Paper 77-26

ODS and Browser Interaction Solutions

Steven Feder, Federal Reserve Board, Washington, D.C.

ABSTRACT

While SAS© ODS readily produces HTML files, some difficulties remain in using the output. Browser interaction in some cases formats default output so that it is difficult to read. Back-end browser manipulation offers solutions, but redirecting the output through PROC REPORT still works better for some applications.

INTRODUCTION

Procedure output to HTML files occasionally runs into snags because the browser on which the output is viewed plays a role in formatting what the end-user sees. Output that makes perfect sense in the HTML file may appear distorted in the browser.

DISTORTED PROC MEANS OUTPUT

The following PROC MEANS example seems simple enough:

```
ods html
    body='ex1_body.html'
    frame='ex1_frame.html'
    contents='ex1_contents.html'
    path='i:\dev\sas8\template\means' (url=none);

data test;
     input a b c d;
     n=_n_;
     label
        a='a One Variable Label'
        b='b Complicated Variable Label that
            Results in Wrapping'
        c='c Another Variable Label'
        d='d One More Variable Label - Medium';
 cards;
 1 2 3 4
 2 3 4 5
 ;
 run;

title 'A SIMPLE PROC MEANS';
title2 'BUT THE LABEL VALUES WRAP';
title3 '(AS VIEWED ON A BROWSER)';

proc means;
run;
```

But, viewed from a frame, the output becomes distorted and unacceptably difficult to understand because a useful PROC MEANS feature interacts with a useful HTML feature to misalign output columns.

The PROC MEANS creates a column of the variable labels, which is convenient for the end-user and works well enough in the standard .LST file. In a listing file with the default linesize for a hardcopy page, the variables and labels are repeated on numerous lines until all the columns have been printed. In this environment output may run into the limits of the printed page, but these limits are clear.

The comparable body of the HTML output is, of course, a table. The browser determines a width and aligns the individual columns. But the width of the columns displayed also depends on the overall width the browser sets for the table. This can result in columns with a width less than the length of any given label value. Then the browser wraps the label text to the next line. While this may seem counter-intuitive, the browser is balancing the drawback of wrapping text with the drawback of text spread out excessively. The text wrapping by itself still would not misalign the rows if there was a set of cells for each variable. Instead, there is one cell for each column, with values separated by line breaks. For example, (after paring away the non-relevant tags for clarity within this paper) the actual data in the first three columns appear as:

```
<TD>
a<br>
b<br>
c<br>
d<br>
</TD>
<TD>
a One Variable Label<br>
b Complicated Variable Label that Results in
Wrapping<br>
c Another Variable Label<br>
d One More Variable Label - Medium<br>
</TD>
<TD>
2<br>
2<br>
2<br>
```

```
2<br>
2</TD>
```

The browser does not compensate for the line wrapping in one column by spacing the other columns because the line breaks do not create an HTML connection between the vertical alignments of the rows.

BROWSER FORMATTING

There are straight-forward if clumsy back-end solutions to the wrapping. Depending on how much the text is wrapping, reducing the browser font may correct the misalignment. For a limited and agreeable target audience this may make perfect sense and even produce more easily interpreted output. But the usual tradeoff between readability and packing information onto a page comes into play. Viewing the body file without going through the frame file, i.e., without the contents on the side, provides a similar effect. For a limited number of pages of relatively simple output this could work.

A more fundamental way to change the appearance is to edit the HTML body file manually to add a width to the table tag so that (with non-relevant tags pared away)

```
<TABLE>
```

becomes

```
<TABLE width=1000>
```

This supersedes the default determination by the browser of the table width. Since, with a little experimentation, this can leave plenty of space for the text without wrapping, this can work for essentially any output. The text may be quite spread out but is easy to interpret. Only the required manual editing prevents this from being an acceptable solution.

IS THERE A PROC TEMPLATE SOLUTION?

