PhUSE 2017

Paper CT11

Using VBScript For Perfecting Statistical Report

Ekaterina Torchinskaya, Data MATRIX Ltd, St. Petersburg, Russia
Andrey Myslivets, Data MATRIX Ltd, St. Petersburg, Russia

ABSTRACT

The clinical SAS® programmer in pharmaceutical industry has the primary responsibility of generating a variety of reports. The programs written in SAS generate output usually in form of tables, figures and listings (TFLs) needed for the analysis and reporting clinical study results. The field of clinical research is a strictly controlled area and TFLs collected in the report have to fulfill clinical regulatory requirements (EMEA, FDA and others local regulatory). Over time, each company defines and reuses the one template with minor modifications for producing all subsequent reports.
Under these conditions, one of the SAS programmer's duties is the automation of the report development process. SAS is one of the most widespread, powerful tools for implementing it but its possibilities are limited, and sometimes it is handy to know some tricks about using call of external programs with other programming languages.
This article demonstrates the usage of VBScript - how to write and to launch the VBScript directly from SAS. The article provides examples of potential challenges in clinical trials report development and examples of the main steps of writing and launching VBScript using SAS.

INTRODUCTION

This article demonstrates the usage of VBScript as SAS code for transformation tables, listings and graphs in the RTF format. The article examines several key issues:

1. When the VBScript can be useful for the Clinical SAS programmers and what problems can be solved using external programs writing?
2. How to write scripts through the SAS system?
3. How to launch VBScript directly from SAS?
4. How to find Visual Basic functions and procedures if you don't know the programming language?

Different pharm companies and contract research organizations (CROs) has non-identical templates, shells and specific layouts for TFLs, statistical reports and clinical study reports. According to the examples exposed in CDISC implementation guide, various open source Clinical Study Reports (CSR) and numerous internal SOPs of global pharma companies, TFLs in the SAR(Statistical Analysis Report) and CSR (Clinical Study Report) should contain additional information such as Sponsor's name, study name, the main table title, population or sample name, the special page numbering, the footnotes etc.
In addition, the structure of SAR includes the table of contents and reference on each object in it. Many but not all of this tasks can be fully performed in SAS. We thought about additional tools to automate the process of collecting TFLs and also to implement such features of RTF-documents which cannot be performed in SAS.

One of such problems, which couldn't be solved in SAS, was that SAS uses section breaks as a separator between pages. This was only in the documents with the landscape orientation. This problem wasn't solved till version SAS 9.4. To perform this, the VBScript, which replaces section breaks with page breaks was created. It is especially convenient that for running this script you don't have to run any other application, write any code to execute it. SAS makes by itself all work for replacement.

WHEN SAS IS NOT ENOUGH

The SAS software uses many tools in addition to just SAS language. It can be as well SAS Macro, SCL, SQL, using dynamic data exchange to work with MS® Office documents. Among all the tools the VBScript must be emphasized. VBScript allows to perform commands in MS Office that cannot be directly accomplished in SAS. When working with RTF documents the usage of VBScript with SAS significantly speeds up the process of documents transformation.

VBScript is a variant of MS Visual Basic language. It is written in text files with a .vbs extension. One of the advantages of VBScript language is understandable for MS Office applications. The use of VBScript can greatly enhance the capabilities of using a SAS program to control Office applications

Andrey Myslivets and Artem Dolgun, Data MATRIX LLC presented a poster in 2016 at the 37th Annual Conference of the International Society for Clinical Biostatistics (ISCB). There they developed the macro - “Universal SAS® macro for generating Statistical Analysis Report in one click for I-IV phase Clinical Trials”. This macro does a big job for creating the appropriate view of statistical reports [1]. The development of a part in this macro responsible for replacing section breaks by page breaks gave us an idea to create this VBScript.

CHANGE THE WORD TO THE BETTER. IT’S TIME TO START “SCRIPTING”

Creating of VBScript starts from creating _NULL_ dataset, where we write the path for this script using operator FILE. Dataset _null_ doesn’t create any output, so it is convenient to use when creating the script:

```
data _null_;
  file "C:\VBS_SAS\replace.vbs";
  put "…";
  …
run;
```

Thus, the external script file is created directly from the SAS system in the specified directory. Further, it is necessary to fill the created file with the Visual Basic script. A full cycle of filling, saving and launching the VBScript will be shown in the example with the section/page break managing in the final MS Word document.

