Paper 020 -2013

Using the SAS® datastep to generate HTML or text based "mark-up"

Matthew T. Karafa, PhD, Cleveland Clinic Foundation, Cleveland, Ohio

Abstract

The author presents macros which produce reports direct to MSWord compliant HTML, thus demonstrating an alternative method to create MSWord documents from SAS. The first step is to create a mock up of the table in an external mark-up editor, then use SAS to produce the text that creates the file, interspersing the required data between the mark-up tags. These macros demonstrate a way to increase the control and flexibility over what is available via the traditional ODS RTF or HTML mechanism. Further, via this method, any text-based mark-up language (HTML, RTF, LaTeX, etc.) can be produced with a minimal effort.

Introduction

ODS has been a great addition to all of our SAS® tool belts. The ability to manipulate the output from specific procedures and piece them together into a table programmatically rather than on the fly helps generate production ready results and lets us systematically repeat the process. However, customizing those tables can be cumbersome and trying to tweak an ODS template can be a daunting task to a new programmer for example trying to produce tables with more complex layouts with spanning columns and the like. However if you know a little HTML, or are willing to make MSWord (or your favorite HTML editor of choice) teach it to you, it is possible to produce very complex tables via Base SAS and the data step. Further since Microsoft as so kindly added in specific tags to make its word documents look spiffy, we can use this technique to create MSWord ready tables as well!

Writing the HTML File

Since HTML is a plain-text markup language, SAS can generate it. Thanks to the `file` and `put` statements. The basic methodology is to use the data step to create a file and "put" our HTML code into the file by brute force thus using a data step to create the text file that contains the HTML. Constructing such a data step is a fairly simple task in SAS the general structure:

```
data _null_;
      file "SomeFileName.Ext";
      put "Line 1 to be put to the output file";
      put "Line 2 to be put to the output file";
      ...
      put "Line K to be put to the output file";
      put "This line is last";
run;
```

Reverse Engineering the HTML

OK so getting the file made is relatively easy code to achieve, but that begs the question what tags and text should be put out to make the HTML table? To determine the needed HTML code, the first step is to create a mock up of the table we wish to create in MSWord. This can be as simple or as complex as we like so long as the cells of interest can easily be found or searched for. In the mock table, generic terms that can later be replaced with SAS macro variables are used rather than actual values. Next, save this file using "Save as Web Page" and open the resulting HTML file in your favorite text editor.

So for example if we wanted to make a table that looked something like this:

Table 1: Comparison of Treatment X vs Y (Continuous Measures)

	Treatment X		Treatment Y				
Factor	N	Median (Q1,Q3)	N	Median (Q1,Q3)	Units	OR (CI)	P value[A]
SAS Factor	25	120 (90, 280)	22	180 (130, 290)	1	1.12 (1.01, 1.20)	P<0.001

A: Wilcoxon Rank Sum P-Value

To make the mock up we would use for coding, replace the actual values with placeholders. Typically I use macro names I plan to use in the code I'm developing, but easy to search for words are also good. The resulting Word table looks like this:

MainTitle

	Gp1_Label		Gp2_Label				
Factor	N	Median (Q1,Q3)	N	Median (Q1,Q3)	Units	OR (CI)	P value
var_ label_i	Gp1SS_i	Gp1Stats_i	Gp2SS_i	Gp2Stats_i	Units_i	OR_i	Pvalue_i

FootNote

A search through the resulting HTML document finds a line that has the form:

```
<p class=MsoNormal>
<span style='font-size:10.0pt;mso-bidi-font-size:12.0pt'>
_MainTitle_
</span>
</p>
```

so in the SAS DATA ta _null_ we would need the following command:

```
put "<p class=MsoNormal>";
put "<span style='font-size:10.0pt;mso-bidi-font-size:12.0pt'>";
put _MainTitle_;
put "</span></p>";
```

If we store the title to be used in a DATA step variable called "_Maintitle_", the data _null_ step will replace that with its value in the resulting text file. Note also that the tags can be split up for legibility. Thus the reader of the program can see the HTML tags that are put into the text file as well as the SAS dataset variables that are used in the process. Searching for the other placeholders and constructing a similar series of put statements can generate the remainder of the table.

