PharmaSUG 2020 - Paper AD-106

Macro to Produce SAS®-Readable Table of Content from TLF Shells

Igor Goldfarb, Accenture ALSS;
Ella Zelichonok, Naxion

ABSTRACT

The goal of this work is to develop a macro that automates a process of reading the shells for tables, listings and figures (TLF) and transforming them into SAS®-readable ordered table of content (TOC). The proposed tool can significantly save time for biostatisticians and lead programmers who have numerously create, revise, update shells documents for TLF.
Development of the shells for TLF for any clinical trial is a time-consuming task. Copying titles and footnotes from the shell document (typically MS Word file) into SAS® program or external source of titles and footnotes (e.g., Excel file) is a tedious process requiring scrupulous work subject to human errors.
The proposed macro (developed in Excel VBA) automates this process. It reads the shell document (Word) and creates/updates SAS®-readable ordered TOC (Excel) in a matter of seconds. The macro identifies common part of the titles and footnotes for all TLF as well as detects differences for specific outputs. The developed tool analyzes all the requests for repeating outputs, updates their sequential numbers and titles and adds them to the TOC. It also allows to change population if required for repeating tables/figures. Finally, the macro generates an Excel file containing ordered TOC that is immediately ready to be used for the final run of SAS® programs to output planned TLF. Any further updates in the shell document can be incorporated in TOC simply by rerunning this macro.

INTRODUCTION

Under current regulations (both national and international) any clinical trial is expected to be analyzed strictly in accordance with Statistical Analysis Plan (SAP). The underlying data and the results of this analysis are typically presented in the form of tables, listings and figures (TLF). The planned structure and appearance of TLF (i.e., shells ⇔ mocks) represent a final step of multiple discussions between clinical specialists, biostatisticians, medical writers and other parties involved in this trial. The shells are a natural part of any SAP and normally contain clear, concise and detailed instructions how to produce every one of the planned outputs. Development of the shells for TLF for any clinical trial is a time-consuming task. Using MS Word to create mocks for planned TLF is a very popular approach in the pharmaceutical industry.

Once a working version of the shells is approved a typical statistician or a lead programmer faces a necessity to transfer the key information characterizing the planned outputs (e.g., their ID, outputs' sequential numbers, outputs' titles, definition of populations, footnotes, company name, protocol number, etc.) into the so called Table of Content (TOC). Typically, TOC contains all the details necessary for the final run of PROC REPORT procedure in the form that can be easily read by SAS®. Excel spreadsheet or SAS® program or another approach (depending on the company's preferences and historical peculiarities) can be used for this purpose, but in both cases, it is a mostly manual copy-paste task. Copying titles, populations and footnotes from the shell document into SAS® program or external source of titles and footnotes (e.g., Excel file) is a tedious process requiring scrupulous work subject to human errors. There is normally a process of ongoing updates to the titles and footnotes through the whole lifetime of the project and even later, when the data are closely analyzed and TLF are developed and output. The last touch the TOC gets when a final clinical study report (CSR) is on the various stages of its writing, review, updates and alterations and medical writers thoroughly evaluate and examine every title and every single line of footnotes of the TLF that are going to become a natural part of the CSR. These numerous revisions (in many cases small ones) significantly complicate the ultimate goal of keeping the titles and footnotes accurate and strictly corresponding to the approved shells (and their later updates). Many statisticians and lead programmers spend substantial part of their time during the development of the shells and later, when a table of content (TOC) is created and constantly updated.

There is a well-accepted understanding in the community of the researchers working in pharmaceutical industry that it would be nice to have a tool that allows to automate this process, to save statisticians time and ultimately reduce time required for processing of results of clinical trials. Unfortunately, SAS® does not provide a procedure to import data directly from Microsoft Word into SAS similar to how PROC IMPORT reads data from Microsoft Excel. This deficiency stimulates SAS programmers and other professionals working in the drug development industry to explore other approaches and to develop the techniques of how to programmatically convert Word documents into SAS data.

LITERATURE REVIEW

To address the challenge, programmers made number of attempts in the past to read data from Word to SAS directly or indirectly. This Section reviews several solutions (the list is far from to be complete) that were suggested for public use and published in the professional literature.

