

SAS5762-2016

That's All Right: More Complex Reports

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

ABSTRACT

Are you living in Heartbreak Hotel because your boss wants different statistics in the SAME column on your report? Need a currency symbol in one cell of your pre-summarized data, but a percent sign in another cell? Large blocks of text on your report have you all shook up because they wrap badly on your report? Have you hit the wall with PROC PRINT? Well, rock out of your jailhouse with ODS, DATA step, and PROC REPORT. This paper is a sequel to the popular 2008 paper "Creating Complex Reports."

The paper presents a nuts-and-bolts look at more complex report examples gleaned from SAS® Community Forum questions and questions from students. Examples will include use of DATA step manipulation to produce PROC REPORT and PROC SGPLOT output as well as examples of ODS LAYOUT and the new Report Writing Interface. And PROC TEMPLATE makes a special guest appearance. Even though the King of Rock 'n' Roll won't be there for the presentation, perhaps we'll hear his ghost say "Thank you very much, I always wanted to know how to do that" at the end of this presentation.

INTRODUCTION

ODS has been around even since SAS 7. And the main report writing procedures (PROC PRINT, PROC REPORT and PROC TABULATE) have been around even longer than that. So what is there new to learn? Well, ODS has not been static over the years. More features, more destinations (here's a shout out to ODS POWERPOINT and ODS EXCEL), new features like ODS LAYOUT and the Report Writing Interface have all been introduced between SAS 7 and SAS 9.4, so there is more than enough in the world of ODS and complex reporting to cover in this paper.

I view this paper as a sequel to my 2008 paper, "Creating Complex Reports" (<http://www2.sas.com/proceedings/forum2008/173-2008.pdf>). What I said in that paper is still true today: really complex reports frequently need some type of pre-processing or DATA step manipulation prior to the report step in order to produce the desired final report.

Before we rock into the first set of examples, I need to provide some qualifying information. This paper was written using SAS 9.4, maintenance release 3. This is important because some features of ODS, such as ODS Excel, were not production until SAS 9.4, maintenance release 3. In addition, ODS GRAPHICS capabilities have been continually enhanced since their introduction in SAS 9.2. Other features, such as ODS LAYOUT and the Report Writing Interface (RWI), were available in earlier releases of SAS but have been enhanced (they were pre-production). So if you download the ZIP file of programs for this paper, you will get the same results only if you use the same version of SAS.

Also, note that some features I show are destination specific. For example, the use of TAGATTR= with an Excel format will not work in ODS PDF or ODS RTF or ODS HTML destinations. ODS LAYOUT is designed to work with Printer family destinations such as ODS PDF, some of the LAYOUT commands will work with ODS HTML, but there are other destinations in which the code will not work.

Finally, there is always more than one way to accomplish something using SAS and ODS. So the examples offered in this paper are not the **only** way to achieve a complex report. They only represent my approach, based on feedback and questions from my students and from working with SAS customer questions.

The code in this paper is aimed at intermediate to advanced SAS programmers. But with enough study (and motivation), the programs are accessible to the beginner. Most of the techniques used in this paper for data manipulation use topics covered in our Programming 1 and Programming 2 classes. Many of the REPORT procedure examples build on topics covered in our Report Writing 1 class. Other topics are covered in papers that you'll find in the reference section at the end of the paper or in the documentation topics for ODS, ODS LAYOUT, ODS GRAPHICS, and the Report Writing Interface. To see the full code, download the ZIP file from support.sas.com at the URL listed at the end of the paper.

COMPLEX EXAMPLES 1: RUSH INTO NEW ODS DESTINATIONS

Sometimes you need a ZIP file that contains files that you know you can create with ODS, and you can't figure out whether SAS can make a ZIP file for you automatically. Let the ODS PACKAGE destination come to the rescue.

Oh, you didn't know about the ODS PACKAGE destination? It's been around for a while now, in various releases. This example was written using SAS 9.4. Basically, you are going to create each of your individual files the way you normally would. The code below used a standard PROC MEANS for the CSV file and a simple PROC REPORT for the RTF file. The "new" part of the program is the ODS PACKAGE set of statements.

Here's the code used to produce the ZIP file:

```
filename meancsv "&mypath./prdsale_mean.csv";
ods csv file=meancsv;
. . . PROC MEANS code . . .
ods csv close;

filename rp "&mypath./prdsale_report.rtf";
ods rtf(id=report) file=rp;
. . . PROC REPORT code . . .
ods rtf(id=report) close;

ods package(myzip) open nopf;
ods package(myzip) add file=meancsv ;
ods package(myzip) add file=rp;
ods package(myzip) publish archive
    properties (archive_name="prdsum.zip"
                archive_path="&mypath");
ods package(myzip) close;
```

The program uses a "helper" macro variable called &mypath. In the download of program code, you will see that there is a %LET statement that provides the path where the output files should all be written. This macro variable is used for the location of the CSV file, the RTF file, and the path of the ZIP archive.

The ODS PACKAGE statements are slightly different from the usual ODS "sandwich" technique. There are ODS PACKAGE OPEN and corresponding ODS PACKAGE CLOSE statements. The NOPF option in the OPEN statement instructs the destination manager to create the ZIP package without inserting any extra structure information in the file. If you have the Publishing Framework licensed, you might need this additional information in the ZIP archive. But for a simple output like this, the extra information is not necessary.

The way that ODS works with most destinations is that you can use an identifying string in parentheses after the destination name. You see this in the ODS RTF example with the use of the (id=report) suboption. If I wanted to create two RTF files simultaneously, using the ID= suboption would allow the ODS controller (also called an ODA – Output Delivery Agent) to keep both files open at once.

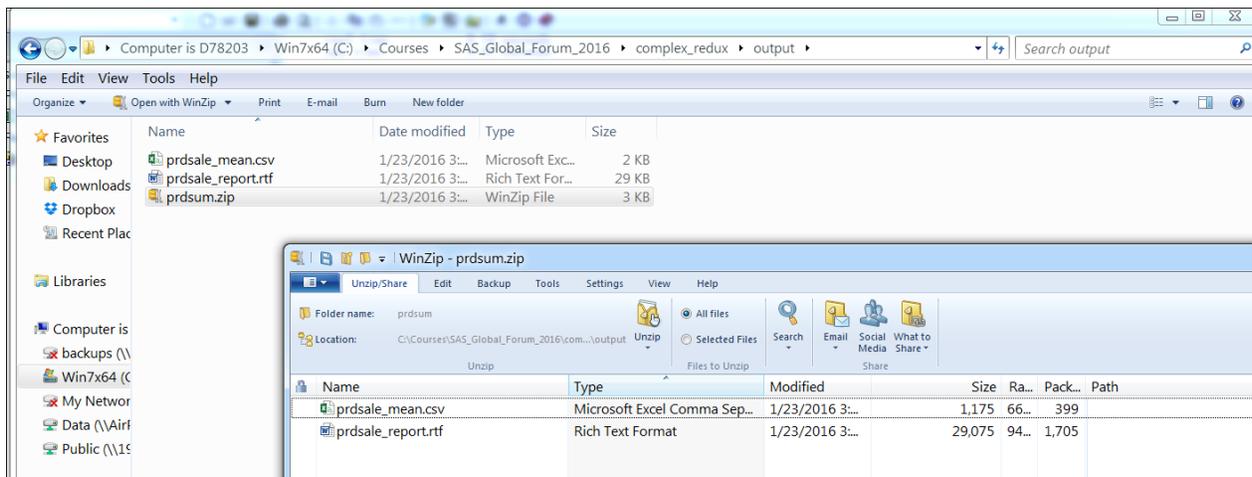
Using slightly different syntax, you can specify a name for your ZIP archive by just using the name in parentheses after the ODS PACKAGE invocation. You don't need to use a name, but it's probably a best practice to always name your ZIP file. And, like your Mom advised, "If you open it, you should close it"; if you do decide to use a name, then you should be consistent and close the same name that you opened. Note that in this example, the name (myzip) is used in each ODS PACKAGE statement. The screen shot of the Windows folder where I created the ZIP archive is shown in Output 1, with a view of the ZIP archive contents.

Also, note that the ZIP archive contains the files that were inserted with the ODS PACKAGE ADD statement. The file= option points to the file reference or fileref for each of the files. In the above example,

each ODS output file was defined in a FILENAME statement, but the direct file location could also be used in the ODS PACKAGE ADD statement, as shown below:

```
ods package(myzip) add file="&mypath./prdsale_mean.csv";
```

But no matter which form of the file path location is used, the output is still the same as that shown in Output 1.



Output 1. View of ZIP Archive Created with ODS PACKAGE

What if your Chief Marketing Officer needs a PowerPoint presentation based on SAS data? Can you do that with ODS? Can you do that if you don't have the SAS Add-in for Microsoft Office? The answer to both questions is yes. Starting with SAS 9.4, ODS PowerPoint destination was production. This destination creates a true PPTX (PowerPoint XML file) that conforms to the Microsoft Office Open XML format.

But, what if, to further complicate the requirements, your PowerPoint document doesn't just need the tables and graphs, it also needs a company logo on each slide. To get a logo on each PowerPoint slide, you basically need to use a style template to create a background for each slide that will "fill" the entire slide area. Your logo will be specified as the background image for the entire "body" of the slide.

