PharmaSUG2011 - Paper TU03

Excel Traffic Lighting and Street Paving Simplified

Steven Black, W. L. Gore & Associates, Inc., Flagstaff, AZ

ABSTRACT

Thanks to several recent additions to the excel xp tagset it is now easy to not only traffic light cells, but also ensure the colors, borders, and dimensions of your excel document are just as you want them, without any post processing. In this paper I will show a simple example of three different methods of traffic lighting, a short template for excel, and a simple method of creating a great looking excel document using the report procedure.

INTRODUCTION

Creating a good looking excel output can be a daunting task, especially when winding your way through proc template and proc report if you have not been down that road before. Equally daunting is the ability to color cells in the document based off either the specific values in that column, or values in other columns. In this paper I illustrate how to simply create a good looking excel document and how to traffic light a single column or multiple columns using SAS® without post processing the excel document. This paper is divided up into five easy sections: 1 – Initial data steps, 2 - formatting, 3 - the template procedure, 4 – the excel xp tagset, and 5 – the report procedure.

Basic reference information: Using SAS V9.2 (TS2M2), Microsoft Office 2003 suite, and Excel XP tagset (SAS 9.1.3, v1.86, 04/15/08).

Throughout the paper I will not discuss every option with each step only those that are relevant to the example provided as there are many other papers and information available to obtain details of each section.

INITIAL DATA STEPS

For ease of use and manipulatibility I have used the sashelp.heart data for this example. The heart data originates from the Framingham Massachusetts Heart Study. I have limited the dataset to females whose height is over 67 inches resulting in 48 observations.

```
data _heart; set sashelp.heart (where=(height ge 68 and lowcase(sex)='female'));
```

In this paper I will show three different methodologies for creating traffic lighting/cell coloring in excel. Two of the methodologies are begun in the data step the third occurs in reporting procedure.

In the first example I want to enable others to see the separation between subjects, this will be done using alternating colors per subject line. I begin this by first creating a subject id and then creating an even/odd status variable using the mod function as shown in the code below:

```
subject_id + 1;

even_odd_status=mod(subject_id,2);
```

The second example shows a traffic light or coloring of a cell based on the subject’s smoking status. This is done by creating a new variable color_status which, when smoking status is noted as very heavy, equals 3, otherwise it equals the even_odd_status variable created prior, see code below:

```
if lowcase(smoking_status)=' very heavy (> 25)' then color_status=3;
else color_status=even_odd_status;
```

Lastly I format the variable cholesterol and drop unwanted variables.

```
format cholesterol cholestfmt.;

drop mrw agechddiag chol_status diastolic systolic;

run;
```

PROC FORMAT

Although the proc format is placed prior to the data step in the program (as to utilize the formats by the data step), it is easier to explain after the data step process description. In this procedure I create three formats: one for the variable *cholesterol* based off of the numerical values of the variable, and two for different color schemes. The *cholestfmt* format and the *colormetoo* format are notably similar with only the quoted text changing, in the *colormetoo* format the quoted text will be the color of the cell based off of the value of the variable. As seen in the data step, I apply the *cholestfmt* to the *cholesterol* variable in the datastep and the *colormetoo* format will be applied during the report procedure.

For both the colorme and colormetoo formats, I am applying a color name to a value or range of values. The color choice and value is preferential. In this example I do not use any hexadecimal colors as I have found using the words: very, light, and the color name sufficient. DelGobbo has created a hexadecimal color guide compatible to use with excel versions 2000-2003, see references at the end of the paper.

```
proc format;

*** create formats for numeric variables ***;

value cholestfmt
low-199 = 'Desirable'
200 - 239 = 'Borderline'
240-high = 'High'
;

*** create color formats ***;

value colorme
0='white'
1='light yellow'
2='light green'
3='light red'
4='light gray'
;

value colormetoo
low-199 = 'very light green'
200 - 239 = 'very light blue'
240-high = 'red'
;

run;
```

