Paper RF-08-2013

Improve Your ODS Experience with These Essential VBScript Tools

Rose Grandy, Abbott Laboratories, Abbott Park, IL

ABSTRACT

While so much is available in version 9.2 and 9.3 of SAS® to make practically perfect submission-ready output using the Output Delivery System (ODS), there are still a few important things that just cannot be done with tools available within SAS. This paper will present some very useful SAS macros that will write and execute VBScript code to: convert ODS RTF files to true DOC files; merge table cells in ODS RTF files; convert RTF or DOC files to PDF; and, convert XML files created using ODS TAGSETS.EXCELXP to true Excel files.

INTRODUCTION

So, you are a pretty experienced SAS programmer. You know how to make a professional looking table with the ODS tools available in SAS. One day an assignment comes your way with a table shell as an example. You look it over – and it seems pretty straight forward. Granted, some sets of columns don't have borders between them but this can be handled with some well placed style statements. There are some bolded rows but these can also be handled with a simple call define statement in a COMPUTE block in PROC REPORT. But then you notice that there are merged cells. This could be a problem. And then, the note…. "*Dear Sally Q. Programmer, Can you please make your output look EXACTLY like this table shell? Also, we'll need a Word and a PDF version of every file. Can you please be sure to send a DOCX file and not an RTF file? The programmer who had your job before you always sent RTF files, it presented tremendous challenges for us to manually convert each file to a DOCX file and make it compatible with the document management system. We just don't have the manpower to do it right now. Also, please send a copy of the data in Excel….*"

If all or part of this has happened to you (or you just want to impress someone with what you can do in SAS), this paper provides some simple tools that you can use.

Each of the SAS macros presented in this paper will write a VBScript program to handle one task, execute that program and then remove the program – all from within a SAS program.

For the sake of space, comments and some of the basic error checking that would normally be included in the example macro code have been removed. All of these tools have been tested for Microsoft Office 2010.

VBSCRIPT: WHAT? WHERE? WHY? HOW?

VBScript is a scripting language developed by Microsoft. Essentially, it is a lite version of Visual Basic that comes with every installation of Microsoft Windows. So, if you're not working in a Microsoft Windows environment, these tools are probably not going to work for you. VBScript files are text files and have a .VBS extension.

VBScript can automatically open Microsoft Word or Microsoft Excel, perform some tasks and then exit out, all accomplished within a SAS session. This makes it possible for SAS programmers to perfom challenging tasks that can not be done with SAS.

These VBScript tools only work if Microsoft Office is installed locally.

VBScript (through CSCRIPT.EXE) can be run from the command line which conveniently allows it to run directly from SAS using CALL SYSTEM(COMMAND) syntax.

Executing a VBScript file from the command prompt will look like this.

```
C:\> START /WAIT Cscript MYFILE.VBS //NOLOGO
```

The "MYFILE.VBS" will be the VBScript program and can be given almost any name, so long as it has the .VBS extension.

TOOL #1: CONVERT RTF TO DOC/DOCX

The first tool to consider is one that will convert ODS RTF output to a true DOC or DOCX file. While having output in RTF format is just fine some of the time, more often than not customers really prefer a true DOC file. Additionally, the .DOC files often take up less space than the .RTF files. Renaming the file from .RTF to .DOC is an option, but Microsoft Word still knows it's an RTF file. Manually opening the file in Microsoft Word and re-saving it as a .DOC file is also an option, but that is laborious and violates the “no touch” rule for output (i.e., all output from SAS should be ready to use when the program is finished and not require any manual adjustments).   The solution is tool #1.

THE SAS MACRO

This tool consists of two macros: %VBSSTART and %VBSCLOSE2DOC.   These macros will be submacros of the macro %VBSMACS just to make them easier to load and manage.

%VBSSTART requires one parameter – RTFNAME. This is the name of the RTF file that will be converted, including the full path. Macro variables can be used to specify RTFNAME but filerefs cannot. The macro first begins writing the VBScript. Lines are written to the VBScript to open Microsoft Word and then to open the RTF file. Now that the VBScript writing has begun, and the code to open the file is in place, it is time to move on to %VBSCLOSE2DOC.

