PhUSE 2017

Paper PP19

A Sample Paper for a PhUSE Author

Toni Rodríguez, Chiltern International Ltd, Madrid, Spain
Sebastià Barceló, Syntax for Science SL, Mallorca, Spain

ABSTRACT

Many times in our daily work, it is needed to import, export or work with an Excel files, functionality that base SAS does not have. To be able to work with Microsoft Excel from SAS you need SAS/ACCESS an extra feature that not everyone is able to purchase. This is the reason why we design a process with SAS macro, able to work with Excel using Visual basic, via Windows command lines. On the one hand, it is a free process, making able to work with those kind of files to the little companies, but on the other hand, this process take more time than using SAS/ACCESS and the macro code is done for Windows. In conclusion, the SAS/ACCESS requirement can be avoided.

Introduction

In many clinical trials, the software used to transfer data is an Excel file because is easy to use and popular tool. There are many SAS users than do not have SAS/ACCESS and always that they need to work with those kind of files they must to do a prior step such as save the file as a .CSV.
This process requires time, not much, but if the data transfer should be done periodically, this imply to do it every time that file is transferred.

IMPORT Excels

A common problem that all SAS users without SAS/ACCESS has suffered is to receive a ".XLS" files and change the format to ".CSV" for each transfer with the time that this means.
To solve that problem, we propose the visual basic approach.

Visual basic

All Microsoft programs, such as Word or Excel, works with Visual Basic. Visual Basic is a programming language developed in the early 90s and it has been used since then in different versions.
This language could be used to define macros or routines to perform tasks; such apply formats to a row, create a table or even export files and save as other format.
Mainly, we can speak of two kinds of Visual Basic Scripts:

1. Visual basic script files ".VBS"
2. Visual basic macros into a document

The first ones are independent files that define its own objects and can be executed independently. The second ones, need to be defined into an Excel or a Word and can use reference as "ThisWorkbook". The language differs a little between two versions.
We are going to focus in the first one on this section.

Visual basic Script

Keeping in mind our goal of importing a ".XLS" we could create a visual basic script that transform our XLS file into a CSV.

```
if WScript.Arguments.Count < 2 Then
            WScript.Echo "Please specify the source and the destination files.
Usage: ExcelToCsv <xls/xlsx source file> <csv destination file>"
            Wscript.Quit
End If

csv_format = 6

Set objFSO = CreateObject("Scripting.FileSystemObject")

src_file = objFSO.GetAbsolutePathName(Wscript.Arguments.Item(0))
dest_file = objFSO.GetAbsolutePathName(WScript.Arguments.Item(1))

Dim oExcel
Set oExcel = CreateObject("Excel.Application")

Dim oBook
```

```
Set oBook = oExcel.Workbooks.Open(src_file)

oBook.SaveAs dest_file, csv_format

oBook.Close False
oExcel.Quit

WScript.Quit 0
```

This Script, expect two parameters, the source file and the destination file. The script call excel and perform the action SaveAs.

If this script file is called XLS2CSV, we can call throw command line the file as

```
Start /Wait XLS2CSV "example.xls" "output.csv"
```

That could be done throw SAS/BASE with a DATA _NULL_ to create the file and the X command.
After that, a simply PROC IMPORT or DATA _NULL_ to read it should be enough. In many cases is useful take into account that sometimes, excel files contain special characters as carrying return that can be deleted. See reference for more details.

Macro process

The main problem of use external scripts, is that you could try to import the CSV files prior to the execution of the VBS script has finished. To prevent that, we have perform a macro that use SLEEP function, always that the file is in use.

```
* Select on &infile. the tab specified.
%Import_XLS(
  infile     = Example_file.xlsx,
  dsout      = Datasets,   /*Name in sas for the imported sheet*/
  tabname    = Sheet2,     /*Name of the sheet that we want to import*/
  path       = U:          /*location of the file*/
);
```

All macro used can be found in the Appendix I.

Export to xls, xlsx or xlsm

We can do a similar approach for the opposite task.
If we want to export a unique SAS dataset, we could create a VBS similar to the one of the previous sections and open a CSV file and save as XLS.

The call is as follows

```
%let Outputspath=U:;

proc export data=sashelp.class (where=(sex='F'))
  outfile="&Outputspath.\Femalelist.csv"
  dbms=csv
  replace;
run;

proc export data=sashelp.class (where=(sex='M'))
  outfile="&Outputspath.\Malelist.csv"
  dbms=csv
  replace;
run;

%CSVToAny(
  Infile = Femalelist.csv,
  Outfile = Class.xlsm,
  Path = &Outputspath.
);
```

Now we have export the Female list into a XLSM (excel file with macro enabled), but in that file does not appear the Malelist and already is a excel file that cannot be edited by SAS/BASE.

