USING VALIDATION TABLES AND MACROS TO AUTOMATE TRAFFIC LIGHTING IN PROCEDURE REPORT

Khoi Dinh To
Office of Planning and Decision Support, Virginia Commonwealth University

ABSTRACT

The flexibility of procedure report, compute blocks, and call define statement allow SAS users to create and format reports in almost any way they want. Traffic lighting--assigning appearance attributes based on predetermined conditions--is a powerful tool that can be applied in procedure report to further customize reports to meet specific requirements.

This presentation shows how validation tables and macros can be used to automate traffic lighting in procedure report, step by step. Automating traffic lighting helps (1) save report writers a significant amount of time from manually writing codes, (2) minimize the possibility of making errors from manual coding, and therefore (3) minimize the maintenance of reports as the automation has taken care of various scenarios that can happen.

This presentation is helpful to analysts who look for a way to automate their work flow with procedure report and traffic lighting. Experience with procedure report, traffic lighting, and macro is desired, but not required.

KEY WORDS

procedure report, compute block, call define statement, traffic lighting, validation table, macro, automation

SYNTAX AND EXPLANATION

Let's have a look at final product that we want to have (Table 1). In this table, General Education courses are highlighted in green, Bottleneck courses in yellow, General Education & Bottleneck courses in red, and other courses in white. Also, courses that have the percentage of D/F/W grades equal or greater than 30% are highlighted in red.

If the report had only a few courses over a small number of years, we could manually code a series of call define statements, one for each course, to achieve the formats we want. However, if we had a report that has fifty or more courses, and these courses may change from one academic term to another; coding these formats manually would

be time-consuming and error-prone. This presentation below shows a way to automate the work flow.

Table 1. Course Grade Summary, Fall 2008 – Fall 2012

	Fall 2008		Fall 2009		Fall 2010		Fall 2011		Fall 2012	
	DFW	Other	DFW	Other	DFW	Other	DFW	Other	DFW	Other
MATH200	30.9	69.1	30.8	69.2	36.1	63.9	35.9	64.1	36.1	63.9
MATH201	41.1	58.9	28.0	72.0	39.6	60.4	37.2	62.8	39.7	60.3
MATH211	38.1	61.9	30.7	69.3	31.0	69.0	22.5	77.5	23.1	76.9
UNIV111	10.4	89.6	9.6	90.4	9.6	90.4	10.4	89.6	10.2	89.8
UNIV112	17.4	82.6	11.7	88.3	12.4	87.6	8.9	91.1	9.3	90.7
UNIV200	0.0	0.0	0.0	0.0	20.9	79.1	16.1	83.9	11.4	88.6
SOCY101	20.1	79.9	15.7	84.3	16.9	83.1	14.0	86.0	10.7	89.3
STAT210	26.6	73.4	23.0	77.0	23.0	77.0	22.6	77.4	21.0	79.0
ACCT202	44.4	55.6	28.0	72.0	35.1	64.9	39.4	60.6	41.4	58.6
BIOZ151	18.8	81.2	20.1	79.9	11.0	89.0	16.2	83.8	13.3	86.7
CHEM101	37.7	62.3	33.2	66.8	31.8	68.2	36.6	63.4	31.9	68.1
ENGL200	28.9	71.1	24.7	75.3	26.5	73.5	0.0	0.0	0.0	0.0
THEA311	2.4	97.6	0.0	100.0	4.0	96.0	0.0	100.0	0.0	100.0
TOTAL	22.7	77.3	20.0	80.0	20.4	79.6	20.0	80.0	18.2	81.8

***Green**: General Education / **Yellow**: Bottleneck / **Red**: General Education & Bottleneck / **White**: Other*

First of all, we need to obtain a validation table that has **all** General Education, Bottleneck, and General Education & Bottleneck courses. This list of courses is determined by an academic advising committee (see Table 2 below).

Table 2. List of all General Education, Bottleneck, and General Education & Bottleneck courses

course_identification	cat	cat_desc
MATH200	1	General Education
MATH201	1	General Education
MATH211	1	General Education
UNIV111	2	Bottleneck
UNIV112	2	Bottleneck
UNIV200	2	Bottleneck
SOCY101	3	General Education & Bottleneck
STAT210	3	General Education & Bottleneck

As we want to highlight General Education courses in green, Bottleneck courses in yellow, General Education & Bottleneck courses in red, and other courses in white, we

use a simple data step below to assign the color to each group (see Table 3). (You can change the color of each group if you wish.)

```
data validation; set validation;
length color $30;
format color $30.;
     if cat = 1 then color = 'Very Light Green';
else if cat = 2 then color = 'Very Light Yellow';
else if cat = 3 then color = 'Very Light Red';
run;
```