%VBSCLOSE2DOC requires one parameter – NEWNAME, and optionally takes a second – TYPE. NEWNAME will be the name the open RTF file will be saved under. TYPE is the format type. If TYPE=DOC the file will be saved as a DOC file (Word 97-2003). If TYPE is missing, or anything other than DOC, it will be saved in the format appropriate for the version of word that is open (e.g., for Microsoft Word 2010 it will be a Word Document (*.DOCX) file).

```
%macro VBSMACS;
 %macro VBSSTART(rtfname=);
    %global docname;
    %let docname=&rtfname;
    %let temp=%scan(&rtfname,1,.);
    options noxwait;

    data _null_;
       length vbscmd temp $ 400;
       file "&docname..vbs";
       temp=''; * INITIALIZE TO AVOID UNINITIALIZED VAR MSG;
       put 'Dim ObjWord';
       put 'set objWord = CreateObject("Word.Application")';
       put 'objWord.Visible = True';
       vbscmd='objWord.Documents.Open("'|| "&docname" ||'")';
       put vbscmd;
 %mend;

 %macro VBSCLOSE2DOC(newname=,type=);
    %let newname=%scan(&newname,1,'.');
    %if %upcase(&type)=DOC %then %do;
       vbscmd='objWord.ActiveDocument.SaveAs "'||"&newname"||'", 0';
    %end;
    %else %do;
       vbscmd='objWord.ActiveDocument.SaveAs "'||"&newname"||'", 16';
    %end;
    put vbscmd;
    put 'objWord.ActiveDocument.Close(False)';
    put 'objWord.Application.Quit(False)';
    run;
    options noxwait;
    data _null_;
       command="START /WAIT CScript &docname..vbs //NoLogo";
       call system(command);
       command="DEL &docname..vbs ";
       call system(command);
    run;
  %mend;
%mend;
```

To call these macros, just a few lines of code are needed after the RTF file is created using ODS RTF/ODS RTF CLOSE and any report writing or graph producing PROCs (e.g., PROC PRINT, PROC REPORT, PROC TABULATE, PROC SGPLOT, etc.). Just to avoid confusion, the RTF file can (and probably should) be removed after it is converted. To save space, &OUTNAME is a macro variable that contains the path and name of the file without the extension. There must be an ODS RTF CLOSE statement before these lines.

```
%vbsmacs;
%vbsstart(rtfname=&outname..rtf);
%vbsclose2doc(newname=&outname,type=DOCX);
options noxwait;
data _null_;
     command="DEL &outname..rtf ";
     call system(command);
run;
```

Because %VBSSTART begins writing to an external file and %VBSCLOSE2DOC continues writing to that same file, there should be no SAS code between invoking these two macros that will impede this (e.g., no RUN; PROC or DATA statements).

Just as an aside, for Microsoft Word 2003, the macro would need to be slightly different: TYPE would not be used and the following line would replace the SAVEAS lines in %VBSCLOSE2DOC.

```
 vbscmd='objWord.ActiveDocument.SaveAs("'||"&newname"||'"),wdFormatDocument';
```

What happens when the program executes these lines? First, %VBSSTART and the first part of %VBSCLOSE2DOC create an external file with a .VBS extension. When this file is done, a CALL SYSTEM(COMMAND) line is executed that opens a window and runs the VBScript. If you are watching it, you will see Microsoft Word open the RTF file and then it will all close. Other open Word sessions will not impact the functioning of the macro. Usually it runs very quickly.

THE VBSCRIPT

It is always a good idea to understand the goal of a macro, so reviewing the expected VBScript is worthwhile. The actual VBScript to convert an RTF file to a DOC file that the SAS macro will produce and execute looks like this.