When the file exists physically in the specified directory, the script writing started using SAS PUT operator. The operator writes the VB commands line by line:

```
put "Dim wrdApp: Set wrdApp = WScript.CreateObject(""Word.Application"")";
```

Here the Windows Scripting Host (WSH) automation object is created. The CreateObject function creates an object of a specified type. In ""Word.Application"" the Server name (Word) and the type/class of the object (in our case it is Application) are specified .

After that, we open the document that to be transformed:

```
put "Dim wrdDoc";

put "Set wrdDoc = wrdApp.Documents.Open(""C:\VBS_SAS\File_to_convert.RTF"")";
```

In Visual Basic variables are declared with the Dim statement followed by the name - wrdDoc in our case, then the keyword AS and the variable type follow. If the type is omitted, then the AS keyword is not specified and the variable is declared as Variant. The second PUT statement opens the specified document and adds it to the Documents collection. As you can see it the expression is enclosed in double quotation marks, because it is necessary for SAS to take the path that enclosed in quotation marks.

LET THE REPLACEMENT BEGIN

Next is the actual replacement of the specified text. Our task now is to find all section breaks occurrences and replace them with page breaks. The wdReplaceAll operator specifies the number of replacements to be made when find and replace is used. The wdReplaceAll = 2 operator replaces all occurrences of the defined string:

```
put "Dim wdReplaceAll";
put "wdReplaceAll = 2";
```

Next comes the beginning of the block, where the replacement will occur:

```
put "With wrdApp.Selection.Find";
put "    .ClearFormatting";
put "    .Replacement.ClearFormatting";
put "    .Text = ""^b""";
put "    .Replacement.Text = ""^m""";
```

```
put "    .Forward = True";
put "    .Wrap = wdFindContinue ";
put "    .Format = False";
put "    .MatchCase = False";
put "    .MatchWholeWord = True ";
put "    .Execute , , , , , , , , , , wdReplaceAll";
put "End With";
```

Write operators inside it:

```
put "    .ClearFormatting";
put "    .Replacement.ClearFormatting";
```

The operators ClearFormatting, .Replacement.ClearFormatting remove text and paragraph formatting from the text specified in a Text operator.

.Text - sets the text to replace. In our case it will be the “^b” special symbol, which means the section break.

.Forward = True – sets find operation to search forward through the document.

.Wrap = wdFindContinue - Returns or sets what happens if the search begins at a point other than the beginning of the document and the end of the document is reached

.Format = False - formats the expression in accordance with the instruction. In this case, no formatting is applied(False).

Next, you can specify any operators. We used the following:

```
.MatchCase = False
.MatchWholeWord = True
```

.MatchCase - the find operation is not case sensitive
.MatchWholeWord - Gets or sets a value indicating whether the search matches whole words only(True).
.Execute , , , , , , , , , , wdReplaceAll – this statement executes the commands and performs the replacement.

OTHER REPLACEMENTS

Using this code any conversion in Word documents can be done. It can be performed by choosing a necessary symbol, for example from the list below.

^p	Paragraph Mark	^c	Clipboard Contents	^g	Graphic
^t	Tab Character	^n	Column Break	^l	Manual Line Break
^?	Any Character	^+	Em Dash	^m	Manual Page Break
^#	Any Digit	^=	En Dash	^~	Nonbreaking Hyphen
^$	Any Letter	^e	Endnote Mark	^s	Nonbreaking Space
^^	Caret Character	^d	Field	^-	Optional Hyphen
^u	Section Character	^&	Find What Text	^b	Section Break
^v	Paragraph Character	^f	Footnote Mark	^w	White Space

In the same way the problem of double Paragraph Mark at the end of each page was solved:

The code for this conversion is:

```
.Text = ""^p^p""";
.Replacement.Text = " ";
```

Later, working in version SAS 9.4, where the automatic replacement of section breaks to page breaks was already implemented, we made some changes to our macro. Now it removes section breaks that are created with page breaks. This is performed by changing a special character.

