Paper 080-2008

The Look: Conforms to Section 508, Looks Like PROC TABULATE

Mikhail Gruzdev & Blake Sanders, U.S. Census Bureau, Wahsington, DC

ABSTRACT

It is possible to create HTML code that conforms with Section 508 requirements with very little trouble. You have to be able to dissect your data and realize the patterns in your tables.

INTRODUCTION

Many people dread the sound of "Section 508". Section 508 requires that federal agencies provide access to data for those with disabilities. This data must be comparable to data provided to those without disabilities. For those who've had no experience with designing for Section 508, it seems like a long and laborious task. With the use of PROC SUMMARY and PROC PRINT to produce results and Cascading Style Sheets (CSS) to manage the display of the data, the result will be tables that are accessible to everyone but look just like you'd see from PROC TABULATE.

PROBLEM

While PROC TABULATE produces some great results, the tables it produces for HTML using the SAS tagset for Section 508 compliance are less than stellar. For instance, the dataset …

Table 1 Sample data set

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
Meat	2005	343	878
Oil	2004	222	555
Oil	2005	777	999

… and the SAS code…

```
proc tabulate data=one;
  class product year;
  var import export;
  table product *(year all='Total Year') all= 'Total Product',
                                    (import export) * (sum);
  run;
```

… using the SAS ODSMARKUP tagset found at http://support.sas.com/faq/ts_qa/ods508.html produce the table…

Table 2 Results from PROC TABULATE with 508 Markup

		IMPORT	EXPORT
		Sum	Sum
PRODUCT	YEAR	999	777
Meat	2004		
	2005	343	878

		IMPORT	EXPORT
		Sum	Sum
	Total Year	1342	1655
Oil	Year	222	555
	2004		
	2005	777	999
	Total Year	999	1554
Total Product		2341	3209

The HTML code behind the table makes it confusing to know how the first numbers in each section will be labeled in a screen reader. "999", under "Meats, 2004, Imports" would be labeled "Import, Sum, Product, Year, Meat, 2004" In an screen reader.

By processing the data the correct way using PROC SUMMARY and PROC PRINT (the code for which we can provide upon request), we can produce a file that is much more straight forward.

Table 3 Data after processing it with PROC SUMMARY and PROC PRINT

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
Meat	2005	343	878
Meat	Total Year	1342	1655
Oil	2004	222	555
Oil	2005	777	999
Oil	Total Year	999	1554
Total Product		2341	3209

Using SAS to produce HTML code, "999" would be labeled "Import, Meat, 2004". That's much easier to understand than the previous version.

This version, however, is cluttered. Do we need to see the name of the product listed multiple times?

No. Cascading Style Sheets (CSS) will help clean this up.

SOLUTION

Take another look at the data file we want to output.

Table 4 Processed data

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
Meat	2005	343	878

Meat	Total Year	1342	1655
Oil	2004	222	555
Oil	2005	777	999
Oil	Total Year	999	1554
Total Product		2341	3209

PRODUCT is the first dimension and YEAR is the second dimension. There are at least two rows of YEAR under each PRODUCT: At least one year row and the TOTAL YEAR row. So, you could say that each group has a TOP record (e.g. "MEAT, 2004") and a BOTTOM record ("Meat, TOTAL YEAR"). If there are additional records, they could be considered MIDDLE records since they're between the TOP and BOTTOM records.

Table 5 Processed data with TOP/BOTTOM flags for category variables.

PRODUCT	FLAG1	YEAR	FLAG2	IMPORT	EXPORT
Meat	*TOP*	2004	*BOTTOM*	999	777
Meat	*MIDDLE*	2005	*BOTTOM*	343	878
Meat	*BOTTOM*	Total Year	*BOTTOM*	1342	1655
Oil	*TOP*	2004	*BOTTOM*	222	555
Oil	*MIDDLE*	2005	*BOTTOM*	777	999
Oil	*BOTTOM*	Total Year	*BOTTOM*	999	1554
Total Product	*BOTTOM*		*EMPTY*	2341	3209

Now, apply that idea to the output produced by PROC TABULATE. A PROC TABULATE table would look like…

Table 6 Example of PROC TABULATE output

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
	2005	343	878
	Total Year	1342	1655
Oil	2004	222	555
	2005	777	999
	Total Year	999	1554
Total Product		2341	3209

How do we get SAS to create a similar table? Take away all of the borders.

Table 7 Processed data without borders.

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
Meat	2005	343	878
Meat	Total Year	1342	1655
Oil	2004	222	555
Oil	2005	777	999
Oil	Total Year	999	1554
Total Product		2341	3209

Now, add a border to the top and right of the table.

Table 8 Add borders to the TABLE

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777

Meat	2005	343	878
Meat	Total Year	1342	1655
Oil	2004	222	555
Oil	2005	777	999
Oil	Total Year	999	1554
Total Product		2341	3209

Add borders to the bottom and left of every cell that's used for a column header.