In the simplest invocation, you use the ODS "sandwich" technique, as shown in the code below:

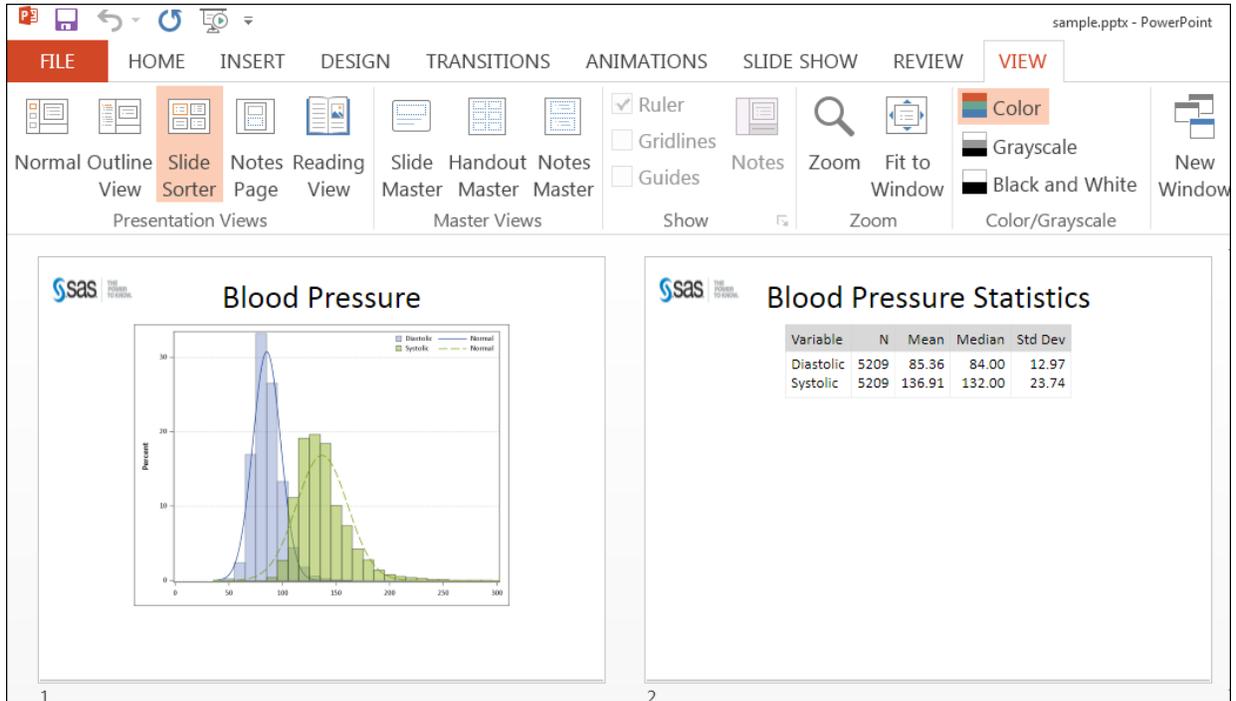
```
proc template;
  define style styles.logo;
    parent=styles.powerpointlight;
    class body /
      backgroundimage = "&mypath./saslogo_pptx.png";
  end;
run;

ods powerpoint file="&mypath./sample.pptx" style=styles.logo nogtitle;

title 'Blood Pressure';
. . . PROC SGPLOT step . . .
run;

ods noptitle;
title 'Blood Pressure Statistics';
. . . PROC MEANS step . . .
ods powerpoint close;
```

The new style template is based on the PowerPointLight style template and adds a background image specification to the BODY style element. The new template is specified on the ODS POWERPOINT statement. In addition, the NOGTITLE option tells ODS to put the title OUTSIDE of any graphics images, under the control of the destination (not under the control of the graphing method). The output from the above code is shown in Output 2.



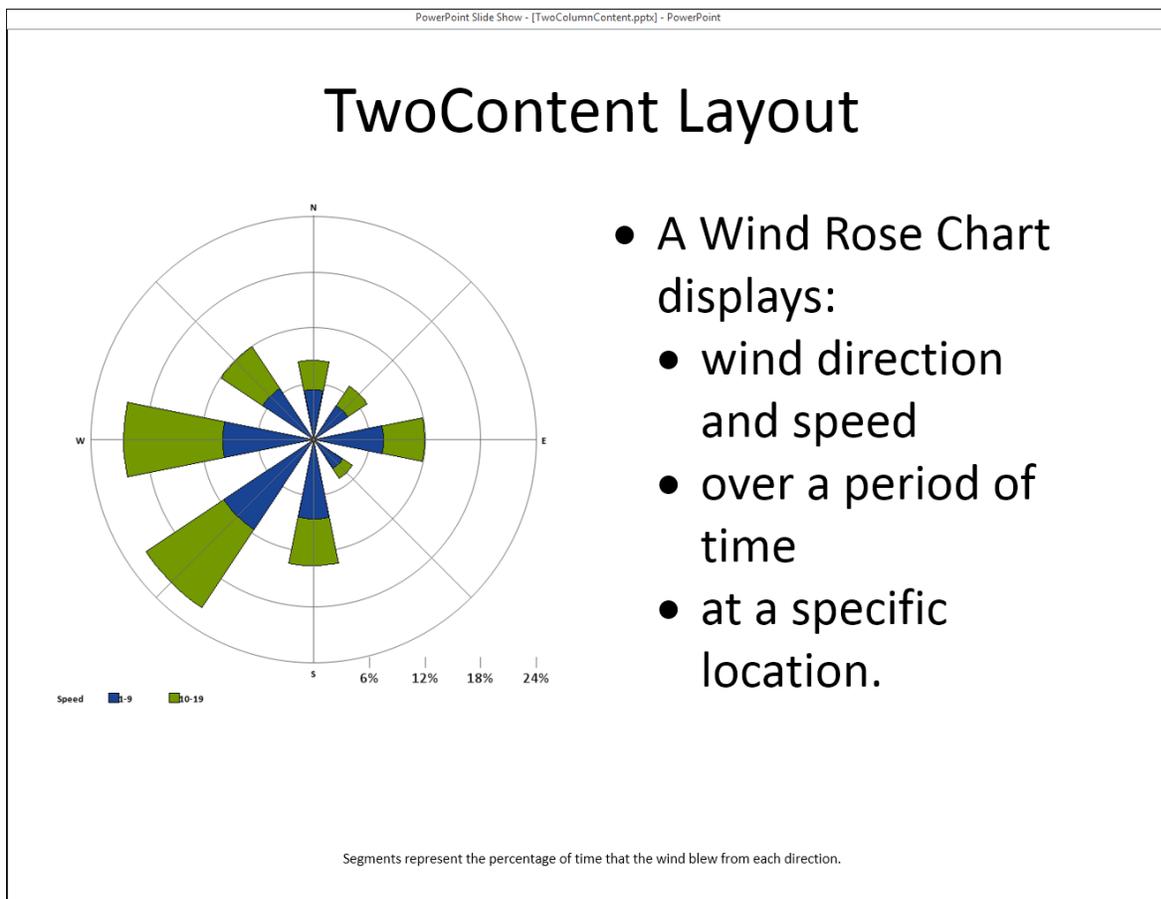
Output 2. PowerPoint Output, with Logo, Shown in SlideSorter View

ODS POWERPOINT can be used with PROC ODSLST to generate a slide with bullet points. New destinations aren't the only new thing about ODS. Two new procedures, PROC ODSLST and PROC ODSSTEXT, make report creation more flexible. You'll have to look at the ODS Tip Sheet and other resources to learn more about PROC ODSSTEXT because this example only uses PROC ODSLST. Here's the code used to produce the ZIP file:

```
ods powerpoint file="&mypath./TwoColumnContentx.pptx" nogtitle
  nogfootnote style=styles.powerpointlight layout=twocontent;

. . . PROC GRADAR step . . .

proc odslist;
  item 'A Wind Rose Chart ^{newline }displays: ';
  item;
  list / style=[bullet=disc];
  item 'wind direction and speed';
  item 'over a period of time';
  item 'at a specific location.';
end;
end;
run;
ods powerpoint close;
```



Output 3. PowerPoint Output, with Side-by-Side Objects, Shown in Reading View

With the LAYOUT=TWOCONTENT specified, the PROC GRADAR graphic image was placed in the left content area and the nested list from PROC ODSLST was placed in the right content area.

Last, but not least, no discussion of new ODS destinations would be complete without a shout out to ODS EXCEL, one of the newest and hottest of the ODS destinations. It provides two new, most desired features: 1) it allows tables and graphs in the same workbook/worksheet file; and 2) the format created conforms to the Microsoft Office Open XML format for Excel, which means the creation of a true XLSX format output file using Base SAS.

The challenge with using other ODS destinations to create output that Excel can open and render is that many of the outputs types (HTML or XML) also come with an annoying pop-up warning message when you open them with Excel, as shown in Figure 1.

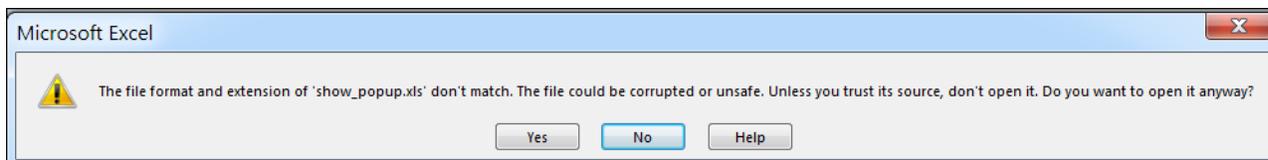


Figure 1. Excel Pop-up Window Warning Message

But with the ODS EXCEL destination, you don't have to worry about the popup warning window from Excel, as long as the appropriate extension is used. With the regular ODS "sandwich" approach, you can combine tables and graphs in one workbook, or, as shown in this example, in one worksheet. The results

