

A SAS® Automation Example for Completing Reporting Templates in Microsoft® Excel

Imelda C. Go, PhD, Palmetto GBA, Columbia, SC and
Abbas S. Tavakoli, DrPH, MPH, ME, University of South Carolina

ABSTRACT

This example resulted from the need to report monthly data processing results by completing an Excel template used for reporting to the contracting party. In this case, the contracting party provided an Excel file with several formatted worksheets (e.g., first and second columns have bolding, cell shading, etc.) with the expectation that we would fill out the second-column cells. Instead of filling out the Excel template the customer provided, SAS creates an Excel file, which is made to look like the template. This example uses a "driver" table that maps each worksheet row to the SAS variable that has the value for the second column cell. The "driver" table also contains information that allows SAS to write into a text file the complete VBA in Excel macros, which can change the appearance of the exported file to look just like the required template. The "driver" can be extended to include columns that are useful for quality control when the programmer's work is checked by another programmer.

INTRODUCTION

Whenever we are required to use a specific format and file to report, the level complexity of the task could increase due to customization, which in turn can lead to quite a bit of hardcoding. Using the "driver" table will help minimize the hardcoding since the table will contain the information needed to direct SAS to write the code we need. Without the driver table, any change in the reporting template could make it more difficult to revise the hardcoding.

The value of this technique becomes more evident when the number of rows is large. A programmer might be able to manage a smaller number of rows with hardcoding, but as the number of rows increases, challenges will be greater. Also, last minute changes can leave less time for quality control. The automation safeguards us from human error provided the "driver" table itself is reliable since the SAS coding only does what the "driver" table tells it to.

REPORTING TEMPLATE

To demonstrate this technique, we use the following contrived example to provide a few examples of the possibilities using a “driver” table, SAS, and VBA in Excel.

Let us suppose we are analyzing international wildlife data and are required to report monthly by completing the two Excel Worksheets below currently title for November 2023.

The first Excel worksheet named Sightings has 14 rows:

	A	B
1	International Wildlife Counts (November 2023)	
2		Number of Sightings
3	Sea Turtle Species	
4	Loggerhead	
5	Green turtle	
6	Leatherback	
7	Hawksbill	
8	Kemp’s ridley	
9	Olive ridley	
10	Flatback	
11		
12	Injured Sea Turtles	
13	Injured	
14	Not Injured	

The second Excel worksheet named Finances has 9 rows:

	A	B
1	Income and Expenses for Sightings Project (November 2023)	
2		Totals in USD
3	Donations Received	
4	Online donations	
5	On-site fund-raising	
6		
7	Expenses	
8	Travel	
9	Room and board	
10		

