PharmaSUG 2017 - Paper BB20

Automating Title and Footnote Extraction Using Visual Basic for Applications (VBA) and SAS™.

Tony Cardozo, Spaulding Clinical

ABSTRACT

When developing tables, figures and listings (TFLs) for clinical trial data we must ensure we are providing outputs that are consistent with the Statistical Analysis Plan (SAP) shells. Titles and footnotes, among other key elements, need a high level of accuracy to minimize review cycles with the statistician and overall development time. Depending on your current process, this may involve copying and pasting every title and footnote into a centralized macro or individual programs. This may be done by multiple programmers and can be a time-consuming process that leaves opportunities for human error. When the TFL shells are created in Microsoft Word, title and footnote extraction can be done programmatically utilizing Visual Basic for Applications (VBA) and SAS™. In this paper, we'll discuss what VBA is, how to set it up and program title and footnote extraction. Also, we'll cover how to process the text streams to catch special characters (≤, ≥, μ, etc) and provide SAS™ Output Delivery System (ODS) friendly equivalents. Additionally, we will go over some assumptions made about the shells and how post processing with SAS™ can make the whole title/footnote process from shells to TFLs dynamic and automated.

INTRODUCTION

Processes that help reduce development time while increasing accuracy are essential to deliver high quality deliverables on time. The process of extracting titles and footnotes from a TFL shell document can be time-consuming and error prone. In this paper, we'll discuss the utilization of VBA and SAS™ to significantly decrease the time needed for title/footnote extraction while increase accuracy. This process will allow for conversion of sub/super scripts and special characters to ODS equivalent values. The VBA UTF-8 friendly text output will be further processed to structure the values into a format that will flow into previously established processes for displaying the titles and footnotes in the corresponding output.

SHELL ASSUMPTIONS

The process outlined in this paper requires a certain structure in the TFL shells. Although the program can be modified to adapt to different shell structures, a certain level of agreement and consistency is necessary in TFL shell development to deliver a consistent extraction output.

GENERAL ASSUMPTIONS

General TFL shell assumptions include:

- TFL shells are developed in MS Word or other software that supports VBA.
- Key words (Table, Listing, Figure) are used in output titles.
- Each new output is in a new section break.
- Title and Footnotes are placed in the Header and Footer sections of the Word document.

Each output shell can have as many titles and footnotes as needed. Superscripts and subscripts are supported and converted dynamically. However, special characters will require additional lines of code as they are used.

VISUAL BASIC FOR APPLICATIONS (VBA)

Visual Basic for Applications is a programming language that is embedded in individual Microsoft applications such as Word and Excel. Its syntax is very similar to Visual Basic (VB), but unlike VB, VBA cannot compile standalone executable programs. VBA is an object based language, meaning different parts of a document are consider their own object. Each object has properties, methods and attributes that can be manipulated with dot notation. Microsoft applications have a built-in VBA editor that can be opened by pressing Alt+F11 in an open application instance. A walkthrough of the editor environment is not within the scope of this paper but can be easily found online.

VBA DOT NOTATION

VBA uses dot notation to separate the various things that can be accessed and manipulated with the programming language. Dot notation is hierarchical, and usually starts with an object followed by a dot. After the dot, it specifies what can be done with the object, or what can be manipulated. The available actions are called methods. The object can also be manipulated via properties or parameters.

MSDN LIBRARY

Starting out, the vast number of different methods and parameters available can be a little overwhelming. Luckily, the Microsoft Development Network (MSDN) has great reference tools and documentation. The VBA MSDN Library (https://msdn.microsoft.com/en-us/library/office/gg264383.aspx) has examples, conceptual overviews, method definitions, how-to topics and language reference that help during the developmental process.

PROGRAMMING TITLE FOOTNOTE EXTRACTION

Once TFL shells are provided that meet the basic assumptions mentioned above we are ready to start the extraction process. The extraction process needs to perform a couple different tasks before a final text file can be output. Since the title and footnotes were placed into the header/footer sections of the document, they are not directly available for some VBA methods. To further complicate it, since each header/footer for a new table is within a new document section so we cannot use some global methods either.