PROC TEMPLATE

In this paper I want to present a quick, simple, and easy to understand example of proc template. Many times I have found examples of proc template to be discouragingly complicated with a lack of description of which code overrules another code applied later. In this example of proc template there are 4 sections which are laid out as follows: The procedure call - including the new style name and parent reference, the table style, the header style, and the data style.

```
proc template;
define style mystyle;
     parent = styles.default;

     style table from table /
         foreground = black
         font_face = 'arial'
         font_size = 10pt
         just = center
         font_weight = bold
         borderwidth = 1
         vjust = center
         bordercolor = black
     ;
     style header from header /
         foreground = black
         font_face = 'arial'
         font_size = 10pt
```

```
            just = center
            font_weight = bold
            borderwidth = 1
            vjust = top
            bordercolor = black
        ;
        style data from data /
            background = white
            foreground = black
            font_face = 'arial'
            vjust = center
            just = center
            font_size = 10pt
            borderwidth = 1
            bordercolor = black
        ;
   end;
   quit;
```

To create a new style template is moderately simple - first create a name for the style, mine happens to be the very common 'mystyle' then determine a parent style which to modify from, I chose 'default' there are over 50 other parent styles that you can choose from, the code '`proc template;list styles;run;`' can be used to see all available styles, however some look better than others. The next section the table style which primarily controls the borders of the table, I set mine to the attributes as seen above, if no change is made then the default value will remain. To see a list of all default values and additional information regarding proc template see the reference paper by Hayworth. The header section defines the headers of the table including the justification of the header labels and the size and color of the header border. The data section modifies all of the cells within the excel table. The vjust option will vertically adjust the data in the cells to whatever you specify (top, center, or bottom). You close the proc template procedure with an 'end;quit;' block of code.

EXCEL XP TAGSET

The relatively new excel xp tagset has greatly increased the ability to create an excel document that requires little or no post production manipulation. The tagset has been updated a number of times and increased in function and scope after each iteration. The tagset can be found at http://support.sas.com/rnd/base/ods/odsmarkup/, this page contains the current version and all previous versions of the excel xp tagset. SAS also provides a very handy quick reference guide which contains all of the options available and a brief description of the option and default values http://support.sas.com/rnd/base/ods/odsmarkup/excelxp_help.html. To use the tagset you must download it and copy/paste into the editor window then run the code. This will update your tagset library with the proper code needed. This paper will only discuss a few of the options used as there excellent additional references have been provided. The tagset itself contains a fuller description of all of the options which will be displayed in the log when the following code is used:

```
   ods tagsets.excelxp file="test.xml" options(doc="help")
```

To begin using the tagset you use the ods statement then reference the tagset and create an output file with a either a .xml or .xls suffix.

```
   ods tagsets.excelxp file="c:\temp\example.xls"
```

Then state the options desired in parenthesis, each option is followed by an equals sign and then the value in single or double quotes. After the close parenthesis the style is referenced. In this example the style 'mystyle' created in the proc template code above is used.

```
   options(sheet_name='Example Data' autofit_height='yes'
   frozen_headers='1'  orientation='landscape'
   default_column_width="7,5,3,5,5,7,12,8,5,17,9"
   print_header_margin='.25' print_footer_margin='.25') style=mystyle;
```

The default _column_width option can be very a handy so you didn't need to resize the columns in the excel file, a general rule is that a value of 1 in SAS is equivalent to a value of 1.4 in the width of the column in excel, many times I'll create the excel format then resize the column widths according to the data or how I want it then I use the widths in excel then divide by 1.4 to get the default_column_width values for SAS.

Other options that I have found to be useful are:

```
   fittopage='yes' pages_fitwidth='1' pages_fitheight='2'
   embedded_footnotes='yes' print_footer='&amp;L&amp;F &amp;C Confidential &amp;R
   Page &amp;P of &amp;N'
```

In these two examples it is important to note that the fittopage option must be yes to use the other two options (pages_fit_width and pages_fitheight), and the embedded_footnotes must be yes to use the print_footer option. These subtleties are described in the help document.

Also within the help document created in the log is the notation of how to use the &amp; special values which make creating headers/footnotes very easy. In the example I use &amp;L (left sided footer) &amp;F (places the filename in footer), &amp;C (center footer), &amp;R (right footer), &amp;P (page number), and &amp;N (total number of pages).

To close and run the tagset the following code is used:

```
   ods tagsets.excelxp close;
   ods listing;
```

As a standard coding I close the ods listing prior to running the tagset and then open it again once run.

PROC REPORT

Within the tagset open and close statements sits the proc report code. To begin using the report procedure you call the procedure and reference the datasource. I use the _heart dataset created previously, then you can use the various options defined below:

```
   proc report data=_heart nowd missing split='$';
```

The missing option considers missing values as valid values for group, order, or across variables. If you omit the MISSING option, then PROC REPORT does not include observations with a missing value for any group, order, or across variables in the report. When you use NOWD or NOWINDOWS, PROC REPORT runs without the REPORT window and sends its output to the open output destinations. The SPLIT= breaks a column heading when it reaches the specified character and continues the heading on the next line.