"DRIVER" TABLE

We take the two columns and all rows from the template and include them in the "Driver" Table. We want a generalized format and we note that Cell A1 in both worksheets contain a title with a date that will change each month. We change the value of "November 2023" to ** in the "Driver" table so that we can change ** to the correct month and year using SAS.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	ColumnA	ColumnB	WSIndex	WSName	Row	DTTable	VarName	SASNumFormat	FontNameA	FontSizeA	FontNameB	FontSizeB	FontBoldA	FontBoldB	InteriorColorA	InteriorColorB
2	International Wildlife Counts (**)						SheetTitle		Times New		Times New				RGB(206,206,206)	
3		Number of Sightings			1 Sightings	1			Roman	14	Roman	14	TRUE			
4	Sea Turtle Species				1 Sightings	2			Calibri	11	Calibri	12		TRUE	RGB(189,215,238)	RGB(189,215,238)
5	Loggerhead				1 Sightings	3			Calibri	11	Calibri	12	TRUE		RGB(231,230,230)	RGB(231,230,230)
6	Green turtle				1 Sightings	4	SeaTurtle	Log	8.0	Calibri	11	Calibri	12			
7	Leatherback				1 Sightings	5	SeaTurtle	Gre	8.0	Calibri	11	Calibri	12			
8	Hawksbill				1 Sightings	6	SeaTurtle	Lea	8.0	Calibri	11	Calibri	12			
9	Kemp's ridley				1 Sightings	7	SeaTurtle	Haw	8.0	Calibri	11	Calibri	12			
10	Olive ridley				1 Sightings	8	SeaTurtle	Kem	8.0	Calibri	11	Calibri	12			
11	Flatback				1 Sightings	9	SeaTurtle	Oli	8.0	Calibri	11	Calibri	12			
12					1 Sightings	10	SeaTurtle	Fla	8.0	Calibri	11	Calibri	12			
13	Injured Sea Turtles				1 Sightings	11			Calibri	11	Calibri	12				
14	Injured				1 Sightings	12			Calibri	11	Calibri	12	TRUE		RGB(231,230,230)	RGB(231,230,230)
15	Not Injured				1 Sightings	13	Injuries	Inj	8.0	Calibri	11	Calibri	12			
16	Income and Expenses for Sightings Project (**)				2 Finances	14	Injuries	Ninj	8.0	Calibri	11	Calibri	12			
17		Total in USD			2 Finances	1		SheetTitle	Times New		Times New				RGB(206,206,206)	
18	Donations Received				2 Finances	2			Roman	14	Roman	14	TRUE			
19	Online donations				2 Finances	3			Calibri	11	Calibri	12		TRUE	RGB(189,215,238)	RGB(189,215,238)
20	On-site fund-raising				2 Finances	4	Donations	OL	14.2	Calibri	11	Calibri	12	TRUE	RGB(231,230,230)	RGB(231,230,230)
21					2 Finances	5	Donations	OS	14.2	Calibri	11	Calibri	12			
22	Expenses				2 Finances	6			14.2	Calibri	11	Calibri	12			
23	Travel				2 Finances	7			14.2	Calibri	11	Calibri	12	TRUE	RGB(231,230,230)	RGB(231,230,230)
24	Room and board				2 Finances	8	Expenses	Tra	14.2	Calibri	11	Calibri	12			
					2 Finances	9	Expenses	Rab	14.2	Calibri	11	Calibri	12			

We also add several columns, which are necessary to direct SAS to write the code for us. The first few additional columns are for SAS to map the SAS variable values to the corresponding row. The rest of the rows are for writing the VBA text that will transform the output into the template.

The following table describes the columns in this "Driver" table. The VBA elements are described briefly. How to use VBA in Excel will not be discussed in this paper.

		"Driver" Table Column Name	Description
		ColumnA	This is the value of the first column from each worksheet. As noted earlier, we changed "November 2023" to "***" to generalize the information. We will use SAS to replace ** to the correct month and year.
		ColumnB	This is the contents of the second column from each worksheet.
		WSIndex	This is the ordinal number for each worksheet as they appear in the reporting template. It is 1 for the Sightings worksheet rows and 2 for the Finances worksheet rows.
		WSName	This is the name of the worksheet the row is from.
		Row	This is the row number of each row in the worksheet. (e.g., row is 1 for the 1 st row of the worksheet, etc.)
		DTTable	This is the name of the data table where the VarName variable for ColumnB is from.
		VarName	This is the name of the variable whose value will be assigned to ColumnB. It is convenient if the VarName values are unique across the different DTTable values. If they are not unique, they must be referenced by using the correct DTTable value. The varname value for row 1 entries is SheetTitle to remind us that the corresponding ColumnA value is the formal worksheet title.
		SASNumFormat	The second column of the worksheets will be populated with a text value. Therefore, we need to control the conversion of the numeric values into character values by specifying a SAS numeric format to use with the conversion.
VBA Elements	FontNameA	This is the font for the ColumnA cell entry.	
	FontSizeA	This is the font size for the ColumnA cell entry.	
	FontNameB	This is the font for the ColumnB cell entry.	
	FontSizeB	This is the font size for the ColumnB cell entry.	
	FontBoldA	The value is TRUE if the ColumnA cell entry is to be bolded.	
	FontBoldB	The value is TRUE if the ColumnB cell entry is to be bolded.	
	InteriorColorA	This is the RGB value of the ColumnA cell entry shading.	
	InteriorColorB	This is the RGB value of the ColumnB cell entry shading.	