SAVE AND CLOSE YOUR WORD DOCUMENT AND SCRIPT

Having manipulations done, save the specified document with a new name or format. Enclose the FileName in a double quotation marks. The default is the current folder and file name. If a document has never been saved, the default name is used (for example, Doc1.doc). If a document with the specified file name already exists, the document is overwritten without the user being prompted first.

```
put "wrdDoc.SaveAS (""&infile."");
put "wrdDoc.Close SaveChanges=True";
```

The second statement closes the specified document. SaveChanges is an optional argument that sets the save action for the document. There can be one of the following WdSaveOptions constants: wdDoNotSaveChanges , wdPromptToSaveChanges, or wdSaveChanges .

Then you need to close the application. This is perfor med by the following command:

```
put "wrdApp.Quit";
```

wrdApp.Quit - quits Microsoft® Word and optionally saves or routes the open documents.

The last thing to be done is to clear the context of all created objects:

```
put "Set wrdApp = Nothing";
put "Set wrdDoc = Nothing";
```

The Nothing keyword in VBScript is used to disassociate an object variable from any actual object. Use the Set statement to assign Nothing to an object variable.

RUN THE SCRIPT

The VBScript can be manually run by double click on the "C:\VBS_SAS\replace.vbs" in the pre-specified folder or the SAS code for VBScript running can be created in SAS:

```
data _null_;
call system("&vbs\&scriptname.vbs");
run;
```

OR

```
filename script "&vbs\&scriptname.vbs";
…
X script;
```

SAS can communicate [2, 4] with the VBScript/VBA programs through the “X” command. It is really useful for MS Office output control, especially when you already have developed your own VBScript/VBA programs library. Calling one of the following string in SAS can produce a great job:

```
X "'&VBS_code.' ""&Input_Excel"" ""&output_excel"" ""&bas_code_path"" ""&vba_module""
""&vba_code"" ";
```

In this way it is possible to run external file directly from SAS.

OTHER TASKS THAT ARE WITHIN THE VB SCRIPT CAPABILITIES

First question after all TFLs are developed is how to merge multiple documents into one. It is possible to deal with that task in different ways such as using Microsoft Office or Adobe Acrobat® tools, but it's more interesting to manage the task using SAS, isn’t it?

Here is a simplest example of the script for merging `table01.rtf`, `table02.rtf` and `table03.rtf` into one document with page breaks between objects:

```
Const wdPageBreak = 7

Set objWord = CreateObject("Word.Application")
objWord.Visible = TRUE
Set objDoc = objWord.Documents.Add
Set objSelection = objWord.Selection

objDoc.Bookmarks("\EndOfDoc").Range.InsertFile "C:\VBS_SAS\TFLs\table01.rtf"
objDoc.Characters.Last.Select
objSelection.InsertBreak(wdPageBreak)
objDoc.Bookmarks("\EndOfDoc").Range.InsertFile "C:\VBS_SAS\TFLs\table02.rtf"
objDoc.Characters.Last.Select
objSelection.InsertBreak(wdPageBreak)
objDoc.Bookmarks("\EndOfDoc").Range.InsertFile "C:\VBS_SAS\TFLs\table03.rtf"
```

This example can be modified into more complicated script using cycles, arrays, kinds of filters, etc. This can be done both at the SAS level (with some macro-variables, %scan and other functions and SAS SQL) and at the VBScript level, using, for example, following if-then statement:

```
Dim wdApp
Dim strErr

wrdPageBreak = 7

On Error Resume Next
Set fso = CreateObject("Scripting.FileSystemObject")
Set folder = fso.GetFolder("C:\VBS_SAS\TFLs\")
Set Files = folder.Files

Set objWord = CreateObject("Word.Application")
objWord.Visible = true

Set objDoc = objWord.Documents.Add
Set objSelection = objWord.Selection
For Each fil In Files
       If (Right(LCase(fil), 4) = ".rtf" AND InStr(fil, "~$") = 0) then
           objDoc.Bookmarks("\EndOfDoc").Range.InsertFile fil.Path
           objDoc.Characters.Last.Select
           objSelection.InsertBreak(wdPageBreak)
       End if
Next
```

The script takes the list of all file names in the specified folder, then launches MS Word and creates a new document; next step is sequentially inserting of all ".rtf" files from this folder in alphabetical order with page breaks between the objects.