The column statement contains all the variables that are to be included in the report and the order in which they will appear.

```
   column (even_odd_status color_status subject_id sex ageatstart height weight
   smoking smoking_status cholesterol status deathcause ageatdeath);
```

The define statement allows you to create the label for the column header and apply any styles needed for that specific column. Labels are created by inserting a '/' and then inserting a label within quotes. In the example I do not want to print the two status variables (even_odd_status and color_status), nor do I care to give them labels, but I do want to apply a color format to these variables, so I use the noprint option and I apply the colorme format to each variable. Some options such as cellwidth do not work in excel nor do rtf codes so these need not be used. To color both character and numeric data the display option should be used within the define statement. For standardization I keep the order in the display section the same as in the column names.

```
   define color_status / display format=colorme. noprint;
   define even_odd_status / display format=colorme. noprint;
   define subject_id / display "Subject ID";
   ...
```

The third form of coloring uses the background color and the *colormetoo* format to color the background of the cell based off the values in the cell. This works for both character and numeric data as long as the correct format is specified for character or numeric data type. In this example I use numeric data. If a value is not contained within the format values then the background color is defaulted to the color specified in the proc template. This style of traffic lighting is only effective when you know the specific values or range of values within the data.

```
   define cholesterol / display "Cholesterol" style={background=colormetoo.};
```

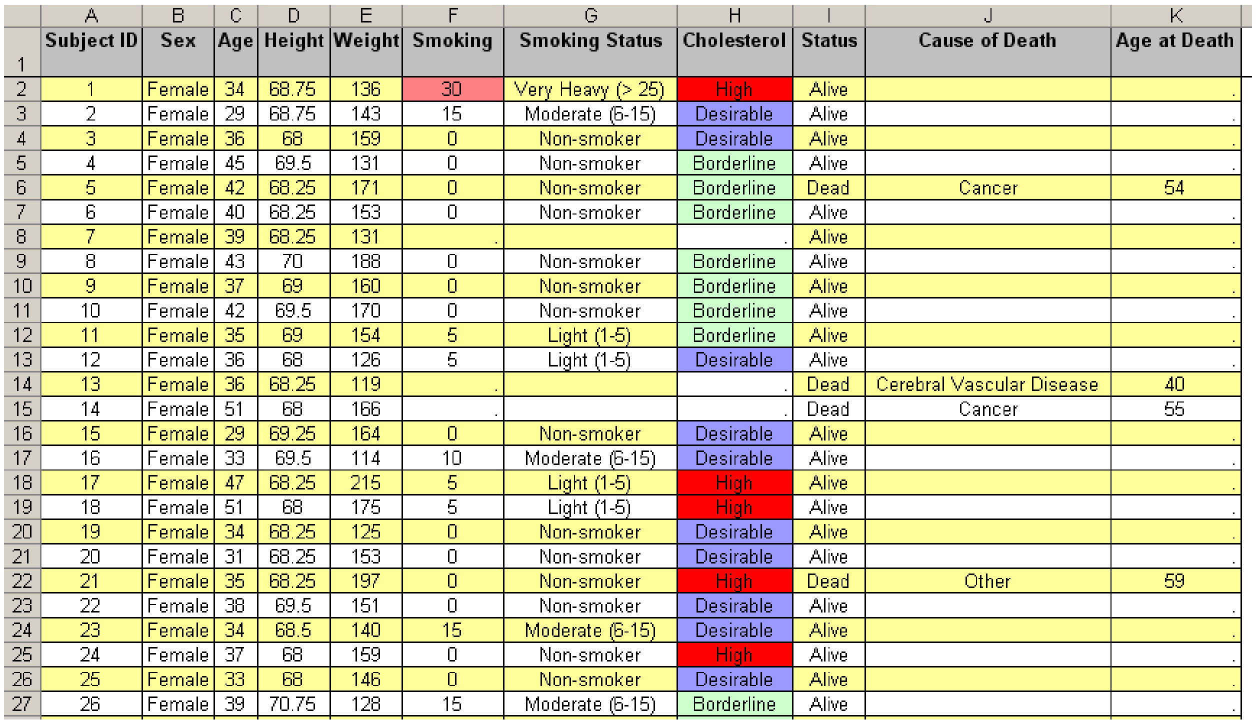
In the above example I demonstrate the format can be applied to both the values of the variable and to the background of the variable based off the values of the variable.

