Paper 066-2012

Create Multi-Sheet Excel Workbook for Large Data Sets Using SAS® and VBA

Chao Huang, Oklahoma State University, Stillwater, OK

ABSTRACT

With fast-growing data volume, transforming large data sets from SAS to multi-sheet Microsoft Excel workbooks becomes challenging. In addition, more than one grouping variables may be specified to separate a SAS data set to sheets in an Excel workbook. This paper describes a new and fast solution to create multi-sheet Excel workbooks, which also allows multiple grouping variables for each sheet. Two examples with SAS's help data sets will be used to illustrate how to use ODS HTML destination and a VBA script to produce Excel workbook. This two-step approach can process very large SAS data sets for multi-sheet Excel reporting in a short time. It is also customizable for special needs such as traffic lighting.

INTRODUCTION

There are two challenges in today's working environment for Excel reporting: first data gets big and second more clients tend to view data in a multi-sheet Excel workbook. Romain compared seven methods for Excel reporting and weighted their strength and weakness [1]. A mutli-sheet Excel workbook from SAS means that the original SAS data set is divided toward Excel spreadsheets according to the levels of the grouping variables. ExcelXP ODS tagset is a popular option to generate multi-sheet Excel workbooks [2]. This method provides ample parameters to produce stylish Excel workbooks. However, ExcelXP transmits data from SAS to Excel by XML format, and therefore may be not very efficient regarding the speed and the final file size [1].

To transform a large SAS data set to a multi-sheet workbook, we can apply a two-step approach by harnessing the powers of SAS's ODS facility and Excel's VBA functionality together. First we split a SAS data set into many XLS files by running a SAS macro *split()* in SAS. Second we merge those XLS files into a multi-sheet Excel workbook by running a VBA subroutine *Merge()* in Excel (Figure 1).

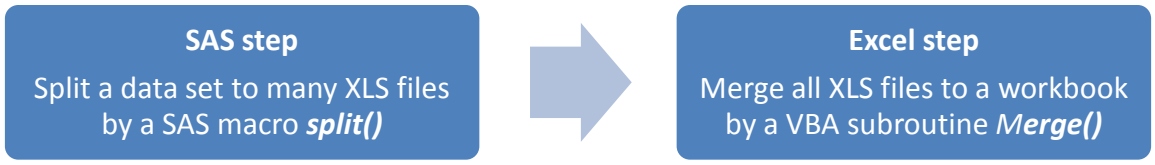


Figure 1. Workflow of the Two-Step Approach

AN EXAMPLE THAT USES ONE GROUPING VARIABLE

SAS's ODS HTML tagset generates HTML structured files, which can be recognized and opened by Excel if their extension is named as XLS. It is a popular solution to produce single-sheet Excel workbooks. To extend this facility to multi-sheet Excel reporting, we can design a simple SAS macro *split()* to output the corresponding XLS files for each level of the grouping variable.

We start with a simple example using SASHELP.CLASS, which is from the SASHELP library and describes the name, age, sex, weight and height information of 19 teenagers. We show how to split this data set by the teenagers' ages to 6 spreadsheets and then merge them in a multi-sheet Excel workbook.

STEP 1 – SPLIT

First we split the raw data set into separated XLS files by ODS HTML destination. For convenience, we can set up an empty directory in the hard disk, such as "c:\demo1". Since the age variable in SASHELP.CLASS has 6 levels, the SAS macro will create 6 XLS files for each age group under the directory (Figure 2).

```
%macro split(data = , dir = , clsvar = );
```

```
/* 1 - Find the levels of the grouping variable */
       proc freq data = &data;
              table &clsvar / out = _tmp01;
       run;
/* 2 - Concatenate all levels of the grouping variable as a macro variable */
       proc sql noprint;
              select &clsvar into: clsvarlist separated by '|'
              from _tmp01;
              select count(*) into: nobs
              from _tmp01;
       quit;
/* 3 - Set some system options for output */
       footnote; title;
       options nocenter nodate nonumber ps = 9000;
/* 4 - Split each level of the grouping variable to a single spreadsheet in a loop */
       %do i = 1 %to &nobs;
              %let clsvarlevel = %scan(&clsvarlist, &i, '|');
              ods html file="&dir\&clsvarlevel..xls" style = minimal;
              proc print data = &data noobs label;
                     where &clsvar = &clsvarlevel;
              run;
       %end;
       ods html close;
%mend;
/* 5 - Apply the macro to SASHELP.CLASS by the variable AGE */
%split(data = sashelp.class, dir = c:\demo1, clsvar = age);
```