The extraction must perform the following task:

- Using a find and replace process, dynamically convert all superscripts and subscripts to ODS commands.
- Determine the number of pages and sections to extract all text in the header/footer sections.
- Convert special characters to the ODS equivalents.
- Output UTF-8 encoded text file.

Please note, the process as outlined in this paper is destructive to the shell document opened. It should be done on a secondary copy or make sure not to save changes to the document after processing is complete. There are ways around this that will be discussed briefly in the Future Development section but those options are outside the scope of this paper.

SETTING UP THE EXTRACTION PROCESS

Before we can start manipulating the different parts of the document, some variables will need to be initialized:

```
Dim oDoc As Word.Document
Dim oSec As Word.Section       1
Dim oPageStart As Word.Range
Dim story As Word.Range           2
Dim iPage As Integer, iTotalPages As Integer, iSection As Integer
Dim sHeader As String, sFooter As String, sFileName As String    3
Dim fso As Object
Set fso = CreateObject("Scripting.FileSystemObject")
Dim oFile As Object                                       4
Set oFile = fso.CreateTextFile("c:\YOUR_PATH/FILENAME.txt", True, True)

Application.ScreenUpdating = False  5

Set oDoc = ActiveDocument  6
```

1. Create a Word Document object so the different parts of the document can be manipulated.
2. Create a Section object so each document section can be accessed.
3. Create a Range object. A Range identifies a specific portion of a document within a section.
4. Create a Scripting.FileSystemObject object to enable text file output. Define output path and set output encoding parameters to True so a UTF-8 file is outputted.
5. Application.ScreenUpdating allows all the process to be done without wasting time updating the screen. This allows for faster processing.
6. Set the oDoc object created in step 1 to the currently opened document.

CONVERTING SUPERSCRIPTS AND SUBSCRIPTS

Unlike special characters that have their own ASCii value, superscripts and subscripts are a format that does not alter the characters ASCii or hexadecimal value. Since the formatting is lost when the text is extracted, a simple replace function will not work. A find and replace function would have to be ran that is looking for only superscripts/subscripts and convert the text to what is needed. To use find and replace (like hitting Ctrl+H) each layer, or story in MS Word terms, must be checked.

This can be done by looping through each story and checking the ones that relate to header and footers:

```
For Each story In oDoc.StoryRanges  1
        Do
            If story.StoryType = wdPrimaryHeaderStory Or _
               story.StoryType = wdPrimaryFooterStory Or _  2
               story.StoryType = wdFirstPageHeaderStory Or _
               story.StoryType = wdFirstPageFooterStory Then

               ReplaceScripts story  3
            End If

            Set story = story.NextStoryRange  4
        Loop Until (story Is Nothing)
    Next;
```

1. Define a loop and loop through each StoryRange object in the active document.
2. Check if the current StoryRange object is one that relates to the headers or footers.
3. Run the ReplaceScripts sub-routine for the current story to replace any super/sub scripts.
4. Move to the next StoryRange object in the collection. Repeat process until all StoryRanges were checked.

Step 3 mentions the ReplaceScripts sub-routine. A sub-routine is a user created routine that is called from the main routine. It can be thought of as a SAS™ Macro. Like macros, a sub-routine can have parameters or not and be called from anywhere in the program. They are useful when the same task needs to be performed multiple times with different attributes.

ReplaceScripts and SubScripts sub-routines:

```
Sub ReplaceScripts(r As Range)  1
    'Replace Subscripts
    SubScripts r, False, True, "", "``{sub ^&}"  2

    'Replace Superscripts
    SubScripts r, True, False, "", "``{super ^&}"  3
End Sub

Sub SubScripts(r As Range, bSuper As Boolean, bSub As Boolean, stxt As
String, rtxt As String)  4
    r.Select
    selection.Find.ClearFormatting
    With selection.Find.Font
        .StrikeThrough = False  5
        .DoubleStrikeThrough = False
        .Hidden = False
        .SmallCaps = False
        .AllCaps = False
        .Superscript = bSuper
        .Subscript = bSub  6
    End With
    selection.Find.Replacement.ClearFormatting
    With selection.Find
        .Text = stxt
        .Replacement.Text = rtxt  7
        .Forward = True
        .Wrap = wdFindContinue  8
        .Format = True
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
    selection.Find.Execute Replace:=wdReplaceAll  9

End Sub
```