Here is where the second Visual basic kind of types become important, the visual basic macros defined into a document.

Visual basic throw macros

We could create visual basic macros into a excel files. Also there is two ways to create a macro. The first one, do not need to know the visual basic language, since we can record our own actions and get the visual basic code. The second one is typing the code, this required to know some visual basic.

For example, given a XLS file, we could create a visual basic macro that import a CSV file in other sheet. The code could be something like as follows.

```
Sub Import_CSV(rootDir As String, FileName As String, SheetName As String)
' filename = CSV filename without directory (test.csv)
' outSheet = name of the worksheet in the current workbook
' where the data should go, will start in A1
Dim outSheet As String
Dim connectionName As String
outSheet = SheetName
connectionName = "TEXT;" & rootDir & "\" & FileName
Sheets.Add.Name = outSheet
With Worksheets(outSheet).QueryTables.Add(Connection:=connectionName,
Destination:=Worksheets(outSheet).Range("A1"))
.Name = FileName
.FieldNames = True
.RowNumbers = False
.FillAdjacentFormulas = False
.PreserveFormatting = True
.RefreshOnFileOpen = False
.RefreshStyle = xlOverwriteCells
.SavePassword = False
.SaveData = True
.AdjustColumnWidth = True
.RefreshPeriod = 0
.TextFilePromptOnRefresh = False
.TextFilePlatform = 437
.TextFileStartRow = 1
.TextFileParseType = xlDelimited
.TextFileTextQualifier = xlTextQualifierDoubleQuote
.TextFileConsecutiveDelimiter = False
.TextFileTabDelimiter = False
.TextFileSemicolonDelimiter = False
.TextFileCommaDelimiter = True
.TextFileSpaceDelimiter = False
.Refresh BackgroundQuery = False
End With
       Sheets(SheetName).QueryTables(SheetName & ".csv").Delete
End Sub
```

Throw Visual basic script language, this macro could add a new tab into an Excel file. We have done a macro to do so. The call is something like

```
data Import_CSV;
format string $1000.;
string =      "Sub Import_CSV(rootDir As String, FileName As String,SheetName As
String)";output;
string =      "' filename = CSV filename without directory (test.csv)";output;
string =      "' outSheet = name of the worksheet in the current workbook";output;
string =      "' where the data should go, will start in A1";output;
string =      " Dim outSheet As String";output;
string =      " Dim connectionName As String";output;
string =      ' outSheet = SheetName';output;
string =      ' connectionName = "TEXT;" & rootDir & "\" & FileName';output;
string =      ' Sheets.Add.Name = outSheet';output;
string =      ' With
Worksheets(outSheet).QueryTables.Add(Connection:=connectionName,
Destination:=Worksheets(outSheet).Range("A1"))';output;
```

```
string =      ' .Name = FileName';output;
string =      ' .FieldNames = True';output;
string =      ' .RowNumbers = False';output;
string =      ' .FillAdjacentFormulas = False';output;
string =      ' .PreserveFormatting = True';output;
string =      ' .RefreshOnFileOpen = False';output;
string =      ' .RefreshStyle = xlOverwriteCells';output;
string =      ' .SavePassword = False';output;
string =      ' .SaveData = True';output;
string =      ' .AdjustColumnWidth = True';output;
string =      ' .RefreshPeriod = 0';output;
string =      ' .TextFilePromptOnRefresh = False';output;
string =      ' .TextFilePlatform = xlMSDOS';output;
string =      ' .TextFileStartRow = 1';output;
string =      ' .TextFileParseType = xlDelimited';output;
string =      ' .TextFileTextQualifier = xlTextQualifierDoubleQuote';output;
string =      ' .TextFileConsecutiveDelimiter = False';output;
string =      ' .TextFileTabDelimiter = False';output;
string =      ' .TextFileSemicolonDelimiter = False';output;
string =      ' .TextFileCommaDelimiter = True';output;
string =      ' .TextFileSpaceDelimiter = False';output;
string =      ' .Refresh BackgroundQuery = False';output;
string =      ' End With';output;
string =      ' Sheets(SheetName).QueryTables(SheetName & ".csv").Delete ';output;
string =     "End Sub";output;
run;

%MacroToExcel(
  Xlpath = &Outputspath.,
  xlname = Class,
  dsmacro = Import_CSV
);
```

And with the same idea, this macro could be executed with next code

```
%execute_vba(
  xlpath = &Outputspath.,
  xlname = Class,
  xlmacro = Import_CSV,
  parameters=%str("&Outputspath.", "Malelist.csv", "Malelist")
);
```