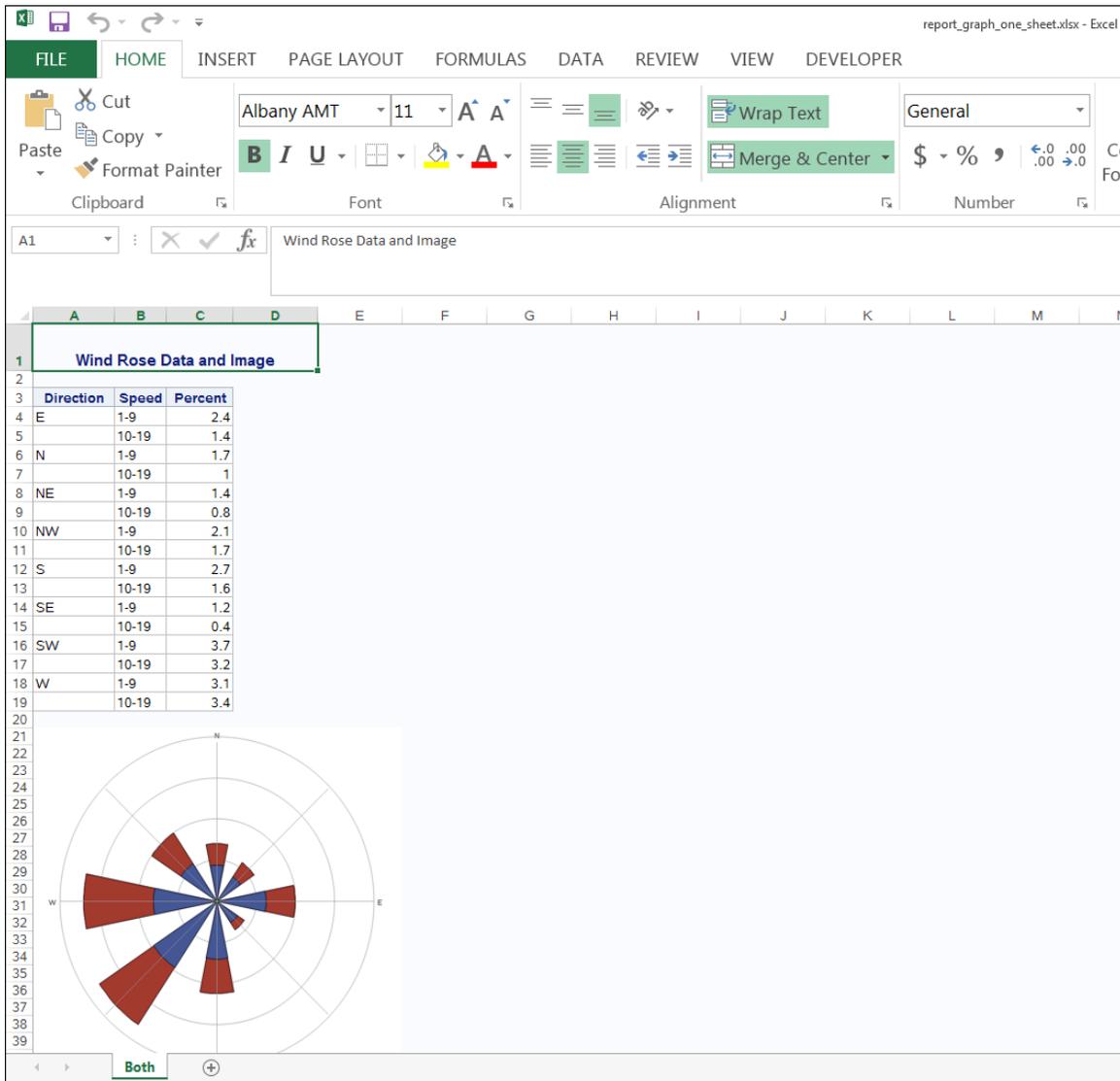
in Output 4 used the same data as the PowerPoint example. The ODS EXCEL statements surround PROC REPORT and PROC GRADAR steps as shown below:

```
ods excel file="&mypath./report_graph_one_sheet.xlsx" style=htmlblue
  options (sheet_interval='none' embedded_titles='on' sheet_name='Both');

. . . PROC REPORT step . . .

. . . PROC GRADAR step . . .

ods excel close;
title; ;
```



Output 4. ODS EXCEL Output with Table and Graph in One Worksheet

The key to making this output work is the SHEET_INTERVAL='NONE' suboption, which instructs ODS that both outputs go into one worksheet.

COMPLEX EXAMPLE 2: CAN'T HELP LOVING PROC REPORT

If Elvis had been a SAS programmer, and responsible for report writing, he would not sing the blues if he used PROC REPORT. As illustrated in my earlier complex reports paper, with some pre-processing of the data, this demographic report can be easily produced using PROC REPORT.

Using PROC REPORT	
<u>Patient Characteristics</u>	
<u>(N= 1,049 Patients)</u>	
<u>Patient Demographics</u>	
Age	69.80 ± 11.409 (32 - 88)
Gender	
Female	473 (45.09)
Male	576 (54.91)
LVEF	32.66 ± 17.762 (5 - 65)
CAD(Required)	
No	283 (26.98)
Yes	766 (73.02)
MI Status	
Without MI	367 (34.99)
With MI	682 (65.01)
Hypertension Status	
No Hypertension	549 (52.34)
Hypertension	500 (47.66)
NYHA Functional Class	
I	376 (35.84)
II	234 (22.31)
III	158 (15.06)
IV	149 (14.20)
UNK	132 (12.58)

Figure 2. Demographic Report Example from 2008 Complex Reports Paper

The 2008 program that created Figure 2 is included in the ZIP file of programs for this paper. Some of the techniques used in this paper should be part of your standard ODS toolkit. In particular, highlights of this code use:

- STYLE=JOURNAL for the overall RTF style
- RTF control strings for the underlined text
- Multiple STYLE= overrides to alter the font, based on “helper” variables created when pre-processing the data
- STYLE= override on the first column for LEFTMARGIN=12pt for the indented text underneath the bold headers
- Macro “Helper” variables for the N= value
- Concatenating several numeric values into one character string for the second column’s values

When you have a truly complex report, you often find that pre-processing the data or making “helper” variables that you hide on the report (with the NOPRINT option) are how you accomplish your task.

But, other times, you will find that the CALL DEFINE statement used in a COMPUTE block will allow you to accomplish your report goals.

Consider data that you get emailed in a CSV file from someone. It is shown in the program below as DATALINES:

```
data diff_fmts;
  length facility var_a $20 var_b 8;
  infile datalines dlm=',' dsd;
  input fnum ordvar var_a $ var_b;
  Facility = catx(' ','Facility',fnum);
return;
datalines;
1,1,"Number of Admissions", 30
1,2,"Percent Covered", .6783
1,3,"Plan Amount",3400
2,1,"Number of Admissions", 50
2,2,"Percent Covered", .7337
2,3,"Plan Amount",5183
3,1,"Number of Admissions", 45
3,2,"Percent Covered", .8451
3,3,"Plan Amount",4325
;
run;
```

The data is pre-summarized, almost ready to go, except your goal is to make a PDF version of the report, an RTF version of the report, and an Excel version of the report. The Number of Admissions row needs to be formatted in the report with only commas if the number exceeds 999. This is no problem with the COMMA. format. Then the percent, of course, needs to be multiplied by 100 and have a trailing percent sign (%), which is also possible with the SAS format for percents (which does an automatic multiply by 100). Then, the Plan Amount needs to be shown with a currency format. As good SAS programmers, we are used to a FORMAT statement to accomplish this or a FORMAT option in the DEFINE statement in PROC REPORT. However, the numeric values all belong to the same variable, VAR_B, so here is where CALL DEFINE comes to the rescue.

Another way to accomplish this would be to make a character variable in the DATA step program and use that in the report instead of using the CALL DEFINE statement. While this would work for pre-summarized data, this approach would not work for detail data that needed to be summarized by PROC REPORT.

```
compute var_b;
  if var_a = 'Number of Admissions' then do;
    call define(_col_,'format','comma6.');
```

The PROC REPORT CALL DEFINE statement uses very predictable syntax:

```
CALL DEFINE(ARG1, ARG2, ARG3);
```

Where ARG1 is the report component you want to change; ARG2 is the attribute you want to change; and ARG3 is the specification for the attribute change. Refer to the PROC REPORT documentation for more information about how these values need to be specified. In the code snippet above, _COL_ identifies the column named in the COMPUTE statement (VAR_B) as the report component that will be changed. The quoted string 'FORMAT' identifies that the format attribute is what will change. Finally, the 3 different format specifications are provided in quoted strings. Because the COMPUTE block supports the use of IF

statements, and because VAR_A is to the left of VAR_B in the COLUMN statement, VAR_A can be used for the condition in the COMPUTE block.

The results of the PROC REPORT, shown in PDF and Excel spreadsheet form, indicate that for the most part, the SAS format was respected in both destinations for the Number of Admissions, the Percent Covered, and the Plan Amount as shown Output 5:

ODS PDF Output

Facility	Column A	Column B
Facility 1	Number of Admissions	30
	Percent Covered	67.83%
	Plan Amount	\$3,400
Facility 2	Number of Admissions	50
	Percent Covered	73.37%
	Plan Amount	\$5,183
Facility 3	Number of Admissions	45
	Percent Covered	84.51%
	Plan Amount	\$4,325

ODS TAGSETS.EXCELXP Output

Facility	Column A	Column B
Facility 1	Number of Admissions	30
	Percent Covered	67.83%
	Plan Amount	\$3,400.00
Facility 2	Number of Admissions	50
	Percent Covered	73.37%
	Plan Amount	\$5,183.00
Facility 3	Number of Admissions	45
	Percent Covered	84.51%
	Plan Amount	\$4,325.00

Output 5. ODS PDF Output and ODS TAGSETS.EXCELXP Output

But the DOLLAR8. format for Plan Amount specified in the Call DEFINE statement was used in the PDF destination, but not used in the ExcelXP destination (and won't be used in the ODS EXCEL destination either).

There is another way to send a format specification to Excel if the destination does not respect the SAS-defined format. The method is to use a Microsoft format in a style override. This method involves changing the TAGATTR style attribute.

```
compute var_b;
  if var_a = 'Number of Admissions' then do;
    call define(_col_, 'style', 'style={tagattr="format:###,##0"}');
  end;
  else if var_a = 'Percent Covered' then do;
    call define(_col_, 'style', 'style={tagattr="format:###0.00%"}');
  end;
  else if var_a = 'Plan Amount' then do;
    call define(_col_, 'style', 'style={tagattr="format:$###,##0"}');
  end;
endcomp;
```

Note how the first argument is the same (_COL_) but the second argument has changed to 'STYLE', which means that the third argument is the TAGATTR style override. With the TAGATTR override, this code can only be used for ODS EXCEL and ODS TAGSETS.EXCELXP destinations.

The use of TAGATTR for XML-based destinations and of HTMLSTYLE for HTML-based destinations was the topic of my 2011 paper entitled "Don't Gamble with Your Output: How to Use Microsoft Formats with ODS" (<https://support.sas.com/resources/papers/proceedings11/266-2011.pdf>). Basically, with TAGATTR or HTMLSTYLE format overrides, you have to have some idea of the way the format is created in Excel before you can provide the value you need for the STYLE override. The 2011 paper has an appendix that shows you how to reverse engineer the format specification in Excel to use in your TAGATTR (or HTMLSTYLE) style override. Output using TAGATTR for the CALL DEFINE is shown in Output 6.