Xu and Zhou (2007) suggested to use the INFILE statement to convert the plain text file programmatically saved by Word via DDE (Dynamic Data Exchange) into SAS. This method can import any Word-readable document into SAS without concerning RTF tagsets. The approach cannot be considered as fully automatic because a manual intervention (click) is necessary to close a pop-up window, if a Word document contains any special symbol characters, including sub-/super-scripts, during the step of saving as a plain text file, because there is no option available in the WordBasic FileSaveAs command to disable the pop-up window.

Huang and Lynn (2008) proposed their own way to obtain titles and footnotes from mockup tables in order to eliminate the step of retyping them in table production. The idea was to import a text file with the INFILE statement into SAS. However, the converting process requires a manual step to edit and save the Word mockup table into a plain text file prior to calling the macro. This manual process could introduce potential human error and be time-consuming when dealing with hundreds of tables.

Zhou (2009) presented a creative way how to import (programmatically) any Word-readable documents, such as .doc, .rtf, .txt, .xml, .html, .wpd, .wps, and .dot files, into SAS. A SAS macro, %word2sas, developed by the author and used for the converting process was introduced. The entire process involves SAS, Word, and Excel which is automated by a SAS macro. This method overcomes any drawback of converting a Word file into plain text. According to the author SAS programmers became able to leverage MS Word and SAS for their daily work with this convenient and reliable solution.

Gupta (2011) introduced a SAS macro %Read_Title_Footnote. This macro was designed to read titles and footnotes from the Microsoft word readable documents using Excel/Word Basic commands via DDE. In order to avoid any typing error caused while entering the text into SAS programs, a variation of this solution can be used to read titles and footnotes from word documents. Also, any further changes in the titles and footnotes can be easily incorporated into a SAS® dataset for the final study report. This convenient and reliable solution can help SAS programmers/statisticians to have better control over the quality of reports and save significant time. The entire process involves SAS®, MS Word, and MS Excel which is automated by a SAS macro called %Read_Title_Footnote. The macro imports the entire content from the title and footnote section of RTF tables, into SAS data.

Bentley (2013) looked at a different approach, not focused on TLF shells. The basic techniques of using SAS to read from (or write to) Word are to assign Bookmarks to specific locations in the document and then use FILENAME statements with the Dynamic Data Exchange (DDE) device-type key to point to those bookmarks. With this done, the bookmark can be accessed as a data set with a single-variable or a tab-delimited text file. The focus of that paper was reading tables in a Word document and then using the data from those tables to generate SAS SQL code.

Burmenko and Gardozo (2014) suggested their vision how shell content and rich text formatting of the shells can be automatically read in from a shell document to create a SAS PROC REPORT mock as a starting point for a table or listing SAS program. They suggested a way to marry the efforts of the table or listing shell creator with the efforts of the table or listing programmer in cases when the shells are designed manually. Their method proposed has three main components: 1) to read in a table or listing

shell document generated in RTF into SAS, 2) to parse the shell for text content and formatting, and 3) to translate that text and formatting into a starter table or listing SAS program.

There are also number of papers closely related to the subject under discussion in this paper, but not focused on the creation TOC based on the shells document. We will mention some of them here.

Hagendoorn *et al*. (2006) developed a utility macro tool that converts RTF-formatted table/listing back to SAS for the purpose of quality control (QC), comparing it against the data independently programmed with PROC COMPARE. There are many limitations with this method. Because the tool reads the RTF file with the INFILE statement in a data step to import the data from the table body based on the RFT control words, intensive maintenance to the macro is needed to remove different special symbols, and sub-/super-scripts, and to fit different column settings and data presentations. Artificially searching the specific RTF tagsets could be cumbersome and unreliable. In addition, table titles, column headers, and footnotes, and special characters are not imported for QC comparison.

Tran (2008) presented a method using Perl Regular Expression to express the RTF Regular Expression in order to strip the RTF tagsets. This Regular Expression approach provides more reliable outcome than the method provided by Hagendoorn *et al*. (2006). It requires version of SAS which provides a set of PRX CALL routines and functions to handle Perl Regular Expression. Potential maintenance to the RTF Regular Expression may be 2 needed due to a new Word release or RTF version change. In addition, Tran's %RTFparser utility tool can be easily improved to parser any Word-readable documents by adding a step with the DDE linkage as employed by Xu and Zhou (2007) to save non-RTF files in RTF prior to pursuing the Regular Expression.