Customize excel

The functionality that we have used until now, allow us to do the same that SAS/ACCESS to work with Excel, but with visual basic, we can increase the functionality, allowing us to do everything that can be done in a Excel, such as create tables to give format
Also we could add macros to perform visual improvements, for example, add tables to each sheet data.

```
Data Presentation;
format string $1000.;
string = 'Sub Presentation()'; output;

string = 'Sheets("Femalelist").Select';output;
string = 'Range("A1").Select';output;
string = 'Range(Selection, Selection.End(xlToRight)).Select';output;
string = 'Range(Selection, Selection.End(xlDown)).Select';output;
string = 'ActiveSheet.ListObjects.Add(xlSrcRange, Selection, , xlYes).Name =
_';output;
string = '"Femalelist"';output;
string = 'Sheets("Malelist").Select';output;
string = 'Range("A1").Select';output;
string = 'Range(Selection, Selection.End(xlToRight)).Select';output;
string = 'Range(Selection, Selection.End(xlDown)).Select';output;
string = 'ActiveSheet.ListObjects.Add(xlSrcRange, Selection, , xlYes).Name =
_';output;
string = '"Malelist"';output;
```

```
string = 'End sub';output;

run;

%MacroToExcel(
  Xlpath = &Outputspath.,
  xlname = Class,
  dsmacro = Presentation
);
%execute_vba(
  xlpath = &Outputspath.,
  xlname = Class,
  xlmacro = Presentation
);
```

Executing all this code pieces is a hard work, this is the reason why, we have create a macro, able to export multiple SAS datasets into differents tabs, applying the table format. The call to the macro is as follows

```
Data malelist;
  set sashelp.class (where=(sex='M'))
run;

Data femalelist;
  set sashelp.class (where=(sex='F'))
run;

%MappingToXLS(
  outfile     = Export,
  infiles     = malelist femalelist,
  path = U:
);
```

Doing that, we obtain the following excel file

	A	B	C	D	E
1	Name	Sex	Age	Height	Weight
2	Alfred	M	14	69	112.5
3	Henry	M	14	63.5	102.5
4	James	M	12	57.3	83
5	Jeffrey	M	13	62.5	84
6	John	M	12	59	99.5
7	Philip	M	16	72	150
8	Robert	M	12	64.8	128
9	Ronald	M	15	67	133
10	Thomas	M	11	57.5	85
11	William	M	15	66.5	112
12					
13					

Malelist | Femalelist

CONCLUSION

In conclusion, with the help of visual basic, we can perform importation and exportations of Excel files or tabs from SAS/BASE. Not only that, we can do with SAS all the task that can be done from an excel file with a little of code.

As we say earlier, it takes more time than SAS/ACCESS, but is free and allow us to do a lot of different task. Also, open the posibility to work with other Microsoft products, since they work with VBA and in the future, maybe be able to modify a mock SAP with final results each time that programs are executed or update Power Points with the updated plot.

References

http://support.sas.com/kb/26/065.html
http://stackoverflow.com/questions/8434994/export-each-sheet-to-a-separate-csv-file.
http://stackoverflow.com/questions/1858195/convert-xls-to-csv-on-command-line
http://stackoverflow.com/questions/11899730/running-vbscript-from-batch-file

http://www.vb-helper.com/howto_csv_to_excel2.html
http://stackoverflow.com/questions/10232150/run-excel-macro-from-outside-excel-using-vbscript-from-command-line

Contact Information

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

Toni Rodríguez
Chiltern International Ltd
Email:Toni.Rodriguez@chiltern.com

Sebastià Barceló
Syntax for Science SL
Email: sbarcelo@syntaxfs.com

APPENDIX I (MACROS)