USE THE "DRIVER" TABLE TO WRITE THE CODE PROGRAMATICALLY

We read the "Driver" Table data and proceed to create the text needed to carry out the text.

For simplicity, we will take advantage of the fact that the variables were named uniquely across all the data sets used.

Code	Description
<pre>/*code with the four required data sets: SeaTurtle, Injuries, Donations, and Expenses*/ ... </pre>	Prepare the data sets that with the information required for completing the reporting template. In this example, the data sets SeaTurtle, Injuries, Donations, and Expenses will be in the SAS workspace memory.
<pre>filename specs "&rootpath./TemplateSpecs.xlsx"; proc import out=specs datafile=specs dbms=xlsx replace; sheet="specs"; getnames=yes; quit; </pre>	Read the "Driver" table and use the column name values as variable names (getnames=yes).
<pre>%let rptdate=November 2023; proc sort data=specs; by WSIndex Row; data prep; length ColumnA \$100; set specs ; by WSIndex Row; if row=1 and lowercase(varname)=:lowercase('SheetTitle') then columnA=tranwrd(columnA,'**',strip("&rptdate")); run; </pre>	Assign the value of the reporting month to rptdate macro variable. Sort the specs by worksheet index and row number. Replace ** to the reporting period in the first column and first row of each worksheet.

Code	Description
<pre>proc sql noprint; select distinct cats(dttable) into: TableList separated by ' ' from specs where dttable ne ''; quit; data AllStats; merge &TableList; run;</pre>	The macro variable TableList is a space-delimited list of the data sets needed for the report. Each data set in the list happens to have one row each so we can merge all the stats as shown.
<pre>filename ColBtxt "&rootpath./ColumnB Assignment.TXT"; **use only the varname records that do not contain Sheet Titles and that are not blank; data _null_; file ColBtxt; set prep (where= (not (varname='SheetTitle') and varname <> ' ')); temp=cats('if strip(varname)="',strip(varname),' then columnB=strip(put(',varname,',',strip(SASNumFor mat),'))'); put temp; keep varname temp WSName WSIndex row; run;</pre>	This creates the file ColBtxt. This code assigns the value to ColumnB. The value to assign to ColumnB is specified by the varname value. Assign the value if it is not SheetTitle and if varname has a value.

ColumnB Assignment.TXT Contains:

```
if strip(varname)="Log" then columnB=strip(put(Log,8.0));
if strip(varname)="Gre" then columnB=strip(put(Gre,8.0));
if strip(varname)="Lea" then columnB=strip(put(Lea,8.0));
if strip(varname)="Haw" then columnB=strip(put(Haw,8.0));
if strip(varname)="Kem" then columnB=strip(put(Kem,8.0));
if strip(varname)="Oli" then columnB=strip(put(Oli,8.0));
if strip(varname)="Fla" then columnB=strip(put(Fla,8.0));
if strip(varname)="Inj" then columnB=strip(put(Inj,8.0));
if strip(varname)="Ninj" then columnB=strip(put(Ninj,8.0));
if strip(varname)="OL" then columnB=strip(put(OL,14.2));
if strip(varname)="OS" then columnB=strip(put(OS,14.2));
if strip(varname)="Tra" then columnB=strip(put(Tra,14.2));
if strip(varname)="Rab" then columnB=strip(put(Rab,14.2));
```

