Paper 18-25

Let There Be Highlights: Data-driven Cell, Row and Column Highlights in %TAB2HTM and %DS2HTM Output

Matthew Flynn and Ray Pass

Introduction

Version 6.12 of the SAS System Technical Support supplied macros TAB2HTM and DS2HTM allow one to easily convert one's SAS output directly to hyper-text markup language - HTML for ease of display over intranets or the Internet.

That ease of use, however, combined with the power of the SAS System, has quickly led us to generate large volumes of HTML output for information delivery to business partners. What we soon needed was a way to quickly identify and highlight particular data cells, rows, columns and entire tables - and we needed a way to make it data-driven.

The solution we arrived at puts a new twist on a very old technique - reading in an output file and post-processing to customize the output. Even though it uses an old tried-and-true technique, it remains a valuable tool in the SAS toolkit.

I learn best by example, so we'll walk though the steps to add highlighting to your SAS HTML output. Hopefully, illustrating the ease and power of the technique along the way,.

Example PROC TABULATE output

Following the example application discussed at the SAS Web Tools web site: TABULATE Formatter Usage Example (http://www.sas.com/rnd/web/format/tab/tabex.html) we will start with traditional PROC TABULATE code & output.

```
title 'World Wide Product Sales
Report';
proc tabulate
data=sashelp.prdsal2
;  table state,
        (actual predict) *
(sum*f=dollar14.2
mean*f=dollar14.2);
   var actual predict;
   class state;
   keylabel sum="Total";
   keylabel mean="Average";
run;
```

Figure 1 shows plain PROC TABULATE output.

Enhancement 1 - Adding TABULATE Formatter Macro to the SAS Program

The first enhancement adds HTML tags to the PROC TABULATE output with two easy commands to 1) mark the start of the data capture:

```
%tab2htm(capture=on,
             runmode=b);
```

and 2) mark the end of the data capture:

```
%tab2htm(
     capture=off,
     runmode=b,
     openmode=replace,
     htmlfile=ex2.html,
brtitle=Tabulate    Formatter
Example,
     center=Y,
     tsize=+3);.
```

And the resulting output:

C:\HTML Highlights\ex1.htm

Enhancement 2: Changing Colors of Output Elements

Next, we modify the tab2htm macro call to manually add color to the table column and row labels, and expanding the table width through the text size.

```
..... clcolor=blue,
      rlcolor=blue,
      tsize=+3);
```

And the resulting output:

C:\HTML Highlights\ex2.htm

Enhancement 3: Adding Images

We can continue dressing up out HTML output by adding background images.

```
..... bgtype=image,
      bg=marble1.jpg,
      encode=N);
```

And the resulting output:

C:\HTML Highlights\ex3.htm

The SAS Web tools TAB2HTM example application next describes editing the HTML output file directly to modify the title line and insert and animated GIF file from:

```
title 'World Wide Product Sales
Report';
```

to:

```
title '<IMG ALIGN=CENTER SRC=
globeanm.gif> World Wide Product
Sales Report';.
```

C:\HTML Highlights\ex4.htm

Enhancement 4: Manual Highlighting of data cells

Continuing on that theme, one can easily edit the HTML file after it is generated with the TAB2HTM macro for a single table and highlight individual cells, rows, or columns. For example, on the line for the Province of Ontario, change the line reading:

```
<TD ALIGN="RIGHT" VALIGN="BOTTOM"
NOWRAP>$-469,832.00</TD>
```

to:

```
<TD ALIGN="RIGHT" VALIGN="BOTTOM"
NOWRAP><FONT COLOR=red>$-
815.68</FONT></TD>.
```

Simply inserting the HTML tag <FONT COLOR=red> to highlight a cell with negative numbers as red text and save the source file as:

C:\HTML Highlights\ex5.htm

and you've now highlighted individual cells.

Powerful Tools

If you've followed along thus far, this technique sounds potentially quite useful for the customization for small numbers of static web pages, but how can one expand & enhance this general process? Better yet, how can one make the table modification data-driven? That is, how can one achieve flexible automated complete formatting control of the generation of your web pages from SAS?