For repeated data points (e.g. the Group 1 and Group 2 data in the above example) we can use looping logic to code more efficiently:

```
    array Events(*)__NEvents __NNonEvents ;
    array Percents(*) P1 P2;

         do i = 1 to dim(Events);
             put "<td valign=top>";
             put "<p class=Textbody align=center " @;
             put " style='text-align:center'>";
             put "<span style='font-size:10.0pt; " @;
             put " mso-bidi-font-size:11.0pt'>" @;
             put Events[ i]  @;
             put "<o:p></o:p></span></p>";
             put "</td>";
             put "<td valign=top>";
             put "<p class=MsoNormal align=center  " @;
             put " style='text-align:center'>";
             put "<span style='font-size:10.0pt; " @;
             put " mso-bidi-font-size:12.0pt'>" @;
             put Percents[ i] @;
             put "<o:p></o:p></span></p>";
             put "</td>";
         end;
```

Admittedly with only 2 columns this is a bit tedious, but the extension possibilities are there. Branching logic is also possible if something is not always going to be in your talble:

```
             %if &Oddsratios = 1 %then %do;
                 put "<td valign=top>";
                 put "<p class=MsoNormal align=center" @;
                 put " style='text-align:center'>";
                 put "<span style='font-size:10.0pt; " @;
                 put " mso-bidi-font-size:12.0pt'>" @;
                 put ORCI @;
                 put "<o:p></o:p></span></p>";
                 put "</td>";
             %end;;
```

Examples of Use
%Cattable() and %Conttable()
The details of these macros usage is published elsewhere [1], but they rely on this method of presenting the final tables. %Cattable is used for categorical factors and has a sample call looking like:

```
    %cattable(ds=Test, GroupVar = Group,
            vlist =Male Black Handedness/E Small,
            DefaultStyle=D, DefaultCutoff=0.15,
            ColP=0, OddsRatios=1,
            ListingFile = T, DelimitedFile = F, HtmlFile =  T,
            owrite_html = T, ofile_html =./SGF2013.html,
            MainTitle=%str(Table 1: Comparison of Group 1 vs. Group 0)
            );
    run;
```

This produces output in the list (`ListingFile=T`) and HTML output (`HTMLFile=T`) stored in the file "./SGF2013.html". Note that in order for the title to correctly parse blanks we need to use `%str()` in the call.

Table 1: Comparison of Group 1 vs. Group 0

Factor	Level	Total	Risk Group N	Risk Group (%)	Reference Group N	Reference Group (%)	Odds Ratio (CI)	P value
Male							9.3 (3.6 ,24.2)	<0.001
	Female	70	6	8.6	64	91.4		
	Male	75	35	46.7	40	53.3		
Black							1.02 (0.30 ,3.4)	0.99F
	White	131	37	28.2	94	71.8		
	Black	14	4	28.6	10	71.4		
Handedness								0.86F
	1:Left	8	3	37.5	5	62.5	1.6 (0.32 ,7.5)	
	3:Right	43	12	27.9	31	72.1	1.0 (REF)	
	2:Both	94	26	27.7	68	72.3	0.99 (0.44 ,2.2)	
Small							3372937.6 (0.00 ,3.1)	<0.001F
	Non-Event	132	28	21.2	104	78.8		
	Event	13	13	100.0	0	0.00		

Similarly, the sample call below produces a nice little table for continuous measures. This mixes mean (SD) and median (quartiles) together well, and uses the appropriate parametric or non-parametric test which corresponds with the asked for summary statistics:

```
      %conttable(ds = Test, GroupVar = Group,
                 vlist = Age+10 LOS Cost/w+25 Satis/w,
                 DefaultStyle = T, DefaultTtype = U, PrintTvalue = 0,
                 MainTitle = %str(Table 2: Comparison of Group 1 vs. Group 0
      [cont.]),
                 OddsRatios = 1, qtldecplace = 1, Estdecplace = 1,
                 Unitdecplace  =  1,
                 ListingFile = T, DelimitedFile = F, HtmlFile = T,
                 owrite_html = F, ofile_html =./SGF2013.html );
      run;
```

Table 2: Comparison of Group 1 vs. Group 0 [cont.]

Factor	Risk Group N	Risk Group Statistics	Reference Group N	Reference Group Statistics	Odds Ratio (CI)	P value
Age	41	37.7(20.3)	104	36.4(21.8)	1.03 (0.87 ,1.2)	0.72S
LOS	41	136.7(230.0)	104	168.7(258.7)	1.00 (1.00 ,1.00)	0.47S
Cost	41	992.0 (975.0, 1024.0)	104	1001.0 (972.5, 1017.0)	1.07 (0.81 ,1.4)	0.79W
Satis	41	3.0 (3.0, 3.0)	104	3.0 (3.0, 3.0)	1.01 (0.72 ,1.4)	0.97W

%Corr()