Another use case of VBScript is to developed PDF-manipulation/control/automate SAS macros, that make possible following process [3]:

- Converting listings and tables in ASCII, or RTF/Word format into PDF documents, or vice versa;
- Merging multiple PDF documents;
- Creating a PDF synopsis for PDF document review;
- Adding protection to a PDF document;
- Obtaining the number of pages of a PDF document;
- Finding blank pages in a PDF document;
- Spell-Checking a PDF document;
- Adding titles, footnotes, page number, and watermarks to the PDF documents being manipulated;
- Getting the number of pages in a PDF report, searching a special PDF page, finding broken web links, and even spell-checking PDF documents;
- Adding various protections to PDF documents, and even setting the PDF printing restrictions.

Additionally VBScript can be used

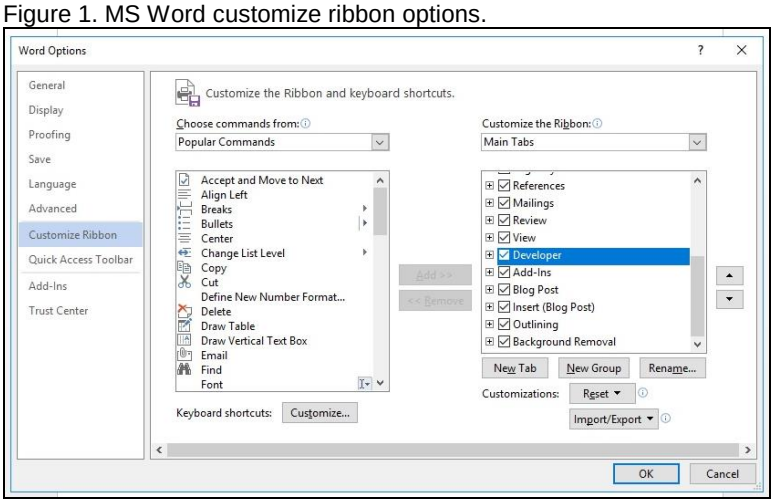
- For automate and control an analysis using JMP on a Windows platform [5];
- For transferring data from SAS into Excel using DDE & ODS with no loss of data integrity (SAS value "1-4" will be exported as Excel value "1 - April", for example) and for creating applications to pull data from SAS to Excel without SAS [6];

Programmers have the ability to automate SAS tasks, traverse the file system, send emails programmatically, manipulate Microsoft Word, Excel, PowerPoint and other files, get web data, and more.
VBScript and SAS are each powerful tools and these two technologies can be combined so that SAS code can call a VBScript program or vice versa [2, 4].

AN EASY WAY TO START WRITING A SIMPLE VBSCRIPTS

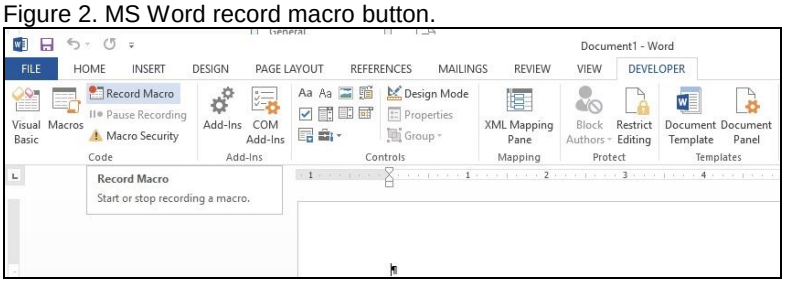
As usual, SAS-programmer does not have sufficient experience in any other programming languages, such as syntax or rules. In this case, it is possible to use the developer tab in your MS Office application. The developer tab is not displayed by default, but can be added to the ribbon as it shown on figure 1.

Figure 1. MS Word customize ribbon options.