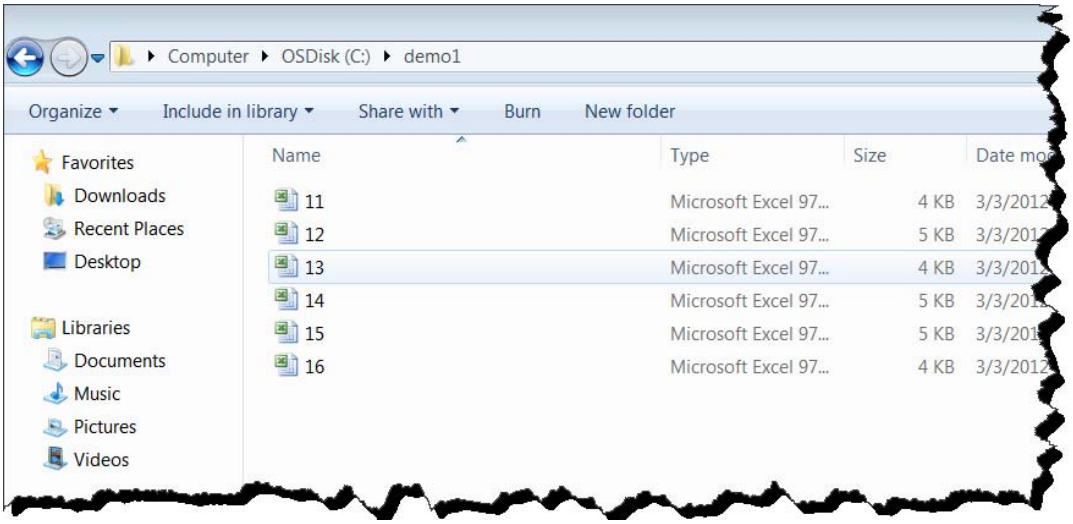


Figure 2. The 6 XLS Files from SASHELP.CLASS Created by SAS's ODS HTML Facility

STEP 2 – MERGE

Next step we move to Microsoft Excel. We open an empty Excel workbook and press ALT+ F11 keys to activate Excel's VBA editor, then copy and paste the VBA subroutine called *Merge()* into the editor. The only place in the VBA subroutine that needs specification is the file path at the second line which is used by the first split step. After submitting this macro (click the RUN button or press F5 key), Excel will sequentially merge all XLS files under this directory. When the job is finished, the user can choose to save it as either an Excel 97-2003 workbook or an Excel 2007 workbook (Figure 3).

```
Sub Merge()
' 1 - Specify the directory where SAS generates individual spreadsheet files
       Dim xlsPath As String
       xlsPath = "c:\demo1"
' 2 - Disable some Excel display options
       Application.DisplayAlerts = False
```

```
        Application.EnableEvents = False
        Application.ScreenUpdating = False
' 3 - Declare four objects for the following loop
        Dim wbDst As Workbook
        Dim wbSrc As Workbook
        Dim wsSrc As Worksheet
        Dim strFilename As String
' 4 - Merge all spreadsheets under the directory to a workbook in a loop
        Set wbDst = Workbooks.Add(xlWBATWorksheet)
        strFilename = Dir(xlsPath & "\*.xls", vbNormal)
        If Len(strFilename) = 0 Then Exit Sub
        Do Until strFilename = ""
                Set wbSrc = Workbooks.Open(Filename:=xlsPath & "\" & strFilename)
                Set wsSrc = wbSrc.Worksheets(1)
                wsSrc.Copy After:=wbDst.Worksheets(wbDst.Worksheets.Count)
                wbSrc.Close False
                strFilename = Dir()
                ActiveWindow.DisplayGridlines = True
        Loop
' 5 - Remove the working worksheet object
        wbDst.Worksheets(1).Delete
' 6 - Enable Excel options disabled previously
        Application.DisplayAlerts = True
        Application.EnableEvents = True
        Application.ScreenUpdating = True
End Sub
```