Import_XLS

```
%macro Import_XLS(
        infile=,
        outfile=,
        sheetname=,
        path=
) / store;

%LOCAL datetime_start opts called_by act_extension filepath i;
%let called_by=&SYSMACRONAME;
%init;
%* --- CHECKS --- *;
%IF %UPCASE(&debug)=Y %THEN %DO;
        %*Mandatory parameters are specified;
        %check_missing_params(AnyToCSV,infile path);
        %if &status=0 %then %goto EXIT;

        %*Infile checks;
        %if %words(&infile,delim=%str(.))>1 %then %do;
                %*Correct extension;
                %let act_extension = %scan(&infile,%words(&infile,delim=%str(.)),%str(.));
                %if %upcase(&act_extension) ne XLS and %upcase(&act_extension) ne XLSX %then %do;
                        %put ERROR: [AnyToCSV] INFILE extension should be .XLS or .XLSX.;
                        %goto EXIT;
                %end;
        %end;
        %else %do;
                %*Error if no extension;
                %put ERROR: [AnyToCSV] INFILE extension should be specified.;
                %goto EXIT;
        %end;

        %*Check that file exist;
        %let filepath=&path./&infile.;
        %if not %sysfunc(fileexist(&filepath)) %then %do;
                %PUT ERROR: [AnyToCSV] Path or filename are incorrect.;
                %let status = 0;
                %goto EXIT;
        %end;

%END;

%*Delete outfile if exist;
%let filepath=&path./&outfile.;
%if %sysfunc(fileexist(&filepath)) %then %do;
        filename _ERASE_ "&filepath";

        data _NULL_;
```

```
                rc=fdelete('_ERASE_');
        run;

        filename _ERASE_ clear;
%end;

data _null_;
        slept=sleep(2);
run;

%let fid=1;
%do %while(&fid ne 0);
        data _null_;
                slept=sleep(1);
        run;
        %let fid=%sysfunc(fileexist(&filepath));
%end;

%* --- END --- *;
%* --- CREATE VBS --- *;

options NOXWAIT NOXSYNC;
        %local filepath tmpfile rc fileref;

        %let tmpfile=C:\Users\TRodriguez\AppData\Local\Temp\Chiltern;
        %let rc = %sysfunc(filename(fileref,&tmpfile));

        %if %sysfunc(fexist(&fileref))=0 %then %do;
                options noxwait;
                %sysexec mkdir "&tmpfile." ;
        %end;

        data _NULL_;
                file "&tmpfile.\XLS2CSV.vbs";
                put 'if WScript.Arguments.Count < 2 Then';
                put '                   WScript.Echo "Please specify the source and the destination files. Usage:
ExcelToCsv <xls/xlsx source file> <csv destination file>"';
                put '                   Wscript.Quit';
                put 'End If';
                put '       ';
                put 'csv_format = 6';
                put '';
                put 'Set objFSO = CreateObject("Scripting.FileSystemObject")';
                put '';
                put 'src_file = objFSO.GetAbsolutePathName(Wscript.Arguments.Item(0))';
                put 'dest_file = objFSO.GetAbsolutePathName(WScript.Arguments.Item(1))';
                put '';
                put 'Dim oExcel';
                put 'Set oExcel = CreateObject("Excel.Application")';
                put 'oExcel.DisplayAlerts = False';
                put '';
                put 'Dim oBook';
                put 'Set oBook = oExcel.Workbooks.Open(src_file,0,False)';
                %if %length(&sheetname.) %then %do;
                        put "oBook.Sheets(""&sheetname."").Activate";
                %end;
                put '';
                put 'oBook.SaveAs dest_file, csv_format';
                put '';
                put 'oBook.Close False';
                put 'oExcel.Quit';
                put '';
                put 'WScript.Quit 0';
        run;

        filename _WAIT_ "&tmpfile.\XLS2CSV.vbs";
```

```
		%let fid=%sysfunc(fopen(_WAIT_,U));
		%do %while(&fid=0);
			%let fid=%sysfunc(fopen(_WAIT_,U));
			data _NULL_;
				slept=sleep(1);
			run;
		%end;
		%let rc=%sysfunc(fclose(&fid));
		filename _WAIT_ clear;

		%* --- END --- *;
%* --- PASSTOCSV --- *;

options NOXWAIT NOXSYNC;
x "cd &tmpfile.";
x "Start /Wait XLS2CSV ""%left(%trim(&path.))/%left(%trim(&infile))"" ""%left(%trim(&path))/_TEMP.CSV"" ";

data _null_;
		slept=sleep(5);
run;

filename _WAIT_ "%left(%trim(&path))/_TEMP.CSV";

%let fid=0;
%do %while(&fid=0);
		data _null_;
			slept=sleep(1);
		run;
		%let fid=%sysfunc(fopen(_WAIT_,U));
%end;
%let rc=%sysfunc(fclose(&fid));

proc import datafile="%left(%trim(&path))/_TEMP.CSV"
    out=&outfile.
    dbms=csv
    replace;
         guessingrows=1000;
    getnames=yes;
run;

filename _ERASE_ "%left(%trim(&path))/_TEMP.CSV";
		data _NULL_;
			rc=fdelete('_ERASE_');
		run;
filename _ERASE_ clear;

%* --- END --- *;

%exit;

%return;

%EXIT:
		%exit;
		%let status=0;


%mend Import_XLS;
```