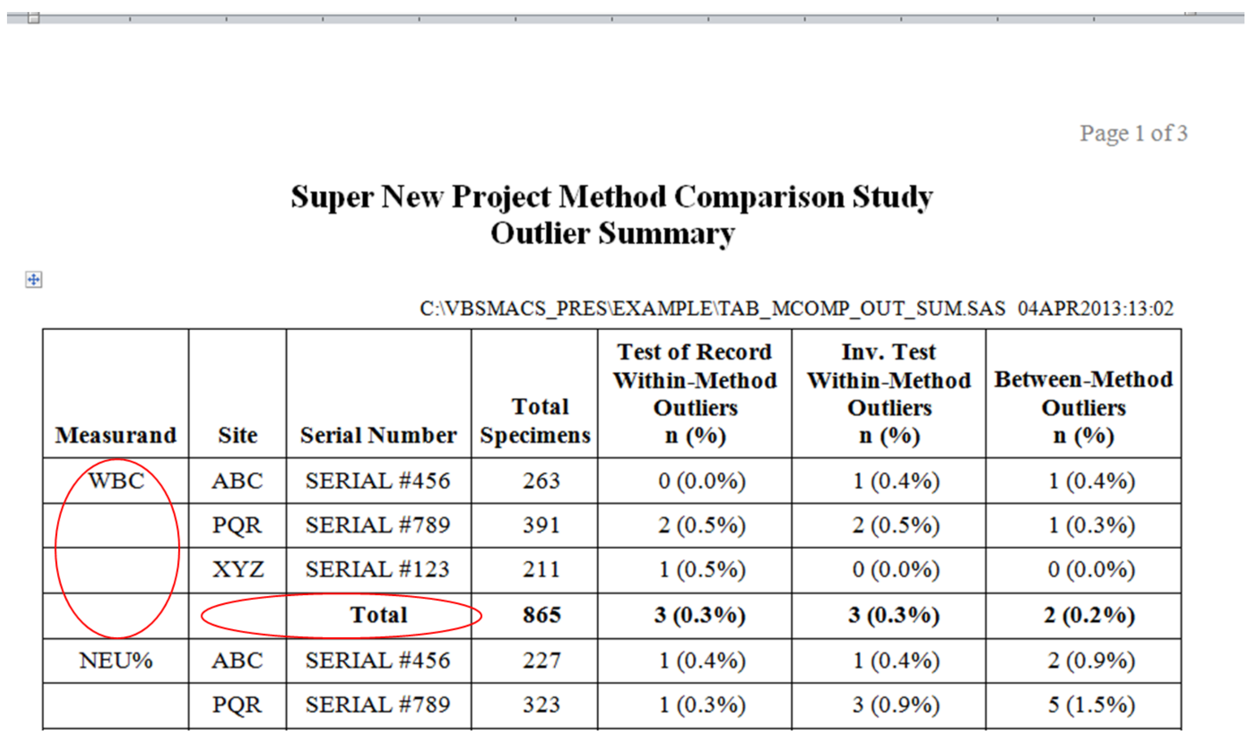
```
Dim ObjWord
set objWord = CreateObject("Word.Application")
objWord.Visible = True
objWord.Documents.Open("C:\VBSMACS_PRES\EXAMPLE\TAB_MCOMP_OUT_SUM.rtf")
objWord.ActiveDocument.SaveAs "C:\VBSMACS_PRES\EXAMPLE\TAB_MCOMP_OUT_SUM", 16
objWord.ActiveDocument.Close(False)
objWord.Application.Quit(False)
```

The first highlighted section is the name of the RTF file to be converted with the full path included. The second highlighted piece is the name of the DOC file that will be created. The extension is not needed as Microsoft Word will assign the correct extension but the full path must be there. The number 16 tells Microsoft Word what format to save it as. In this case, 16 is "Word default document file format" (i.e., DOCX for Word 2010). Alternatively, the number 0 could have been used which will save it as a "Microsoft Word 97-2003" formatted file. A list of other valid numbers can be found by doing a web search for "WdSaveFormat Enumeration." The other lines in this code are standard VBScript to open a session of Microsoft Word, open the RTF document, save and close the document, and finally exit out of Microsoft Word.

TOOL #2: MERGE CELLS IN ODS RTF

The first tool converted an RTF file to a DOC/DOCX file. While the RTF file is open, between %VBSSTART and %VBSCLOSE2DOC, other changes can be made to the RTF file. One of the most valuable changes would be merging cells in report tables.

Consider Figure 1. This is a snippet of a larger table created with PROC REPORT. There are several sets of empty cells and the table would look a lot nicer if these can be merged together. The solution is tool #2.