The macro mSHELL2TOC presented in this paper allows total automation of reading the shells for TLF (organized in the form of the regular Word file) and transforming them into SAS®-readable ordered table of content (TOC, Excel spreadsheet). This automatic process takes into account repeated tables (and adds them to the TOC in an order defined by its number), updates definition of the population (if appropriate requirement exists in the shells), etc. The proposed tool can significantly save time for biostatisticians and lead programmers who have to numerously create, revise, update shells documents for TLF.

GENERAL DESCRIPTION

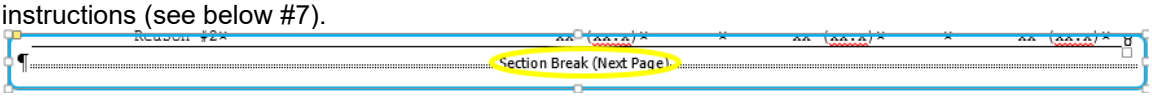
Macro mSHELL2TOC was developed in a framework of internal project of automation conducted by the Department of Clinical Programming and Statistics, Accenture Life Science Solutions. The ultimate goal of the proposed macro was to automate a process of initial creation and numerous late updates of the Table of Content (TOC) used to output tables, listings and Figures (TLF) according to the shells provided by the project statistician. The macro was created using Excel Visual Basic for Applications (VBA) and requires running the appropriate module within VBA environment. The macro is residing in the regular EXCEL file, it reads the shell document (WORD) and creates TOC within the original Excel file. TOC displays every document mentioned in the shells appears in the ascending order within its group (tables, listings, figures).
The macro was developed within environment of the MS Office 2013 that is currently installed on the SAS server (ALSS). It is possible, therefore, that after next upgrade of the MS Office the macro will require customized tuning to address the changes in VBA that were introduced by this upgrade.

SHELLS - WORD DOCUMENT

In order to allow the developed macro to run smoothly the Word file containing the shells for TLF should be correctly prepared in some special way. The instructions below describe a structure and peculiarities of the shell document which should be maintained to achieve this goal.

Title and footnotes of the projected TLF should be entered as header and footer of the WORD document, correspondingly (refer to Figure 1 below).

1. Every shell must be confined within its own Section and is separated from other shells by Section Break (Go to “Page Layout”->”Breaks”->”Section Breaks”->“Next Page”; see screenshot below). The macro reads distinct Sections within the Word document and assign title and footnotes to the appropriate output. The first Section of the Shells document is normally containing general instructions (see below #7).



2. Once a new Section is created enter the header of the WORD document, go to “Header and Footer Tools”->”Link to Previous” and make sure that this option (”Link to Previous”) is unmarked/unchecked for both header and footer. It allows to revise title and footnotes for a new shell without their changes in the previous mock.

3. The general information appearing in the header of every one of the outputs (e.g., company name, protocol number, etc.) must be the same (i.e., absolutely identical) in every one of the shells (it can take one, two or more lines, but they must be same for all outputs – macro is looking for repeating parts of the titles and add them to the beginning of the TOC as it is accepted).

4. Title of the output is presumed to contain three different parts located on the separate lines:
    a. Output type and number (e.g., Table 14.1.1, Listing 16.2.1, Figure 14.2.1.1). This part must start with one of these three words - “Table”, “Listing”, “Figure”.
    b. Title of the output (e.g., “Subject Disposition and Analysis Population by Treatment Group”).
    c. Population definition (e.g., “Safety Population”). This part can be missing.
    d. Every one of these three parts should be separated from each other by the paragraph mark (use “Ctrl +” to see hidden formatting symbols).
    e. If the title of the output is longer than an available line, the title should be subdivided into two parts on two subsequent lines. Each line should end with the paragraph mark.

