Traffic Lighting: The Next Generation

Julie VanBuskirk, Baylor Health Care System, Dallas, TX
Jennifer S. Harper, Baylor Health Care System, Dallas, TX

Abstract:

Traffic lighting is a tool intended to let a reader quickly evaluate data and at-a-glance sort out good versus bad performance. As the name implies, traditional traffic lighting generally separates data into three categories highlighted with red, yellow, and green based on performance. However, if all the cells in a table are boldly colored in primary colors, the reader is unable to achieve their goal of easily sorting between good and bad results. In addition, colorblind users will not be able to even distinguish between colors.

In this paper we will present the visual design changes made to our output and the SAS techniques behind them. We sought to reduce the amount of decoration in our tables and focus on what was important, the data! We employed a more subtle use of cell shading and borders to effectively draw the reader's eye to the results that most need their attention. Typically, traffic lighting is done using PROC FORMAT, however our data cells contained text strings which required a more complex use of style options, COMPUTE blocks, flag variables, and macros to implement. However all this is quite do-able in SAS and the results were a much more effective table delivered to our partners.

Introduction

Effective presentation is a key component of any analytic work. The message of tables and graphs must be clear and concise. The technique of traffic lighting has been used to draw attention to the performance of individual area on a table so that performance can be assessed at a glance. The idea is a good one – readers should be able to very easily find the values they most need to focus on.

However, we seem to have gotten traffic lighting happy in our report production. Bright red, yellow, and green colors are showing up in graphs, maps, tables, and dashboards. With all this traffic lighting the message in the data itself is getting lost. If everything is highlighted then nothing stands out. This paper discusses our methods as we transformed output from the traditional bold red/yellow/green version of traffic lighting to a new design that is more in line with visualization best practices.

The "before" table shown below is based on the traditional form of traffic lighting where red, yellow and green indicate relative performance. The symbolism is obvious as this is a very literal metaphor of the ubiquitous traffic light. The problem with this method is that it creates a table so bright that it is distracting. It becomes difficult to even read the data contained within the cells, let alone to actually understand the message of the data. Also, we need to consider colorblind readers. Approximately 10 percent of men and 1 percent of women are colorblind. For those users with the most common form, the red and green colors can be difficult to distinguish. To make sure colorblind users are able to discern the important information in a table, make sure information is not communicated with color alone.

Replacing Traffic Lighting with Performance Signals

What we wanted to do was to highlight data in a table that needed focus, but in a more effective way. To do this, we used some principles of visual perception. Specifically we used enclosure and color. We enclosed the data cells in solid borders as well as by using subtle shading. The colors we used were subtle (gray background) when used for enclosure – or separating cells of interest from others. The use of a stronger red color served to alert the user to particularly poor performance.

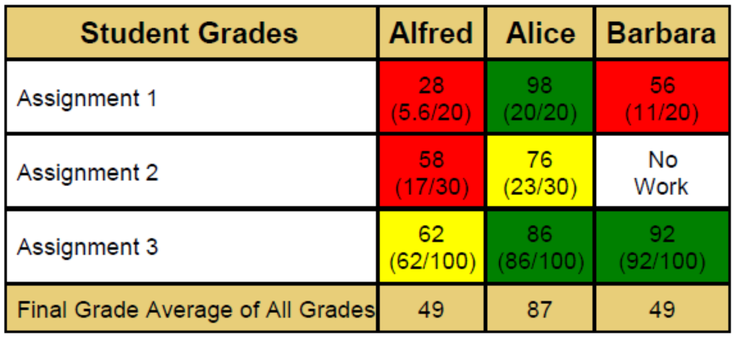
Using just enclosure and color, we were able to create three levels of performance signals. Since our purpose was to highlight values that need attention, we don't need to do any extra formatting to "good" values so we decided to leave those alone. In an effort to draw attention to mid and low performance, we used a subtle, but visible light gray background. This draws the reader's attention, but doesn't obscure the ability of the reader to read the actual text values. To differentiate low performers from the middle, we used the technique of enclosure and color by putting a red box around each cell. The red draws the users attention to these cells, but the because no other cells have strong borders, the use of enclosure ensures even colorblind readers can identify those cells. The design also reproduces well in black-and-white.

In addition to designing more effective highlighting, we made general design changes to make the table more readable. These include lighter border lines and the use of subtle color highlighting to differentiate the summary row from the individual data cells. One other change was to restructure the table. A user will often first want to know the overall score then look at the values that feed into that value. To encourage this reading, we moved the summary row to the top of the table.