Page 1 of 3

Super New Project Method Comparison Study
Outlier Summary

C:\VBSMACS_PRES\EXAMPLE\TAB_MCOMP_OUT_SUM.SAS 04APR2013:13:02

Measurand	Site	Serial Number	Total Specimens	Test of Record Within-Method Outliers n (%)	Inv. Test Within-Method Outliers n (%)	Between-Method Outliers n (%)
WBC	ABC	SERIAL #456	263	0 (0.0%)	1 (0.4%)	1 (0.4%)
	PQR	SERIAL #789	391	2 (0.5%)	2 (0.5%)	1 (0.3%)
	XYZ	SERIAL #123	211	1 (0.5%)	0 (0.0%)	0 (0.0%)
		Total	**865**	**3 (0.3%)**	**3 (0.3%)**	**2 (0.2%)**
NEU%	ABC	SERIAL #456	227	1 (0.4%)	1 (0.4%)	2 (0.9%)
	PQR	SERIAL #789	323	1 (0.3%)	3 (0.9%)	5 (1.5%)

Figure 1. Output from PROC REPORT with cells to be merged

THE VBSCRIPT

For this tool, it may be helpful to look at what the actual VBScript will look like before looking at the SAS macro that will produce it. The VBScript that will convert an RTF file to a DOC file and merge these two sets of cells together looks like this code.

```
Dim ObjWord
set objWord = CreateObject("Word.Application")
objWord.Visible = True
objWord.Documents.Open("C:\VBSMACS_PRES\EXAMPLE\TAB_MCOMP_OUT_SUM.rtf")
With objWord.ActiveDocument.Tables.Item(1).Cell(3,1).Merge(.Cell( 6,1))
End With
With objWord.ActiveDocument.Tables.Item(1).Cell(6,2).Merge(.Cell( 6,3))
End With
objWord.ActiveDocument.SaveAs "C:\VBSMACS_PRES\EXAMPLE\TAB_MCOMP_OUT_SUM", 16
objWord.ActiveDocument.Close(False)
objWord.Application.Quit(False)
```

The highlighted number 1 is the table number. Tables are "numbered" from the beginning of the document, excluding the header and footer sections. Generally, each page will correspond to a table number unless titles and footnotes are in their own tables or multiple tables are included on a single page. If there is more than one table on a given page, it may take some careful review of the RTF file to be sure that the table numbers are correct in the call(s) to %VBSMERGECELL. The bold red numbers are cells determined by the ROW (first number) and the COLUMN (second number) position in the table. The first set of numbers is a starting position, the second is an ending position – everything in between will be merged (and they must be in a straight line).

THE SAS MACRO

This tool consists of one additional macro added to %VBSMACS described in tool #1: %VBSMERGECELL.

%VBSMERGECELL requires five parameters.

- TABLE is the table number of the cells to be merged.
- ROW1 is the row # of the starting cell.
- COL1 is the column # of the starting cell.
- ROW2 is the row # of the ending cell.
- COL2 is the column # of the ending cell.

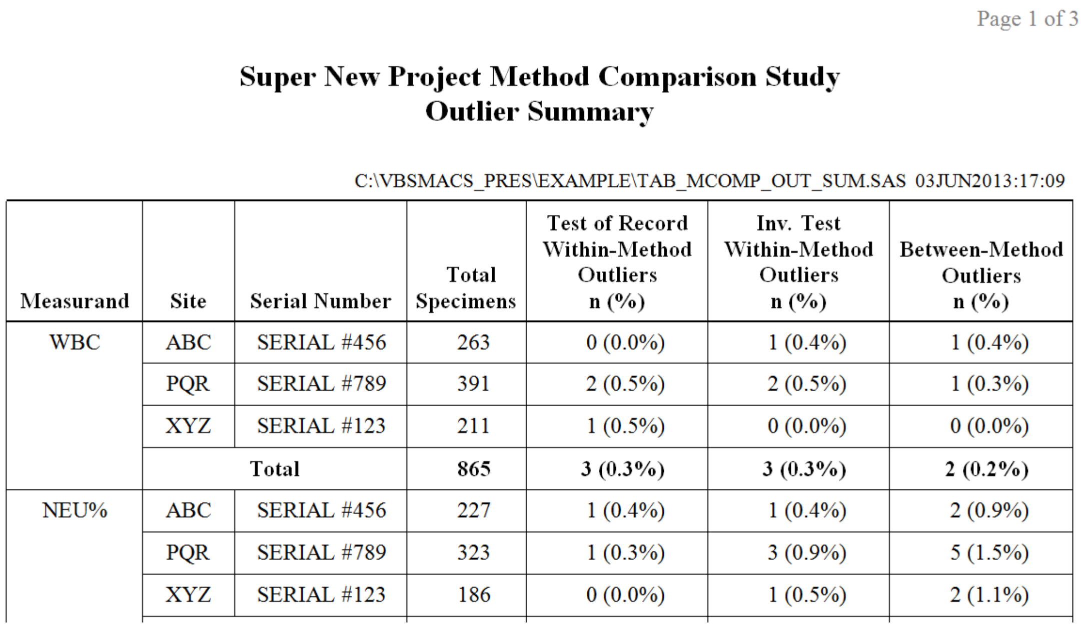
```
    %macro vbsmergecell(table=,row1=,col1=,row2=,col2=);
        vbscmd='With objWord.ActiveDocument.Tables.Item(' || "&table" || ')';
        put vbscmd;
        vbscmd='.Cell(' || "&row1" || ',' || "&col1" || ').Merge(.Cell( ' ||
            "&row2" || ',' || "&col2" || '))';
        put vbScmd;
        put 'End With';
     %mend vbsmergecell;
```