Code	Description
<pre>proc sql; create table Allinfo as select a.*, b.* from Allstats as a CROSS JOIN prep as b; quit; data AllInfo; length ColumnB \$25; set AllInfo; %include ColBtxt; run;</pre>	Join the single-record in Allstats to each record in data set prep with a CROSS JOIN. Create the allinfo data set which is allinfo with columnB assigned by text in filename ColBtxt.
<pre>filename LoadWS "&rootpath./Worksheet List.TXT"; proc sql; create table WSlist as select distinct WSIndex, WSName from specs; quit; data _null_; length temp \$100; file LoadWS; set WSlist; temp=cats("%","LoadWS(WSIndex=",put(WSIndex,8.),", WSName=",WSName,"");"); put temp; run;</pre>	Using the information in specs, create the macro call statements that will load the Excel file.

Worksheet List. txt contains:

```
%LoadWS(WSIndex=1,WSName=Sightings);
%LoadWS(WSIndex=2,WSName=Finances);
```

Code	Description
<pre>filename XL "&rootpath./Report.xlsx"; %macro LoadWS(WSIndex,WSName); proc sql; create table temp as select columnA, columnB, WSIndex, row from AllInfo where WSIndex=&WSIndex order by row; quit; proc export data=temp(keep=columnA ColumnB) outfile=XL dbms=xlsx replace; sheet="&WSName."; putnames=NO; run; %mend LoadWS; %include LoadWS;</pre>	Define the Excel file with final results. Write the macro to write ColumnA and ColumnB values into each worksheet based on WSIndex and row order. The worksheets appear in WSIndex order and the rows are presented in row order. Call the macros by using the list of macro calls.

Code	Description
<pre> data _null_; length temp \$300; file "&rootpath./VBA Code.TXT"; set prep end=lastrecord; by WSIndex Row; if first.WSIndex then do; temp= cats('Sub FormatWorkSheets',put(WSIndex,2.)); put temp / ' Dim Cell as Range'; end; **make active a worksheet; if first.WSIndex then do; temp=cats('Worksheets(',put(WSIndex,2.),').Select'); put temp; **set column width of columns A and B; put 'Columns ("A:A").ColumnWidth=80'; put 'Columns ("B:B").ColumnWidth=25'; end; temp= cats ('Set Cell=Range("A',put(row,2.),'")'); put temp; put 'Cell.Select'; if FontItalicA ne '' then do; temp= cats ('Selection.Font.Italic=',FontItalicA); put temp; end; if FontColorA ne '' then do; temp= cats ('Selection.Font.Color=',FontColorA); put temp; end; if FontNameA ne '' then do; temp= cats ('Selection.Font.Name="",FontNameA,''); put temp; end; if FontsizeA ne '' then do; temp= cats('Selection.Font.Size=',put(FontSizeA,2.)); put temp; end; if FontboldA ne '' then do; temp= cats ('Selection.Font.Bold=',FontBoldA); put temp; end; if InteriorColorA ne '' then do; temp= cats('Selection.Interior.Color=', InteriorColorA); put temp; end; temp= cats ('Set Cell=Range("B',put(row,2.),'")'); put / temp; put 'Cell.Select'; </pre>	This section creates the complete VBA macro codes necessary to create the cosmetic changes for this example. One macro was created for each worksheet. Excel has a 64kb limit on VBA code per macro. If the code exceeds 64kb, Excel is unable to execute the code. The column widths for Columns A and B were pre-determined and assigned. For this contrived example, each cell that needed to be processed was selected as a single-cell range. Each cell's properties are then changed based on the values in the "Driver" table. The example illustrates simple string construction to create an external file with the code to be used in Excel.

<pre> if FontNameB ne '' then do; temp= cats ('Selection.Font.Name=',FontNameB,''); put temp; end; if FontsizeB ne '' then do; temp= cats ('Selection.Font.Size=', put(FontSizeB,2.)); put temp; end; if FontboldB ne '' then do; temp= cats ('Selection.Font.Bold=',FontBoldB); put temp; end; if InteriorColorB ne '' then do; temp= cats ('Selection.Interior.Color=', InteriorColorB); put temp; end; if last.WSIndex then put 'End Sub'//; run; </pre>	
---	--

VBA Code.Txt can contains a few hundred lines for this example. The following is an excerpt of the two complete macros that will perform the cosmetic changes to the default file produced by PROC EXPORT.