Since we have now abandoned the usual red/yellow/green traffic lighting metaphor, we decided we needed a new name. In discussions of how to replace traffic lighting in our language with Kathy Rowell, a proponent of better data visualization in health care data, she suggested the term "**performance signals**". We hope this term catches on as a way to encourage designers to create tables that successfully *signal the performance* shown in the table rather than maintain an ineffective, if catchy-named, metaphor.

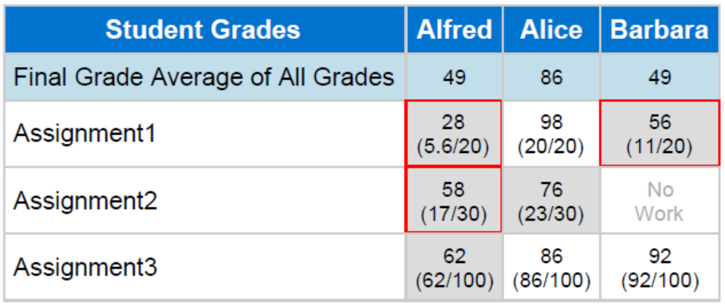
For more on data visualization best practices see the references under "Further Reading".

Before – using conventional red/yellow/green highlighting



Student Grades	Alfred	Alice	Barbara
Assignment 1	28 (5.6/20)	98 (20/20)	56 (11/20)
Assignment 2	58 (17/30)	76 (23/30)	No Work
Assignment 3	62 (62/100)	86 (86/100)	92 (92/100)
Final Grade Average of All Grades	49	87	49

After – using better practices to highlight performance



Student Grades	Alfred	Alice	Barbara
Final Grade Average of All Grades	49	86	49
Assignment1	28 (5.6/20)	98 (20/20)	56 (11/20)
Assignment2	58 (17/30)	76 (23/30)	No Work
Assignment3	62 (62/100)	86 (86/100)	92 (92/100)

Implementing New Performance Signals in SAS

Converting from conventional traffic lighting to our new "performance signals" required a new method of coding. The usual method involves using a PROC FORMAT of ranges that is applied to the usually numeric variable of interest within the PROC REPORT. This method works well for numeric variables, however report requirements often require a text string for the presentation of the value (example: adding the numerator and denominator such as "55% (110/200)"). To create a complex text string we concatenate several variables into a character string. Since these text strings will be complex and vary cell-by-cell, applying a SAS format is not a feasible way to add the desired highlighting to data values. We had to find an automated, data-driven methodology.

Flagging Performance without PROC FORMAT

Creating the text string

The specifications for this project required that data be displayed as a character string as shown in the sample table cell shown below. The table has student names across the top, and Assignments as rows with the last row being the Final Grade (the average of the assignments). (*Note: sample data and complete code are in the appendix.*) Creating the displayed cell contents required concatenating several numeric variables so the assignment score could be displayed along with the numerator and denominator. We also had to add other text elements like parentheses and a line break within the cell.

Student Grades	Alfred	Alice
Assignment 1	28 (5.6/20)	98 (19.6/20)
Assignment2	58 (17.4/30)	76 (22.8/30)

To create the necessary text we used the CATS function to the individual rows for each assignment (excluding the final grade data rows).

```
        if assignment_name ne 'Final Grade' then
           All_Grades = cats(put(Grade, 3.1),
                             '^n(',
                             (put(Numerator, 3.1),
                             '/',
                             put(Denominator, 3),  ')'  )
```

Identifying Performance

Because this new variable is a complex character string, it cannot be easily formatted with PROC FORMAT. To identify which cells should be highlighted with a performance signal, we decided to create a new flag variable that could be used within the report as an indicator of the cell's eventual color. The flag variable was created based on the original numeric grade values using a PROC FORMAT. The format allowed for easy separation between the varying degrees of grades, and allowed us to account for any values that occur out of the expected range.

```
proc format;
      value grades
          low -59   = 1 /* low */
      60 - 78   = 2 /* mid */
      79 - high = 3 /* satisfactory */ ;
run;
```

Creating the table

We used PROC REPORT rather than other reporting procedures because we needed to display the complex text rather than automated summaries. In PROC REPORT the table layout involves using student names as an ACROSS variable, the flag and created grade text string compose the columns under the across variable. Below is the generic structure of the COLUMN statement for PROC REPORT. It sets up the assignment names as the row headers and labels the column, then sets up the student names as the column headers. Nested below each column header are the performance flag and the cell text.

```
column ('column1header' column1)( acrossvar,( flagvar  complextext ));
```