1. Define the ReplaceScripts sub-routine as expecting a Range object (**blue text**) to be passed in.
2. Call the SubScripts sub-routine using the Range object passed in and defining bSuper (**brown text**) as False and bSub (**green text**) as True. This will replace all the sub scripts in the document, since stxt (**purple text**) is null, with rtxt (**red text**) ``{sub ^&} where ^& represents a wildcard for the text found. This allows find and replace to be dynamic since the actual subscripted text does not need to be known.
3. Same as step 2 except defined for search for the superscripts, bSuper as True and bSub as False.
4. Define SubScripts sub-routine with range, Boolean and text parameters. Select the current range, like highlighting in Word, so it can be manipulated.
5. Set the attributes of selection.Find.Font so that it only searches for sub or super scripts.
6. Use bSuper and bSub defined above to toggle the correct search criteria.

7. With the actual find function, define the search text (.Text) and replacement text (.Replacement.Text) using stxt and rtxt defined in sub-routine call.
8. Set .Forward = True so the find replace function works as "replace all." If set to false, only the first instance of a sub/super script would be replaced.
9. Replace the text in the range selection with the text defined in the Replacement.Text attribute.

At this point, we have all the sub and scripts replaced with parsing anchor '``' and OBS sub/super script command.

LOOPING THROUGH EACH PAGE AND HEADER SECTION

The next step is to loop through each page and section to extract the actual text. This can be done by defining a couple variables to determine the upper and lower bound of the loops.

The following code defines these values, loops through the pages, extracts the text, converts special characters and outputs the text file:

```
iTotalPages = oDoc.ComputeStatistics(wdStatisticPages) 1

  With oDoc  2

      iSection = 0 3

      For iPage = 1 To iTotalPages  4

          Set oPageStart = oDoc.GoTo(what:=wdGoToPage, _
      5                          Which:=wdGoToAbsolute, Count:=iPage)
          Set oSec = oPageStart.Sections(1)

          If (iSection < oSec.Index) And _
             (oSec.PageSetup.DifferentFirstPageHeaderFooter) Then
              sHeader = oSec.HeaderS(wdHeaderFooterFirstPage).Range.Text
     6        sFooter = oSec.Footers(wdHeaderFooterFirstPage).Range.Text
          Else
              sHeader = oSec.HeaderS(wdHeaderFooterPrimary).Range.Text
              sFooter = oSec.Footers(wdHeaderFooterPrimary).Range.Text
          End If

          iSection = oSec.Index

          sHeader = Replace(sHeader, Chr(7), "") 7
          sFooter = Replace(sFooter, Chr(7), "")

          sFooter = Replace(sFooter, ChrW(8805), "^R'\ {\uc2\u8805 >=}'")
      8   sFooter = Replace(sFooter, ChrW(8804), "^R'\ {\uc2\u8804 <=}'")

          sHeader = Replace(sHeader, "``{", "^{")  9
          sFooter = Replace(sFooter, "``{", "^{")

          oFile.WriteLine "Page~" & iPage & "~Section~" & iSection
     10   WriteHeadFootLines sHeader, oFile, "Header~"
          WriteHeadFootLines sFooter, oFile, "Footer~"
 11   Next
      oFile.Close
      Set fso = Nothing  12
      Set oFile = Nothing
  End With
```

1. Create iTotalPages using the ComputeStatistics method to determine the total number of pages in the document.
2. Define that the succeeding processes are to be performed on the oDoc object.
3. Define iSection to track the sections that have been processed.
4. Prepare a for loop to through all the pages from 1 to the value of iTotalPages.
5. Go to the current page getting processed using the GoTo method and defining the page as Count:=iPage. Once on the correct page, select it's content for processing.
6. Determine if the current section is the first section within the page range. Since Word treats the first page within a section differently, the page sections need to be determined to use the correct attributes for text extraction. If it is the first page, use (wdHeaderFooterFirstPage).Range.Text to extract the text from the oSec.Header and oSec.Footer objects. Otherwise use (wdHeaderFooterPrimary).Range.Text.