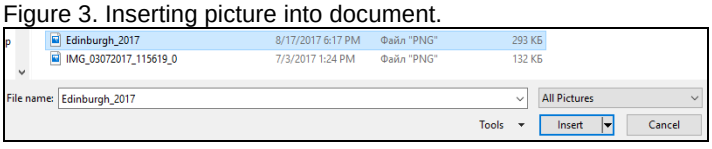
It is possible to write macros using the developer tab, which is the first step in VBScript writing.

Figure 2. MS Word record macro button.



The next step is to make the necessary action with the document (formatting, adding something, etc.). For example, if we need to add the picture in the document and get the VBScript for the adding, after the record macro is started it is sufficient just to add the real picture in the document, then stop the macro writing.

Figure 3. Inserting picture into document.



Then the macro recording should be stopped and it can be edited in MS Visual Basic for application that is available out of the box in MS Office.

Figure 4. List of macros written in the MS Word for picture adding.

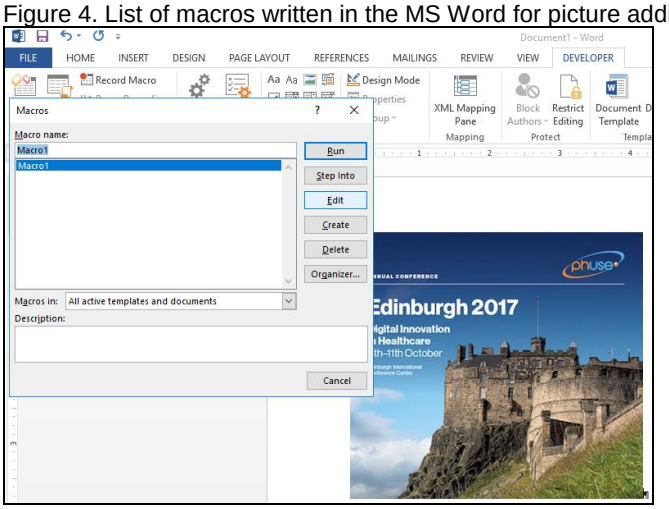
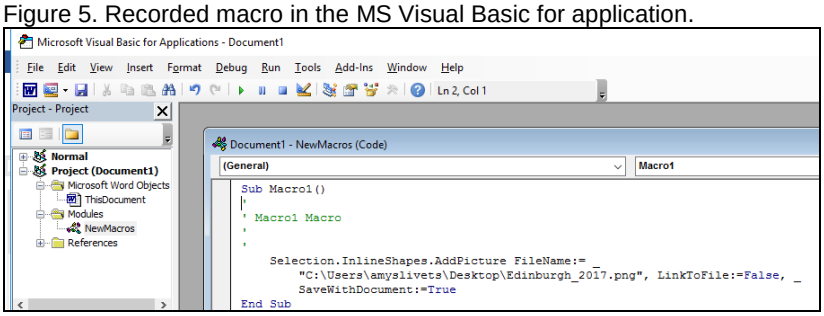


Figure 5. Recorded macro in the MS Visual Basic for application.



The basic operators/scripts and syntax for future usage of VBScripts in SAS can be taken from the algorithm above.

CONCLUSION

Don't despair if you have "unresolved" problems, which SAS cannot overcome, during the creating of statistical reports. The big advantage of SAS is the possibility of its interaction with other scripting languages, in particular its ability to set and run VB commands. And VBScript, in turn, is a great tool for working with Microsoft Office applications. As a result, automate all process programmatically without manual work is possible with a wide range of tasks in general and creating reports is simplified and takes much less time in particular.

REFERENCES

1. Data Matrix Ltd. (2016) Andrey Myslivets, Artem Dolgun; Universal SAS® macro for generating Statistical Analysis Report in one click for I-IV phase Clinical Trials (ISCB 2016, Birmingham, UK) (https://dm-matrix.com/universal-sas-macro-for-generate-statisitcal-analysis-reportt-2016.pdf)

2. Christopher Johnson; Integrating Microsoft® VBScript and SAS® (SESUG 2015, Savannah, Georgia) (http://www.lexjansen.com/sesug/2015/9_Final_PDF.pdf)

3. Lei Zhang; Developing PDF-Manipulation Macros for eSubmission Automation (PharmaSUG 2006, Bonita Springs, Florida) (http://www.lexjansen.com/pharmasug/2006/applicationsdevelopment/ad01.pdf)