The flag variable will be used to indicate which color the following grade variable should be. We used a CALL DEFINE because the number of columns in the table would vary by the number of students and we will have hidden flag columns. Since we did not want to manually count the number and names of each column generated by across variable, we used pre-processing data step to feed into a MACRO created by Allison McMahill in her paper 'Beyond the Basics: Advanced PROC REPORT Tips and Tricks'**.** This macro uses a data step to calculate the number of variables under the across and sets it as a global macro variable. This data step calculates the column number (_c2_, _c3_, etc.) of each value of the across variable. This is done by sorting the data by the across variable, in this case student_names, next a data step is used to create a count +1 of the total number of occurrences of that variable. This will allow for the calculation of the column number (_*cn*_) that is used in the report macro to automatically move through the columns.

```
proc sort data=class2; by student_name; run;
data class2;
    set sorted_name;
       by student_name;
        if first.student_name then count+1;
        call symput('num', strip(count) )  ;
run;
```

The macro uses %LET to set the values for several macro variables. These include the position of the starting column under the across, the number of variables before the across, and the number of variables under the across. In this case, where the assignment name is the only variable before the across the starting column is two. The varsleft value is the number of variables before the across (in our example, this is example only one – the assignment variable). The varsunder variable is the number of variables that are under the across variable. In this case, this includes the performance and final_grades variables.

```
    /*the position of the first computed var under the across*/
    %let startcol = 2;
    %let varsleft = 1; /*Number of variables before the across*/
    %let varsunder = 2; /*Number of variables under the across*/
```

The MACRO then uses the value of the performance flag to highlight the students' grades. This method uses the COMPUTE block to assign the column identifiers for processing. The DO block starts at the created start column value and goes to a calculated end value. The performance signals are created through a series of CALL DEFINE statements based on the formatted values of the grades variable.

```
%macro create ;
    /* highlight all_grades based on value of performance */
    compute All_Grades;
        %do i=&startcol %to %eval((&num*&varsunder)+&varsleft)
            %by &varsunder;
            /* lower student grades  - gray with red border */
            if _c%eval(&i)_ = 1 and count ne 1 then
                call define ("_c%eval(&i+1)_" ,
                             "style", "style = [background = gainsboro
                                                bordertopcolor = red
                                                borderbottomcolor = red
                                                borderleftcolor = red
                                                borderrightcolor= red" );
            /* mid-level student grades - gray background */
            if _c%eval(&i)_ = 2 and count ne 1 then
                call define ("_c%eval(&i+1)_",
                             "style", "style = [background = gainsboro]");
            /* no highlighting for satisfactory peformance */
        %end;
    endcomp;
%mend create;
%create ;
```

To create a table that better focuses on the data, we used subtle grey shading (SAS color gainsboro) for all student grades that needed improvement. We emphasized those that need immediate efforts at improvement by adjusting the cell border colors to enclose only those cells in red. To avoid creating signals for the summary grade row, we used count not equal to avoid any shading on that row because our goal was to monitor individual assignments, not the final score.

A New Way to Traffic Light

Moving beyond the conventional red, yellow, green traffic lighting to a new design that highlights data without hiding the details allows the person reading a table

The challenges in SAS come in two forms – how to flag cells for highlighting if the cell values are complex character strings and how to apply a more complex style to an unknown table layout. This paper demonstrated moderately complex PROC REPORT techniques using a hidden flag variable to signal performance and compute blocks to assign formatting. The macro we used to manage the counting of an unknown number of columns is also complex, but quite extensible to many situations.

Once again we see that SAS can do it all – boldly going where few tables have gone before.

References

Beyond the Basics: Advanced PROC REPORT Tips and Tricks. McMahill, Allison. SAS Institute Inc. Cary, NC; SAS Global Forum 2007 (Paper 276-2007).

Carpenter's Complete Guide to the SAS Report Procedure. Carpenter, Art. SAS Publishing, 2007.

Kathy Rowell, Kathy Rowell and Associates. Personal Communication.

STOP! WAIT! GO!: See What Traffic-Lighting Can Do For You! Hadden, Louise. Abt Associates, Inc., Cambridge, MA. SUGI 31 (Paper 142-31).

Further Reading

Information Dashboard Design. Stephen Few. (ISBN-13 978-0596100162) www.perceptualedge.com

PDF Can be Pretty Darn Fancy – Tips and Tricks for the ODS PDF Destination. Lund, Pete. Looking Glass Analytics, Olympia, WA. SUGI 31 (Paper 092-31).