CHECKING FOR SPECIAL CHARACTERS

Now that the text is extracted for the current section some post extraction processing can be performed.

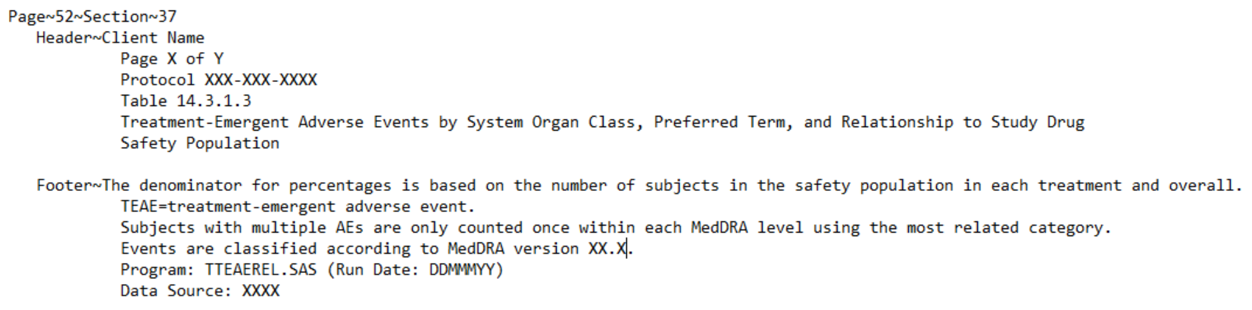
7. Replace carriage returns that will not import correctly to SAS™ with a blank.
8. Using the ASCii value, replace special characters with the SAS™ ODS equivalent. The ChrW method takes the ASCii value of the character being replace.
9. Replace any other text that needs replacing. For example, the '^' was the desired ODS escape character but it could not be directly set in the SubScripts sub-routine because '^' is a VBA operator in the Find/Replace method. The SubScripts sets '``{' which can now be replaced to the desired '^{' value without undesirable effects.

OUTPUTTING UTF-8 ENCODED TEXT FILE

10. Using the Scripting.FileSystemObject object oFile, the contents of sHeader and sFooter for the current section/page are written to the text file defined during object creation. Adding the Page~, ~Section~, Header~ and Footer~ text will allow for easier post extraction processing.
11. Continue to the next page until all pages are processed.
12. Close the text file and clear it from memory.

POST PROCESSING WITH SAS™

At this point, there is a text file that contains all the headers and footnotes for a TFL Shell document. The text file will look like the screen shot in Display 1.



```
Page~52~Section~37
   Header~Client Name
            Page X of Y
            Protocol XXX-XXX-XXXX
            Table 14.3.1.3
            Treatment-Emergent Adverse Events by System Organ Class, Preferred Term, and Relationship to Study Drug
            Safety Population

   Footer~The denominator for percentages is based on the number of subjects in the safety population in each treatment and overall.
            TEAE=treatment-emergent adverse event.
            Subjects with multiple AEs are only counted once within each MedDRA level using the most related category.
            Events are classified according to MedDRA version XX.X
            Program: TTEAEREL.SAS (Run Date: DDMMMYY)
            Data Source: XXXX
```

Display 1: Example of outputted text file from extraction process.

Depending how titles and footnotes are handled, additional processing may be used to re-structure the output as needed. The process outlined below re-structures the text in preparation to be read into a macro called in the TFL programs directly. Although processes for handling titles and footnotes may differ between companies, the same overall logic outlined below may apply in part or in its entirety.