%VBSMERGECELL can be called multiple times between %VBSSTART and %VBSCLOSE2DOC. Here is the example code that will merge the two sets of cells from Figure 1.

```
%vbsmacs;
%vbsstart(rtfname=&outname..rtf);
%vbsmergecell(table=1,row1=3,col1=1,row2=6,col2=1);
%vbsmergecell(table=1,row1=6,col1=2,row2=6,col2=3);
%vbsclose2doc(newname=&outname,type=DOCX);
options noxwait;
data _null_;
     command="DEL &outname..rtf ";
     call system(command);
run;
```

Figure 2 shows the result. In this table there were many additional sets of cells that needed to be merged. This can easily be accomplished with some simple macro code such as the following show below. The dataset here has been processed earlier in the program so that the macro field &LASTPAGENO contains the total number of pages (or tables in this case because there is one table per page). The macro fields &&NUMRECS&PAGE are the number of records on each page, also created earlier in the program.

```
%vbsmacs;
%vbsstart(rtfname=&outname..rtf);
%macro fixit;
    %do page=1 %to &lastpageno;
       %let tnum=&page; * ONE TABLE PER PAGE IN THIS CASE;
       %let numrecs=&&numrecs&page;
       %do irow=3 %to &numrecs %by 4;
          %let end=%eval(&irow+3);
          %vbsmergecell(table=&tnum,row1=&irow,col1=1,row2=&end,col2=1);
          %vbsmergecell(table=&tnum,row1=&end,col1=2,row2=&end,col2=3);
       %end;
     %end;
%mend;
%fixit;
%vbsclose2doc(newname=&outname,type=DOCX);
data _null_;
     command="DEL &outname..rtf ";
     call system(command);
run;
```



Page 1 of 3

Super New Project Method Comparison Study
Outlier Summary

C:\VBSMACS_PRES\EXAMPLE\TAB_MCOMP_OUT_SUM.SAS 03JUN2013:17:09

Measurand	Site	Serial Number	Total Specimens	Test of Record Within-Method Outliers n (%)	Inv. Test Within-Method Outliers n (%)	Between-Method Outliers n (%)
WBC	ABC	SERIAL #456	263	0 (0.0%)	1 (0.4%)	1 (0.4%)
	PQR	SERIAL #789	391	2 (0.5%)	2 (0.5%)	1 (0.3%)
	XYZ	SERIAL #123	211	1 (0.5%)	0 (0.0%)	0 (0.0%)
	Total		**865**	**3 (0.3%)**	**3 (0.3%)**	**2 (0.2%)**
NEU%	ABC	SERIAL #456	227	1 (0.4%)	1 (0.4%)	2 (0.9%)
	PQR	SERIAL #789	323	1 (0.3%)	3 (0.9%)	5 (1.5%)
	XYZ	SERIAL #123	186	0 (0.0%)	1 (0.5%)	2 (1.1%)