Unleash Your Inner Healthcare Data. "Red Yellow Green is SO Last Season (7/8/11 Kathy Rowell and Associates Newsletter) http://ksrowell.com/newsletters/2011_07_08_healthcare_data.html

Keywords: Traffic Lighting, PROC REPORT, Compute block, OD

Appendix: Full Code

```
/* Code to get the modified class dataset used in this example */
data class;
    length assignment $11;
    infile datalines delimiter = ',' dsd;
    input student_name $ assignment $ grade numerator denominator ;
    datalines;
Alfred,Assignment1,28,5.6,20
Alfred,Assignment2,58,17.4,30
Alfred,Assignment3,62,62,100
Alfred,Final Grade,49,.,.
Alice,Assignment1,98,19.6,20
Alice,Assignment2,76,22.8,30
Alice,Assignment3,86,86,100
Alice,Final Gradem,86,.,.
Barbara,Assignment1,56,11.2,20
Barbara,Assignment2,.,.,30
Barbara,Assignment3,92,92,100
Barbara,Final Grade,49,.,.
;
run;
proc format;
    /*Format all missing work*/
    value missing
        . = 'No^nWork' ;
    /*Color change for presentation*/
    value $nocases
        'No^nWork' = 'light grey';
    /* Range for basis of traffic lighting */
    value grades
        low -59  = 1
        60 - 78  = 2
        79 - high = 3;
run;
/* Data changes necessary for desired table structure */
data class2; set class;
    if assignment ne 'Final Grade' then
/* Create string character for printing*/
    All_Grades = cats(put(Grade,3.1),
                      '^n(',
                      put(Numerator,3.1),
                      '/',
                      put(denominator,3.1), ')'   );
/* Final grades printed without numerators and denominators */
    else All_Grades = put(Grade, 3.1);
/* Format missing data for correct printing */
    if grade = . then all_grades = put(grade, missing.);
/* Creation of a flag based on student grades */
    performance = put(grade, grades.);
/* Creation of bundle flag to control print order of final grades */
    if assignment = 'Final Grade' then bundle = 1;
    else bundle = 0;
run;
/* Sort by student name for calculation of variables under the across */
proc sort data=class2; by student_name; run;
/* Data step for calculation of macro variable num */
```

```
data class2; set class2;
    by student_name;
    /* Count of observations per student */
    if first.student_name then count+1;
    /* Set num as count per student */
    call symput('num', trim(left(count)));
run;
/* Sort by student name, then descending bundle to ensure final grade prints
     first then assignments */
proc sort data=class2 out=class3; by student_name descending bundle; run;
ods listing close;
ods pdf file= "C:\\your location" style=sasweb ;
proc report data= class3 completecols nowd style={frame=void}
        /* Header style modifications */
        style(header) = {background = cx0073cf
                         font_size = 10pt
                         foreground = white
                         fontfamily='Arial'}  ;
    column ('Student Grades' Assignment)
           ( Student_name,( performance All_grades ));
    /* Student names on top of report */
    define Student_name / Across ' ' order=data ;
    /* Student Assignments List */
    define Assignment / group ' ' order = data
                        style(column) = {just=Left vjust = center};
    /* -- Variables under the across var -- */
    /* Performance is the hidden formatting flag */
    define performance / group   missing ' ' noprint ;
    /* the displayed complex text of students grades */
    define All_grades / group missing ' '  center
                        style(column)={font_size = 8pt
                        /* formated to print gray if missing */
                        foreground = $nocases.
                        just=center vjust=center } ;
    /* Count to shade the final grade light blue */
    compute Assignment;
        count+1;
        if count = 1 then do;
            call define(_row_, "style","style=[background=cxc2deea]");
        end;
    endcomp;
%let startcol = 2; /* position of the first computed var under the across */
%let varsleft = 1; /* Number of variables before the across */
%let varsunder = 2; /* Number of variables under the across */
```

```
/* Macro create calculates the number of across rows and highlights based on
     these numbers */
%macro create ;
    /* traffic light student grades based on value of performance */
    compute All_grades;
        %do i=&startcol %to %eval((&num*&varsunder)+&varsleft)
            %by &varsunder;
            /* low performance - gray with red border */
            if _c%eval(&i)_ = 1 and count ne 1 then
                call define ("_c%eval(&i+1)_" ,
                             "style", "style=[background = gainsboro
                                              bordertopcolor = red
                                              borderbottomcolor = red
                                              borderleftcolor = red
                                              borderrightcolor = red" ) ;
            /* Not so great facility numbers - gray background */
           if _c%eval(&i)_ = 2 and count ne 1 then
               call define ("_c%eval(&i+1)_",
                            "style", "style = [background = gainsboro]" ) ;
        %end;
    endcomp;
%mend create;

%create ;
run; quit;
ods pdf close;
```