The manual table-editing process above should be by now screaming for a tool to automate the tedious HTML editing. Powerful tools for this process include HTML-editors and UNIX regular expressions. There are lots of

ways to do this. First, we will briefly discuss (GASP!) non-SAS options. Then we present the advantages of the power of the SAS System to really drive this process to give you dynamic data-driven great looking SAS output ready for the web.

Option 1. Interactive HTML Editors

HTML Editors or any good text editor allow for powerful search-and-replace functions. For example, to change all negative numbers in a table to display as red, simply global change '>$-' to '><FONT COLOR=red>$-'.

Step 2. UNIX SED utility

In a UNIX production environment of mass-produced web pages one can use the powerful UNIX text-processing commands SED and AWK to issue the global search-and-replace function. What's even better, they can be called directly from your SAS program after generating the output HTML file. SED, is a Stream-oriented, non-interactive EDitor.

```
  x 'sed "s/>\$->/<FONT
COLOR=red>$-/g"  ex3.htm >
ex8.htm';
```

The SED command 's/target/replace/g' filename > newfilename

is used to search for /target text/ and insert /replacement text/, the 'g' option says do all occurrences on each line. The backslash '\' is required because '$' is a keyword in SED saying to only look on the last line. The '\' says take the next character as a literal.

More Powerful Tools (SAS)

Now, make it data-driven

The last technique of reading the data back into SAS for text string manipulation is a great improvement to the process. Now we want to make the process data-driven - Allowing us to highlight any cell, row, column or table ***based on the data values in the cells*** (or in any of the cells in a particular row or columns). These data values do not have to be pre-determined or known in advance.

How can we do it? There must be several ways, but one that I have found to be very powerful, yet easily interpretable is to generalize the above search-and-replace tools with a SAS MACRO and use the power of SAS Formats to identify individual cells. As is often the case, this technique can best be explained though an example or two.

Step 1. Identify target cells

An easy, flexible, and powerful way to quickly identify target cells for highlighting is though picture formats. The examples below insert a character into the display of data values meeting certain pre-set criteria.[1] For example, the numeric picture format 'above' identifies all data values above 50,000.

```
   proc format;
    picture above
    low-50000='000,000,009'
    50000<-high='000,000,009'
(prefix='*');
```

The next picture format 'cfmt' identifies all data values in a particular column.

```
    picture cfmt
   low-high='00,000,009.99'
(prefix='*');


   proc print data=x;
    var x1;
   format above.;
```

[1] This arbitrary character that is inserted is swapped or removed before final display.

The 'above' format produces output (like the above example) where selected numbers (all those above the mean) display as '*9,999', for example. The cfmt format, like typical formats is applied to all the values in a column and can be therefore be used to easily highlight entire columns in the HTML table. Now that we have identified the targeted cells to highlight though a data-driven procedure, we now can automate the process easily with a SAS Macro.

Step 2. Insert HTML Formatting Tags

The SAS System's ease of the input and output of flat files combined with powerful text string manipulation functions allow quick work of HTML formatting changes. Simply read in the HTML file, do a global search-and replace and then write out the file with a data _null_ step. The infile, file and put statements in the data step do the file handling while the TRANWRD function does the text swapping.[2]

```
filename in
 'C:\HTML Highlights\ex3.htm';

filename out
'C:\HTML Highlights\ex5.htm';

%let string = $-;

data _null_;
 infile in missover
length=flen;
 file out;
 length line $92 flag1 3;
 input line $varying.
   flen;
flag1 = index(line,"*");

line=tranwrd(line,trim("$"),
      "<FONT COLOR=red>$-");
  if flag1 then
line=tranwrd(line, "</TD>",
             "</FONT></TD>");
   put line;
  run;
```

[2] Documented in one of the tan manuals - SAS Technical Report P-222, p.67.

Enhancement - highlighting entire rows based on cell values

The next enhancement would be to select not just individual data cells but to highlight entire rows based upon data values in particular cells. Problem: HTML displayed table rows are described in the source file over multiple lines. As a reminder, the structure of a HTML table definition looks like:

```
<TR>
  <TH ……(table header for
         the row label)
 <TD ……(cells definition)
 <TD …… (as many <TD's as
         columns)
   .
   .
   .
</TR>
```

One way to identify logical rows would be to find the targeted cell (with the swap character, '*'), flag the following '<TD' records until hitting an end of the table row, '/TR' and reversing the row order with PROC SORT until the beginning of the table row, '<TR'. But that is less than satisfactory because it involve two passes through the data, which might be acceptable for a small number of HTML tables, but would affect performance with large automated production of lots of web pages.