ODS Excel Output

	A	B	C
1	Facility	Column A	Column B
2	Facility 1	Number of Admissions	30
3		Percent Covered	67.83%
4		Plan Amount	\$3,400
5	Facility 2	Number of Admissions	50
6		Percent Covered	73.37%
7		Plan Amount	\$5,183
8	Facility 3	Number of Admissions	45
9		Percent Covered	84.51%
10		Plan Amount	\$4,325
11			
12			

ODS TAGSETS.EXCELXP Output

	A	B	C
1	Facility	Column A	Column B
2	Facility 1	Number of Admissions	30
3		Percent Covered	67.83%
4		Plan Amount	\$3,400
5	Facility 2	Number of Admissions	50
6		Percent Covered	73.37%
7		Plan Amount	\$5,183
8	Facility 3	Number of Admissions	45
9		Percent Covered	84.51%
10		Plan Amount	\$4,325
11			

Output 6. ODS Excel Output and ODS TAGSETS.EXCELXP Output

The bottom line is that if you need different cells to have different formats, you can accomplish that with PROC REPORT and CALL DEFINE.

Another attribute you can change, other than STYLE or FORMAT, is the URL attribute. If, for example, you wanted to make a dynamic URL, you could do something like this in a COMPUTE block, which would make a URL dynamically from the values of COUNTRY and PRODTYPE:

```
compute prodtype;
  length uvar $75;
  uvar = catt('https://www.google.com/#q=',holdcountry,'+',prodtype);
  call define(_col_, 'url', uvar);
endcomp;
```

If the value of COUNTRY was Canada and the value of PRODTYPE was Furniture, then the ODS HTML destination would build an ANCHOR tag like this:

```
<a href="https://www.google.com/#q=Canada+Furniture">
```

Of course this URL uses www.google.com as the main part of the anchor tag HREF= value. But in a production environment, the main part of the URL would usually be an address on your company server, where the detail files would be loaded from. The ZIP file has a bonus program with a URL example that shows PROC REPORT generating a set of drill-down reports using CALL DEFINE in a COMPUTE block.

COMPLEX EXAMPLE 3: SGPLOT IS ALWAYS ON MY MIND

If you haven't used any SAS version higher than 9.1, you don't really know about the ODS Graphics procedures (SG procedures). Suffice it to say that ODS GRAPHICS and the SG procedures represent an exciting new way to produce graphic images with SAS.

One of the challenges with classic SAS/GRAPH was understanding the PATTERN, SYMBOL and other global statements and controlling them so that the same color was always used for the same bar or plot line. Through the use of data attribute maps, the SGPLOT procedure can provide this level of style control.

The style template can also be used for this task, by changing the style elements GDATA1 through GDATA12 and GCDATA1 through GCDATA12 for grouped data. The style template assigns the colors associated with GDATA1 to the first value in a group and then the second value uses the colors for GDATA2, etc. Although you can control the colors associated with those style elements, you cannot always control which of your data values will be considered in the first group and which in the second group (depending on the data and missing values and missing observations). So, if you want to always have particular presentation attributes (like fill color, line color, etc) used, then an ODS GRAPHICS attribute map will do the job.

If you have ever created a CNTLIN dataset for PROC FORMAT, then you are halfway to understanding how attribute maps work with ODS GRAPHICS. In order to use an attribute map with ODS GRAPHICS,

you need to make a SAS dataset with variable names that conform to a specific naming convention. The ODS GRAPHICS documentation outlines all the possible variables that you could specify. In our fake data, we need to ensure that the value for the TYPE variable is illustrated with the right colors based on the corporate color specification. The PinkLady brand is always shown on charts with the approved shade of pink (CXd98cb3) and the VerdantIvy brand has an approved shade of green (CX2e852e). In this case, an attribute map will ensure the correct colors:

```
data myattrmap;
  length linecolor $ 9 fillcolor $ 9 value $15;
  input ID $ value $ linecolor $ fillcolor $;
datalines;
myid PinkLady CXd98cb3 CXd98cb3
myid VerdantIvy CX2e852e CX2e852e
;
run;
```

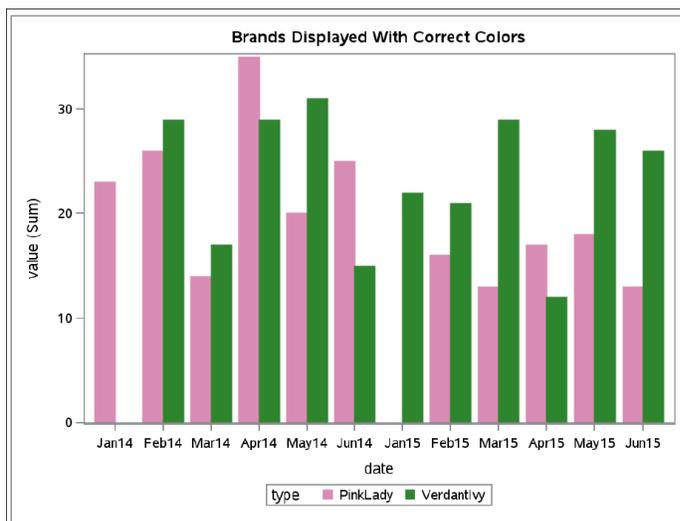
As with the CNTLIN dataset, the ID, VALUE, LINECOLOR and FILLCOLOR variables are required variable names for discrete attribute maps. The values for the variables in the attribute map, however, are specific to the data. The value for ID is the way to give a name to a particular set of group values. The ID value, in this case, “myid” will be used as a link when specified in the ATTRID= option in a plot statement.

You will notice that the VALUE variable holds the name of the two possible values for TYPE: PinkLady and VerdantIvy. FILLCOLOR and LINECOLOR are the attributes that are being linked to the group variable and their values are the colors to use. In this case, the fill color and line color are being set to the same values, but they could be created separately.

But, like a user-defined format, the program above only makes a dataset, with the attribute information, which has to be used. The code below shows how this dataset is used with the SGPLOT procedure:

```
proc sgplot data=attrbar dattrmap=myattrmap;
  title 'Brands Displayed With Correct Colors';
  vbar date / group=type groupdisplay=cluster response=value
  stat=sum attrid=myid;
  format date monyy5.;
run;
```

In the PROC SGPLOT procedure, the attribute dataset is specified in the DATTRMAP= option. In the VBAR statement, the ATTRID= option specifies MYID as the link to value of the TYPE variable so that the correct color values will be used. Output from the above program is shown in PDF results in Output 7.

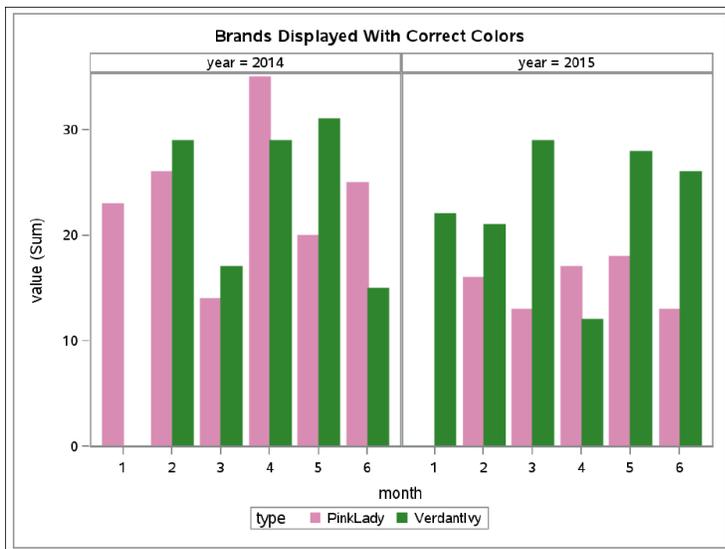


Output 7. PDF Results SGPLOT using Attribute Map

And, with the SGPANEL procedure, the DATTRMAP= and ATTRID= options can also be used to produce paneled output, using the code below:

```
proc sgpanel data=attrbar dattrmap=myattrmap;
  title 'Brands Displayed With Correct Colors';
  panelby year / layout=columnlattice;
  vbar date / group=type groupdisplay=cluster response=value
    stat=sum attrid=myid;
  format date monyy5.;
run;
```

The PDF results, with the correct colors, are shown in Output 8.



Output 8. PDF Results SGPANEL using Attribute Map

Attribute maps can also be used with the SGSCATTER procedure. In addition, be aware that not all plot statements in SGPLOT, SGPANEL, and SGSCATTER support the use of discrete attribute maps. You'll have to refer to the documentation to see the complete list of statements. Generally, the plot statements that support the use of grouping also support discrete attribute maps.

COMPLEX EXAMPLE 4: WISE MEN SAY USE WIDTH

In my 2014 SAS Global Forum paper with Scott Huntley, there is an extensive discussion about how to deal with reports that have many columns and might not fit on a PDF or RTF page. This type of report is not really an issue with HTML output, since HTML output can be as wide as it needs to be. Similarly, when you create output files for Excel, wide reports are not an issue because a spreadsheet can be very wide (based on your version of Excel). However, one question I frequently get asked about concerning width on reports is how to make cells higher so that the text in the cell will “flow” or “wrap” in a more pleasing way in all of these destinations.

The usual reason behind this question is that the report writer has tried to make a cell higher as a means to control long text. But, with SAS procedures like PRINT and REPORT, the way to control text flow or wrapping is through adjustment of the column width. For example, if you drag a spreadsheet column wider, the text will flow wider and the height on that row will decrease; conversely, if you drag a column narrower, the row height will increase and the text will flow into the new, narrower, area. For reports created with SAS, making a cell higher does not cause the text to flow differently.

For purposes of this example, a long variable (named, LONGVAR) was created to hold a very long text string. Some other variables were created that contain strings of differing lengths. The variable SHORTVAR, for example, contains a string of 26 characters. The variable SHORTVAR_NL contains the

same 26 characters, but with an ESCAPECHAR function to insert a line break at a specific break points in the variable:

```
shortvar = 'abcdefghijklmnopqrstuvwxyz';
shortvar_nl = 'abcdefghijklmnop~{newline 1}hijklmnop~{newline 1}qrstuvwxyz';
```

As you can see in Output 9, PDF (and RTF) will work within the limitations set by the destination, as determined by the current margin option settings that were in effect when the output was created.