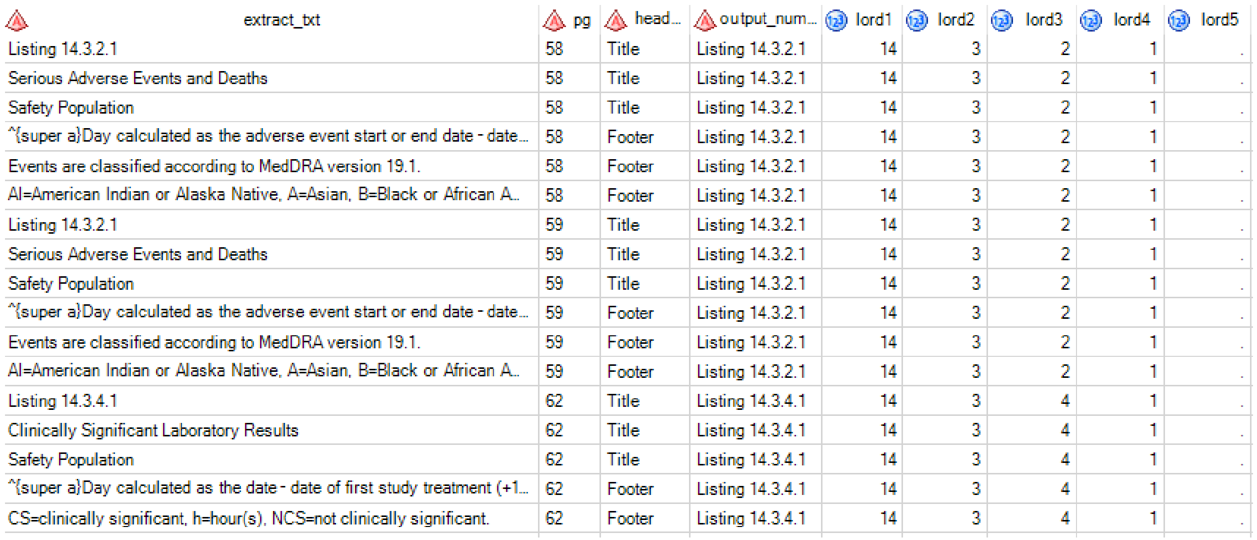
The following code segment was used to pares the extraction text file as needed for internal application:

```
filename shellex "&dir_path.\&import_filename" lrecl=39562 ;

data Shell_extract;    1
   length extract_txt $5000;
   infile shellex dlm='|' truncover;
   input extract_txt $;
   list;

   *Compress formating characters like tabs, section breaks, etc;
   extract_txt=strip(compress(extract_txt, ,'H'));    2
   if extract_txt ne '';
run;

data Shell_extract2(where=(cnt >0 and program_name = '' and extract_txt ne ''))
     prg(where=(program_name ne '' and output_number ne '') keep=output_number
program_name);
   set Shell_extract;  3
   retain pg sec header_footer_flg output_number cnt;    4

   length pg sec header_footer_flg output_number program_name _temp
output_number_char $100;

   if _n_ = 1 then do; pg=''; sec=''; header_footer_flg=''; output_number='';
cnt=0; end;

   if index(extract_txt,'Program:') then
program_name=strip(scan(scan(extract_txt,2,':'),1,'.'));
   if index(extract_txt,'Source: ') then delete;
   if index(extract_txt,'Page~') then
      do;
         header_footer_flg='';
         pg=strip(scan(extract_txt,2,'~'));    5
         output_number='';
         cnt=0;
      end;
   if index(extract_txt,'Section~') then sec=strip(scan(extract_txt,4,'~'));
   if index(extract_txt,'Header~') and index(extract_txt,'Page')=0  then
      do;
         cnt=0;
         header_footer_flg='Title';  6
      end;
   if index(extract_txt,'Footer~') and index(extract_txt,'Page')=0  then
      do;
         cnt=0;
         header_footer_flg='Footer';
         extract_txt=substr(extract_txt,8);  7
      end;

   if strip(compress(extract_txt,'1234567890. "'))
      in('Table' 'Figure' 'Appendix'  'Listing') then    8
         output_number=strip(extract_txt);

   if output_number ne '' then cnt+1;

   if output_number ne '' then
      do;                        9
         _temp=strip(compress(scan(output_number,2,''),'.'));
         output_number_char=strip(scan(output_number,1,''));
```

```
            nlen=length(_temp);

            num_levels = length(strip(scan(output_number,2,'')))-
                         length(strip(compress(scan(output_number,2,''),'.')));

            array sort_levels[*] lord1-lord6;

            do __i=1 to (num_levels+1);
              sort_levels[__i]=
                    input(strip(scan(strip(scan(output_number,2,'')),__i,'.')),best.);
            end;

            output_number_num=input(_temp,best.);
         end;
   run;
```