Table 9 Add borders to the column headers.

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
Meat	2005	343	878
Meat	Total Year	1342	1655
Oil	2004	222	555
Oil	2005	777	999
Oil	Total Year	999	1554
Total Product		2341	3209

Place borders on the bottom and left of every cell that's a BOTTOM of it's group. Also, right justify the data in the analysis variable fields.

Table 10 Add borders to the BOTTOM cells.

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
Meat	2005	343	878
Meat	Total Year	1342	1655
Oil	2004	222	555
Oil	2005	777	999
Oil	Total Year	999	1554
Total Product		2341	3209

In the cells that are TOP or MIDDLE of their group, place a border on the left side of the cell.

Table 11 Add borders to the TOP and MIDDLE cells.

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
Meat	2005	343	878
Meat	Total Year	1342	1655
Oil	2004	222	555
Oil	2005	777	999
Oil	Total Year	999	1554
Total Product		2341	3209

Finally, if we have an empty cell at the bottom of the table, place a border at the bottom of the cell.

Table 12 Add a border to the EMPTY cell.

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777

Meat	2005	343	878
Meat	Total Year	1342	1655
Oil	2004	222	555
Oil	2005	777	999
Oil	Total Year	999	1554
Total Product		2341	3209

Finally, let's display all but the first (i.e. TOP) products in white text on a white background.

Table 13 Display all of the PRODUCTS (except for TOP) in white text on a white background.

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
	2005	343	878
	Total Year	1342	1655
Oil	2004	222	555
	2005	777	999
	Total Year	999	1554
Total Product		2341	3209

All of this is taken care of with the Cascading Style Sheet. You just need to know which cells need which styles.

CODE

To flag the cell with the right type as exemplified in Table 5, we ran the following code:

```
data temp;
    retain product flag1 year flag2 import export;
    length flag1 flag2 $6;

/* Set the data that's been processed. */

    set processeddata;
    by product year;

/* Set values for FLAG1 – PRODUCT status. */

    if first.product then flag1 = 'TOP';
    else if not (first.product or last.product) then flag1 = 'MIDDLE';
    else if last.product then flag1 = 'BOTTOM';
    if product = "" then flag1 = 'EMPTY';

/* Set values for FLAG2 – YEAR status. */

    if first.year then flag2 = 'TOP';
    else if not (first.year or last.year) then flag2 = 'MIDDLE';
    else if last.year then flag2 = 'BOTTOM';
    if year = "" then flag2 = 'EMPTY';

run;
```

Since we know that the analysis fields of IMPORT and EXPORT are BOTTOM cells, we don't have to worry about explicitly labeling them as such.

To generate the HTML code with the appropriate styles, we ran this code:

```
data _null_;
```

```
    set temp end=eof;
    file _webout;
    if _n_ = 1 then do;
       put '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
                 4.01 Transitional//EN">';
       put '<html>';
       put '<head>';
       put '<title> 508 Accessible table </title>';
       put '<meta http-equiv="Content-Type"
                 content="text/html; charset=iso-8859-1">';
       put '<link href="ftd-basic.css"
                 rel="stylesheet" type="text/css">';
       put '<link href="dataapplication.css"
                 rel="stylesheet" type="text/css">';
       put '<link href="migtest.css"
                 rel="stylesheet" type="text/css" media="all"/>';
       put '<link href="migprint.css"
                 rel="stylesheet" type="text/css" media="print"/>';
       put '</head>';
       put '<body>';
       put "<table width='100%' border='0' cellpadding='5'
                 cellspacing='0' class='datatable'>";
       put "<tr>";
       put "<th scope='col'>PRODUCT</th>";
       put "<th scope='col'>YEAR</th>";
       put "<th scope='col'>IMPORT</th>";
        put "<th scope='col'>EXPORT</th>";
       put "</tr>";
   end;

   put "<tr>";

/* Process the PRODUCT data of the record first. */

    if flag1 = 'TOP' then do;
          put "<td scope='row' class='migleft'>" product"</td>";
    end;
    else if flag1 = 'MIDDLE'  then do;
          put "<td scope='row' class='migleft'>
                <span class ='blankcell'> " product" </span>
                %nrstr( )</td>";
    end;
    else if flag1 = 'BOTTOM' then do;

          if product = “TOTAL PRODUCT” then do;
                put "<td scope='row' class='migbottomleft'> " product" </td>";
          end;
                else
          do;
                put "<td scope='row' class='migbottomleft'>
                      <span class ='blankcell'> " product" </span>
                      %nrstr( )</td>";
          end;

    end;

/* Process the YEAR data of the record next. */

    if flag2 = 'TOP' then do;
          put "<td scope='row' class='migleft'>" year "</td>";
    end;
```

```
      else if flag2 = 'MIDDLE'  then do;
            put "<td scope='row' class='migleft'>
                  <span class ='blankcell'> " year " </span>
                  %nrstr( )</td>";
      end;
      else if flag2 = 'BOTTOM' then do;

            put "<td scope='row' class='migbottomleft'> " year " </td>";

      end;
      else if flag2 = 'EMPTY' then do;

            put "<td scope='row' class='migbottom'>%nrstr( )</td>";

      end;

   /* Display the data fields. */

      put "<td scope='row' class='migbottomleft datafield'>"  import "</td>";
      put "<td scope='row' class='migbottomleft datafield'>"  export "</td>";

      put "</td>";
      put "</tr>";

      if eof then do;
            put "</table>";
            put "</body>";
            put "</html>";
      end;

   run;
```