1) see all the defaults

Name	shortvar	shortvar_nl	longvar
Alfred	abcdefghijklmnopqrstuvwxyz	abcdefghijklmnop qrstuvwxyz	Twas brillig and the slithy toves. Did gyre and gimble in the wabe All mimsy were the borogroves And the mome raths outgrabe. Beware the Jabberwock my son The jaws that bite, the claws that snatch. Beware the jubjub bird and shun the frumious Bandersnatch.
Alice	abcdefghijklmnopqrstuvwxyz	abcdefghijklmnop qrstuvwxyz	xxxxx



26
characters
no line
break



26 characters
with
ESCAPECHAR
line break

Output 9. PDF Results Created with Default Cellwidth

ODS ESCAPECHAR capability has been around ever since Version 8 of SAS. The ESCAPECHAR syntax changed to more like a function invocation starting in Version 9.2 of SAS. But when control over wrapping and cell width are important, the use of WIDTH= and ESCAPECHAR provides you with the control you need.

Let's see what happens when using just the HEIGHT= override, because this is usually the first thing that most folks try. This code uses the same variables and the following code to specify the HEIGHT= style override for all the columns on the report:

```
proc report data=longtxt nowd style (column)={height=2in};
```

As shown in Output 10, the new report looks the same as Output 9, except that the rows are higher.

2) adjust height without changing width default width

Name	shortvar	shortvar_nl	longvar
Alfred	abcdefghijklmnopqrstuvwxyz	abcdefghijklmnop qrstuvwxyz	Twas brillig and the slithy toves. Did gyre and gimble in the wabe All mimsy were the borogroves And the mome raths outgrabe. Beware the Jabberwock my son The jaws that bite, the claws that snatch. Beware the jubjub bird and shun the frumious Bandersnatch.
Alice	abcdefghijklmnopqrstuvwxyz	abcdefghijklmnop qrstuvwxyz	xxxxx

Output 10. PDF Results Created with Changed Value for Cell Height

Taking the width approach is as easy as using the CELLWIDTH= or WIDTH= option on an override for the particular cell, based on the code below. In this example, using PROC REPORT's alias feature, the LONGVAR variable is shown in the report two times, once with a width of 2 inches and once with a width of 4 inches. Also note, the unreasonable attempt to make the SHORTVAR column very, very narrow:

```
proc report data=longtxt nowd;
  column name shortvar shortvar_nl longvar longvar=longvar2;
  title '3) notice how width impacts text wrapping and height';
  define name / display style(column)={just=c};
  define shortvar/ display
    style(column)={cellwidth=.10in};
  define shortvar_nl/display;
  define longvar/ display
    style(column)={cellwidth=2in};
  define longvar2/display "Longvar Diff Width"
    style(column)={cellwidth=4in};
run;
```

3) notice how width impacts text wrapping and height

Name	shortvar	shortvar_nl	longvar	Longvar Diff Width
Alfred	abcdefghijklmnopqrstuvwxy	abcdefg hijklmnop qrstuvwxy	Twas brillig and the slithy toves. Did gyre and gimble in the wabe All mimsy were the borogroves And the mome raths outgrabe. Beware the Jabberwock my son The jaws that bite, the claws that snatch. Beware the jubjub bird and shun the frumious Bandersnatch.	Twas brillig and the slithy toves. Did gyre and gimble in the wabe All mimsy were the borogroves And the mome raths outgrabe. Beware the Jabberwock my son The jaws that bite, the claws that snatch. Beware the jubjub bird and shun the frumious Bandersnatch.
Alice	abcdefghijklmnopqrstuvwxy	abcdefg hijklmnop qrstuvwxy	xxxxx	xxxxx

Output 11. PDF Results Created with Changed Value for Cell Width

The neat thing about changing cell width is that the width is respected by most of the ODS destinations (except for LISTING and the CSV-based destinations). Notice how the “impossible” width of .10 for SHORTVAR was ignored by ODS PDF. Some destinations have a width that they deem impossible to implement and, in this case, the “impossible” width is ignored by ODS PDF. And, also notice how the line break ESCAPECHAR function was respected for SHORTVAR_NL. In the code below, the rest of the report stayed the same, but SHORTVAR had cell width adjusted:

```
proc report data=longtxt nowd;
  column name shortvar shortvar=svar2
    shortvar_nl shortvar_nl=snl2 longvar ;
  title '4) notice how ESCAPECHAR works in narrower and wider cells';
  define name / display style(column)={just=c};
  define shortvar/ display "Default Width";
  define svar2 / display
    style(column)={cellwidth=.5in};
  define shortvar_nl/display
    style(column)={cellwidth=.5in};
  define snl2/display
    style(column)={cellwidth=1.5in};
  define longvar/ display
    style(column)={cellwidth=3in};
run;
```

The results would be as shown in Output 12.

4) notice how ESCAPECHAR works in narrower and wider cells

Name	Default Width	shortvar	shortvar_nl	shortvar_nl	longvar
Alfred	abcdefghijklmnopqrstuvwxy	abcde fghij klmno pqrst vwxyz	abcde fghij klmno pqrst vwxyz	abcde fghij klmno pqrst vwxyz	Twas brillig and the slithy toves. Did gyre and gimble in the wabe. All mimsy were the borogroves And the mome raths outgrabe. Beware the Jabberwock, my son The jaws that bite, the claws that snatch. Beware the jubjub bird and shun the frumious Bandersnatch.
Alice	abcdefghijklmnopqrstuvwxy	abcde fghij klmno pqrst vwxyz	abcde fghij klmno pqrst vwxyz	abcde fghij klmno pqrst vwxyz	xxxxx

Without line break, value uses 4 lines at .5 in for cell width.

With ESCAPECHAR command for line break, the break occurs at the 'g' and 'p' no matter how wide or narrow the cell width.

Output 12. PDF Results Created with Changed Value for Cell Width

And, although the Excel-based destinations do have alternate methods to control cell width, my recommendation is that you try the WIDTH= style override first, because it is a technique that enables you to send your output to other destinations without code changes. The previous code, when submitted for ODS EXCEL, is shown in Output 13.

	A	B	C	D	E	F
1	Name	Default Width	shortvar	shortvar_nl	shortvar_nl	longvar
2	Alfred	abcdefghijklmnopqrstuvwxy	abcde fghij klmno pqrst vwxyz	abcde fghij klmno pqrst vwxyz	abcde fghij klmno pqrst vwxyz	Twas brillig and the slithy toves. Did gyre and gimble in the wabe. All mimsy were the borogroves And the mome raths outgrabe. Beware the Jabberwock my son The jaws that bite, the claws that snatch. Beware the jubjub bird and shun the frumious Bandersnatch.
3	Alice	abcdefghijklmnopqrstuvwxy	abcde fghij klmno pqrst vwxyz	abcde fghij klmno pqrst vwxyz	abcde fghij klmno pqrst vwxyz	xxxxx
4						
5						
6						

Output 13. ODS EXCEL Results Created with Same WIDTH= Values as Output 12

Another question, related to wide reports is how to create a reports that are a mixture of paragraphs that span the entire page and other report objects, possibly graphs or tables, which take up less space.

Prior to ODS LAYOUT and the ODS Report Writing Interface, the choices for creating this kind of output were somewhat limited to ODS TEXT and PROC REPORT. For example, the PDF file shown, annotated, in Output 14, was created from a combination of PROC REPORT for large blocks of text with a poem (also produced with PROC REPORT) and some PROC FREQ output.

The full text for each paragraph is stored in a very large character variable. You can look at the program in the ZIP file. Each PROC REPORT step produces a block of text from a different dataset. Because the ODS invocation uses the STARTPAGE=NO option, the normal page break before each procedure is suppressed. A regular SAS TITLE statement makes the top title line (in black), but the two lines that

appear as subtitles in a smaller font are actually part of the paragraph data. Using a similar technique, the red text that ends the page is not from a FOOTNOTE statement. Instead, it is part of the last paragraph about Piet Mondrian.

This is an example of mixing text and tables

Moby Dick - Chapter 1 - Loomings
by Herman Melville

Call me Ishmael. Some years ago - never mind how long precisely - having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off - then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in this. If they but knew it, almost all men in their degree, some time or other, cherish very nearly the same feelings towards the ocean with me.

There now is your insular city of the Manhattoes, belted round by wharves as Indian isles by coral reefs - commerce surrounds it with her surf. Right and left, the streets take you waterward. Its extreme down-town is the battery, where that noble mole is washed by waves, and cooled by breezes, which a few hours previous were out of sight of land. Look at the crowds of water-gazers there.

Country	
COUNTRY	Frequency
CANADA	480
GERMANY	480
U.S.A.	480

Product type	
PROTOTYPE	Frequency
FURNITURE	576
OFFICE	864

Jabberwocky
by Lewis Carroll

Twas brillig and the slithy toves
Did gyre and gimble in the wabe.
All mimsy were the borogroves,
And the mome raths outgrabe.
Beware the Jabberwock, my son!
The jaws that bite, the claws that snatch!
Beware the Jubjub bird and shun
The frumious Bandersnatch.

From Alice in Wonderland

Modernist Mondrian

After he moved to Paris, Piet Mondrian started painting in an abstract style. His most famous paintings showed a grid of black lines on a primarily white canvas. However, the painted white shapes on the grid were interspersed with the three primary colors, in differing arrangements or compositions. In his later work, he would tilt the square canvases 45 degrees into a diamond shape that he called Lozenges. Mondrian has had an eclectic influence on popular culture. Two programming languages (Piet and Mondrian) are named after him. Yves Saint Laurent designed dresses based on Mondrian paintings. Even cycling uniforms and sneakers have been influenced by his modernist vision.

This Is The End. This Is The End, My Friend.

TITLE Statement

PROC REPORT

PROC FREQ (2)

PROC REPORT

PROC REPORT

Output 14. ODS PDF Results and the Procedures That Placed Output on the Page

Although this type of text-based report is possible using Base SAS and ODS techniques, a better way to produce this or a similar report would be to move into the newest features of ODS, which are the topics of the final sections in this paper.