Sub FormatWorkSheets1

```

Dim Cell as Range
Worksheets(1).Select
Columns ("A:A").ColumnWidth=80
Columns ("B:B").ColumnWidth=25
Set Cell=Range("A1")
Cell.Select

```

```

Selection.Font.Name="Times New Roman"
Selection.Font.Size=14
Selection.Font.Bold=1
Selection.Interior.Color=RGB(206,206,206)

```

```

Set Cell=Range("B1")
Cell.Select
Selection.Font.Name="Times New Roman"
Selection.Font.Size=14

```

...

End Sub

Sub FormatWorkSheets2

```

Dim Cell as Range
Worksheets(2).Select
Columns ("A:A").ColumnWidth=65
Columns ("B:B").ColumnWidth=25
Set Cell=Range("A1")
Cell.Select

```

```

Selection.Font.Name="Times New Roman"
Selection.Font.Size=14
Selection.Font.Bold=1
Selection.Interior.Color=RGB(206,206,206)

```

```

Set Cell=Range("B1")
Cell.Select

```

```

Selection.Font.Name="Times New Roman"
Selection.Font.Size=14

```

...

End Sub

EXCEL OUTPUT PRODUCED BY THE "DRIVER" TABLE

The preceding SAS code produced two worksheets, which are simple and not formatted. Note that the ColumnB entries are in text format and are all left-justified. The text format can handle non-numeric values as well in Column B. By default, text is left-justified in Excel. You could stop here and just take the column B values since they are in the right order and can be cut and paste into the original template. But if you want to go further, you can run the VBA code to transform the appearance of the original Excel output from SAS.

Completed Sightings Worksheet:

	A	B
1	International Wildlife Counts (November 2023	
2		Number of Sightings
3	Sea Turtle Species	
4	Loggerhead	567
5	Green turtle	1232
6	Leatherback	359
7	Hawksbill	12
8	Kemp's ridley	23
9	Olive ridley	12
10	Flatback	34
11		
12	Injured Sea Turtles	
13	Injured	238
14	Not Injured	2001

Completed Finances Worksheet:

	A	B
1	Income and Expenses for Sightings Project (No	
2		Total in USD
3	Donations Received	
4	Online donations	138123.00
5	On-site fund-raising	12345.12
6		
7	Expenses	
8	Travel	23782.00
9	Room and board	6230.12

EXCEL OUTPUT MODIFIED BY VBA IN EXCEL

The program produces a VBA in Excel macro. Copy and paste the text into the VBA editor and run it and voila, the two worksheets transform to have the same appearance as the original template.

Sightings Worksheet:

	A	B
1	International Wildlife Counts (November 2023)	
2		Number of Sightings
3	Sea Turtle Species	
4	Loggerhead	567
5	Green turtle	1232
6	Leatherback	359
7	Hawksbill	12
8	Kemp's ridley	23
9	Olive ridley	12
10	Flatback	34
11		
12	Injured Sea Turtles	
13	Injured	238
14	Not Injured	2001

Finances Worksheet:

	A	B
1	Income and Expenses for Sightings Project (November 2023)	
2		Total in USD
3	Donations Received	
4	Online donations	138123.00
5	On-site fund-raising	12345.12
6		
7	Expenses	
8	Travel	23782.00
9	Room and board	6230.12

PARALLEL PROCESSING AND AUTOMATING THE VALIDATION OF ENTRIES

Now that we see the automation potential when such a "Driver" Table exists, let us take it a step further by providing an example where we automate the validation of the entries during parallel processing. We add two additional columns (highlighted below) to the "Driver" Table. The first one is chkrDTTable and the second one is chkrVarname. These two columns indicate the name of the data table and the variable name for each value that needs to be checked. To simplify things, the chkrVarnames are unique across DTable values.