CSVToAny

```
%macro CSVToAny(
		infile	=,
		outfile	=,
		path	=
) / minoperator;
```

```
	%*Delete outfile if exist;
	%let filepath=&path.\&outfile.;
	%if %sysfunc(fileexist(&filepath)) %then %do;
		filename _ "&filepath";

		data _NULL_;
			rc=fdelete('_ERASE_');
		run;

		filename _ERASE_ clear;
	%end;
%* --- END --- *;
%* --- PASSTOCSV --- *;

	%let act_extension		=		%scan(&outfile,%words(&outfile,delim=%str(.)),%str(.));

	%local filepath tmpfile rc fileref;

	%let tmpfile=C:\Users\TRodriguez\AppData\Local\Temp\Chiltern;
	%let rc = %sysfunc(filename(fileref,&tmpfile));
	%if %sysfunc(fexist(&fileref))=0 %then %do;
		 options noxwait;
		 %sysexec mkdir "&tmpfile." ;
	%end;

	data _NULL_;
		file "&tmpfile.\CSV2XL.vbs";
		put 'if WScript.Arguments.Count < 2 Then';
	  put '	WScript.Echo "Please specify the source and the destination files. Usage: CSV2XL <CSV
source file> <destination file>"';
	  put '	Wscript.Quit';
		put 'End If';
		put '	';
		put 'Set objFSO = CreateObject("Scripting.FileSystemObject")';
		put '	';
		put 'src_file = objFSO.GetAbsolutePathName(Wscript.Arguments.Item(0))';
		put 'dest_file = objFSO.GetAbsolutePathName(WScript.Arguments.Item(1))';
		put '	';
		put 'Dim oExcel';
		put 'Set oExcel = CreateObject("Excel.Application")';
		put '	';
		put 'Dim oBook';
		put 'Set oBook = oExcel.Workbooks.Open(src_file)';
		put '	';
		%if %upcase(&act_extension) eq XLS %then %do;
			put 'oBook.SaveAs dest_file, 1';
		%end;
		%else
		%if %upcase(&act_extension) eq XLSX %then %do;
			put 'oBook.SaveAs dest_file, 51';
		%end;
		%else
		%if %upcase(&act_extension) eq XLSM %then %do;
			put 'oBook.SaveAs dest_file, 52';
		%end;
		put '	';
		put 'oBook.Close False';
		put 'oExcel.Quit';
	run;

	filename _ERASE_ "&tmpfile.\CSV2XL.vbs";
	%let fid=%sysfunc(fopen(_ERASE_,U));
	%do %while(&fid=0);
		%let fid=%sysfunc(fopen(_ERASE_,U));
		data _null;
```

```
                        slept=sleep(1);
                run;
        %end;
        %let rc=%sysfunc(fclose(&fid));
        filename _ERASE_ clear;

        option noxwait XSYNC;
        x "cd &tmpfile.";
        x "Start /Wait CSV2XL ""%left(%trim(&path.))\%left(%trim(&infile))""
""%left(%trim(&path))\%left(%trim(&outfile))"" ";

        filename _ERASE_ "&outputspath.\&outfile.";
        %let fid=%sysfunc(fopen(_ERASE_,U));
        %do %while(&fid=0);
                %let fid=%sysfunc(fopen(_ERASE_,U));
                data _null;
                        slept=sleep(1);
                run;
        %end;
        %let rc=%sysfunc(fclose(&fid));
        filename _ERASE_ clear;

        filename _ERASE_ "&tmpfile.\CSV2XL.vbs";
        data _NULL_;
                rc=fdelete('_ERASE_');
        run;
        filename _ERASE_ clear;

%* --- END --- *;

        %exit;

        %return;

        %EXIT:
                %exit;
                %let status=0;


%mend CSVToAny;
```