COMPLEX EXAMPLE 5: FOUND A NEW PLACE TO DWELL IN REPORT WRITING

One of the newest features of ODS is the Report Writing Interface, which you use in a DATA step program to create output. For example, two simple tables that use the SASHELP.IRIS data, are shown in Output 15:

1) Partial Detail Report

Species	Span 3 Header		Span Last 3		
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	New Var
Setosa	50	33	14	2	99
Setosa	46	34	14	3	97
Setosa	46	36	10	2	54
Setosa	51	33	17	5	106
Setosa	55	35	13	2	105
Setosa	48	31	16	2	97
Setosa	52	34	14	2	102
Setosa	49	36	14	1	100
Setosa	44	32	13	2	91
Setosa	50	35	16	6	107

2) Average Report

Averages by Species					
Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	SWIPW
Setosa	50.06	34.28	14.62	2.46	13.93
Versicolour	59.36	27.7	42.6	13.26	2.09
Virginica	65.88	29.74	55.52	20.26	1.47

Output 15. ODS HTML Results Using the Report Writing Interface

One report is a detail report, showing every observation, created with a DATA step program and the Report Writing Interface and the other report is a summary report created with a DATA step program. The code that produced this report is much more verbose than what might be needed in a report generated with PROC PRINT or PROC REPORT.

But although you can create basic tabular reports quite easily with PROC REPORT and PROC PRINT, you do not have full control over row and column spanning in those two procedures. But before getting into more complex topics like row spanning or column spanning, let's look at the things that have to happen for one table:

- Table creation starts
 - Row with column headers starts
 - One or more than one column header is written out
 - End of column header row(s)
 - Beginning of data rows starts
 - Write one row for each observation (in a simple report) and keep writing data rows based on program logic
 - End of all data rows
- End of table creation

Some of these actions happen only once. For example, starting the table usually happens only 1 time conceptually at the top of your program (or when `_N_ = 1`). Then the table ends at the bottom of the program (or when the last observation is reached). SAS programs have a way to detect the last observation through the use of the `END=` option.

The code that created the detail report follows the above pattern. A skeleton of this code is shown:

```
title '1) Partial Detail Report';
data _null_;
  set SASHELP.IRIS(obs=10) end=last;
  if _N_ = 1 then do;
    dcl odsout obj();
    obj.table_start();
    obj.head_start();
    ** Header row 1;
    obj.row_start(type:"Header");
    ** statements for header cells;
    obj.row_end();
    ** Repeat similar pattern for other header rows;

    obj.head_end();
  end;
  ** row for every obs;
  newvar = sum(of _numeric_);
  obj.row_start();
  obj.format_cell(data: varname );
  ** repeat for every variable column;
  obj.row_end();

  if last then do;
    obj.table_end();
  end;
run;
```

The basic program, as far as writing the headers and report rows will not change much if you want to switch from a detail report to a summarized report. The part of the program that has to change is the part that summarizes the data. BY group processing needs to be turned on if the summarization will take place in the program, so FIRST.byvar and LAST.byvar can be used:

```

data _null_;
  set IRIS end=last;
  by species;
  retain spc_cnt;
  if first.species then do;
    spc_cnt = 0;
    sl_tot = 0;
    sw_tot = 0;
    pl_tot = 0;
    pw_tot = 0;
  end;
  spc_cnt + 1;
  sl_tot + SepalLength;
  sw_tot + sepalwidth;
  pl_tot + PetalLength;
  pw_tot + PetalWidth;
  if last.species then do;
    sl_avg = sl_tot / spc_cnt;
    sw_avg = sw_tot / spc_cnt;
    pl_avg = pl_tot / spc_cnt;
    pw_avg = pw_tot / spc_cnt;
    newvar = round(sw_avg/pw_avg, .01);
  end;
  if _N_ = 1 then do;
    dcl odsout obj();
    obj.table_start();
    *** RWI code to write out headers and report rows;
  end;
  if last then do;
    obj.table_end();
  end;
run;
ods _all_ close;

```

The logic to summarize and calculate the average is just standard DATA step code. The variables are initialized to 0 and retained, the total is calculated and at the end of a BY group, the averages and NEWVAR are calculated. Results for the Summary Report are shown in Output 16.

2) Average Report					
Averages by Species					
Species	SepalLength	SepalWidth	PetalLength	PetalWidth	SW/PW
Setosa	50.06	34.28	14.62	2.46	13.93
Versicolor	59.36	27.7	42.6	13.26	2.09
Virginica	65.88	29.74	55.52	20.26	1.47

Output 16. Summary Report Results Using the Report Writing Interface

So far, the summary report is close to the type of report that could be created with PROC REPORT. However, the advantage of the Report Writing Interface comes when you need to do either more complicated column spanning or more complicated row spanning, particularly in the body of the table. For example, consider Output 17, which shows the CSS (Corrected Sum of Squares) statistic in the last column on the report.

3) Row Spanning

Averages by Species					Overall CSS
Species	SepalLength	SepalWidth	PetalLength	PetalWidth	
Setosa	50.06	34.28	14.62	2.46	SepalLength: 10216.83 SepalWidth: 2830.69 PetalLength: 46432.54 PetalWidth: 8656.99
Versicolor	59.36	27.7	42.6	13.26	
Virginica	65.88	29.74	55.52	20.26	

Output 17. ODS RTF Summary Report With Row Spanning using the Report Writing Interface

The CSS statistic was written to a SAS dataset by a PROC MEANS step. However, this is just an arbitrary example of using two datasets in one DATA step program with the Report Writing Interface. The overall structure of the program, with only the relevant syntax, is shown below:

```
data _null_;
  ** SET, BY, LENGTH, RETAIN statements;
  ** same code for accumulating total and calculating averages;
  if _N_ = 1 then do;
    set allcss;
    longvar = catx(' ', 'SepalLength:', round(cssl, .01),
                  '*SepalWidth:', round(cssw, .01),
                  '*PetalLength:', round(cspl, .01),
                  '*PetalWidth:', round(cspw, .01));

    dcl odsout obj();
    obj.table_start();
    obj.head_start();
    ** Header row 1;
    obj.row_start(type:"Header");
    obj.format_cell(text: "Averages by Species", column_span:5, . . .);
    obj.format_cell(text: "Overall*CSS",
                  split: '*', row_span:2,
                  style_attr:"vjust=m color=black backgroundColor=CXd98cb3");
    obj.row_end();
    ** 2nd Header row;
    obj.head_end();
  end;
  ** row for every obs;
  if species='Setosa' then do;
    obj.format_cell(data: longvar, split: '*',
                  row_span:3, style_attr:"just=r vjust=m");
  end;

  ** rest of program same as detail report;
run;
```

The LONGVAR variable is created when `_N_ = 1` because the CSS statistics only need to be written out one time. You could use any other procedure or process to get the data from two or more datasets ready for report writing. Notice that the header cell for "Overall CSS" spans 2 header rows, and the CSS values held in the LONGVAR variable span 3 data rows. Because the Report Writing Interface provides a way to "split" text based on a split character, the header cell for CSS and the data cell for CSS use the same split character of * (asterisk). Notice how the SPLIT: argument is specified in the FORMAT_CELL method.

The IF statement that tests the value of SPECIES then writes the LONGVAR value and spans 3 rows (when Species = Setosa). The first species value is Setosa, so that was chosen for the row_span:3 instruction as a result of the IF statement.

To see the full code, download the ZIP file from support.sas.com at the URL listed at the end of the paper.

Another interesting use of the Report Writing Interface involves creating a table and a long paragraph somewhat similar to Output 14. Consider this report, completely written using a DATA_NULL_step and the Report Writing Interface. Results and notes are shown in Output 18.

Homage to Piet Mondrian			
Row 1 column_span: 2, height:'1.6875in'			
Row 2 column_span: 2, height:'0.875in'			
Row 3 column_span: 1, height:'1.125in'			
Row 4 column_span: 1, height:'0.25in'			

Modernist Mondrian

After he moved to Paris, Piet Mondrian started painting in an abstract style. His most famous paintings showed a grid of black lines on a primarily white canvas. However, the painted white shapes on the grid were interspersed with the three primary colors, in differing arrangements or compositions. In his later work, he would tilt the square canvases 45 degrees, into a diamond shape that he called Lozenges. Mondrian has had an eclectic influence on popular culture. Two programming languages (Piet and Mondrian) are named after him. Yves Saint Laurent designed dresses based on Mondrian paintings. Even cycling uniforms and sneakers have been influenced by his modernist vision.

This is the End. This is the End, My Friend.

DATA_null_step begins

Declare first table:
Table with no header row. Otherwise, the table has 4 data rows and 4 columns. The first 2 rows span 2 columns. The last 2 rows each span 1 column. The borders on col1, col2 and col4 are "inhibited" on the top and the bottom.

On last observation, end first table, then, declare second table:
Second table of paragraph text has 3 rows.

DATA step ends

Output 18. ODS PDF Table and Text Report using the Report Writing Interface

Both tables in this report were created in one DATA step program. The top report is a table of 4 rows and 4 columns without a header. Because this is an homage to Piet Mondrian, the top report specified row heights in the approximate proportions of one of Mondrian's compositions. If you do not specify row heights, then each row height is based on the contents of the cell. If you don't know about Piet Mondrian, he was a modernist artist. After he moved to Paris, Piet Mondrian started painting in an abstract style. His most famous paintings showed a grid of black lines on a primarily white canvas. When I started thinking

of row spanning and column spanning, I was faced with either creating a completely arbitrary example or taking an example from the art world and Mondrian's famous color arrangements inspired dresses, shoes and now, a SAS program.

The first challenge I had, however, was deciding how many rows and how many columns were in the whole table. The final decision was that there were 4 rows and 4 columns. At first, I wanted to say there were just 3 rows and 4 columns, but in the end, decided that there really were 4 rows, with that pesky white cell underneath the blue cell being the only cell with borders on row #4.