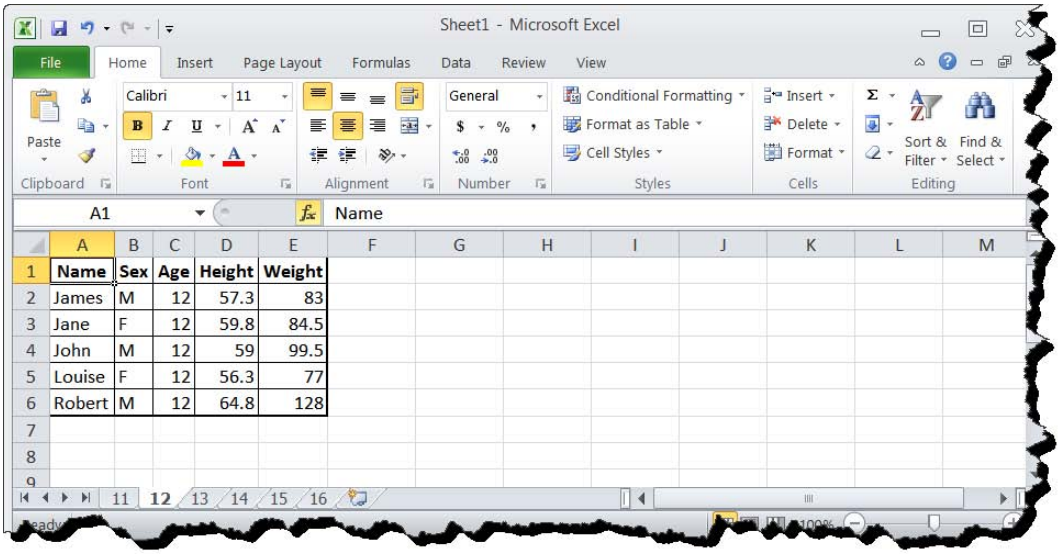


Figure 3. The Resulting Multi-sheet Excel Workbook for SASHELP.CLASS

AN EXAMPLE THAT USES TWO GROUPING VARIABLES AND TRAFFIC LIGHTING

Furthermore we show a more complicated illustration. First, we utilize a relatively large data set, SASHELP.PRDSAL2, which is also from the SASHELP library. It records the furniture sales in 64 states of the three countries from 1995 to 1998, and has total 23,040 observations and 11 variables. Second, instead of one variable, we use two variables, STATE and YEAR, to partition SASHELP.PRDSAL2 for individual spreadsheets in an Excel workbook. The variable STATE has 16 levels, and the other variable YEAR has 4 levels. Therefore we will create 64 spreadsheets by all the combination levels. Third, we add the "traffic lighting" or conditional variable highlighting feature for one variable PREDICTED that is the predicted sales in this data set. “Traffic lighting” applies distinctive colors to the variables to indicate the ranges [3]. In this example, we create a user-defined format to assign three colors for the three intervals by the FORMAT procedure (Table 1).

```
/* 0 - Make a user-defined format for traffic lighting */
proc format;
value range
      2000 - high = '#ffffcc'
      400 -< 2000 = 'yellow'
      other = '#ff9900';
run;
```

Table 1 shows the background colors corresponding to the values of the PREDICTED variables.

Values of the variable PREDICTED	Background color
More than $2,000	#ffffcc
Between $400 and $2,000	yellow
Less than $400	#ff9900