MacroToExcel

```
%macro MacroToExcel(
        path    =,
        infile  =,
        inds    =
);
        ******************************;
        ***    Create VBS code     ***;
        *** to be placed into xlsm ***;
        ******************************;

        %local rc fid;

        data _NULL_;
                file "%sysfunc(strip(&path.))\Create_macro.vbs";
                set &inds end=EoF;
                if _N_=1 then do;
                        put 'Set objExcel = CreateObject("Excel.Application")';
                        put 'objExcel.Visible = False';
                        put 'objExcel.DisplayAlerts = False';
                        put "Set objWorkbook =
objExcel.Workbooks.Open(""%sysfunc(strip(&path.))\&infile..xlsm"")";
                        put '       Set xlmodule = objworkbook.VBProject.VBComponents.Add(1) ';
```

```
                put '       strCode = _';
        end;
        if not EoF then do;
                string2=tranwrd(string,'"','""');
                put '"' string2  '" & vbCr & _ ';
        end;
        else if EoF then do;
                string2=tranwrd(string,'"','""');
                put '"' string2  '"';
                put '       xlmodule.CodeModule.AddFromString strCode';
                put "objWorkbook.SaveAs ""%sysfunc(strip(&path.))\&infile..xlsm"" ";
                put 'objWorkbook.Close';
                put 'objExcel.Visible = True';
                put 'objExcel.DisplayAlerts = True';
                put 'objExcel.Quit';
                put 'Set objExcel = Nothing';
        end;
run;

**********************************;
*** Check when VBS is editable ***;
**********************************;
%*It means that is finished and can be executed;

filename _WAIT_ "%sysfunc(strip(&path.))\Create_macro.vbs";
%let fid=0;
%do %while(&fid=0);
        %let fid=%sysfunc(fopen(_WAIT_,U));
        data _null_;
                slept=sleep(1);
        run;
%end;
%let rc=%sysfunc(fclose(&fid));
filename _WAIT_ clear;
filename _WAIT_ "%sysfunc(strip(&path.))\&INFILE..xlsm";
%let fid=0;
%do %while(&fid=0);
        %let fid=%sysfunc(fopen(_WAIT_,U));
        data _null_;
                slept=sleep(1);
        run;
%end;
%let rc=%sysfunc(fclose(&fid));
filename _WAIT_ clear;

option noxwait XSYNC;
************************;
*** Execute VBS code ***;
************************;
x "Start /Wait %sysfunc(strip(&path.))/Create_macro.vbs";

filename _WAIT_ "&path.\&infile..xlsm";
%let fid=0;
%do %while(&fid=0);
        %let fid=%sysfunc(fopen(_WAIT_,U));
        data _null_;
                slept=sleep(1);
        run;
%end;
%let rc=%sysfunc(fclose(&fid));
filename _WAIT_ clear;


filename _ERASE_ "%sysfunc(strip(&path.))\Create_macro.vbs";
data _NULL_;
        rc=fdelete('_ERASE_');
```

```
	run;
	filename _ERASE_ clear;

%mend;
```

execute_vba

```
%macro execute_vba(
	path		=,
	infile		=,
	xlmacro		=,
	PARAMS			=
);

****************************************;
*** VBS code to execute macro on xlsm ***;
****************************************;
   %*Generate VBS code;
	data _NULL_;
		*generate a temporary file called Excel_macro to execute;
		file "%sysfunc(strip(&path.)).\Excel_macro.vbs";
		put 'Set objExcel = CreateObject("Excel.Application")';
		put 'objExcel.DisplayAlerts = False';
		put 'objExcel.Visible = False';
		put "Set objWorkbook = objExcel.Workbooks.Open(""&path.\&infile..xlsm"")";
		put 'objExcel.Application.Run "' @;
		%if %length(&PARAMS) %then %do;
			put '"&path.\&infile..xlsm'!&xlmacro." "', _';
			*Codify double quotation mark to a double quotation mark, since it must go into
quotation marks;
			PARAMS="%BQUOTE(&PARAMS)";
			put  PARAMS;
		%end;
		%else %do;
			put '"&path.\&infile..xlsm'!&xlmacro." """;
		%end;
		put 'objWorkbook.Save';
		put 'objWorkbook.Close';
		put 'objExcel.DisplayAlerts = True';
		put 'objExcel.Visible = True';
		put 'objExcel.Application.Quit';
		put 'Set objExcel = Nothing';
	run;

****************************************;
*** Check if VBS is finished to execute **;
****************************************;

filename _WAIT_ "&path.\Excel_macro.vbs";

%let fid=0;
%do %while(&fid=0);
	%let fid=%sysfunc(fopen(_WAIT_,U));
	data _null_;
		slept=sleep(1);
	run;
%end;
%let rc=%sysfunc(fclose(&fid));
filename _WAIT_ clear;

**********************************;
*** Check if excel is prepared to**;
**********************************;

filename _WAIT_ "&path.\&infile..xlsm";
```

```
%let fid=0;
%do %while(&fid=0);
	%let fid=%sysfunc(fopen(_WAIT_,U));
	data _null_;
		slept=sleep(1);
	run;
%end;
%let rc=%sysfunc(fclose(&fid));
filename _WAIT_ clear;

******************;
*** Execute VBS ***;
******************;

option noxwait XSYNC;
x "Start /Wait %sysfunc(strip(&path.))./Excel_macro.vbs";

****************************************;
*** Wait until excel is prepared again ***;
****************************************;
filename _WAIT_ "&path.\&infile..xlsm";
%let fid=0;
%do %while(&fid=0);
	data _null_;
		slept=sleep(1);
	run;
	%let fid=%sysfunc(fopen(_WAIT_,U));
%end;
%let rc=%sysfunc(fclose(&fid));
filename _WAIT_ clear;

**********************;
*** Delete VBS code ***;
**********************;
%*Delete file after use it;
%let filepath=%sysfunc(strip(&path.)).\Excel_macro.vbs;
%if %sysfunc(fileexist(&filepath)) %then %do;
	filename _ERASE_ "&filepath";

	data _NULL_;
		rc=fdelete('_ERASE_');
	run;

	filename _ERASE_ clear;
%end;
%* --- END --- *;
%mend;
```