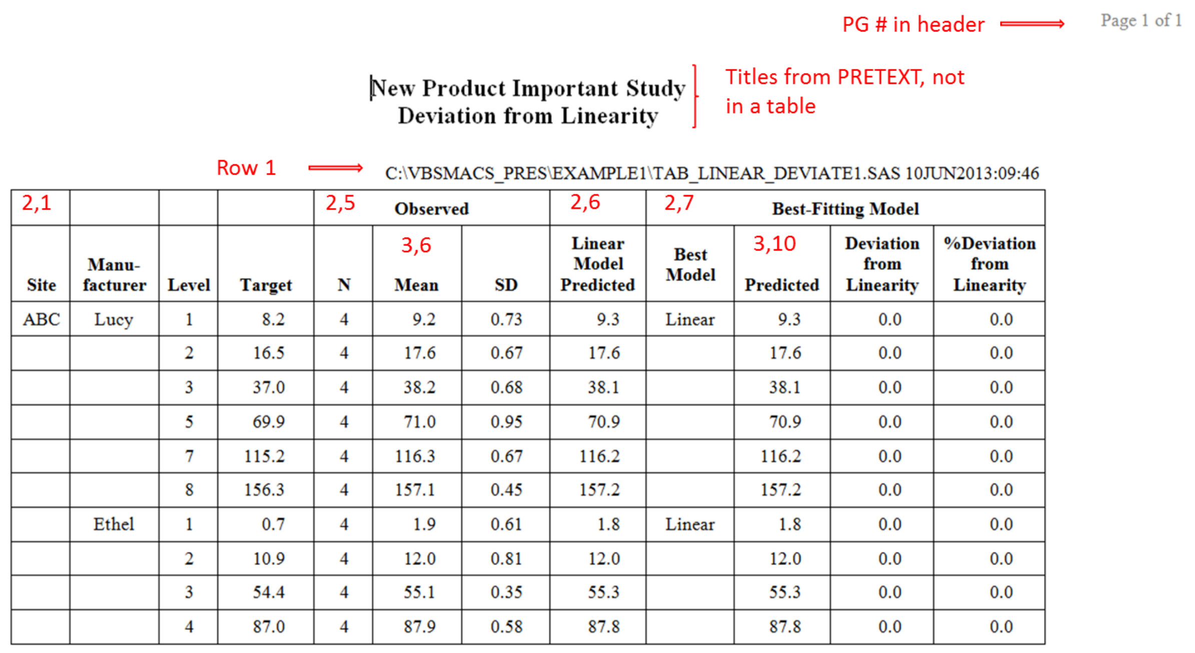
Figure 2. Table after %VBSMERGECELL

%VBSMERGECELL is a bit more complicated to use than just %VBSSTART and %VBSCLOSE2DOC. If, for example, the table column headers (in PROC REPORT, as an example) have spanning headers, it can be difficult to count the rows and the columns. Merging cells in the column headers when there are spanning column headers can be very beneficial because the spacing can be improved, but the adjacent columns need to have clear empty cells above them available to merge. This is easily accomplished in the PROC REPORT column statement by making sure every column has a spanning (albeit empty) column above it. Code like this simple example will accomplish this:

```
column ('^S={background=_undef_}' site )
       ('^S={background=_undef_}' manufacturer)
       ('^S={background=_undef_}' level )
       ('^S={background=_undef_}' target)
       ('Observed' n mean sd )
```

Figure 3 is another table where merging cells is needed, including in the header. In this case, the red numbering indicates the row and column number of some of the cells to make it easier to understand how to merge. For example, to merge the cell above “Linear Model Predicted” in the header, it would be necessary to merge cell 2,6 with 3,8. To merge the first set of cells under “Best Model” between the cell that says “Cubic” and the last empty cell before the next “Cubic”, it would be necessary to merge cell 4,9 through cell 9,9.

It is sometimes beneficial to start merging cells from the bottom of the table up, as the merging of cells can change how Microsoft Word “views” the cell row and column numbering which can impact the next command to merge cells. Also, if incorrect information is passed into the script (such as a set of row and column coordinates that do not exist or trying to merge cells that are not in a straight line), the Microsoft Word session will usually hang and need to be closed manually.