	A	B	C	D	E	F	G	H	I
1	ColumnA	ColumnB	WSIndex	WSName	Row	DTable	VarName	chkrDTTable	chkrVarname
2	International Wildlife Counts (**)								
3		Number of Sightings	1	Sightings	1		SheetTitle		
4	Sea Turtle Species		1	Sightings	2				
5	Loggerhead		1	Sightings	3				
6	Green turtle		1	Sightings	4	SeaTurtle	Log	chkrST	Lo
7	Leatherback		1	Sightings	5	SeaTurtle	Gre	chkrST	Gr
8	Hawksbill		1	Sightings	6	SeaTurtle	Lea	chkrST	Le
9	Kemp's ridley		1	Sightings	7	SeaTurtle	Haw	chkrST	Ha
10	Olive ridley		1	Sightings	8	SeaTurtle	Kem	chkrST	Ke
11	Flatback		1	Sightings	9	SeaTurtle	Oli	chkrST	OI
12			1	Sightings	10	SeaTurtle	Fla	chkrST	FI
13	Injured Sea Turtles		1	Sightings	11				
14	Injured		1	Sightings	12				
15	Not Injured		1	Sightings	13	Injuries	Inj	chkrINJ	NumInj
16			1	Sightings	14	Injuries	Ninj	chkrINJ	NumNinj
17	Income and Expenses for Sightings Project (**)								
18		Total in USD	2	Finances	1		SheetTitle		
19	Donations Received		2	Finances	2				
20	Online donations		2	Finances	3				
21	On-site fund-raising		2	Finances	4	Donations	OL	chkrDon	OL
22			2	Finances	5	Donations	OS	chkrDon	OS
23	Expenses		2	Finances	6				
24	Travel		2	Finances	7				
25	Room and board		2	Finances	8	Expenses	Tra	chkrExp	Tra
26			2	Finances	9	Expenses	Rab	chkrExp	Rab

We will carry out the following steps:

1. Read each worksheet from the completed Excel template.
2. Perform a UNION ALL operation on all the worksheets in Step 1.
3. Join Step 2 data with specs data to include the chkrDTTable and chkrVarnam columns.
4. Create a data set with a single record containing all variables listed in chkvarnames. We will transpose the data to produce a data set with the first column being the variable name in chkrvarname and the second column being the value for the

corresponding chkvarname variable that we want to check against the value in the completed Excel template.

5. Join the transposed data in Step 4 to the data in Step 3, which will allow us to compare the value in the Excel file against the values of the corresponding chkr variable.

Code	Description
<pre>%macro LoadWS(WSName,WSIndex); data _null_; length dsn \$32.; dsn=compress("&WSName",' ','ak'); call symputx('dsn',dsn); run; proc import out=&WSName datafile=XL dbms=xlsx replace; sheet="&WSName"; getnames=no; quit; data &WSName; set &WSName; row=_n_; WSName="&WSName"; run; %mend LoadWS; %include LoadWS;</pre>	Read all the worksheets in the completed Excel file that was output earlier using Filename XL. We will reuse the macro calls used earlier to create the worksheets in Filename XL. The _data_ null step assigns a data set name to macro variable dsn. The value is derived by removing spaces and keeping only alphabetic characters in the data set name. SAS data set names cannot have spaces among other things. Adjust this derivation as needed. But in this %LoadWS macro, we will use getnames=no instead of getnames=yes. Also create the variable WSName (worksheet name) and the variable row (row number within the worksheet).