RemoveMacros

```
%macro RemoveMacros(
	path	= ,
	infile	=
);
	%local filepath path rc fileref;

	data _NULL_;
		file "&path.\Delete_macro.vbs";
			put 'Set objExcel = CreateObject("Excel.Application")';
			put '		objExcel.Visible = False';
			put '		objExcel.DisplayAlerts = False';
			put '		Dim objWorkbook';
			put '		Dim x';
```

```
                        put "               Set objWorkbook =
objExcel.Workbooks.Open(""&path.\&infile..xlsm"")";
                        put '               With objWorkbook.VBProject';
                        put '                       For x = .VBComponents.Count To 1 Step -1';
                        put '                                   Select Case .VBComponents(x).Type';
                        put "                                                   Case 1, 2, 3 'standard
module, class module, and userform";
                        put '
.VBComponents.Remove .VBComponents(x)';
                        put '                                   End Select';
                        put '                       Next';
                        put '                       For x = .VBComponents.Count To 1 Step -1';
                        put '
.VBComponents(x).CodeModule.DeleteLines _';
                        put '                                                                   1,
.VBComponents(x).CodeModule.CountOfLines';
                        put '                       Next';
                        put '               End With ';
                        put "               objWorkbook.SaveAs ""&path.\&infile..xlsx"", 51 ";
                        put '               objExcel.Visible = True';
                        put '               objExcel.DisplayAlerts = True';
                        put '               objExcel.Quit';
                        put 'Set objExcel = Nothing';
        run;

        filename _WAIT_ "&path.\Delete_macro.vbs";
        %let fid=0;
        %do %while(&fid=0);
                %let fid=%sysfunc(fopen(_WAIT_,U));
                data _null_;
                        slept=sleep(1);
                run;
        %end;
        %let rc=%sysfunc(fclose(&fid));
        filename _WAIT_ clear;

        *options NOXWAIT NOXSYNC;
        option noxwait XSYNC;
        x "Start /Wait &path.\Delete_macro.vbs";

        filename _WAIT_ "&path.\&infile..xlsm";
        %let fid=0;
        %do %while(&fid=0);
                %let fid=%sysfunc(fopen(_WAIT_,U));
                data _null_;
                        slept=sleep(1);
                run;
        %end;
        %let rc=%sysfunc(fclose(&fid));
        filename _WAIT_ clear;

        filename _ERASE_ "&path.\Delete_macro.vbs";
        data _NULL_;
                rc=fdelete('_ERASE_');
        run;
        filename _ERASE_ clear;

%mend;
```