4. William E Benjamin Jr; Extend the Power of SAS® to Use Callable VBS and VBA Code Files Stored in External Libraries to Control Excel Formatting Routines (PharmaSUG 2013, Chicago, Illinois) (http://www.lexjansen.com/pharmasug/2013/TF/PharmaSUG-2013-TF10.pdf)

5. Matthew Flynn; Automating JMP via Scripting tools and MS Windows COM (NESUG 2009, Burlington, Vermont) (http://www.lexjansen.com/nesug/nesug09/ap/AP09.pdf)

6. Timothy Adlington; Using VBA and BASE SAS to Get Data from SAS to Excel Without Data Integrity Issues (PhUSE 2005, Heidelberg, Germany)
(http://www.lexjansen.com/phuse/2005/as/as11.pdf)

7. Word VBA reference
(https://msdn.microsoft.com/en-us/vba/word-vba/articles/find-execute-method-word
https://msdn.microsoft.com/en-us/vba/word-vba/articles/document-close-method-word)

CONTACT INFORMATION

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

Ekaterina Torchinskaya
Data MATRIX Ltd
14 Nekrasova Street, Let. A
Saint Petersburg, 191014
+7 (812) 449 86 33 Ext 2194
+7 (812) 449 86 35
ktorchynska@dm-matrix.com
https://dm-matrix.com/

Andrei Myslivets
Data MATRIX Ltd
14 Nekrasova Street, Let. A
Saint Petersburg, 191014
+7 (812) 449 8633 Ext:2108
+7 (926) 812 7088,
amyslivets@dm-matrix.com
https://dm-matrix.com/

Appendix: PAGEBR macro

```
/*****************************************************************/
/* MACRO NAME: PAGEBR                                            */
/* AUTHOR:                                                       */
/* DATE:                                                         */
/* SAS VERSION: 9.3                                              */
/* DESCRIPTION: Generate VBS script to convert section breaks    */
/*              to page breaks                                   */
/* PARAMETERS: infile- the name of the .RTF file which will be   */
/*                  converted                                    */
/* MACRO CALL: %PAGEBR(infile="C:\VBS_SAS\File_to_convert.RTF")  */
/*****************************************************************/

%macro PAGEBR(infile= );

data _null_;
  file "C:\VBS_SAS\replace.vbs";

  put "Dim wrdApp: Set wrdApp = WScript.CreateObject(""Word.Application"")";
  put "Dim wrdDoc";
  put "Dim wdReplaceAll";
  put "Set wrdDoc = wrdApp.Documents.Open("&infile.")";

  put "wdReplaceAll = 2";
  put "wrdDoc.Select";

  put "With wrdApp.Selection.Find";
  put "    .ClearFormatting";
  put "    .Replacement.ClearFormatting";
  put "    .Text = ""^b""";
  put "    .Replacement.Text = "" """;
  put "    .Forward = True";
  put "    .Wrap = wdFindContinue ";
  put "    .Format = False";
  put "    .MatchCase = False";
  put "    .MatchWholeWord = True ";
  put "    .Execute , , , , , , , , , , wdReplaceAll";
  put "End With";

  put "With wrdApp.Selection.Find";
  put "    .ClearFormatting";
  put "    .Replacement.ClearFormatting";
  put "    .Text = ""^p^p""";
  put "    .Replacement.Text = ""^p""";
  put "    .Forward = True";
  put "    .Wrap = wdFindContinue";
  put "    .Format = False";
  put "    .MatchCase = False";
  put "    .MatchWholeWord = False";
  put "    .Execute , , , , , , , , , , wdReplaceAll";
  put "End With";

  put "wrdDoc.SaveAS (""&infile."");
  put "wrdDoc.Close SaveChanges=True";
  put "wrdApp.Close";
  put "wrdApp.Quit";
  put "Set wrdApp = Nothing";
  put "Set wrdDoc = Nothing";
run;

*Running the .VBS script;
data _null_;
  call system(" C:\VBS_SAS\replace.vbs");
run;

%mend PAGEBR;
%PAGEBR(infile="C:\VBS_SAS\File_to_convert.RTF");
```