5. Footnotes.
    a. It is strongly recommended to use for title and especially for footnotes in the shells the same size and font that will be used for actual outputs. It will allow to manage available space (especially for the footnotes) most effectively. The macro will read the footer and put the content of every line (of the footnotes) – exactly as they appear - into appropriate cell of the Excel TOC file.
    b. There is no need to add line of the type “Program: outid.sas Programmer: xxx Data Source: ADSL, ADLES, ADPE ddmmmyyyy hh.mm SAS 9.4” to the TOC as it will be added automatically at the end of every footnote as part of the established internal process.
    c. Every line of the footnotes (as a statistician wants to see it in the actual output printed on the paper) should end with the paragraph mark (use “Ctrl +” to see hidden formatting symbols).
    d. If there are no meaningful footnotes (excluding the automatic one with information about program name, programmer, date and time stamp, etc.) in the shell then no footnotes will be copied to TOC.

6. Repeated outputs (i.e., one that should be created based on the shell for another TLF).
    a. Instructions to use the same layout for another output are normally presented in the part of the shell called “Notes to programmers” (body of the Word document, normally right after any clarifying comments to programmers (“Notes to Programmers”) regarding the proposed shell).
    b. Every instruction to use the same layout for another output must start with the words “Repeat this Table” and read like {Repeat this Table for the Table 14.1.6.2 “Cancer Diagnosis by Gender and Treatment Group”} (double quotes must be used).

c. An instruction to use the same layout for another output can be about Table, Listing or Figure (Repeat this Listing or Repeat this Figure, correspondingly).
d. The title of a new output (to be created based on the shell for another TLF) must be in double quotes (regular Word double quotes - “”). By default a definition of a population will be copied from the sample shell.
e. If there is a need in updating population in a new output then the request for this change should follow the description of the new title. This request must start with the words “Change the population to” followed by the description of the new population in regular Word double quotes (e.g., “Safety Population”). Here is an example of the request for Repeat this figure for Figure 14.2.2.5 “Overall Survival (Safety Population) by Treatment Group”. Change the population to “Safety Population”.

7. The first Section of the shells document is reserved for general information for a person working on this project. This Section typically contains instructions for programmers that are not output-specific and are applicable to the whole set of TLF for this study. The list includes, but is not limited to, size and name of the font to be used in the actual outputs, number of columns in the tables, bold/regular titles, case of the text in the Descriptor column. Sometimes this Section includes list of planned TLF from the SAP, summary of unique outputs vs repeat ones, etc. This Section is excluded from the macro consideration. To make sure that it is the case two conditions must be fulfilled:
    a. Both header and footer for this Section are empty.
    b. The text (body of the document; not header and/or footer) contains the phrase “General Notes for Programmers”.

TOC – EXCEL FILE

This part of the Instructions and Rules describes how the macro should run and how we need to understand a structure and results of its application.

1. Open EXCEL file where the macro is residing and clean up the Sheet 1, i.e. erase everything that is probably remaining there from previous exercises.