The above table tag solution would work adequately if the SAS program could specify the width for the table tag. The obvious place for this would be in a PROC TEMPLATE modification of the standard PROC MEANS HTML output. Once created and stored the modified template could be referred to with little complication. Unfortunately, no option for specifying table width is available in PROC TEMPLATE. Also, no option to set overall table font size is available.

STANDARD CODE WORK-AROUNDS

One way around the PROC MEANS table problem is to avoid the PROC MEANS entirely. Transpose the data and run a PROC REPORT on the output dataset:

```
proc transpose data=test out=tpose label=label;
   by n;
   var a b c d;
run;

title 'PROC REPORT REPLACES
        PROC MEANS';
title2 'BUT FIRST TRANSPOSE';
title3 '(AS VIEWED ON A BROWSER)';

proc report nowd;
    col _name_ label col1, (n mean std min max);
    define _name_ / group ' ';
    define label  / group ' ';
    define col1   / ' ';
    define n      / format=3.;
    define std    / format=10.8;
run;
```

This produces clean output and highlights the advantage of having each value in a separate cell as produced by PROC REPORT. The browser still wraps the long labels, but the values for each row are in a series of HTML-defined cells: (with non-relevant tags pared away)

```
<TR>
<TD>a</TD>
<TD>a One Variable Label></TD>
<TD>  2</TD>
<TD>      1.5</TD>
<TD>0.70710678</TD>
<TD>        2</TD>
</TR>
<TR>
<TD>b</TD>
<TD>b Complicated Variable Label that Results
in Wrapping</TD>
<TD>  2</TD>
<TD>      2.5</TD>
<TD>0.70710678</TD>
<TD>        2</TD>
<TD>        3</TD>
</TR>
```

The rows remain aligned. Though effective in this case, the PROC TRANSPOSE code could become complicated with a large number of variables, and a large enough dataset could present an efficiency problem.

An alternative approach uses the PROC MEANS to generate a dataset, transposes the dataset, and runs a PROC REPORT:

```
proc means data=test noprint;
   output out=temp;
run;

proc transpose data=temp
   out=temp2 name=VARIABLE
   label=LABEL;
   id _stat_;
   idlabel _stat_;
run;
title 'PROC MEANS';
title2 'FOLLOWED BY A TRANSPOSE';
title3 '(AS VIEWED ON A BROWSER)';

proc report nowd data=temp2;
   where variable not in('_TYPE_' '_FREQ_');
   col variable label n mean min max std;
   define variable / display 'VARIABLE';
   define label / display 'LABEL';
   define n / display ;
   define mean / display;
   define min /  display ;
   define max /  display ;
   define std /  display ;
run;
```

This produces an aligned table similar to the last example, but the code is more readily adapted to future datasets; since the PROC MEANS output dataset is standard, the PROC TRANSPOSE can be standard. This solution is ideal except for the repetition of this extra coding necessary if multiple datasets are to be analyzed.

REWRITING THE OUTPUT FILE

A trick to avoid work-around code entirely is to add a datastep that will read in the HTML body file and rewrite it with the width specified for all the table tags:

```
data temp;
   length c $200;
   infile
   'i:\dev\sas8\template\means\ex1_body.html'
      pad missover;
   input c $ 1-200;
   c=tranwrd(c,'<TABLE','<TABLE
      width="1000"');
   file 'i:\dev\sas8\template\means\
      ex1_body_rewrite.html'  ;
   put c;
run;
```

This is still burdensome, but if many PROC MEANS tables are in one file, this reduces the number of extra steps.

CONCLUSION

To keep the browser from distorting the appearance of your HTML output, experimenting with browser and HTML settings as well as with code work-arounds leads to a greater understanding of the output formatting. This can help to select the solution most effective for your application.

TRADEMARK INFORMATION

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries.

CONTACT INFORMATION

Steven Feder
Federal Reserve Board, Mail Stop 157
Washington, D.C. 20551
202-452-3144
email: steven.h.feder@frb.gov