier",9pt,Italic)
  'FixedStrongFont' = ("Courier",9pt,Bold)
  'FixedHeadingFont' = ("Courier",9pt,Bold)
  'BatchFixedFont' = ("Courier",6.7pt)
  'FixedFont' = ("Courier",9pt);
```

The resulting output:



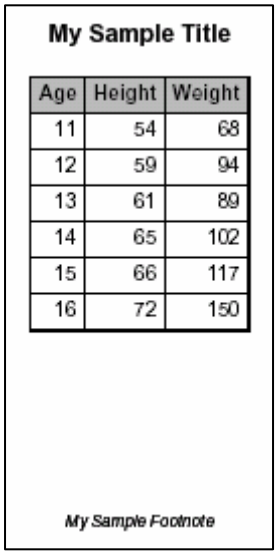
➢ CHANGING THE WEIGHTS AND STYLES

In addition to typefaces and font sizes, you can also modify the font weight (Medium, Bold, Light) and the font style (Italic, Roman, Slant). You may also be able to control the font width, though few fonts honor settings like Compressed or Expanded. To use any of these settings, just list the appropriate keyword(s) after the font size specification. Generally, the only two settings that you will want to add are Bold and/or Italic. If you leave this setting blank, the fonts are set to Medium Roman.

Try changing some of these settings to see what happens. A sample modification is shown below. This code removes the italics from the titles, and adds italics to the footnotes. Notice how an additional comma was needed in order to add the Italic style to the footnote font specification.

```
replace fonts /
  'TitleFont' = ("Arial",12pt,Bold)  /* Titles from TITLE statements */
  'TitleFont2' = ("Arial",12pt,Bold) /* Proc titles ("The XX Procedure")*/
  'StrongFont' = ("Arial",10pt,Bold)
  'EmphasisFont' = ("Arial",10pt,Italic)
  'headingEmphasisFont' = ("Arial",10pt,Bold Italic)
  'headingFont' = ("Arial",10pt,Bold)       /* Table column and row headings */
  'docFont' = ("Arial",10pt)                /* Data in table cells */
  'footFont' = ("Arial",8pt, Italic)              /* Footnotes from FOOTNOTE statements */
  'FixedEmphasisFont' = ("Courier",9pt,Italic)
  'FixedStrongFont' = ("Courier",9pt,Bold)
  'FixedHeadingFont' = ("Courier",9pt,Bold)
  'BatchFixedFont' = ("Courier",6.7pt)
  'FixedFont' = ("Courier",9pt);
```

The resulting output:



➢ CHANGING THE TABLE RULES AND BORDERS

The next thing we will modify is the table rules and borders. The lines around the table and between rows and columns within the table are controlled by two style attributes: frame and rules.

The frame attribute specifies whether there will be any borders around the outside of your tables. The frame is currently set to box, which generates a border around the entire table. Another setting to try is void, which removes all of the borders around the table. There are also settings that let you have borders on the top and bottom, on both sides, or on any individual edge.

Try changing the frame setting. Do not worry about the line width right now; we will get to that later. A sample modification:

```
replace Table from Output /
  frame = hsides       /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = all          /* internal borders: none, all, cols, rows, groups */
  cellpadding = 4pt    /* the space between table cell contents and the cell border */
  cellspacing = 0.25pt    /* the space between table cells, allows background to show */
  borderwidth = 0.75pt    /* the width of the borders and rules */;
```

The resulting output is below. Notice that now the frame is only at the top and bottom of the table. The sides have no borders.

My Sample Title

Age	Height	Weight
11	54	68
12	59	94
13	61	89
14	65	102
15	66	117
16	72	150

My Sample Footnote

The rules attribute controls the lines that appear inside your tables. This attribute is currently set to none, so there are no lines at all. Other settings to try are all and groups. All turns on all possible lines, creating a table grid. Groups puts a border between row and column headers and footers and the rest of the table body. Other settings include rows and cols, which include only row dividers or column dividers.

Try changing the rules setting. You may also want to experiment with combinations of frame and rules settings. A sample modification:

```
replace Table from Output /
  frame = hsides       /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = groups          /* internal borders: none, all, cols, rows, groups */
  cellpadding = 4pt   /* the space between table cell contents and the cell border */
  cellspacing = 0.25pt    /* the space between table cells, allows background to show */
  borderwidth = 0.75pt    /* the width of the borders and rules */;
```

The resulting output:

My Sample Title

Age	Height	Weight
11	54	68
12	59	94
13	61	89
14	65	102
15	66	117
16	72	150

My Sample Footnote

Now that you have all of the lines you want, we can look at the width for those lines. This is controlled by the borderwidth attribute. Changing this setting will not have any effect unless you have specified some lines for your table. With frame=void and rules=none, the border width is irrelevant.

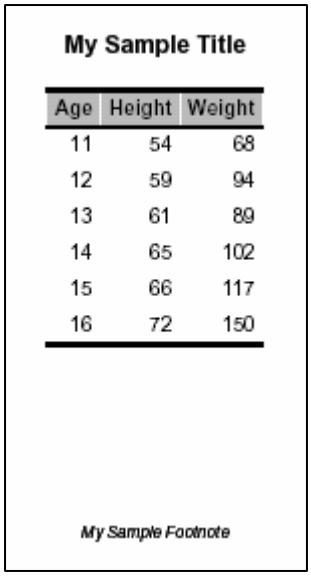
Borderwidth is simply the line width. It affects the width of the table border, but not the rules. Use a number followed by “pt” to set the width in points. Try experimenting with the borderwidth setting.

If your custom style does not have any lines, go ahead and turn on frame=box and rules=all so that you can at least see how it works. You can reset frame and rules later.

A sample modification:

```
replace Table from Output /
  frame = hsides       /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = groups          /* internal borders: none, all, cols, rows, groups */
  cellpadding = 4pt    /* the space between table cell contents and the cell border */
  cellspacing = 0.25pt    /* the space between table cells, allows background to show */
  borderwidth = 2pt    /* the width of the borders and rules */;
```

The resulting output:



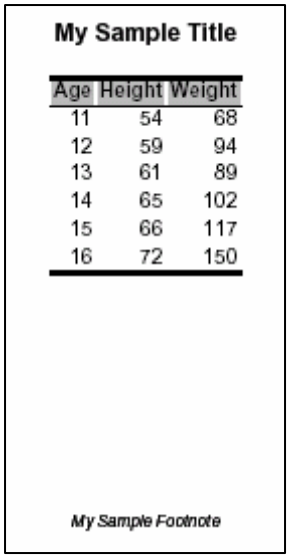
➢ CHANGING THE TABLE SPACING

The final thing we will modify is the table spacing. There are two aspects of table spacing that are controlled by the style template. Cell padding and cell spacing.

Cell padding is the amount of space that is left between table cell contents (your results) and the top, bottom, left, and right sides of the cell. To make your table readable, you want a large value. However, to squeeze more information on the page, you probably want a smaller value. The attribute that controls this spacing is called cellpadding. The example program uses a value of 4. Experiment with various values to see what you like. One thing to note here is that you have to have the same amount of space on all sides. You cannot add more space left and right and less space above and below. A sample modification:

```
replace Table from Output /
  frame = hsides       /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = groups          /* internal borders: none, all, cols, rows, groups */
  cellpadding = 2pt    /* the space between table cell contents and the cell border */
  cellspacing = 0.25pt    /* the space between table cells, allows background to show */
  borderwidth = 2pt    /* the width of the borders and rules */;
```

The resulting output is shown below. If you look closely, you can see the change from the previous output.



My Sample Title

Age	Height	Weight
11	54	68
12	59	94
13	61	89
14	65	102
15	66	117
16	72	150

My Sample Footnote

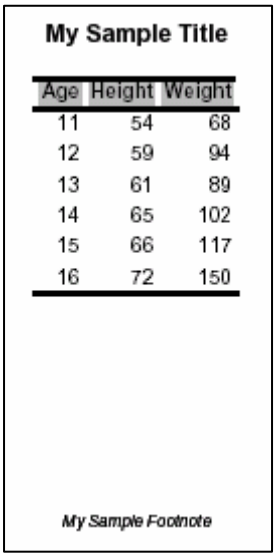
The cellpadding setting controls the space inside the cells. There is also an attribute that controls the space between the table cells. This attribute is called cellspacing.

If you look closely at the header of the sample table, you can see the cell spacing. There are narrow white lines between the column headers. These are the result of a cellspacing setting of 0.25 inches, which allows a quarter inch slice of the white background to show through. There is a similar spacing between the cells of the table body, but this space cannot be seen because the background color for the page and for the table cells is the same – white. By setting the cellspacing to zero, these white "lines" will go away. Alternatively, it is possible to widen the cellspacing so that more of the white shows through.

The following example widens the cellspacing:

```
replace Table from Output /
  frame = hsides       /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
  rules = groups          /* internal borders: none, all, cols, rows, groups */
  cellpadding = 2pt    /* the space between table cell contents and the cell border */
  cellspacing = 2pt    /* the space between table cells, allows background to show */
  borderwidth = 2pt    /* the width of the borders and rules */;
```

The resulting output is shown below. Now, there are significant white spaces showing through in the header.



➢ CHANGING THE COLORS

Changes to the fonts and lines are subtle. This next section lets you make big bold changes to your output. This section considers the color scheme.

ODS allows you to set the foreground (text) colors and background colors of every part of your output. These colors are set by defining a color scheme in the color_list style element.

In the example program, each color is identified by name. Appendix D lists the color names you can use. This gives you a palette of 216 colors.

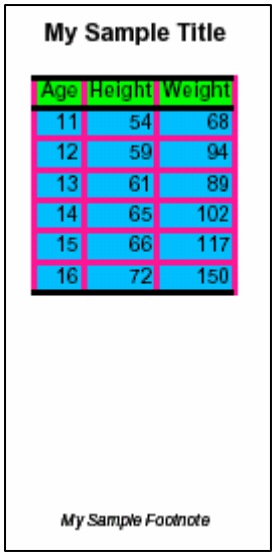
You also have the option of specifying custom colors by using their RGB values given in hexadecimal. For example, white would be cxFFFFFF, and black would be cx000000 (the “cx” tells SAS that the following value is a hexadecimal color). For the purposes of this workshop, we will stick to the named colors.

When you modify the colors, notice that some of the names start in “fg” and represent foreground colors. Others start in “bg” and represent background colors. These colors work in pairs, and you need to be sure that you pick pairs of colors that will be readable. For example, pink foreground text on a red background would be a problem.

Try creating a new color scheme. See how it looks. A sample modification:

```
replace color_list /
  'link' = blue              /* links */
  'bgH' = lime               /* row and column header background */
  'bgT' = deeppink           /* table background */
  'bgD' = deepskyblue        /* data cell background */
  'fg' = black               /* text color */
  'bg' = white;              /* page background color */;
```

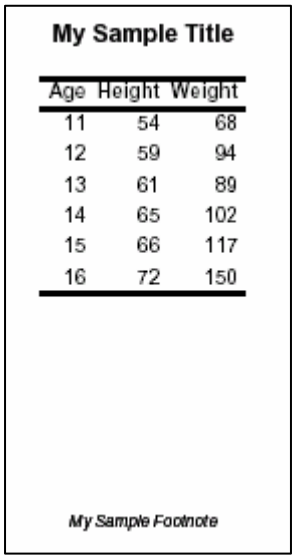
The resulting output:



The previous example uses color. If your company primarily produces black and white output, then you may wish to limit your choices to black, white, and various shades of gray. Here's another sample modification:

```
replace color_list /
  'link' = blue              /* links */
  'bgH' = white              /* row and column header background */
  'bgT' = white              /* table background */
  'bgD' = white              /* data cell background */
  'fg' = black               /* text color */
  'bg' = white;              /* page background color */;
```

The resulting output:



If you are not very creative, there is a web site that will help you design an attractive color scheme. Go to http://www.colorschemer.com/online/ and click on a color that you like. The web site will generate a group of 16 related colors that create an attractive color scheme. You can then copy down the hex codes for these colors and use them in your style. Another way to pick colors for your scheme is to use colors from your corporate logo. Ask your graphics department for the correct color codes. They should be able to give you the RGB values (you can find an RGB/hex converter on the web).

➢ PDF TWEAKS: PAGE NUMBERS

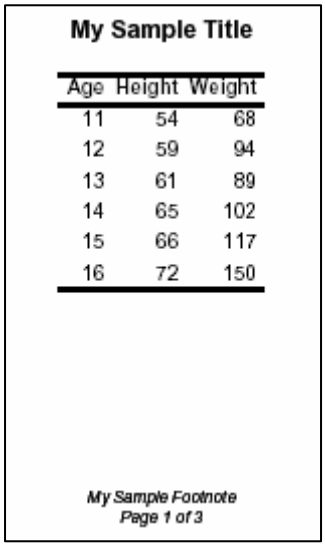
SAS has always supported putting page numbers on ODS output. However, there was little control over the format of the numbers. The only option was a single page number on the top right corner of the page. However, many users like to use numbering in the format “Page X of Y” so that their customers can be sure they have all of the pages. In addition, it is often preferable to have the page numbers at the bottom of the page, rather than the top.

With version 9.1, new features have been added to ODS to support both of these functionalities. The following code in PROC TEMPLATE will add centered page numbers in “Page X of Y” format at the bottom of the page:

```
Style PageNo from TitlesAndFooters /
   font = fonts("footFont")
   cellpadding = 0
   cellspacing = 0
   pretext = "Page "
   posttext = " of ^{lastpage}"
   just=c
   vjust=b;
```

Note: this example assumes that the statement ODS ESCAPECHAR=”^”; has also been added to the program before the PROC TEMPLATE call. The “^{lastpage}” is an in-line formatting command, and relies upon the ESCAPECHAR setting.

To test this out, copy the PROC REPORT code a couple of times so that there are three PROC REPORT calls between the ODS PDF statements. This will create three pages of output, so you can see the effect of the “Page X of Y” numbering. The revised output:

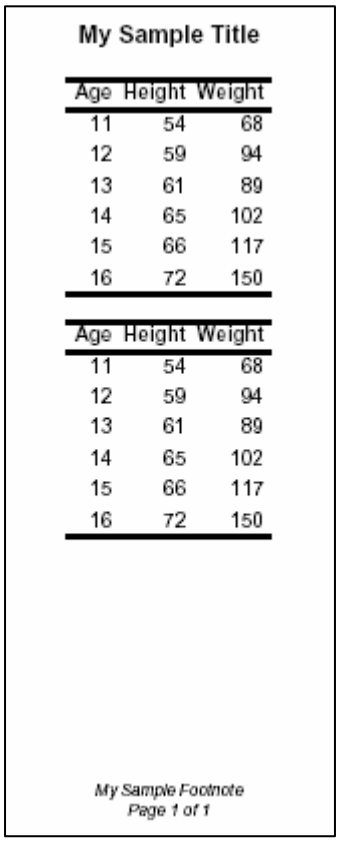


PDF TWEAKS: PAGE BREAKS

By default, ODS puts a page break after each procedure's output. If you run two PROC REPORTs, the RTF file will have the first table, then a page break, and then the second table. For small tables, this can waste a lot of space. You can prevent the page breaks from being generated by adding a STARTPAGE option. The syntax is:

```
ods pdf startpage=never;
```

This code should be added before the two PROC REPORT calls. You can try this in the example program by copying the PROC REPORT code so it runs twice, and then inserting the STARTPAGE call before the two. The new output will have the two tables on the same page.



My Sample Title

Age	Height	Weight
11	54	68
12	59	94
13	61	89
14	65	102
15	66	117
16	72	150

Age	Height	Weight
11	54	68
12	59	94
13	61	89
14	65	102
15	66	117
16	72	150

My Sample Footnote
Page 1 of 1

To turn page breaks back on, you can issue another statement with STARTPAGE=YES. Alternatively, to insert a single page break but keep the automatic breaks turned off, use STARTPAGE=NOW.

➢ SAVING YOUR STYLE

Once you have created your custom style, you can save the program that creates the style. This will allow you to regenerate it at any time. However, you do not need to run this PROC TEMPLATE every time you want to use your new style. SAS has saved the style for you in the sasuser library.

If this style is for you alone, this will work just fine. If you want to share your style, you will need to make a couple of changes. First, set up a libname for your custom style in a commonly accessible area. Then, you will need to learn about the ODS PATH statement, which you can use to route your custom style to this libname. Other users can set up the same ODS PATH statement in their programs to reference this libname and access your style.

CONCLUSIONS

This workshop has been a short cut to using some basic style functionality. If you just need make a quick and simple style modification, this may be enough.

However, this template only allows you to modify certain aspects of your output. You may find that you want to control other aspects. To do that, you are going to have to learn more about PROC TEMPLATE syntax.

➢ RESOURCES

PROC TEMPLATE documentation is in the References chapter of:

Guide to the Output Delivery System in (version 8.2 online documentation) http://v8doc.sas.com/sashtml/

SAS Output Delivery System Users Guide (version 9.1.3 online documentation) http://support.sas.com/onlinedoc/913/docMainpage.jsp

A CD of additional sample styles you can modify to meet your needs:

Bernadette Johnson*, Instant ODS: Style Templates for the SAS Output Delivery System*, © 2003, BBU Press, Cary, NC, USA.

Preliminary documentation of new features and sample programs can be found at:

http://www.sas.com/rnd/base/index-ods-resources.html.

Web site for downloading Acrobat Reader:

http://www.adobe.com/products/acrobat/readstep2.html

My book on ODS has a number of chapters on modifying ODS styles:

Haworth, Lauren, *Output Delivery System: The Basics*, ©2001, SAS Institute Inc., Cary, NC, USA.

➢ ACKNOWLEDGEMENTS

➢ CONTACTING THE AUTHOR

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

APPENDIX A

```
proc template;
  define style Styles.Custom;
  parent = Styles.Printer;

     replace fonts /
         'TitleFont' = ("Times Roman",13pt,Bold Italic)  /* Titles from TITLE statements */
         'TitleFont2' = ("Times Roman",12pt,Bold Italic) /* Proc titles ("The XX Procedure")*/
         'StrongFont' = ("Times Roman",10pt,Bold)
         'EmphasisFont' = ("Times Roman",10pt,Italic)
         'headingEmphasisFont' = ("Times Roman",11pt,Bold Italic)
         'headingFont' = ("Times Roman",11pt,Bold)       /* Table column and row headings */
         'docFont' = ("Times Roman",10pt)                /* Data in table cells */
         'footFont' = ("Times Roman",13pt)               /* Footnotes from FOOTNOTE statements */
         'FixedEmphasisFont' = ("Courier",9pt,Italic)
         'FixedStrongFont' = ("Courier",9pt,Bold)
         'FixedHeadingFont' = ("Courier",9pt,Bold)
         'BatchFixedFont' = ("Courier",6.7pt)
         'FixedFont' = ("Courier",9pt);

     replace color_list /
         'link' = blue              /* links */
         'bgH' = grayBB             /* row and column header background */
         'bgT' = white              /* table background */
         'bgD' = white              /* data cell background */
         'fg' = black               /* text color */
         'bg' = white;              /* page background color */;

     replace Table from Output /
         frame = box      /* outside borders: void, box, above/below, vsides/hsides, lhs/rhs */
         rules = all      /* internal borders: none, all, cols, rows, groups */
         cellpadding = 4pt   /* the space between table cell contents and the cell border */
         cellspacing = 0.25pt    /* the space between table cells, allows background to show */
         borderwidth = 0.75pt    /* the width of the borders and rules */;