2. Go the left side of the bar and press the tab “Macros”.

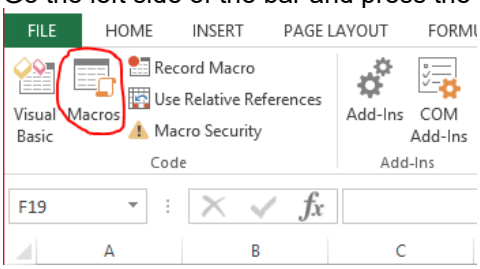
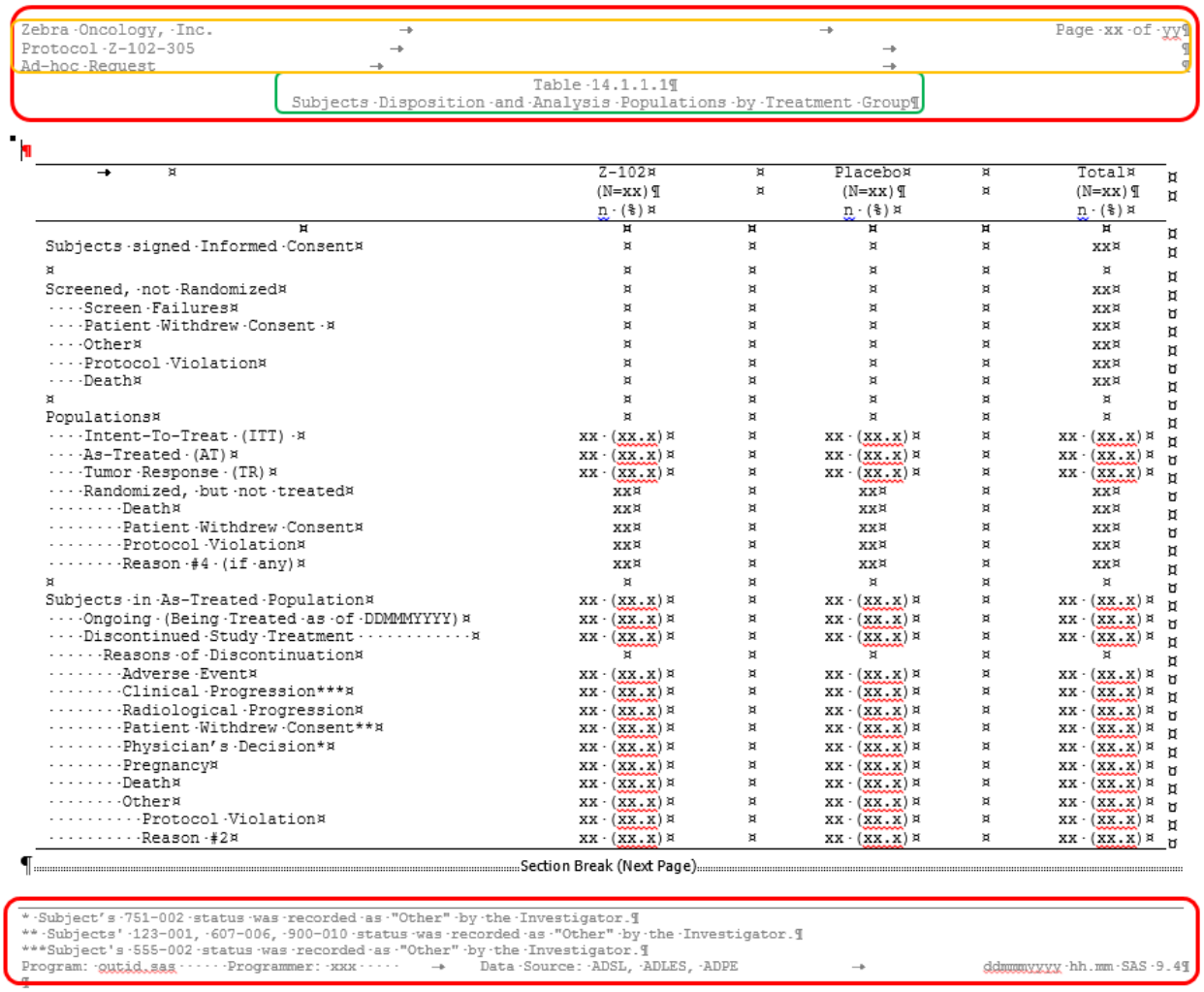


Figure 1. Location on the bar (Excel) to start the macro



Zebra Oncology, Inc. → → Page xx of yy
Protocol Z-102-305 → → →
Ad-hoc Request → → →