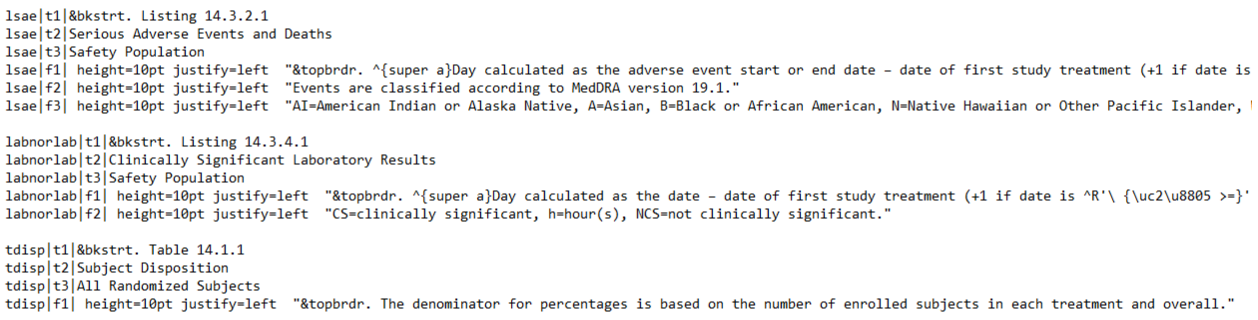
1. Import file into SAS™.
2. Remove any formatting characters that were extracted but not needed.
3. Process the imported data and create 2 output data sets. One contains all the title and footnote information, the other has all the program names and associated table numbers for later use.
4. Create and initialize retain variables so each section can be processed and grouped as needed per output per header and footnote.
5. Use the page anchor text (Page~) to determine when a new page begins. Set Page number variable accordingly and re-set other retained variables to 0 or null.
6. Use the header anchor text (Header~) to determine the start of a header section and update header_footer_flg accordingly.
7. Use the footer anchor text (Footer~) to determine the start of a Footer section and update header_footer_flg accordingly.
8. Determine if the line being processed contains the output number.
9. For accurate sorting, break down the character table number to its individual numeric parts. For example, 'Table 14.10.2.3' would break down to lord1=14, lord2=10, lord3=2 and lord4=3.

The above code will produce a data set that looks like below.



extract_txt	pg	head...	output_num...	lord1	lord2	lord3	lord4	lord5
Listing 14.3.2.1	58	Title	Listing 14.3.2.1	14	3	2	1	.
Serious Adverse Events and Deaths	58	Title	Listing 14.3.2.1	14	3	2	1	.
Safety Population	58	Title	Listing 14.3.2.1	14	3	2	1	.
^{super a}Day calculated as the adverse event start or end date - date...	58	Footer	Listing 14.3.2.1	14	3	2	1	.
Events are classified according to MedDRA version 19.1.	58	Footer	Listing 14.3.2.1	14	3	2	1	.
AI=American Indian or Alaska Native, A=Asian, B=Black or African A...	58	Footer	Listing 14.3.2.1	14	3	2	1	.
Listing 14.3.2.1	59	Title	Listing 14.3.2.1	14	3	2	1	.
Serious Adverse Events and Deaths	59	Title	Listing 14.3.2.1	14	3	2	1	.
Safety Population	59	Title	Listing 14.3.2.1	14	3	2	1	.
^{super a}Day calculated as the adverse event start or end date - date...	59	Footer	Listing 14.3.2.1	14	3	2	1	.
Events are classified according to MedDRA version 19.1.	59	Footer	Listing 14.3.2.1	14	3	2	1	.
AI=American Indian or Alaska Native, A=Asian, B=Black or African A...	59	Footer	Listing 14.3.2.1	14	3	2	1	.
Listing 14.3.4.1	62	Title	Listing 14.3.4.1	14	3	4	1	.
Clinically Significant Laboratory Results	62	Title	Listing 14.3.4.1	14	3	4	1	.
Safety Population	62	Title	Listing 14.3.4.1	14	3	4	1	.
^{super a}Day calculated as the date - date of first study treatment (+1...	62	Footer	Listing 14.3.4.1	14	3	4	1	.
CS=clinically significant, h=hour(s), NCS=not clinically significant.	62	Footer	Listing 14.3.4.1	14	3	4	1	.