Table 1. The Variable Values and Their Background Colors

STEP 1 – SPLIT

To accommodate the two grouping variables, we make some modifications to the SAS macro *split()* used in the first illustration. Thus, the 64 XLS files are created according to the location variable STATE and the time variable YEAR (Figure 4).

```
%macro split(data = , dir = , clsvar1 = , clsvar2 = );
/* 1 - Find the combination levels of the two grouping variable */
       proc freq data = &data;
              table &clsvar1 * &clsvar2 / out = _tmp01;
       run;
/* 2 - Concatenate all levels of the grouping variables as a macro variable */
       proc sql noprint;
              select cats(&clsvar1, '__', &clsvar2) into: clsvarlist separated by '|'
              from _tmp01;
              select count(*) into: nobs
              from _tmp01;
       quit;
/* 3 - Set some system options for output */
       footnote; title;
       options nocenter nodate nonumber ps = 9000;
/* 4 - Split each level of the grouping variables to a single spreadsheet in a loop
       and apply the RANGE highlighting format */
       %do i = 1 %to &nobs;
              %let clsvarlevel = %scan(&clsvarlist, &i, '|');
              ods html file="&dir\&clsvarlevel..xls" style = minimal;
              proc print data = &data noobs label;
                     where &clsvar1 = "%scan(&clsvarlevel, 1, '__')"
                            and &clsvar2 = %scan(&clsvarlevel, 2, '__');
                     var _all_;
                     var predict / style = [background = range.];
              run;
       %end;
       ods html close;
%mend;
/* 5 - Apply the macro to SASHELP.PRDSAL2 by the variables STATE and YEAR */
%split(data = sashelp.prdsal2, dir = c:\demo2, clsvar1 = state , clsvar2 = year);
```

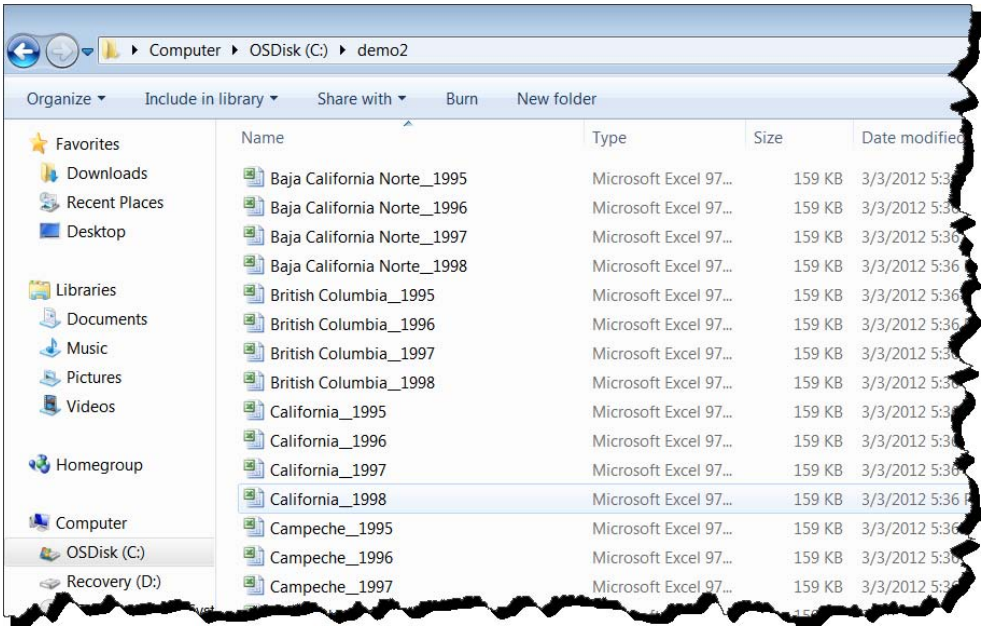
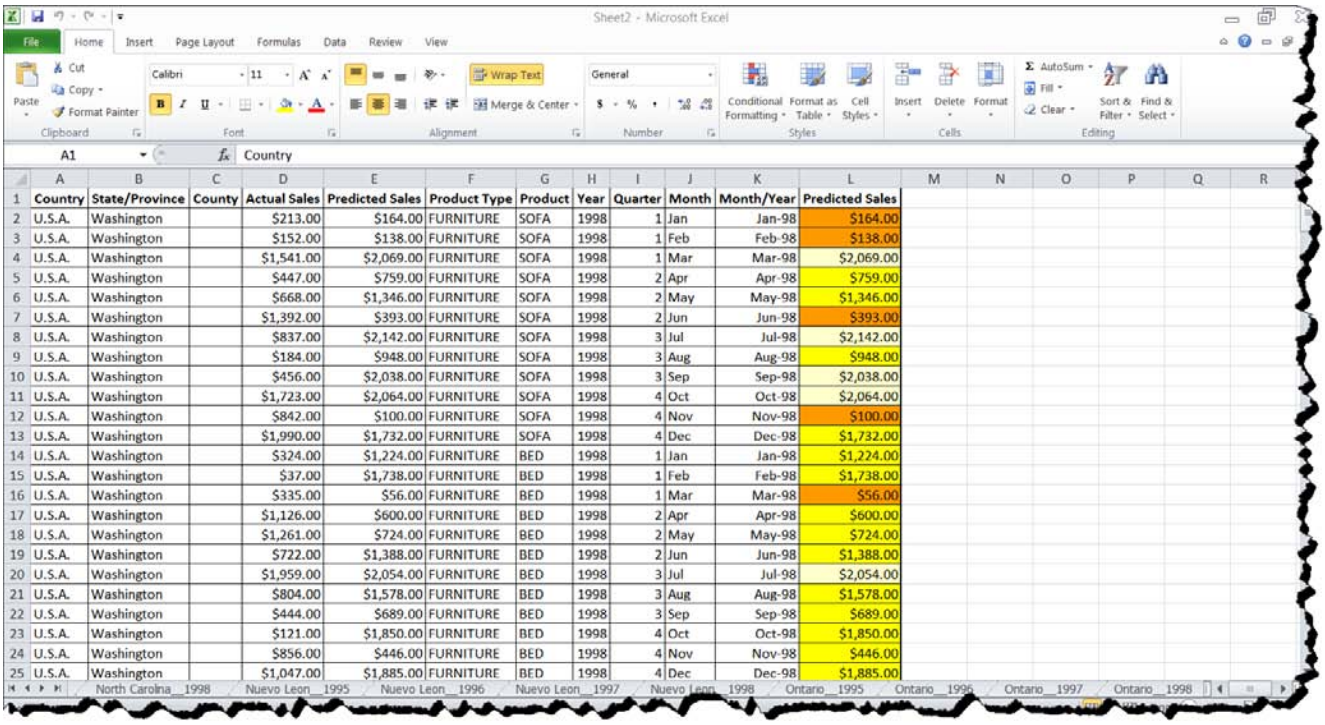