To apply the color to a cell that is not dependant upon the values within the cell, compute and call define statements must be used. To begin you state compute and then put the variable name of the last column that you want affected or colored. Then in your call define statement you place define a bracket and then the variable within quotes a coma then the attribute that you are going to define and the how your are going to define it, in our example we are looking at redefining the style attribute and we change the background to the color of the specified variable, using the put statement. A call define background color syle overrides a style background color in the define statement.

```
compute ageatdeath;
call
define('subject_id','style','style=[background='||put(even_odd_status,colorme.)
||']');
...
call
define('smoking','style','style=[background='||put(color_status,colorme.)||']');
...

endcomp;
run;
```

Once run the excel output will contain all of the columns in the proc report and will color the cells according to formats created previously.



	A	B	C	D	E	F	G	H	I	J	K
1	Subject ID	Sex	Age	Height	Weight	Smoking	Smoking Status	Cholesterol	Status	Cause of Death	Age at Death
2	1	Female	34	68.75	136	30	Very Heavy (> 25)	High	Alive		.
3	2	Female	29	68.75	143	15	Moderate (6-15)	Desirable	Alive		.
4	3	Female	36	68	159	0	Non-smoker	Desirable	Alive		.
5	4	Female	45	69.5	131	0	Non-smoker	Borderline	Alive		.
6	5	Female	42	68.25	171	0	Non-smoker	Borderline	Dead	Cancer	54
7	6	Female	40	68.25	153	0	Non-smoker	Borderline	Alive		.
8	7	Female	39	68.25	131	.		.	Alive		.
9	8	Female	43	70	188	0	Non-smoker	Borderline	Alive		.
10	9	Female	37	69	160	0	Non-smoker	Borderline	Alive		.
11	10	Female	42	69.5	170	0	Non-smoker	Borderline	Alive		.
12	11	Female	35	69	154	5	Light (1-5)	Borderline	Alive		.
13	12	Female	36	68	126	5	Light (1-5)	Desirable	Alive		.
14	13	Female	36	68.25	119	.		.	Dead	Cerebral Vascular Disease	40
15	14	Female	51	68	166	.		.	Dead	Cancer	55
16	15	Female	29	69.25	164	0	Non-smoker	Desirable	Alive		.
17	16	Female	33	69.5	114	10	Moderate (6-15)	Desirable	Alive		.
18	17	Female	47	68.25	215	5	Light (1-5)	High	Alive		.
19	18	Female	51	68	175	5	Light (1-5)	High	Alive		.
20	19	Female	34	68.25	125	0	Non-smoker	Desirable	Alive		.
21	20	Female	31	68.25	153	0	Non-smoker	Desirable	Alive		.
22	21	Female	35	68.25	197	0	Non-smoker	High	Dead	Other	59
23	22	Female	38	69.5	151	0	Non-smoker	Desirable	Alive		.
24	23	Female	34	68.5	140	15	Moderate (6-15)	Desirable	Alive		.
25	24	Female	37	68	159	0	Non-smoker	High	Alive		.
26	25	Female	33	68	146	0	Non-smoker	Desirable	Alive		.
27	26	Female	39	70.75	128	15	Moderate (6-15)	Borderline	Alive		.

CONCLUSION

This paper has demonstrated how to create a traffic lighted and paved Excel spreadsheet with minimal coding. Including coloring cells based off values of other cells and coloring cells based on the value of those cells. Using five simple steps: data steps, formats, proc template, excel xp tagset, and proc report. This paper presents several of the options in each step and provides references for additional resources. Once these basic principles are understood a myriad of additional options/techniques can be used to increase the usefulness of the excel output.