Now we can drop repeat records by output, merge on program names, add footnote records for the outputs with 0 footnotes or any other additional processing needed for desired output.



```
lsae|t1|&bkstrt. Listing 14.3.2.1
lsae|t2|Serious Adverse Events and Deaths
lsae|t3|Safety Population
lsae|f1| height=10pt justify=left  "&topbrdr. ^{super a}Day calculated as the adverse event start or end date - date of first study treatment (+1 if date is
lsae|f2| height=10pt justify=left  "Events are classified according to MedDRA version 19.1."
lsae|f3| height=10pt justify=left  "AI=American Indian or Alaska Native, A=Asian, B=Black or African American, N=Native Hawaiian or Other Pacific Islander,

labnorlab|t1|&bkstrt. Listing 14.3.4.1
labnorlab|t2|Clinically Significant Laboratory Results
labnorlab|t3|Safety Population
labnorlab|f1| height=10pt justify=left  "&topbrdr. ^{super a}Day calculated as the date - date of first study treatment (+1 if date is ^R'\ {\uc2\u8805 >=}'
labnorlab|f2| height=10pt justify=left  "CS=clinically significant, h=hour(s), NCS=not clinically significant."

tdisp|t1|&bkstrt. Table 14.1.1
tdisp|t2|Subject Disposition
tdisp|t3|All Randomized Subjects
tdisp|f1| height=10pt justify=left  "&topbrdr. The denominator for percentages is based on the number of enrolled subjects in each treatment and overall."
```

Display 2: Example of fully processed headers and footers ready for TFL production.

FURTHER DEVELOPMENT

Right now, the VBA code requires Word to have the Shell document actively open. This makes for the possibility of the changes made during the find and replace process to be accidently saved. In future versions, the VBA code could have a Graphic User Interface (GUI) added where a file is selected, oDoc document object is created, assigned, processed and closed without saving any changes.

The VBA code could also be transferred in a standalone VB application with its own executable file. This would allow for an .exe file to be ran independent of MS Word.

Java could also be leveraged to run the script from the command prompt allowing for future version to be ran completely from SAS™.

CONCLUSION

The process of extracting header and footnotes from TFL shell documents can be a time-consuming process that leaves room for human error. The process outlined in this paper can handle dynamic mapping of superscripts, subscripts and special characters to ODS equivalent values. In future versions, a GUI interface may be added or it can be made into a standalone application using VB or Java. Utilizing a shell document that follows certain criteria, VBA and SAS™ a process can be developed that automates the extraction process while eliminating copy/paste and human errors thus reducing TFL production/review time.