PG # in header
Titles from PRETEXT, not in a table
Row 1
2,1
2,5
2,6
2,7
3,6
3,10

Page 1 of 1

New Product Important Study
Deviation from Linearity

C:\VBSMACS_PRES\EXAMPLE1\TAB_LINEAR_DEVIATE1.SAS 10JUN2013:09:46

				Observed				Best-Fitting Model			
Site	**Manu-facturer**	**Level**	**Target**	**N**	**Mean**	**SD**	**Linear Model Predicted**	**Best Model**	**Predicted**	**Deviation from Linearity**	**%Deviation from Linearity**
ABC	Lucy	1	8.2	4	9.2	0.73	9.3	Linear	9.3	0.0	0.0
		2	16.5	4	17.6	0.67	17.6		17.6	0.0	0.0
		3	37.0	4	38.2	0.68	38.1		38.1	0.0	0.0
		5	69.9	4	71.0	0.95	70.9		70.9	0.0	0.0
		7	115.2	4	116.3	0.67	116.2		116.2	0.0	0.0
		8	156.3	4	157.1	0.45	157.2		157.2	0.0	0.0
	Ethel	1	0.7	4	1.9	0.61	1.8	Linear	1.8	0.0	0.0
		2	10.9	4	12.0	0.81	12.0		12.0	0.0	0.0
		3	54.4	4	55.1	0.35	55.3		55.3	0.0	0.0
		4	87.0	4	87.9	0.58	87.8		87.8	0.0	0.0

Figure 3. Finding Row and Column Positions

TOOL #3: CONVERT RTF/DOC/DOCX TO PDF

Now that cells have been merged, and the RTF file has been saved as a true DOC file, it might be nice to make a PDF version of the final product. While ODS PDF could have been used to create a PDF version of the original table, it would not include the merged cells and any RTF inline formatting that may have been used for the original table. The solution is tool #3.

THE SAS MACRO

In this case, no action needs to be taken between opening the Word file and saving it as a PDF, so one simple all-in-one macro will suffice. %VBSMKPDF requires two arguments: WORDNAME is the name of the file to be opened and NEWNAME is the name of the new PDF file. WORDNAME requires a file extension but NEWNAME does not (although it will still work if the PDF extension is included).

```
%macro vbsmkPDF(wordname=,newname=);
        data _null_;
          length vbscmd $ 400;
          file "temp.vbs";
          put 'Dim ObjWord';
          put 'set objWord = CreateObject("Word.Application")';
          put 'objWord.Visible = True';
          vbscmd='objWord.Documents.Open("'|| "&wordname" ||'")';
          put vbscmd;
          vbscmd='objWord.ActiveDocument.SaveAs "'||"&newname"||'", 17';
          put vbscmd;
          put 'objWord.ActiveDocument.Close(False)';
          put 'objWord.Application.Quit(False)';
       run;
```

```
        options noxwait;
        data _null_;
           command="START /WAIT CScript temp.vbs //NoLogo";
           call system(command);
           command2="DEL temp.vbs ";
           call system(command2);
        run;
  %mend;
```

This all-in-one macro makes it quite straightforward to create a program that will find all RTF, DOC and DOCX files in a directory and convert them all. Simple code like the following will find all of the RTF, DOC or DOCX files in a directory indicated by the macro variable &PATHNAME. (Before this section of code, there would need to be code that finds %VBSMKPDF and assigns PATHNAME).

```
options noxwait;
filename doclst pipe "dir &pathname.\*.rtf &pathname.\*.doc* /b";

data docs; length fname $ 200; infile doclst pad missover;
     input fname $ 1-200;
     n+1;
     call symput('count',compress(n));
run;
proc sql noprint;
     select fname
     into : docs1- :docs&count
     from docs;
run;
%macro mkpdfs;
  %do i=1 %to &count;
    %let mypdf=%scan(&&docs&i,1,'.');
    %vbsmkpdf(docxname=&pathname.\&&docs&i,newname=&pathname.\&mypdf..pdf);
   %end; * END OF DO LOOP;
%mend;

%mkpdfs;
run;
```

THE VBSCRIPT

The VBScript that will convert an RTF, DOC or DOCX file to a PDF file looks like this code. It is almost identical to the VBSCRIPT for converting an RTF to a DOC except the number 16 (save as Word Document (*.DOCX)) is now 17 (save as PDF). The option to save a file as a PDF this way is not available in versions of Microsoft Word before 2007.

```
Dim ObjWord
set objWord = CreateObject("Word.Application")
objWord.Visible = True
objWord.Documents.Open("C:\VBSMACS_PRES\EXAMPLE\TAB_MCOMP_OUT_SUM.docx")
objWord.ActiveDocument.SaveAs "C:\VBSMACS_PRES\EXAMPLE\TAB_MCOMP_OUT_SUM", 17
objWord.ActiveDocument.Close(False)
objWord.Application.Quit(False)
```

TOOL #4: CONVERT XML TO XLS/XLSX

Now, let's move away from Microsoft Word and apply the same ideas to Microsoft Excel. Advancements in ODS TAGSETS.EXCELXP make creating Excel versions of reports or data easy. There is one problem though – ODS TAGSETS.EXCELXP creates an XML file that Excel can recognize and open but it is not a true Excel file. With the addition of one more VBSCRIPT tool, tool #4, this too can be remedied.