REFERENCES

- Lauren Haworth - PROC TEMPLATE: The Basics http://www2.sas.com/proceedings/sugi31/112-31.pdf

- Vincent DelGobbo Creating AND Importing Multi-Sheet Excel Workbooks the easy way with SAS http://www2.sas.com/proceedings/sugi31/115-31.pdf
- Options for Report Procedure http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a002473620.htm

ACKNOWLEDGMENTS

The great work of Marc Mucatel, Bruce Spotts, and Vincent Del Gobbo

CONTACT INFORMATION

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

Name: Steve Black
Enterprise: W. L. Gore & Associates, Inc.
Address: 3250 W. Kiltie Lane
City, State ZIP: Flagstaff AZ 86001
Work Phone: 928-864-4203
Fax: 928-864-4301
E-mail: sblack@wlgore.com
Web: http://goremedical.com/

APPENDIX

```
proc format;

*** create formats for numeric variables ***;

value cholestfmt
low-199 = 'Desirable'
200 - 239 = 'Borderline'
240-high = 'High'
;

*** create color formats ***;

value colorme
0='white'
1='light yellow'
2='light green'
3='light red'
4='light gray'
;

value colormetoo
low-199 = 'very light blue'
200 - 239 = 'very light green'
240-high = 'red'
;

run;

*** create _heart dataset from sashelp.heart, limiting to females 69 inches or taller ***;

data _heart; set sashelp.heart (where=(height ge 68  and lowcase(sex)='female'));

*** create a subject id ***;

subject_id+1;

*** create an even odd status based of mod function of subject_id ***;
```

```
even_odd_status=mod(subject_id,2);

*** create a color_status variable for smoking ***;

if lowcase(smoking_status)='very heavy (> 25)' then color_status=3;
else color_status=even_odd_status;

*** format all new variables based off formats created above ***;

format cholesterol cholestfmt.;

*** drop unwanted variables ***;

drop mrw agechddiag chol_status diastolic systolic;

run;


*** create style template for excel data called mystyle ***;

proc template;
define style mystyle;
     parent = styles.default;
     style table from table /
         foreground = black
         font_face = 'arial'
         font_size = 10pt
         just = center
         font_weight = bold
         borderwidth = 1
         vjust = center
         bordercolor = black
     ;
     style header from header /
         foreground = black
         font_face = 'arial'
         font_size = 10pt
         just = center
         font_weight = bold
         borderwidth = 1
         vjust = top
         bordercolor = black
     ;
     style data from data /
         background = white
         foreground = black
         font_face = 'arial'
         vjust = center
         just = center
         font_size = 10pt
         borderwidth = 1
         bordercolor = black
     ;
end;
quit;

*** close listing to not show data in output ***;

ods listing close;

title;

*** create excel output file using excelxp tagsets ***;

ods tagsets.excelxp file="c:\temp\example.xls"
options(sheet_name='Example Data' autofit_height='yes'
frozen_headers='1'  orientation='landscape' default_column_width="7,5,3,5,5,7,12,8,5,17,9"
print_header_margin='.25' print_footer_margin='.25') style=mystyle;

*** pull in _heart data ***;

proc report data=_heart nowd missing split='$';
```

```
*** list column variables ***;

    column (even_odd_status color_status subject_id sex ageatstart height weight
       smoking smoking_status cholesterol status deathcause ageatdeath);

*** define each variable ***;

       define color_status / display format=colorme. noprint;
       define even_odd_status / display format=colorme. noprint;
       define subject_id / display "Subject ID";
       define deathcause / display "Cause of Death";
       define ageatstart / display "Age";
       define height / display "Height";
       define weight / display "Weight";
       define smoking / display "Smoking";
       define ageatdeath / display "Age at Death";
       define cholesterol / display "Cholesterol" style={background=colormetoo.};
       define sex / display "Sex";
       define status / display "Status";
       define smoking_status / display "Smoking Status";

*** add color background using call define statement ***;

  compute ageatdeath;
       call define('subject_id','style','style=[background='||
       put(even_odd_status,colorme.)||']');
       call define('deathcause','style','style=[background='||
       put(even_odd_status,colorme.)||']');
       call define('ageatstart','style','style=[background='||
       put(even_odd_status,colorme.)||']');
       call define('height','style','style=[background='||put(even_odd_status,colorme.)||']');
       call define('weight','style','style=[background='||put(even_odd_status,colorme.)||']');
       call define('smoking','style','style=[background='||put(color_status,colorme.)||']');
       call define('ageatdeath','style','style=[background='||
       put(even_odd_status,colorme.)||']');
       call define('status','style','style=[background='||put(even_odd_status,colorme.)||']');
       call define('sex','style','style=[background='||put(even_odd_status,colorme.)||']');
       call define('smoking_status','style','style=[background='||
       put(even_odd_status,colorme.)||']');
  endcomp;
  run;

*** close the tagset and reset ods listing ***;

ods tagsets.excelxp close;
ods listing;
```