Figure 4. The 64 XLS Files from SASHELP.PRDSAL2 Created by SAS's ODS HTML Facility

STEP 2 – MERGE

This time the VBA subroutine *Merge()* does not need further change, except that we may specify a different directory that stores the XLS files, such as "c:\demo2". The result is showed as Figure 5.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Country	State/Province	County	Actual Sales	Predicted Sales	Product Type	Product	Year	Quarter	Month	Month/Year	Predicted Sales
2	U.S.A.	Washington		$213.00	$164.00	FURNITURE	SOFA	1998	1	Jan	Jan-98	$164.00
3	U.S.A.	Washington		$152.00	$138.00	FURNITURE	SOFA	1998	1	Feb	Feb-98	$138.00
4	U.S.A.	Washington		$1,541.00	$2,069.00	FURNITURE	SOFA	1998	1	Mar	Mar-98	$2,069.00
5	U.S.A.	Washington		$447.00	$759.00	FURNITURE	SOFA	1998	2	Apr	Apr-98	$759.00
6	U.S.A.	Washington		$668.00	$1,346.00	FURNITURE	SOFA	1998	2	May	May-98	$1,346.00
7	U.S.A.	Washington		$1,392.00	$393.00	FURNITURE	SOFA	1998	2	Jun	Jun-98	$393.00
8	U.S.A.	Washington		$837.00	$2,142.00	FURNITURE	SOFA	1998	3	Jul	Jul-98	$2,142.00
9	U.S.A.	Washington		$184.00	$948.00	FURNITURE	SOFA	1998	3	Aug	Aug-98	$948.00
10	U.S.A.	Washington		$456.00	$2,038.00	FURNITURE	SOFA	1998	3	Sep	Sep-98	$2,038.00
11	U.S.A.	Washington		$1,723.00	$2,064.00	FURNITURE	SOFA	1998	4	Oct	Oct-98	$2,064.00
12	U.S.A.	Washington		$842.00	$100.00	FURNITURE	SOFA	1998	4	Nov	Nov-98	$100.00
13	U.S.A.	Washington		$1,990.00	$1,732.00	FURNITURE	SOFA	1998	4	Dec	Dec-98	$1,732.00
14	U.S.A.	Washington		$324.00	$1,224.00	FURNITURE	BED	1998	1	Jan	Jan-98	$1,224.00
15	U.S.A.	Washington		$37.00	$1,738.00	FURNITURE	BED	1998	1	Feb	Feb-98	$1,738.00
16	U.S.A.	Washington		$335.00	$56.00	FURNITURE	BED	1998	1	Mar	Mar-98	$56.00
17	U.S.A.	Washington		$1,126.00	$600.00	FURNITURE	BED	1998	2	Apr	Apr-98	$600.00
18	U.S.A.	Washington		$1,261.00	$724.00	FURNITURE	BED	1998	2	May	May-98	$724.00
19	U.S.A.	Washington		$722.00	$1,388.00	FURNITURE	BED	1998	2	Jun	Jun-98	$1,388.00
20	U.S.A.	Washington		$1,959.00	$2,054.00	FURNITURE	BED	1998	3	Jul	Jul-98	$2,054.00
21	U.S.A.	Washington		$804.00	$1,578.00	FURNITURE	BED	1998	3	Aug	Aug-98	$1,578.00
22	U.S.A.	Washington		$444.00	$689.00	FURNITURE	BED	1998	3	Sep	Sep-98	$689.00
23	U.S.A.	Washington		$121.00	$1,850.00	FURNITURE	BED	1998	4	Oct	Oct-98	$1,850.00
24	U.S.A.	Washington		$856.00	$446.00	FURNITURE	BED	1998	4	Nov	Nov-98	$446.00
25	U.S.A.	Washington		$1,047.00	$1,885.00	FURNITURE	BED	1998	4	Dec	Dec-98	$1,885.00