One nifty solution to this problem utilizes a technique that has been an occasional topic of discussion on the web SAS email discussion list -SAS-L. It uses PROC SQL to identify targeted records based on user input select criteria. It then conveniently stores either the targeted value in a macro variable, or, stores another associated variable for that selected row. In this case, we can store the targeted HTML table row label value, and use it to turn on the highlighting for the entire row.

```
proc sql;
 select state, actual
    format=comma12.
 into :rows separated by '#'
 from (select sum(actual)
        as actual, state
          from user.predsal
          group by state)
 having actual
           gt mean(actual);
quit;
   %put _user_;
```

This procedure grabs the value for the row label for each of the targeted rows (selected based on the table data values), and stores the one (or many) row labels so we can then highlight the entire row in one pass through the data! This technique can be easily made more flexible using macro variables.

An Automatic Flexible MACRO tool

In the spirit of the powerful, flexible, easy to use TAB2HTM macros supplied by SAS Institute, we've build a macro that automates the table cell, column and row highlighting process. It is available from the authors. After inserting a target swap character in your HTML, the macro allows the insertion of the wide variety of HTML formatting tags for individual data cells, rows & columns.

Next Challenge - Uses (and abuses)

A word of caution, however, it is easy to go quickly overboard with easy flexible table formatting - for example, inserting animated GIFs in multiple table cells. Remember, too much highlighting makes it harder to pick out data elements than no highlighting at all!

The remaining challenges are yours - the user - to use restrained table highlighting to draw the report reader's eye and enhance or emphasize the points you want to make. Several more examples of uses of highlighting might be:

1.) general page formatting - every N rows. or every K columns.
2.) outlier or exception reporting.
3.) picking out statistically significant variables.
4.) following a particular variable though the results of several analyses or reports.

Version 7

A word of caution, however, it is easy to go quickly overboard with easy and flexible table highlighting. For table formatting tips, visit the ODS Style Gallery at http://www.sas.com/rnd/base/index-gallery.html to view examples of good SAS generated HTML pages.

Chris Olinger, in SUGI 24 Paper 56 'Twisty Little Passages, All Alike - ODS Templates Exposed' describes using the Version 7 CELLSTYLE-AS statement in an ODS Template to format individual data cells based on the data values. It does not, however, highlight the row or column conditional on the data values in a cell.

SUGI 24 Paper 190 'Getting Stylish with Version 7 Base Reporting' by David Kelly and Sandy McNeill describe using traffic lighting and adding hyperlinks with Version 7 PROCs TABULATE and REPORT. The techniques they describe are very close those discussed here for Version 6.12 - using PROC FORMAT to define data ranges to highlight with PROC TABULATE. PROC REPORT, however, offers the power of data set programming to accomplish the same task.

The main difference in Version 7 HTML formatting is the modification and use of templates and styles to achieve data-driven cell formatting.

Also newly available in Version 7 PROC REPORT is a new _ROW_ symbol used to highlight entire rows as we have done here.

Author Contact Information

Matt Flynn
The Hartford Personal Lines
400 Executive Blvd.
Southington, CT 06489

email: Matt.Flynn@TheHartford.com
phone: (860) 620-6891
fax: (860) 276-2855

Ray Pass
Ray Pass Consulting
5 Sinclair Place
Hartsdale, NY 10530

email: raypass@att.net
phone: (914) 693-5553

World Wide Product Sales Report
Selection of sum(actual) by state for hl.prdsal3, condition: ge mean

	Actual Sales		Predicted Sales	
	Total	Total	Total	Total
State/Province				
Baja California Norte	35,760	35,760	40,226	40,226
British Columbia	50,527	50,527	52,234	52,234
California	60,807	60,807	69,920	69,920
Campeche	25,940	25,940	34,257	34,257
Colorado	58,297	58,297	69,895	69,895
Florida	52,094	52,094	42,438	42,438
Illinois	321,054	321,054	333,362	333,362
Michoacan	33,820	33,820	38,187	38,187
New York	59,219	59,219	64,296	64,296
North Carolina	72,663	72,663	69,179	69,179
Nuevo Leon	28,949	28,949	28,540	28,540
Ontario	48,223	48,223	-46,720	-46,720
Quebec	42,564	42,564	49,159	49,159
Saskatchewan	58,026	58,026	64,046	64,046