There is no border between row 3 and row 4 on columns 1, 2 and 4, because of the INHIBIT attribute:

```
obj.row_start(style_attr:"bordercolor=black borderwidth=2");
obj.format_cell(data: col1, height:"1.125in", width:".4375in",
  inhibit:'B', style_attr:"color=yellow backgroundcolor=yellow");
obj.format_cell(data: col2, height:"1.125in", width:"1.3125in",
  inhibit:'B', style_attr:"color=white backgroundcolor=white");
obj.format_cell(data: col3, height:"1.125in", width:"1.1875in",
  style_attr:"color=cx0000ff backgroundcolor=cx0000ff");
obj.format_cell(data: col4, height:"1.125in", width:".9375in",
  inhibit:'B', style_attr:"color=white backgroundcolor=white");
obj.row_end();
```

On row 3, the bottom border is inhibited (or turned off) with INHIBIT:'B'. Then on row 4, the INHIBIT attribute is set to 'T' to suppress the top border for row 4, columns 1, 2 and 4. As you can see in the code above, the COL3 variable does not use the INHIBIT attribute, so the bottom border from row 3 and the top border from row 4 for column 3 is dividing the blue area from the white area.

The code to produce this report is quite lengthy and you can see it in its entirety in the ZIP archive of programs. But if you need extensive control over row spanning and/or column spanning and you are comfortable with the DATA step, the additional effort to learn the Report Writing Interface syntax will be well worth the time you spend. Even though the methods and "object-dot" syntax are new, the way to write out a table is very straightforward and will add a very valuable skill to your report writing toolbox.

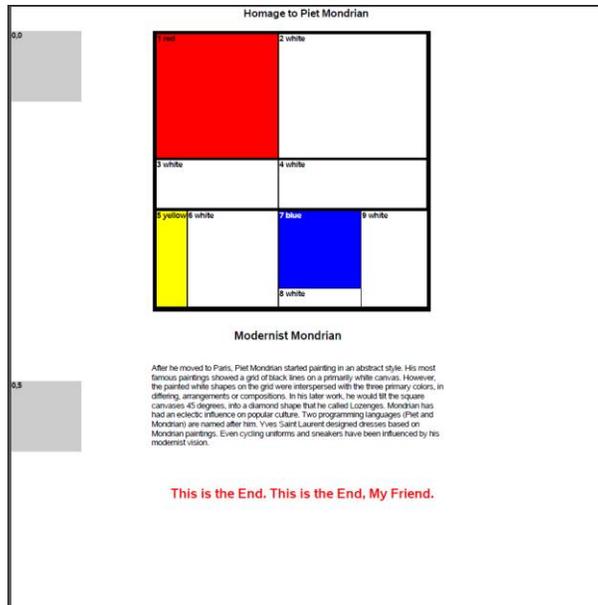
COMPLEX EXAMPLE 6: ALL SHOOK UP OVER LAYOUT

So where else is there to go with complex reports? Sometimes you need to produce tri-fold brochures or annual reports or dashboards that need the kind of pixel-perfect placement for text and images that you can get with either the SAS/GRAPH or ODS GRAPHICS ANNOTATE capability. There are two ways to accomplish this type of output. Depending on what you need to do, you can use absolute layout and region control with a DATA step program and you can also use ODS LAYOUT and ODS REGION to control placement of output objects.

A lot has been written about ODS LAYOUT and ODS REGION statements. Actually, there are a lot of people who have blazed the trail for you in the area of arranging output objects using ODS LAYOUT and ODS REGION statements. Look for links to their papers in the Reference section at the end of this paper. And for many, many more hits, search on www.lexjansen.com for other user group papers.

Within the DATA step, however, you can also use the LAYOUT_ABSOLUTE method as part of the Report Writing Interface. This feature gives you complete control to write output where you want on a page. Absolute layout is best used, in my opinion, for one page report production, where the layout of the page (perhaps a title page, or a handout with text and graphics) is fixed and will never exceed the boundaries of the physical page

The basic outline for starting the DATA step program and declaring the ODS object is the same as those previous syntax snippets, where the "housekeeping" steps happens when `_N_ = 1`. But you only need to test for `_N_` if you are looping through data with a SET statement. In the program that produced Output 19, the DATA step program is not reading any data. So the statements in the program will only be executed one time to produce the output.



Output 19. Partial ODS PDF Table and Text Report using Absolute Layout in a DATA Step Program

The full code that produced this output is shown below:

```

title 'Homage to Piet Mondrian';

options leftmargin=.001in rightmargin=.001in
        topmargin=.001in bottommargin=.001in nodate nonumber;
ods pdf file="&mypath.\complex_6_piet.pdf";

data _null_;
  length pietpara $1500;

  pietpara="&p1.&p2.&p3.&p4.&p5";
  dcl odsout o();
  o.layout_absolute();

  /* where is x=0, y=0 given the above margins and title */
  o.region(x:"0in", y:"0in",width:"1in", height:"1in");
  o.format_text(data: '0,0', style_attr:". . .");

  ** Repeat o.region and o.format for every box in the homage;

  ** paragraph at bottom;
  o.region(x: "2in", y:"4.25in", width: "3.9475in" );
  o.format_text(data: "Modernist Mondrian", just: "c",
               style_attr: ". . .");

  o.region(x: "2in", y:"4.75in", width: "4.5in" );
  o.format_text(data: pietpara, style_attr:". . .");

  o.region(x: "2in", y: "6.5in", width: "4.5in");
  o.format_text(data: "This is the End. This is the End, My Friend.",
               just: "c", style_attr: ". . ." );

o.layout_end();

```

```
run;
ods pdf close;
```

With absolute layout, every region has to be defined. The boxes all had to line up on the Y value and the X value. It was easy to calculate, but tedious to do the coding, according to the following table for every box on the page:

```
o.region(x:"?in", y:"?in", width:"?in", height:"?in" , style_attr:". . .");
o.format_text(data: '???' , style_attr:". . .");
```

Box/Region	Row/col	color	X (in)	Y (in)	Width(in)	Height (in)
1	r1/c1	red	2.0725	0.045	1.725	1.75
2	r1/c2	white	3.8225	0.045	2.09125	1.75
3	r2/c1	white	2.0725	1.845	1.725	0.675
4	r2/c2	white	3.8225	1.845	2.09125	0.675
5	r3/c1	yellow	2.0725	2.5725	0.4175	1.355
6	r3/c2	white	2.52	2.5725	1.2825	1.355
7	r3/c3	blue	3.8225	2.5725	1.15	1.095
8	r4/c3	white	3.8225	3.6875	1.15	0.24
9	r3/c4	white	5	2.5725	0.9255	1.355

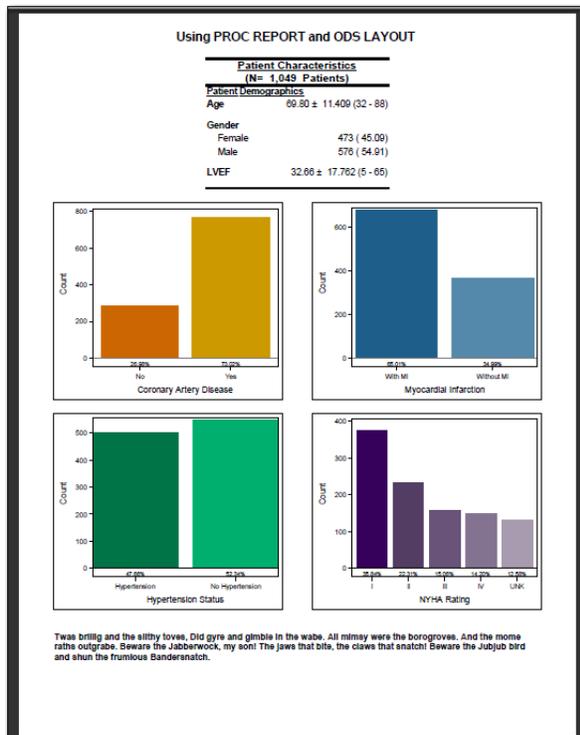
Table 1. Absolute Layout X, Y, WIDTH and HEIGHT values

A 1 inch by 1 inch gray box was placed at x=0 and y=0; x=0 and y=5; and x=0 and y=10 to illustrate the point that the 0 value for Y starts at the upper left corner of the layout area. Then, Y increases from top to bottom, just as X increases from left to right. This may prove slightly disconcerting for SAS/GRAPH programmers, since the usual location of x=0 and y=0 is the bottom left of the graph area.

But, your choices with Report Writing Interface, LAYOUT options and the new method and object syntax will allow you to get creative with your reports.

COMPLEX EXAMPLE 7: SOME THINGS ARE MEANT TO BE

The last example brings you back to a “real world” example, using several of the techniques used in other examples in the paper, and showing ODS LAYOUT and ODS REGION statements too. This example is a reworking of the demographic example from my 2008 paper, as shown in Output 20.



Output 20. ODS PDF Table, Graph and Text Report using Different Techniques

The report is composed of multiple regions. The top region is created by a PROC REPORT program based on the 2008 paper, but only for 3 of the categories, AGE, GENDER and LVEF. Then 4 of the variables are grouped into 2 rows of side-by-side graphs. Finally, a closing paragraph (in this case, a nonsense paragraph) is written to the bottom of the page.

The graphs were all produced with PROC SGPLOT. In this instance, instead of using an attribute map to control the colors, a STYLEATTRS statement is used in each SGPLOT to control the fill colors and outline colors for each graph using a color scheme for each graph (gold, blue, green, and purple).

Rather than use the Report Writing Interface, this report uses ODS LAYOUT and ODS REGION statements to structure the report. The report starts with 1 column that spans the entire page. Then, there is a layout area composed of 2 columns and 2 rows. This means that 4 SGPLOT steps will populate these 4 areas. Of course, that means you have to plan ahead of time for the number of regions on the report and the number of rows/columns that you will need.