Mapping_To_Excel

```
%macro MappingToXLS(
        outfile    = ,
        infiles    = ,
```

```
        path      =
);

%local actual_word i j rc fid lrecl;

        %let lrecl          =       %sysfunc(getoption(lrecl));

        options lrecl=MAX;

        %*For each file on the list;
        %do i=1 %to %sysfunc(countw(&infiles,%str( )));

                %let actual_word=%scan(&infiles,&i.,%str( ));

                proc export data=&actual_word.
    outfile="&path.\&actual_word..csv"
    dbms=csv
    replace;
                run;

                %*If first, then create XLSM file;

                %if &i=1 %then %do;
                        %CSVToXL(infile=%scan(&infiles,&i,%str( )).csv,outfile=&outfile..xlsm,path=&path.);

                        data _Import_CSV;
                                format string $1000.;
                                string    =       "Sub Import_CSV(rootDir As String, FileName As
String,SheetName As String)";output;
                                string    =       "' filename = CSV filename without directory
(test.csv)";output;
                                string    =       "' outSheet = name of the worksheet in the current
workbook";output;
                                string    =       "'          where the data should go, will start in A1";output;
                                string    =       "wscript.echo ""rootDir: "" & WScript.Arguments.Item(0)";
                                string    =       "wscript.echo ""FileName: "" &
WScript.Arguments.Item(1)";
                                string    =       "wscript.echo ""SheetName: "" &
WScript.Arguments.Item(2)";
                                string    =       "   Dim outSheet As String";output;
                                string    =       "   Dim connectionName As String";output;
                                string    =       '   outSheet = SheetName';output;
                                string    =       '   connectionName = "TEXT;" & rootDir & "\" &
FileName';output;
                                string    =       '   Sheets.Add.Name = outSheet';output;
                                string    =       '   With
Worksheets(outSheet).QueryTables.Add(Connection:=connectionName,
Destination:=Worksheets(outSheet).Range("A1"))';output;
                                string    =       '       .Name = FileName';output;
                                string    =       '       .FieldNames = True';output;
                                string    =       '       .RowNumbers = False';output;
                                string    =       '       .FillAdjacentFormulas = False';output;
                                string    =       '       .PreserveFormatting = True';output;
                                string    =       '       .RefreshOnFileOpen = False';output;
                                string    =       '       .RefreshStyle = xlOverwriteCells';output;
                                string    =       '       .SavePassword = False';output;
                                string    =       '       .SaveData = True';output;
                                string    =       '       .AdjustColumnWidth = True';output;
                                string    =       '       .RefreshPeriod = 0';output;
                                string    =       '       .TextFilePromptOnRefresh = False';output;
                                string    =       '       .TextFilePlatform = 437';output;
                                string    =       '       .TextFileStartRow = 1';output;
                                string    =       '       .TextFileParseType = xlDelimited';output;
                                string    =       '       .TextFileTextQualifier =
xlTextQualifierDoubleQuote';output;
                                string    =       '       .TextFileConsecutiveDelimiter = False';output;
```

```
                                    string  =       '       .TextFileTabDelimiter = False';output;
                                    string  =       '       .TextFileSemicolonDelimiter = False';output;
                                    string  =       '       .TextFileCommaDelimiter = True';output;
                                    string  =       '       .TextFileSpaceDelimiter = False';output;
                                    string  =       '       .Refresh BackgroundQuery = False';output;
                                    string  =       '   End With';output;
                                    string  =       '   Sheets(SheetName).QueryTables(SheetName &
".csv").Delete ';output;
                                    string  =       "End Sub";output;
                            run;

                                    %MacroToExcel(path=&path.,infile=&outfile.,inds=_Import_CSV);

                    %end;
                    %else %do;
                            %*Import visits to Excel;


%execute_vba(path=&path.,infile=&outfile.,xlmacro=Import_CSV,PARAMS=%str("&path.", "&actual_word..csv",
"&actual_word."));

                    %end;

                    %*Delete auxiliary CSV;
                            filename _ERASE_ "&path./&actual_word..csv";
                            %let fid=0;
                            %do %while(&fid=0);
                                    %let fid=%sysfunc(fopen(_ERASE_,U));
                                    data _null_;
                                            slept=sleep(1);
                                    run;
                            %end;
                            %let rc=%sysfunc(fclose(&fid));
                            data _NULL_;
                                    rc=fdelete('_ERASE_');
                            run;
                            filename _ERASE_ clear;

                    *If last, delete macros and pass from XLSM to XLS ;
                    %if &i=%sysfunc(countw(&infiles,%str( ))) %then %do;

                            data _Sort;
                            format string $1000.;
                                    string  =       "Sub Sort()";output;
                                    %do j=1 %to %sysfunc(countw(&infiles,%str( )));
                                            %let actual_word=%scan(&infiles,&j.,%str( ));
                                            string  =       " Sheets(""&actual_word"").Select";output;
                                            string  =       " Sheets(""&actual_word"").Move
Before:=Sheets(&j.)";output;
                                            string  =       "
Sheets(""&actual_word"").Range(""A1"").Select";output;
                                    string  =       "   Sheets(""&actual_word"").Range(Selection,
Selection.End(xlToRight)).Select";output;
                                    string  =       "   Sheets(""&actual_word"").Range(Selection,
Selection.End(xlDown)).Select";output;
                                    string  =       "   Sheets(""&actual_word"").ListObjects.Add(xlSrcRange,
Selection, , xlYes).Name = _";output;
                                            string  =       "                                       ""&actual_word""
";output;
                                            string  =       "Range(""&actual_word.[#All]"").Select";output;
                                            string  =       "
Sheets(""&actual_word"").Range(""A1"").Select";output;
                                            %end;

                                            string  =       "Application.DisplayAlerts = False";output;
```

```
                        string          =       "'---> Save Current workbook just in case latest additions not
saved";output;
                        string          =       "ActiveWorkbook.Save";output;
                                string      =       "End Sub";output;
                    run;

                    %MacroToExcel(path=&path.,infile=&outfile.,inds=_Sort);

                    %execute_vba(path=&path.,infile=&outfile.,xlmacro=Sort,PARAMS=);

                    %RemoveMacros(path=&path.,infile=&outfile);

                    filename _ERASE_ "&path./&outfile..xlsm";
                    data _NULL_;
                            rc=fdelete('_ERASE_');
                    run;
                    filename _ERASE_ clear;
                    filename _ERASE_ "&path./&outfile..vbs";
                    data _NULL_;
                            rc=fdelete('_ERASE_');
                    run;
                    filename _ERASE_ clear;
            %end;

    %end;

    proc datasets lib=work nolist;
     delete _SORT _Import_CSV;
    quit;

    options lrecl=&lrecl;
%mend;
```