The native correlation procedure in SAS produces fine output, but sometime you need it formatted a little differently, the macro provided here uses the HTML table technique to produce a table that provides confidence limits around the estimated rho values: This macro I use takes a list of vars and a list of "with variables" and produces pairwise correlations between each of the possible pairs. It can handle both Spearman and Pearson correlations, as well as partial correlations, correlations by sub group and weighted data as well. The sample call looks like this:

```
%corr(ds = Test, vlist = Age LOS Cost, wlist=Satis sati2,
      CorrType=S, alpha=.05, Estdecplace =2, CI_decplace =2,
      MainTitle = %str(Table 3: Correlation with Satisfaction
Measurements),
      FootNote  = %str(A: Spearman Correlation Coefficients),
      ListingFile=T, DelimitedFile = F, HtmlFile = T,
      owrite_html= F, ofile_html =./SGF2013.html );
```

Table 3: Correlation with Satisfaction Measurements

With	**Var**	**N**	**rho**	**95% CI**	**P value**[A]
Satis					
	Age	145	-0.15	(-0.31,0.01)	0.071
	LOS	145	-0.00	(-0.17,0.17)	0.99
	Cost	145	0.03	(-0.14,0.19)	0.75
Satisfaction					
	Age	145	-0.15	(-0.31,0.01)	0.071
	LOS	145	-0.00	(-0.17,0.17)	0.99
	Cost	145	0.03	(-0.14,0.19)	0.75

A: Spearman Correlation Coefficients

%DSI()

%DSI() takes a comma separated data file defining data “rules” with fields including variable name, type, valid values and ranges. These metadata are then applied to the data set using internal macros that report the records and values that violate the given set of rules. These are then organized using the HTML techniques described here into a fairly concise, MSWord compliant HTML document, which can be returned to the client for action. For more details on %DSI check out [2].

```
%DSI(DS= MyData, Rules= "C:\data\datadict.csv",
     RulesDlm=%str(','), ListingFile=T, HtmlFile=T,
     owrite_html=F, ofile_html=./Temp.html, PageBreakAtSplit=F,
     AppendToPrevRuns=F, ReportTitle=, DEBUG = 0);
```

The sample report below is thus produced (Note: I have changed the ID’s and blacked out dates to protect the innocent.) The nice thing about this report is unlike so many we produce it sorts things by ID so that all the troubles for a given ID can be addressed at once. In our field, this is likely how physicians will address problems in the data.

Table 4: Sample of 3 ID’s with problems from %DSI()

Problem ID #001

Problem	Variable Involved	Value of Involved Variable
0 : Missing Values	dcg_dt	.
4 : Gender Male or Female	gender	6
7 : Check Valid DOB	dob	12JUN[redacted]

Problem ID #007

Problem	Variable Involved	Value of Involved Variable
6 : Age between 12 and 35	age	55
2 : Check Valid Date	admit_dt	17DEC[redacted]
3 : Check Valid Date	dcg_dt	26DEC[redacted]
7 : Check Valid DOB	dob	25OCT[redacted]

Problem ID #042

Problem	Variable Involved	Value of Involved Variable
0 : Missing Values	dbp	.
5 : Race White Black Hispanic Other	race	K
6 : Age between 12 and 35	age	63
2 : Check Valid Date	admit_dt	15AUG[redacted]
3 : Check Valid Date	dcg_dt	24AUG[redacted]
7 : Check Valid DOB	dob	16JUL[redacted]
7 : Check Valid DOB	dob	13MAR[redacted]

The Next Steps
It is a short leap from the HTML data to other forms of text based mark up language (xml, LaTeX, etc.) Once the markup tags delimiters that go between the data points are known, the file and put statements give the user a lot of power to create text-based documents to be taken in to any form of document.

If you strip away all the special meaning from the markup tags, they become simply another form of delimiter between the data that you wish to present in Tabular form. Thus we SHOULD be able to parse that "delimited" data set and generate it using the datastep we all know and love. Thus, the ultimate goal would be to design a SAS macro that could read a "mock table" based on HTML or other markup rules as a SAS dataset. The trick would be to then use key words within the HTML document (e.g. _Title_1_, _Title_2_, _Data_Sub_I_, _FOOTNOTE_) and create the table without ever needing to hard code the HTML data directly.

Something to ponder for SGF 2014!

References

1. Karafa MT, "Illustrating generation of MSWord tables via HTML with the %cattable and %contable macros." www.mwsug.org/proceedings/2006/stats/MWSUG-2006-SD02.pdf
2. Karafa MT, Thornton J, "Data Set Investigator - Automated Exception Reporting from an electronic data dictionary with %DSI()." www.mwsug.org/proceedings/2008/appdev/MWSUG-2008-A02.pdf

Contact Information
Matthew T. Karafa, PhD
Quantitative Health Sciences, Cleveland Clinic Foundation
9500 Euclid Avenue
Cleveland, Ohio 44195
Phone: (216) 445 – 9556
Fax: (216) 444 - 8021
Email: karafam@ccf.org

Copies of the macros listed available at: www.sascommunity.org/wiki/User:Mkarafa