Code	Description						
<pre> proc sql; create table TableList as select distinct WSName from specs; quit; filename RPTUnion "&rootpath./RPT union.TXT"; data _null_; file RPTUnion; set tablelist end=eof; if _n_=1 then do; put 'proc sql;'; put 'create table RPTUnion as ' @/; end; put 'select * from ' @; put WSName; if not eof then put 'union all'; if eof then put ';' / 'quit;'; run; %include RPTUnion; </pre>	Create a table with the unique worksheet names in specs. <table border="1"> <thead> <tr> <th></th><th>WSName</th></tr> </thead> <tbody> <tr> <td>1</td><td>Finances</td></tr> <tr> <td>2</td><td>Sightings</td></tr> </tbody> </table> Write the SAS code to carry out the UNION ALL JOIN involving all the worksheets read in previous step. The Union ALL JOIN has the advantage of being able to set the length of character variables so that data will not be truncated. (Truncation can happen easily when using the SET statement to concatenate data sets.) The text written is: <pre> proc sql; create table RPTUnion as select * from Finances union all select * from Sightings ; quit; </pre>		WSName	1	Finances	2	Sightings
	WSName						
1	Finances						
2	Sightings						

Data set RPTUnion:

Obs	A	B	row	wsname
1	Income and Expenses for Sightings Project (November 2023)		1	Finances
2		Total in USD	2	Finances
3	Donations Received		3	Finances
4	Online donations	138123.00	4	Finances
5	On-site fund-raising	12345.12	5	Finances
6			6	Finances
7	Expenses		7	Finances
8	Travel	23782.00	8	Finances
9	Room and board	6230.12	9	Finances
10	International Wildlife Counts (November 2023)		1	Sightings
11		Number of Sightings	2	Sightings
12	Sea Turtle Species		3	Sightings
13	Loggerhead	567	4	Sightings
14	Green turtle	1232	5	Sightings
15	Leatherback	359	6	Sightings
16	Hawksbill	12	7	Sightings
17	Kemp's ridley	23	8	Sightings
18	Olive ridley	12	9	Sightings
19	Flatback	34	10	Sightings
20			11	Sightings
21	Injured Sea Turtles		12	Sightings
22	Injured	238	13	Sightings
23	Not Injured	2001	14	Sightings

Code	Description
<pre>proc datasets lib=work memtype=data noprint; modify RPTUnion; attrib _all_ format=; attrib _all_ label=''; attrib _all_ informat=; contents data=RPTUnion; run;</pre>	This removes any formats, labels, and informats in the RPTUnion data set. This prevents confusion regarding the new attributes of the variables in RPTUnion, which is a consolidation of variables from different data sets where SAS resolves selects the final attributes, such as variable length. For example, the new length of a variable may be longer than what is indicated in the Format attribute.
<pre>proc sql noprint; select distinct cats(chkrdtttable) into: chkrTableList separated by ' ' from specs where chkrdtttable ne ''; order by chkrdtttable; quit; data chkrAllStats; merge &chkrTableList; run; proc transpose data=chkrAllStats out=tranStats (rename=(col1=b _name_=varname) drop=_label_); run;</pre>	Create a macro variable with the space-separated list of tables in chkrdtttable. Then merge the data in these single-record data sets. Transpose the data set to create a data set with two columns: chkrVarname value and the corresponding variable's value, which will be called chkrColB.

Data set chkrAllStats:

	OL	OS	Tra	Rab	NumInj	NumNin	Lo	Gr	Le	Ke	Ha	FI
1	12	12345	23782.00	6230.12	238	2001	567	1232	359	23	0	34

Data set tranStats: (Varname is in the first column and is labelled by SAS as shown.)