Now that we have the code in place, let's talk about how the Cascading Style Sheets (CSS) and their associated style affect the appearance of the page.

FTD-BASIC.CSS contains:

The BODY tag ensures that data is displayed in black Arial text on a white background.

```
body {
      font-family: Arial, Helvetica, sans-serif;
      font-size: medium;
      color: #000000;
      background-color: #FFFFFF;
}
```

DATAAPPLICATION.CSS contains:

All areas with a "datafield" class will right justify their contents.

```
.datafield {
     text-align: right;
}
```

MIGTEST.CSS contains (in the order they were mentioned earlier in the paper):

"datatable" displays a border at the top and right of the table.

```
.datatable {
```

```
        border-top-width: thin;
        border-right-width: thin;
        border-top-style: solid;
        border-right-style: solid;
        border-top-color: #000000;
        border-right-color: #000000;
}
```

“datatable th” displays a border on the bottom and left of any table cell that’s a column header (“<th>”).

```
.datatable th {
        border-bottom: thin solid #000000;
        border-left: thin solid #000000;
}
```

“migbottomleft” displays a border on the bottom and left of any BOTTOM cell.

```
.migbottomleft {
        border-bottom-width: thin;
        border-left-width: thin;
        border-bottom-style: solid;
        border-left-style: solid;
        border-bottom-color: #000000;
        border-left-color: #000000;
}
```

“migleft” displays a border on the left of any TOP or MIDDLE cell.

```
.migleft {
        border-left-width: thin;
        border-left-style: solid;
        border-left-color: #000000;
}
```

“migbottom” displays a border at the bottom of the empty cell on a TOTAL line.

```
.migbottom {
        border-bottom-width: thin;
        border-bottom-style: solid;
        border-bottom-color: #000000;
}
```

“blankcell” displays the contents of a cell in white text on a white background.

```
.blankcell {
 color: #FFFFFF;
 background-color: #FFFFFF;
}
```

When you print the table as-is, almost everything shows up as expected except for the “blankcell” fields.

Table 14 White-on-white "blankcells" when printed.

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
Meat	2005	343	878
Meat	Total Year	1342	1655
Oil	2004	222	555
Oil	2005	777	999
Oil	Total Year	999	1554
Total Product		2341	3209

The “blankcell” areas are printed in gray instead of white on white.

MIGPRINT.CSS is the last style sheet loaded, and it only contains one style definition.

```
.blankcell {
      display: none;
}
```

Since this is the last CSS loaded (and it only applies when the output is printed), it overwrites the first “blankcell” definition and just plain does not display the additional instances of PRODUCT.

Table 15 Using the MIGPRINT style sheet, the "blankcells" are just not displayed.

PRODUCT	YEAR	IMPORT	EXPORT
Meat	2004	999	777
	2005	343	878
	Total Year	1342	1655
Oil	2004	222	555
	2005	777	999
	Total Year	999	1554
Total Product		2341	3209

These are cells we’ve also inserted a non-breaking space. When the PRODUCT is not displayed, we need something to be in the cell. Otherwise, the browser will not display anything (even the borders).

CONCLUSION

With planning and a little imagination, you can produce the same polished results that come from other procedures (e.g. PROC TABULATE) with better results. Using CSS and HTML, you can produce tables that are both accessible and professional looking.

REFERENCES

- To see this concept in action, take a look at the Census Bureau’s RELATED PARTY database application at http://sasweb.ssd.census.gov/relatedparty/
- Section 508: http://www.section508.gov/
- Web Accessibility Initiative: http://www.w3.org/WAI/
- SAS Federal Government Accessibility: http://www.sas.com/govedu/accessibility.html

CONTACT INFORMATION

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

Mikhail Gruzdev
U.S. Census Bureau / Foreign Trade Division
4600 Silver Hill Rd.
6K502B
Washington, DC 20233
Work Phone: 301-763-2206
E-mail: mikhail.g.gruzdev@census.gov

Blake Sanders
U.S. Census Bureau / Foreign Trade Division
4600 Silver Hill Rd.
6K505
Washington, DC 20233
Work Phone: 301-763-6965
E-mail: blake.r.sanders@census.gov