PharmaSUG 2021 - Paper EP-013

Reduce Review Time: Using VB to Catch Spelling Errors & Blank Pages in TLFs

Corey Evans, Qingshi Zhou, and Ting Su, LLX Solutions, LLC

ABSTRACT

Checking blank pages and spelling errors is usually time-consuming and monotonous. We developed a macro using VB embedded in SAS that will assist with checking for unwanted blank pages generated from SAS ODS procedures. With one click, the macro will run and return an excel file with the desired results populated for all the outputs under one folder. Also, the utility can help to check spelling errors and populate unique errors/counts in the excel file.

INTRODUCTION

Manually verifying that there are no blank pages or spelling issues across a large-scale project is an incredibly time-consuming task for all parties involved. In a fast-paced work environment, any tool that can significantly reduce the time spent on a task is highly valuable. This macro was developed using Visual Basic with that goal in mind.

Standard review of outputs requires the individual to scroll through each page to ensure that there are no blank pages, page break issues or obvious spelling errors. In studies with even a moderate number of subjects enrolled, there are listings that can easily exceed 1,000 pages. Manually reviewing an output that long is cumbersome and is prone to human error.

Automated review of outputs to ensure there are no blank pages, page break issues or spelling errors allows the reviewer more time to focus on things of greater importance, like cross-checking values between outputs. Additionally, this automated review can be implemented at any step in the output creation process; production programmers can use it to ensure there are no issues early on, as well as validation programmers. Reduction of time spent on these “low-value” tasks allows everyone involved in the process to spend more time on the “high-value” project tasks, which allows for a high-quality product to be delivered using the same (or less) resources.

OUTPUT REVIEW

For programmers as well as statisticians, output review is a critical step to ensuring that the produced outputs are high-quality and meet internal as well as any external standards. Output review can be broken out into several categories; checking for consistency between outputs, ensuring the output matches the shells, and ensuring that the overall formatting/spelling in the output is correct. While the first two categories can be considered as high-value, low-time tasks, the last category has less inherent value in comparison, and requires a large amount of time to accurately check for those type of issues.

In the programming world, development of macros that serve to automate certain aspects of a job can provide incredible value. It allows a programmer to spend more time on tasks that require focus and critical thinking, allowing the simple, repeatable tasks to be handled by the macro(s). The process of searching an output for any blank pages, page break issues or spelling errors is a simple, repeatable task, making it an extremely suitable candidate for automation. Using Visual Basic, a macro was developed to automate this task.

The following code checks all outputs within its respective folder for spelling errors, blank pages, and page breaks:

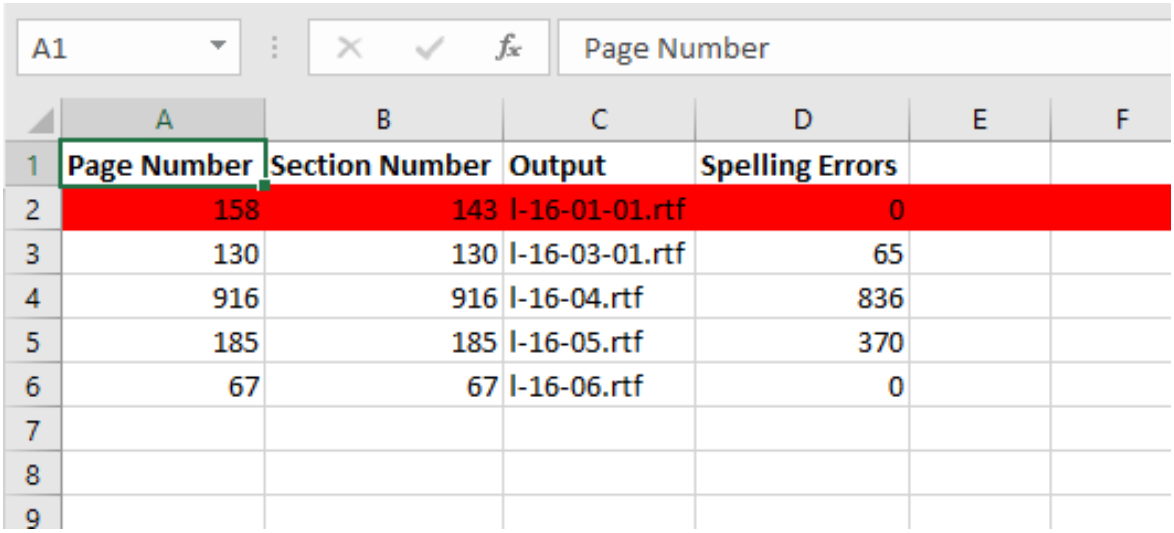
```
Set objExcel = CreateObject("Excel.Application")
objExcel.Visible = True
objExcel.DisplayAlerts=False
Set wb = objExcel.Workbooks.Add
wb.Activesheet.range("A1").Value="Page Number"
```

```
wb.Activesheet.range("B1").Value="Section Number"
wb.Activesheet.range("C1").Value="Output"
wb.Activesheet.range("D1").Value="Spelling Errors"
wb.Activesheet.range("A1:D1").Font.Bold = True
n = 2
Set obj = CreateObject("Scripting.FileSystemObject")
Set myPath = obj.GetFolder(".")
For Each file In myPath.Files
If Right(file, 3) = "rtf" then
Set myDoc = GetObject(file.path)

wb.Activesheet.cells(n,1).Value=myDoc.Windows(1).Panes(1).Pages.Count
wb.Activesheet.cells(n,2).Value=myDoc.Sections.Count
wb.Activesheet.cells(n,3).Value=myDoc.name
wb.Activesheet.cells(n,4).Value=myDoc.SpellingErrors.Count
if wb.Activesheet.cells(n,1) <> wb.Activesheet.cells(n,2) then
wb.Activesheet.cells(n,1).entirerow.interior.ColorIndex=3
'For Each range in myDoc.SpellingErrors
'wb.Activesheet.cells(n,x).Value=range.Text & vbNewLine
'x = x + 1
'Next
n = n + 1
myDoc.close
End If
Next
wb.Activesheet.Columns("A:D").Autofit() 'Auto fit columns width
```