After this layout area for the graphs is ended, there is still a one-column layout area for the final paragraphs. The final paragraph used a PROC REPORT step to write out a paragraph, as shown in the example in Output 14. The basic outline of the program is shown below:

```

** run the program from 2008 to make the finaldata file;
** complex2008_demog_data_macro.sas ;

** create data files used by program;
options leftmargin=.25in rightmargin=.25in
        topmargin=.25in bottommargin=.25in nonumber nodate;

ods pdf file="&mypath.\sgplot_complex.pdf" notoc
        startpage=no style=journal;

title font=Helvetica bold h=14pt 'Using PROC REPORT and ODS LAYOUT';

ods escapechar='^';

```

```

ods layout gridded columns=1 column_widths=(8.00in);
ods region;

** First PROC REPORT;

ods layout end;
title; footnote;

ods layout gridded rows=2 columns=2 column_widths=(3.75in 3.75in);

ods region;
ods graphics / height=3in width=3.5in;;
** SGPLOT for Coronary Artery Disease;
ods region;
ods graphics / height=3in width=3.5in;;
** SGPLOT for Myocardial Infarction;

ods region;
ods graphics / height=3in width=3.5in;;
** SGPLOT for Hypertension;

ods region;
ods graphics / height=3in width=3.5in;;
** SGPLOT for NYHA Status;

ods layout end;

ods layout absolute y=8.75in x=.25in width=7.5in;
ods region;
** now write out the last paragraph using PROC REPORT;
ods layout end;

ods _all_ close;

```

Notice how both gridded and absolute layout were used in the ODS LAYOUT statements in the code. If the final paragraph were dynamic and might vary in the amount of text, then GRIDDED would have been a better choice or possibly the Report Writing Interface for the final paragraph. In fact, the Report Writing Interface could also have been used for the top report too. However, one of the goals of this last example is to show that you don't have to abandon existing code or data that you already have working.

You can combine the very new ODS features with older ODS techniques to produce your reports. This will give you a little breathing room to learn and experiment.

CONCLUSION

Some people say that rock 'n roll never forgets and that data never sleeps. Other people say that there's always more than one way to accomplish report writing tasks using SAS. I hope this paper has showed you that there are many new and different ways to write complex reports. From new destinations to new capabilities, ODS doesn't disappoint.

You will be able to find this paper and the zip file of programs by going to <http://support.sas.com/resources/papers/proceedings16/> and search for the paper number. Papers and zip files should be available after the conference.

REFERENCES

- Dorinski, S.M. 2008. "Using ODS Object Oriented Features To Produce a Formatted Record Layout" *Proceedings of North East SAS Users Group, 2008*. SAS Institute Inc., Cary, NC. Available at <http://www.lexjansen.com/nesug/nesug08/bb/bb02.pdf>
- Feder, Steven. 2010. "Integrating Tables and Graphs with ODS LAYOUT." *Proceedings of the SAS Global Forum 2010 Conference*. SAS Institute Inc., Cary, NC. <http://support.sas.com/resources/papers/proceedings10/230-2010.pdf>
- Herbison, R. 2010. "Using the Data Step to Create Bar Charts: The ODS Report Writing Interface" *Proceedings of North East SAS Users Group, 2010*. Baltimore, MD. Available at <http://www.lexjansen.com/nesug/nesug10/bb/bb05.pdf>
- Huff, Gina, Helbig Ph.D., Tuesdi, and James, Chris. 2011. "Absolutely Fabulous: Tips on Creating a Publication-Ready Report using ODS Absolute Layout Functionality." *Proceedings of the SAS Global Forum 2011 Conference*. SAS Institute Inc., Cary, NC. Available at <http://support.sas.com/resources/papers/proceedings11/293-2011.pdf>
- Huntley, Scott, and Lawhorn, Bari. 2010. "Getting the Right Report (Again): Your Compatibility Guide for ODS PDF 9.2." *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings10/035-2010.pdf>
- Huntley, Scott. 2015. "An Insider's Guide to ODS LAYOUT Using SAS® 9.4." *Proceedings of the SAS Global 2015 Conference*. SAS Institute Inc., Cary, NC <http://support.sas.com/resources/papers/proceedings15/SAS1836-2015.pdf>
- Ladan, Annette I. 2006. "The Absolute Nitty-griddy of ODS Layout: Part I." *Proceedings of the South East SAS Users Group 2006*. SAS Institute Inc., Cary, NC. http://analytics.ncsu.edu/sesug/2006/PO02_06.PDF
- Lund, Pete. 2011. "You Did That Report in SAS@!?: The Power of the ODS PDF Destination." *Proceedings of the SAS Global Forum 2011 Conference*. SAS Institute Inc., Cary, NC. <http://support.sas.com/resources/papers/proceedings11/247-2011.pdf>
- Koopmann Jr., Richard. 2007. "Experimenting with the ODS DATA Step Object (Part I)". *Proceedings of the Pacific Northwest SAS Users Group 2007*. SAS Institute Inc., Cary, NC. Available at <http://www.lexjansen.com/pnwsug/2007/Richard%20Koopmann%20-%20Experimenting%20with%20the%20ODS%20DATA%20Step%20Object.pdf>
- Koopmann, Jr. Richard. 2008. "Experimenting with the ODS DATA Step Object (Part II)" *Proceedings of the SAS Global Forum 2008 Conference*. SAS Institute Inc., Cary, NC. Available at <http://www2.sas.com/proceedings/forum2008/261-2008.pdf>
- Kummer, Daniel. 2014. "Toe to Toe: Comparing ODS LAYOUT and the ODS Report Writing Interface." *Proceedings of the SAS Global Forum 2014 Conference*. SAS Institute Inc., Cary, NC. Available at <https://support.sas.com/resources/papers/proceedings14/SAS330-2014.pdf>
- Lund, Pete. 2006. "PDF Can be Pretty Darn Fancy - Tips and Tricks for the ODS PDF Destination". *Proceedings of the 31st SAS Users Group International Conference*. SAS Institute Inc., Cary, NC. Available at <http://www2.sas.com/proceedings/sugi31/092-31.pdf>
- Lund, Pete. 2010. "More to it than Meets the Eye: Creating Custom Legends that Really Tell a Story." *Proceedings of the Midwest SAS Users Group 2010*. Available at <http://www.mwsug.org/proceedings/2010/dataviz/MWSUG-2010-60.pdf>
- Lund, Pete. 2013. "Have it Your Way: Creating Reports with the Data Step Report Writing Interface." *Proceedings of the SAS Global Forum 2013 Conference*. SAS Institute Inc., Cary, NC. Available at <http://support.sas.com/resources/papers/proceedings13/040-2013.pdf>
- Lund, Pete. 2015. "Something Old, Something New...Flexible Reporting with DATA Step-based Tools." *Proceedings of the SAS Global Forum 2015 Conference*. SAS Institute Inc., Cary, NC. Available at <http://support.sas.com/resources/papers/proceedings15/3496-2015.pdf>

- Mays, Rich. 2007. "ODS LAYOUT is Like an Onion." *Proceedings of the 31st SAS Users Group International Conference*. SAS Institute Inc., Cary, NC. Available at <http://www2.sas.com/proceedings/sugi31/159-31.pdf>
- Nelson, Rob. 2010. "ODS LAYOUT to Create Publication-Quality PDF Reports of STD Surveillance Data." *Proceedings of the SAS Global Forum 2010 Conference*. SAS Institute Inc., Cary, NC. Available at <http://support.sas.com/resources/papers/proceedings10/216-2010.pdf>
- 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*. SAS Institute Inc., Cary, NC. Available at <https://support.sas.com/resources/papers/proceedings09/043-2009.pdf>
- O'Connor, Daniel. 2003. "Next Generation Data _NULL_ Report Writing Using ODS OO Features". *Proceedings of the Twenty-Eighth SAS Users Group International 2003*. SAS Institute Inc., Cary, NC. Available at <http://www2.sas.com/proceedings/sugi28/022-28.pdf>
- O'Connor, Daniel. 2008. "SAS Graphics on ODS 9.2 Performance-Enhancing Steroids". *Proceedings of SAS Global Forum 2008*. SAS Institute Inc., Cary, NC. Available at <http://www2.sas.com/proceedings/forum2008/254-2008.pdf>
- O'Connor, Daniel. and Huntley, Scott. 2009. "Breaking New Ground with SAS 9.2 ODS Layout Enhancements." *Proceedings of SAS Global Forum 2009*. SAS Institute Inc., Cary, NC. Available at <http://support.sas.com/resources/papers/proceedings09/043-2009.pdf>
- O'Connor, Daniel. 2009. "The Power to Show: Ad Hoc Reporting, Custom Invoices, and Form Letters." *Proceedings of the SAS Global Forum 2009*. SAS Institute Inc., Cary, NC. Available at <http://support.sas.com/resources/papers/proceedings09/313-2009.pdf>
- O'Connor, Daniel. 2013. "Take Home the ODS Crown Jewels: Master the New Production Features of ODS LAYOUT and Report Writing Interface Techniques." *Proceedings of the SAS Global 2013 Conference*. SAS Institute Inc., Cary, NC. Available at <https://support.sas.com/resources/papers/proceedings13/015-2013.pdf>
- Okerson, Barbara B. 2008. Combining Text and Graphics with ODS LAYOUT and ODS REGION". *Proceedings of the Southeast SAS Users Group 2008*. Available at <http://analytics.ncsu.edu/sesug/2008/SIB-097.pdf>
- Okerson, Barbara B. 2009. "Pleasing the Client: Creating Custom Reports with SAS® ODS LAYOUT and Proc REPORT." *Proceedings of the Southeast SAS Users Group 2009*. Available at <http://analytics.ncsu.edu/sesug/2009/RV008.Okerson.pdf>
- Zender, Cynthia L. 2008. "Creating Complex Reports." *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. Available at <http://www2.sas.com/proceedings/forum2008/173-2008.pdf>
- Zender, Cynthia L. 2010. "SAS® Style Templates: Always in Fashion." *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings10/033-2010.pdf>
- Zender, Cynthia L. 2009. "Tiptoe through the Templates." *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/proceedings09/227-2009.pdf>
- Zender, Cynthia L. 2011. "Don't Gamble with Your Output: How to Use Microsoft Formats with ODS." *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC: SAS Institute Inc. Available at <https://support.sas.com/resources/papers/proceedings11/266-2011.pdf>
- SAS Institute Inc. 2008. "Using Style Elements in the REPORT and TABULATE Procedures." Available at <http://support.sas.com/resources/papers/stylesinprocs.pdf>

ACKNOWLEDGMENTS

The author would like to thank paper reviewers, Chevell Parker, Bari Lawhorn, Jane Eslinger, Linda Jolley, Dan O'Connor, and Michele Ensor for their review and comments. Many thanks to my editor, Jeanette Bottitta, whose help and expertise have improved this paper and made it easier to read.

CONTACT INFORMATION

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

Cynthia L. Zender
100 SAS Campus Drive
Cary, NC 27513
SAS Institute, Inc.
Email: Cynthia.Zender@sas.com
<http://www.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.