Figure 1 - Standard PROC TABULATE output

	Actual Sales		Predicted Sales	
	Total	Average	Total	Average
State/Province				
Baja California Norte	$289,473.00	$502.56	$337,230.00	$585.47
British Columbia	$445,087.00	$772.72	$474,946.00	$824.56
California	$585,194.00	$1,015.96	$625,879.00	$1,086.60
Campeche	$294,193.00	$510.75	$328,204.00	$569.80
Colorado	$587,695.00	$1,020.30	$633,825.00	$1,100.39
Florida	$564,396.00	$979.85	$592,433.00	$1,028.53
Illinois	$2,968,170.00	$1,030.61	$3,095,395.00	$1,074.79
Michoacan	$295,024.00	$512.19	$326,319.00	$566.53
New York	$592,185.00	$1,028.10	$619,317.00	$1,075.20
North Carolina	$596,188.00	$1,035.05	$610,335.00	$1,059.61
Nuevo Leon	$293,945.00	$510.32	$316,780.00	$549.97
Ontario	$433,359.00	$752.36	$469,832.00	$815.68
Quebec	$448,790.00	$779.15	$492,740.00	$855.45
Saskatchewan	$449,338.00	$780.10	$468,602.00	$813.55
Texas	$598,661.00	$1,039.34	$593,099.00	$1,029.69
Washington	$571,753.00	$992.63	$612,197.00	$1,062.84

Figure 3 TAB2HTM output with some manual enhancements

	Actual Sales		Predicted Sales	
	Total	Average	Total	Average
State/Province				
Baja California Norte	$289,473.00	$502.56	$337,230.00	$585.47
British Columbia	$445,087.00	$772.72	$474,946.00	$824.56
California	$585,194.00	$1,015.96	$625,879.00	$1,086.60
Campeche	$294,193.00	$510.75	$328,204.00	$569.80
Colorado	$587,695.00	$1,020.30	$633,825.00	$1,100.39
Florida	$564,396.00	$979.85	$592,433.00	$1,028.53
Illinois	$2,968,170.00	$1,030.61	$3,095,395.00	$1,074.79
Michoacan	$295,024.00	$512.19	$326,319.00	$566.53
New York	$592,185.00	$1,028.10	$619,317.00	$1,075.20
North Carolina	$596,188.00	$1,035.05	$610,335.00	$1,059.61
Nuevo Leon	$293,945.00	$510.32	$316,780.00	$549.97
Ontario	$433,359.00	$752.36	$469,832.00	$815.68
Quebec	$448,790.00	$779.15	$492,740.00	$855.45
Saskatchewan	$449,338.00	$780.10	$468,602.00	$813.55
Texas	$598,661.00	$1,039.34	$593,099.00	$1,029.69
Washington	$571,753.00	$992.63	$612,197.00	$1,062.84

Figure 2- TAB2HTM output in web browser

Figure 4 - TAB2HTM with cell highlights

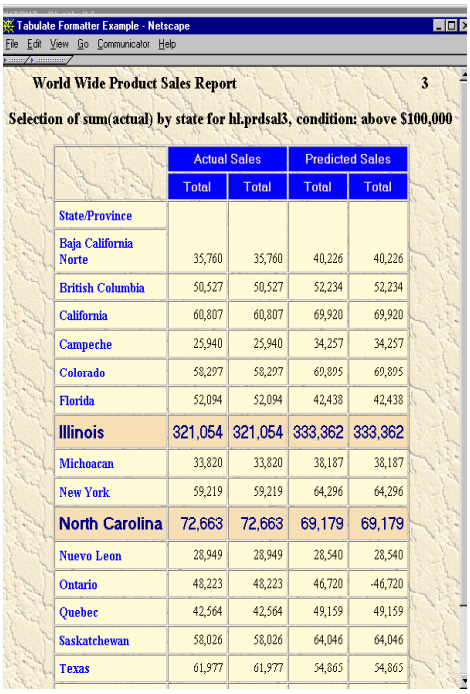

Figure 5 - TAB2HTM with row highlights