CONTACT INFORMATION

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

Tony Cardozo
Spaulding Clinical, West Bend, Wisconsin
Antonio.Cardozo@spauldingclinical.com

APPENDIX 1: VBA AUTOTITLFOOT() CODE:

```
Sub WriteHeadFootLines(sHeader As String, oFile As Object, sPrefix As
String)

    Dim arrHeader() As String
    Dim i As Integer

    arrHeader() = Split(sHeader, Chr(13))

    For i = LBound(arrHeader) To UBound(arrHeader)
        If i = LBound(arrHeader) Then
            oFile.WriteLine "   " & sPrefix & arrHeader(i)
        Else
            oFile.WriteLine "            " & arrHeader(i)
        End If
    Next i

End Sub

Sub ReplaceScripts(r As Range)
    'Replace Supscripts
    SubScripts r, False, True, "", "``{sub ^&}"

    'Replace Superscripts
    SubScripts r, True, False, "", "``{super ^&}"
End Sub

Sub SubScripts(r As Range, bSuper As Boolean, bSub As Boolean, stxt As
String, rtxt As String)
    r.Select
    selection.Find.ClearFormatting
    With selection.Find.Font
        .StrikeThrough = False
        .DoubleStrikeThrough = False
        .Hidden = False
        .SmallCaps = False
        .AllCaps = False
        .Superscript = bSuper
        .Subscript = bSub
    End With
    selection.Find.Replacement.ClearFormatting
    With selection.Find
        .Text = stxt
        .Replacement.Text = rtxt
        .Forward = True
        .Wrap = wdFindContinue
        .Format = True
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
    selection.Find.Execute Replace:=wdReplaceAll
End Sub
```

```
Sub AutoTitlFoot()
    Dim oDoc As Word.Document
    Dim oSec As Word.Section
    Dim oPageStart As Word.Range
    Dim story As Word.Range
    Dim iPage As Integer, iTotalPages As Integer, iSection As Integer
    Dim sHeader As String, sFooter As String, sFileName As String

    Dim fso As Object
    Set fso = CreateObject("Scripting.FileSystemObject")
    Dim oFile As Object
    Set oFile = fso.CreateTextFile("C:\PATH\FILENAME.txt", True, True)

    Application.ScreenUpdating = False

    Set oDoc = ActiveDocument

    For Each story In oDoc.StoryRanges
        Do
            If story.StoryType = wdPrimaryHeaderStory Or _
               story.StoryType = wdPrimaryFooterStory Or _
               story.StoryType = wdFirstPageHeaderStory Or _
               story.StoryType = wdFirstPageFooterStory Then
               Debug.Print "In story loop= "; story.StoryType
                ReplaceScripts story
            End If

            Set story = story.NextStoryRange
        Loop Until (story Is Nothing)
    Next

    iTotalPages = oDoc.ComputeStatistics(wdStatisticPages)

    With oDoc

        iSection = 0

        For iPage = 1 To iTotalPages
            Set oPageStart = oDoc.GoTo(what:=wdGoToPage, _
                                       Which:=wdGoToAbsolute, Count:=iPage)
            Set oSec = oPageStart.Sections(1)

            If (iSection < oSec.Index) And _
               (oSec.PageSetup.DifferentFirstPageHeaderFooter) Then
                sHeader = oSec.HeaderS(wdHeaderFooterFirstPage).Range.Text
                sFooter = oSec.Footers(wdHeaderFooterFirstPage).Range.Text
            Else
                sHeader = oSec.HeaderS(wdHeaderFooterPrimary).Range.Text
                sFooter = oSec.Footers(wdHeaderFooterPrimary).Range.Text
            End If

            iSection = oSec.Index

            sHeader = Replace(sHeader, Chr(7), "")
            sFooter = Replace(sFooter, Chr(7), "")

            sFooter = Replace(sFooter, ChrW(8805), "^R'\ {\uc2\u8805 >=}'")
```

```
                sFooter = Replace(sFooter, ChrW(8804), "^R'\ {\uc2\u8804 <=}'")

                sHeader = Replace(sHeader, "``{", "^{")
                sFooter = Replace(sFooter, "``{", "^{")

                oFile.WriteLine "Page~" & iPage & "~Section~" & iSection
                WriteHeadFootLines sHeader, oFile, "Header~"
                WriteHeadFootLines sFooter, oFile, "Footer~"
            Next

            oFile.Close
            Set fso = Nothing
            Set oFile = Nothing
        End With

        Application.ScreenUpdating = True
    End Sub
```