Execution of this macro opens an Excel spreadsheet detailing any issues that are found within an output. This excel sheet allows you to quickly scan and see what outputs are highlighted as having issues, and which outputs have passed through the macro with no outstanding issues.

Figure 1 is an Excel spreadsheet of the output from running the macro.



	A	B	C	D	E	F
1	Page Number	Section Number	Output	Spelling Errors		
2	158	143	l-16-01-01.rtf	0		
3	130	130	l-16-03-01.rtf	65		
4	916	916	l-16-04.rtf	836		
5	185	185	l-16-05.rtf	370		
6	67	67	l-16-06.rtf	0		
7						
8						
9						

Figure 1. Excel Output

As shown in Figure 1, rows highlighted in red have been flagged for page break/blank page issues. Page Number indicates the total number of pages in each output. Section Number indicates the number of pages you have specified in your SAS data set when you generated the output using Proc Report. If Page Number and Section Number are not equal, then the output must be checked manually, as it is highly

likely there are page break issues or blank pages. Output simply refers to the output the Visual Basic code is being run on. Spelling Errors indicates the number of spelling errors the macro has identified in the output.

PRACTICAL APPLICATION & LIMITATIONS

It is all too common for programmers to be under tight deadlines to complete a large set of outputs. While quality is of the utmost importance, short timelines can sometimes compromise the review time that is allotted for a project. A practical approach to utilizing this macro would be to run it once outputs have been validated, but before statistical review has begun, which would cut down on review time in multiple ways. Both the production and validation programmers do not have to scroll the outputs to check for page break/blank page issues. The statistical reviewer does not have to worry about it either during review, as there is a spreadsheet that indicates that there are no page break/blank page issues present in the outputs. For any spelling errors that are present, the production programmer can quickly identify and assess the issues and address them as needed.

While this tool is useful in the tasks it is designed to streamline, there are limitations to it. The macro has the ability to check for spelling errors, but clinical trial data sets tend to have words that are not dictionary-defined. This can lead to the tool showing spelling errors, when in fact there are none. As a result, end users of this tool must take the result in the Spelling Errors column with a grain of salt. There is no perfect system when identifying spelling errors, so some manual review in regard to that is necessary, despite what the tool may show you.

Another major limitation with this macro is that it requires that the programmer utilize a page breaking variable within their SAS program when producing the output. Without a page break variable, the macro will be unable to compare the number of sections (that were specified in the SAS dataset) with the number of pages in the document. To ensure this macro runs perfectly, programmers should always specify a page break variable in their programs.

Lastly, if there are too many outputs within the specified folder, and some of the outputs are very large, the macro may end up with a runtime error. The simplest way around this is to break the number of outputs within a folder into smaller groups, which will allow the macro to run more efficiently. You may find with very large outputs that you will need to run the macro just on that one output in order for it to function. With large outputs, the macro may take a long time to run, depending on the processing power of the device you are using.

Overall, use of this tool should simplify the review process significantly. You may find that implementation of this macro eases the overall burden of statistical review, something every statistician that you work with will likely appreciate. While there are limitations with the macro, we believe that the benefits of it far outweigh any of the limitations it may have.

WHY VISUAL BASIC

Visual Basic is a programming language developed by Microsoft. While it is now considered “legacy”, it still has many practical uses even with all the modern tools at our disposal. The biggest consideration in creating this macro and utilizing Visual Basic is that RTF files are native to Microsoft Word, making it easy to utilize Visual Basic to parse out the information of interest. Using any other programming language complicates the task, as the programming language is not native to the object of interest (the output being reviewed), which can make the task of parsing out what is needed challenging. Try reading an RTF file into SAS, it is a challenge to make heads or tails of the resulting dataset. Taking the path of least resistance leads to Visual Basic for a task such as this. 30 lines of code produces a clean and clear Excel sheet that tells you everything you need to know in regard to blank pages, page breaks and spelling errors in an output, or a series of outputs.

There are likely fancier ways than using Visual Basic to automate this task. But quick and clean get the job done, and that is what is important here.

CONCLUSION

Output review is one of the simpler but tedious steps that is required in projects. The consideration of whether to automate a task should be based both on the complexity of the task as well as the standard time spent on the task. The ability to give time back to programmers or statisticians that they would normally spend on this portion of review cannot be overstated. The result of utilizing this macro should be higher quality outputs, with an overall decrease in time spent working on them, a positive outcome for everyone involved. Let some work be completed by automated tools, you will thank yourself for it later.

ACKNOWLEDGMENTS

Jin Shi for her support on this paper as well as her technical knowledge.

CONTACT INFORMATION

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

Corey Evans
Corey.Evans@llxsolutions.com

Any brand and product names are trademarks of their respective companies.