Figure 5. The Resulting Multi-Sheet Excel Workbook for SASHELP.PRDSAL2

Running this VBA subroutine will permanently remove the embedded HTML feature of the single XLS files from SAS's ODS HTML facility, and therefore the resulting Excel workbook is streamlined. The size of the original SAS data set, SASHELP.PRDSAL2, is 2,725 KB, and the total size of the 64 XLS files is 10,176 KB (159 KB × 64) after running the SAS *split()* macro. Finally the size of the final Excel 2007 workbook is 1,956 KB after running the VBA *Merge()* subroutine. This speed of the two-step approach is rather fast. In our test, the time cost for splitting and merging SASHELP.PRDSAL2 as a multi-sheet Excel workbook is less than 30 seconds on a standard desktop.

CONCLUSION

This two-step approach introduced in this paper has a few advantages. First, this "split-then-merge" method works with any size of SAS data set. Second, this method only requires SAS/BASE. It is compatible with SAS 9.1 or later. And it works with Microsoft Excel 97 or later. Second, it is highly customizable. Any SAS procedure with listing output, such as PROC REPORT, PROC TABULATE, PROC SQL and the statistical procedures, can be integrated with the *split()* macro. The styles of the Excel report can be further defined by PROC TEMPLATE [3]. In conclusion, this approach may prove to be an efficient way to create multi-sheet Excel workbooks.

REFERENCES

1. Romain Miralles, "Creating an Excel report: A comparison of the different techniques". SAS Global Forum Proceeding 2011.
http://support.sas.com/resources/papers/proceedings11/074-2011.pdf
2. Vincent DelGobbo, "Creating Stylish Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS". SAS Global Forum Proceeding 2011.
http://support.sas.com/resources/papers/proceedings11/170-2011.pdf
3. Michael Davis, "Traffic Lighting Your Multi-Sheet Microsoft Excel Workbooks the Easy Way with SAS". SAS Global Forum Proceeding 2010.
http://support.sas.com/resources/papers/proceedings10/153-2010.pdf

ACKNOWLEDGMENTS

We would like to thank Tricia Aanderud, the author of the book "Building Business Intelligence Using SAS: Content Development Examples", for valuable comments and careful revision.

CONTACT INFORMATION

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

Chao Huang
Office of Institution Research and Information Management
221 PIO Building
Stillwater, OK. 74075
Email: hchao8@gmail.com
Web: www.sasanalysis.com