Table 3. List of all General Education, Bottleneck, and General Education & Bottleneck courses and their assigned colors

course_identification	cat	cat_desc	color
MATH200	1	General Education	Very Light Green
MATH201	1	General Education	Very Light Green
MATH211	1	General Education	Very Light Green
UNIV111	2	Bottleneck	Very Light Yellow
UNIV112	2	Bottleneck	Very Light Yellow
UNIV200	2	Bottleneck	Very Light Yellow
SOCY101	3	General Education & Bottleneck	Very Light Red
STAT210	3	General Education & Bottleneck	Very Light Red

Create a few formats that will be used later:

```
proc format;
/* the following format puts grades into two groups: DFW and OTHER */
value $dfw (notsorted)
'D','F','W' = 'DFW'
OTHER       = 'Other'
;
value grades (notsorted)
/* the following format highlights in red those courses having DFW
rates equal to or greater than 30% */
.,0     = 'Black'
0 <- 30 = 'Black'
OTHER   = 'Red'
; run;
```

Second, merge raw data set with validation table so that we can sort raw data by cat and course identification. That way, general education courses come first, then bottleneck courses, then general education & bottleneck courses. Other courses come at the end of the report without being highlighted in color.

```
proc sql;
create table courses as
select distinct A.*, B.cat
           from vasug.&data_set. A
left outer join validation B
             on A.course_identification = B.course_identification
; quit;

data courses; set courses;
if cat = . then cat = 4;
/* other courses that will not be highlighted in color */
run;
```

Third, read two cells “course_identification” and “color” of each record in Table 3 into macro variables so that they can be used in proc report later. There are a few ways we can read a data cell into a macro variable, and in this presentation we will use proc SQL to do it.

```
proc sql noprint;
select count(*) into: anum /* total number of records in table_3 */
  from table_2
;
select course_identification into :avar_1 - :avar_%trim(&anum.)
  from table_2
;
select color into :color_1 - :color_%trim(&anum.)
  from table_2
; quit;
```

We also need to read into a macro variable the total number of columns that will appear in the final report (Table 1). (The total number of columns = the total number of academic terms multiplied by 2. This is because in each academic term the data are divided into two columns--one for DFW grades and the other for OTHER grades.)

```
proc sql noprint;
select (count(distinct academic_period)*2) into: colnum from courses
; quit;
```

Fourth, we use proc tabulate and its out= option to aggregate data. We also calculate the percentages of DFW grades and OTHER grades in each course. (This step can also be done in proc report, using compute block.)

```
proc sort data=courses; by cat course_identification; run;

ods select none;
proc tabulate data=courses missing out=summary_data;
format final_grade $dfw.;
class academic_period_desc course_identification / order=data;
class final_grade / order=data preloadfmt;
table course_identification='' all='TOTAL'
    , academic_period_desc=''*final_grade=''*pctn<final_grade
all>=''*format=comma5.1;
run;
ods select all;

data summary_data; set summary_data;
if course_identification = ' ' then do;
   course_identification = 'TOTAL';
   pctn_110 = pctn_100;
   cat      = 5; /* the total line */
end;
run;
```

Up to this point, everything is ready and we can start build up the report we wanted (Table 1), using data from the aggregate data set above.

```
%macro do_this;
title 'Example 1 - Traffic Lighting with Proc Report';
footnote1 height=3 justify=left 'Green: General Education courses';
footnote2 height=3 justify=left 'Yellow: Bottleneck courses';
footnote3 height=3 justify=left 'Red: General Education & Bottleneck
courses';
footnote4 height=3 justify=left 'White: Other courses';
proc report data=summary_data missing LIST LS=100;
/* LIST LS=100 will print out in the log the commands executed */
format final_grade $dfw.;
column course_identification
academic_period_desc,final_grade,pctn_110;
define course_identification / ' ' group  order=data
style(column)={font_weight=bold};
define academic_period_desc  / ' ' across order=data;
define final_grade           / ' ' across order=data;
define pctn_110              / ' ' analysis sum format=comma5.1;

compute course_identification;  /* compute block (1) */
%do i = 1 %to &anum.;
if course_identification = "&&avar_&i." then call
define(_row_,'style',"style={background=&&color_&i.}");
/* this tiny macro is the key point!!! */
%end;
endcomp;
```

```
compute pctn_110; /* compute block (2) */
%do j = 2 %to &colnum. %by 2;
if pctn_110 < 30 then call define("_c&j._", 'style',
'style={foreground=grades.}');
/* this tiny macro is the key point!!! */
%end;
endcomp;
%mend;

%do_this;
```

When executed, the macro %do_this above will produce the final report as shown in Table 1. If you checked the log, you would see that compute blocks (1) and (2) resolved into the following code lines:

Compute block (1):

```
COMPUTE  COURSE_IDENTIFICATION / CHAR LENGTH=63 ;

if course_identification = "ENGL220" then call define(_row_, 'style',
"style={background=Very Light Green}");
if course_identification = " ENGL225" then call define(_row_, 'style',
"style={background=Very Light Green}");
if course_identification = " ENGL230" then call define(_row_, 'style',
"style={background=Very Light Green}");
if course_identification = "MATH200" then call define(_row_, 'style',
"style={background=Very Light Green}");
if course_identification = "MATH201" then call define(_row_, 'style',
"style={background=Very Light Green}");
if course_identification = "MATH211" then call define(_row_, 'style',
"style={background=Very Light Green}");

if course_identification = "UNIV111" then call define(_row_, 'style',
"style={background=Very Light Yellow}");
if course_identification = "UNIV112" then call define(_row_, 'style',
"style={background=Very Light Yellow}");
if course_identification = "UNIV200" then call define(_row_, 'style',
"style={background=Very Light Yellow}");
if course_identification = "UNIV220" then call define(_row_, 'style',
"style={background=Very Light Yellow}");

if course_identification = "SOCY100" then call define(_row_, 'style',
"style={background=Very Light Red}");
if course_identification = "SOCY101" then call define(_row_, 'style',
"style={background=Very Light Red}");
if course_identification = "STAT210" then call define(_row_, 'style',
"style={background=Very Light Red}");
ENDCOMP;
```

Compute block (2):

```
COMPUTE  PctN_110;
if pctn_110 < 30 then call define("_c2_", 'style',
'style={foreground=grades.}');
if pctn_110 < 30 then call define("_c4_", 'style',
'style={foreground=grades.}');
if pctn_110 < 30 then call define("_c6_", 'style',
'style={foreground=grades.}');
if pctn_110 < 30 then call define("_c8_", 'style',
'style={foreground=grades.}');
if pctn_110 < 30 then call define("_c10_", 'style',
'style={foreground=grades.}');
if pctn_110 < 30 then call define("_c12_", 'style',
'style={foreground=grades.}');
if pctn_110 < 30 then call define("_c14_", 'style',
'style={foreground=grades.}');
ENDCOMP;
```

EXAMPLE 2: USING MOD FUNCTION WITH COMPUTE BLOCK

MAJOR	Fall 2008			Fall 2009			Fall 2010		
	UG	PR	GR	UG	PR	GR	UG	PR	GR
Accountancy			21			18			19
Accounting	20			14			13		
Addiction Studies			8			29			29
Adult Learning			3			4			2
African American Studies				2			2		
African-American Studies	6								
Aging Studies			1						1
Anatomy and Neurobiology			26			4			
Anthropology	27			29			27		
Applied Social Research			1						1
Art Education	6		12	6		9	4		7
Art Foundation	462			446			479		
Art History	23		11	25		8	15		11
Athletic Training						1			
Autism Spectrum Disorders						1			5
Biochemistry			38			10			6
Bioinformatics	16		6	11		3	16		4
Biology	474		11	474		13	564		13
Biomedical Engineering	54		11	67		10	71		14
Biostatistics			5			9			9
Business	120		22	54		33	19		20
Business Administration			114			104			95

In the above table, we highlight rows in white and gray, alternately, for easier reference. This can be done easily by using MOD function in proc report. MOD function determines whether a number is odd or even. Let's examine the codes below.

First, we sort data and count the number of majors:

```
proc sort data=students; by major_desc1 academic_period_desc; run;

data students; set students;
by major_desc1;
count + 0;
if first.major_desc1 then count + 1;
run;
```

When building the report, we use MOD function to determine whether a row (i.e., a major) is odd or even. If it is odd, we highlight it in gray; if it is even, we highlight it in white.

```
proc report data=students missing;
column major_desc1 count academic_period_desc,student_level1;
define major_desc1          / group 'MAJOR' style=[font_weight=bold];
define count                / group noprint;
define academic_period_desc / across '';
define student_level1       / across '' descending;

compute count;
/* even numbers */
if mod(count,2)  = 0 then call
define(_row_,'style','style={background=white}');
/* odd numbers */
if mod(count,2) ^= 0 then call
define(_row_,'style','style={background=very pale gray}');
endcomp;
run;
```

CONCLUSION

SAS users can combine procedure report, compute block, and call define statement to produce reports and customize traffic lighting features at their own wishes. However, when the data get larger and change every time the reports are produced, manually coding the traffic lighting formats can pose two problems--more time-consuming and error-prone. Using validation tables and macros in those situations would help solve the problems as the work flow would be fully automated. The reports, therefore, are produced without errors and require a minimal amount of time for maintenance.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please contact me at:

Khoi Dinh To, PhD
Office of Planning and Decision Support, Virginia Commonwealth University
901 W Franklin Street
Richmond, VA 23284-2527
Email: kdto@vcu.edu, todinhkhoi@yahoo.com