     * Leave code below this line alone ;
     style SystemFooter from SystemFooter /
         font = fonts("footFont");
     style Data from Data /
         background=color_list('bgD');

  end;
run;

options nodate nonumber papersize="4x6 Card" orientation=landscape;
ods noptitle;

ods pdf file='c:\sample.pdf' style=Custom;
  title 'My Sample Title';
  footnote 'My Sample Footnote';
  proc report data=sashelp.class nowd;
     column age height weight;
     define age / group;
     define height / mean f=8.;
     define weight / mean f=8.;
  run;
ods pdf close;
```

APPENDIX B

Font Style	Portion of Output it Controls
TitleFont	Titles generated with TITLE statement
TitleFont2	Titles for procedures (“The _____ Procedure”)
StrongFont	Strong (more emphasized) table headings and footers, page numbers
EmphasisFont	Titles for table of contents and table of pages, emphasized table headings and footers
headingFont	Table column and row headings and footers, by-group headings
docFont	Data in table cells
footFont	Footnotes generated with FOOTNOTE statement

APPENDIX C

Fonts to Try
Times Roman
Arial
Arial Black
Book Antigua
Comic Sans MS
Verdana
Clarendon
Letter Gothic
Marigold
New Century Schoolbook
Palatino

APPENDIX D

White	Cornsilk	Antiquewhite	Seashell	Linen	Ivory	Floralwhite
Snow	Azure	Mintcream	Ghostwhite	Honeydew	Aliceblue	Beige
Oldlace	Bisque	Moccasin	Wheat	Navajowhite	Blanchedalmond	Tan
Gray	Lightgrey	Darkgray	Dimgray	Gainsboro	Silver	Whitesmoke
Black	**Darkslategray**	Slategray	Lightslategray	Lemonchiffon	Khaki	Darkkhaki
Brown	Sienna	Chocolate	Saddlebrown	Sandybrown	Burlywood	Peru
Red	Tomato	Darkred	Indianred	Mistyrose	Lavenderblush	Firebrick
Crimson	Maroon	Peachpuff	Goldenrod	Darkgoldenrod	Palegoldenrod	Lavender
Orange	Darkorange	Orangered	Forestgreen	Greenyellow	Lime	Lightgoldenrodyellow
Yellow	Lightyellow	Gold	Springgreen	Darkolivegreen	Olive	Limegreen
Green	Lightgreen	Darkgreen	Mediumseagreen	Mediumspringgreen	Palegreen	Olivedrab
Lawngreen	Chartreuse	Yellowgreen	Paleturquoise	Darkseagreen	Aquamarine	Mediumaquamarine
Teal	Lightseagreen	Seagreen	Darkblue	Mediumturquoise	Turquoise	Darkturquoise
Darkcyan	Cyan	Lightcyan	Mediumslateblue	Lightskyblue	Skyblue	Deepskyblue
Blue	Lightblue	Mediumblue	Steelblue	Darkslateblue	Powderblue	Cornflowerblue
Royalblue	Dodgerblue	Slateblue	Plum	Cadetblue	Mediumorchid	**Darkorchid**
Navy	Midnightblue	Lightsteelblue	Mediumvioletred	Orchid	Thistle	Rosybrown
Purple	Mediumpurple	Indigo	Fuchsia	Palevioletred	Magenta	Darkmagenta
Violet	Darkviolet	Blueviolet	Salmon	Deeppink	Coral	Lightcoral
Pink	Lightpink	Hotpink	Lightsalmon	Darksalmon		