	NAME OF FORMER VARIABLE	b
1	OL	12
2	OS	12345.12
3	Tra	23782
4	Rab	6230.12
5	NumInj	238
6	NumNin	2001
7	Lo	567
8	Gr	1232
9	Le	359
10	Ke	23
11	Ha	0
12	FI	34

Code	Description
<pre>proc sql; create table RPT_Specs_chkr as select a.a as RPTColA, a.b as RPTColB, b.WSIndex, b.WSName, b.row, b.dtttable as pgmrddtable, b.varname as pgmrvarname, b.chkrddtable, b.chkrvarname, b.SASNumFormat, c.b as chkrColB from RPTUnion as a LEFT Join specs as b on a.WSName=b.WSName and a.row=b.row LEFT JOIN transtats as c on upcase(b.chkrvarname)=upcase(c.varname) where b.varname ne '' and b.chkrvarname ne '' order by b.WSIndex, b.row; quit;</pre>	Join the data from RPTUnion, specs, and transtats.
<pre>data finalcheck; set rpt_specs_chkr; difference=RPTColB-chkrColB; run;</pre>	RPTColB in this data set is a character variable. SAS will convert RPTColB to a number to calculate the difference. We want the difference to be 0.

Data set finalcheck:

	RPTColA	RPTColB	WSIndex	WSName	Row	DTTable	VarName	chkrDTTable	chkrVarname	SASNumFormat	chkrColB	difference
1	Loggerhead	567	1	Sightings	4	SeaTurtle	Log	chkrST	Lo	8.0	567	0
2	Green turtle	1232	1	Sightings	5	SeaTurtle	Gre	chkrST	Gr	8.0	1232	0
3	Leatherback	359	1	Sightings	6	SeaTurtle	Lea	chkrST	Le	8.0	359	0
4	Hawksbill	12	1	Sightings	7	SeaTurtle	Haw	chkrST	Ha	8.0	0	12
5	Kemp's ridley	23	1	Sightings	8	SeaTurtle	Kem	chkrST	Ke	8.0	23	0
6	Olive ridley	12	1	Sightings	9	SeaTurtle	Oli	chkrST	Ol	8.0	12	0
7	Flatback	34	1	Sightings	10	SeaTurtle	Fla	chkrST	Fl	8.0	34	0
8	Injured	238	1	Sightings	13	Injuries	Inj	chkrINJ	NumInj	8.0	238	0
9	Not Injured	2001	1	Sightings	14	Injuries	Ninj	chkrINJ	NumNinj	8.0	2001	0
10	Online donations	138123.00	2	Finances	4	Donations	OL	chkrDon	OL	14.2	12	138E3
11	On-site fund-raising	12345.12	2	Finances	5	Donations	OS	chkrDon	OS	14.2	12E3	0
12	Travel	23782.00	2	Finances	8	Expenses	Tra	chkrExp	Tra	14.2	24E3	0
13	Room and board	6230.12	2	Finances	9	Expenses	Rab	chkrExp	Rab	14.2	6230	0

Code	Description
<pre> proc sql; select count(*) into :Ndifferences from finalcheck where difference ne 0; quit; %macro FinalMessage; data _null_; title; file print; put _page_; if &Ndifferences=0 then do; put "NO DIFFERENCES FOUND"; put "PROCEED WITH REPORT SUBMISSION";end; else do; put "DIFFERENCES FOUND"; PUT "DO NOT PROCEED WITH REPORT SUBMISSION"; end; run; %mend FinalMessage; %FinalMessage; </pre>	We count the number of records where the difference is not zero. If there are no differences, we have success.

We found two records with differences:

	RPTColA	RPTColB	WSIndex	WSName	Row	DTTable	VarName	chkrDTTable	chkrVarname	SASNumForma	chkrColB	difference
1	Hawksbill	12	1	Sightings	7	SeaTurtle	Haw	chkrST	Ha	8.0	0	12
2	Online donations	138123.00	2	Finances	4	Donations	OL	chkrDon	OL	14.2	12	138111

CONCLUSION

The SAS programming language offers different tools to help us with our work. We can develop solutions that help us achieve the desired result faster and with less effort.

CONTACT INFORMATION

Imelda C. Go
Palmetto GBA
imelda.go@palmettogba.com

Abbas S. Tavakoli
University of South Carolina
abbas.tavakoli@sc.edu

TRADEMARK NOTICE

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.