THE SAS MACRO

As with %VBSMKPDF, no action needs to be taken between opening the XML file and saving it as an XLS/XLSX file. One simple all-in-one macro will suffice. %VBSMKXLS requires two arguments and will optionally use a third: FILE is the name of the [XML] file to be opened and NEWFILE is the name of the new, true, Excel file. FILE requires an extension but NEWFILE does not (it will work if the extension is included because the macro will strip it off). Optionally, TYPE can be specified. If TYPE=XLS, an Excel 97-2003 Workbook will be created (extension XLS). Any other value will result in an XLSX file.

```
%macro VBSMKXLS(file=,newfile=,type=);
    %let docname=&file;
    %let newfile=%scan(&newfile,1,'.');
    data _null_;
      length vbscmd $ 200;
      file "&docname..vbs";
      put 'Dim ObjExcel';
      put 'set ObjExcel = CreateObject("Excel.Application")';
      put 'ObjExcel.Visible = True';
      put 'ObjExcel.Application.DisplayAlerts = False';
      vbscmd='ObjExcel.Workbooks.Open("'|| "&file" ||'")';
      put vbscmd;

    %if %upcase(&type)=XLS %then %do;
      vbscmd='ObjExcel.ActiveWorkbook.SaveAs "'|| "&newfile" ||'",1';
    %end;
    %else %do;
      vbscmd='ObjExcel.ActiveWorkbook.SaveAs "'|| "&newfile" ||'",51 ';
    %end;
      put vbscmd;
      put 'ObjExcel.ActiveWorkbook.Close';
      put 'ObjExcel.Application.Quit';
    run;
  options noxwait;
  data _null_;
     command="START /WAIT CScript &docname..vbs //NoLogo";
     call system(command);
     command="DEL &docname..vbs ";
     call system(command);
  run;
%mend;
```

THE VBSCRIPT

The VBScript that will convert the XML file to an XLS file looks like the following code. The first highlighted part is the path and name of the XML file to be converted. The second highlighted part is the path and name of the XLS file that will be created. The number 51 is the enumeration code for WorkBookDefault that can be figured out by doing a web search for “XLFileFormat Enumeration.” (Although it is not always easy to find, 1 is the code to save as an Excel 97-2003 Workbook.)

```
Dim ObjExcel
set ObjExcel = CreateObject("Excel.Application")
ObjExcel.Visible = True
ObjExcel.Application.DisplayAlerts = False
ObjExcel.Workbooks.Open("C:\VBSMACS_PRES\TEST1\MYWORKBOOK.XML")
ObjExcel.ActiveWorkbook.SaveAs "C:\VBSMACS_PRES\TEST1\NEWWORKBOOK",51
ObjExcel.ActiveWorkbook.Close
ObjExcel.Application.Quit
```

CONCLUSION

These SAS macros provide a simple and straightforward way to generate and execute VBScript to overcome some common issues encountered when using ODS RTF or ODS TAGSETS.EXCELXP. The four issues that we identified have all been addressed.

Is there more we can do with VBScript? Yes, there is! Most of the customization that is needed for either RTF output or Excel can be done in SAS with some creative application of the tools already there. However, when there is something that just cannot be done in SAS, there is usually a VBScript solution. For the adventurous among us, tasks such as adding paragraph spacing or removing individual hard returns between tables are definitely doable.

ACKNOWLEDGMENTS

Many thanks to my Abbott colleagues Hope Knuckles, Jennifer Yen, Jeremy Desiron and Justin Rogers for their constructive comments and editing help.

CONTACT INFORMATION

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

Rose Grandy
Abbott Laboratories
100 Abbott Park Road, Dept 07B2, CP1
Abbott Park, IL 60064
Phone: (847) 935-1596
E-mail: rose.grandy@abbott.com