Table 14.1.1.1
Subjects Disposition and Analysis Populations by Treatment Group

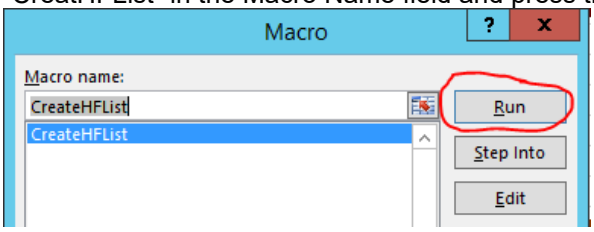
	Z-102 (N=xx) n (%)	Placebo (N=xx) n (%)	Total (N=xx) n (%)
Subjects signed Informed Consent			xx
Screened, not Randomized			xx
Screen Failures			xx
Patient Withdrew Consent			xx
Other			xx
Protocol Violation			xx
Death			xx
Populations			
Intent-To-Treat (ITT)	xx (xx.x)	xx (xx.x)	xx (xx.x)
As-Treated (AT)	xx (xx.x)	xx (xx.x)	xx (xx.x)
Tumor Response (TR)	xx (xx.x)	xx (xx.x)	xx (xx.x)
Randomized, but not treated	xx	xx	xx
Death	xx	xx	xx
Patient Withdrew Consent	xx	xx	xx
Protocol Violation	xx	xx	xx
Reason #4 (if any)	xx	xx	xx
Subjects in As-Treated Population	xx (xx.x)	xx (xx.x)	xx (xx.x)
Ongoing (Being Treated as of DDMMMYYYY)	xx (xx.x)	xx (xx.x)	xx (xx.x)
Discontinued Study Treatment	xx (xx.x)	xx (xx.x)	xx (xx.x)
Reasons of Discontinuation			
Adverse Event	xx (xx.x)	xx (xx.x)	xx (xx.x)
Clinical Progression***	xx (xx.x)	xx (xx.x)	xx (xx.x)
Radiological Progression	xx (xx.x)	xx (xx.x)	xx (xx.x)
Patient Withdrew Consent**	xx (xx.x)	xx (xx.x)	xx (xx.x)
Physician's Decision*	xx (xx.x)	xx (xx.x)	xx (xx.x)
Pregnancy	xx (xx.x)	xx (xx.x)	xx (xx.x)
Death	xx (xx.x)	xx (xx.x)	xx (xx.x)
Other	xx (xx.x)	xx (xx.x)	xx (xx.x)
Protocol Violation	xx (xx.x)	xx (xx.x)	xx (xx.x)
Reason #2	xx (xx.x)	xx (xx.x)	xx (xx.x)

Section Break (Next Page)

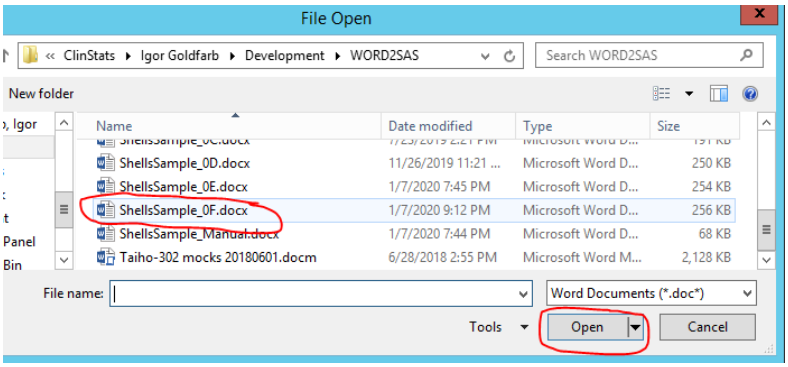
* Subject's 751-002 status was recorded as "Other" by the Investigator.
** Subjects' 123-001, 607-006, 900-010 status was recorded as "Other" by the Investigator.
*** Subject's 555-002 status was recorded as "Other" by the Investigator.
Program: outid.sas Programmer: xxx → Data Source: ADSL, ADLES, ADPE → ddmmmyyyy hh.mm SAS 9.4

Figure 2. Typical shell for disposition table. Red color marks title (header in Word document) and footnotes (footer in Word document), green color denotes table number and its name. Sand color marks part of the title that is repeated (identically) in every output (protocol number, legal name of the client, etc.)

3. When a small window opens please make sure that you see an internal name of the macro "CreatHFList" in the Macro Name field and press the tab "Run" in the right upper corner



4. The file dialog window opens where only Word files are shown. Select an appropriate file and press the tab "Open"



5. Macro starts to run. It automatically cleans the Sheet 1 and after that copies there all the titles and footnotes. They are combined in ordered groups corresponding to their original shell and are ordered according to their numbers.

6. Check the content of the Sheet 1 – it now contains TOC that was created based on the shells that were developed in appropriate way (i.e., strictly following the rules that were described in the Section “Shells - WORD document” above) and combined all together in the LiveMOCK document. Make sure that a content of TOC is corresponding to the source file – LiveMOCK.

7. Structure of TOC.
    a. TOC is created based on the information collected from the shell document (LiveMOCK). The structure and content of the TOC follows the way that is well accepted by the standard working process of the Department of Clinical Programming at Accenture ALSS.
    b. TOC represents an Excel file with meaningful information in the first 3 columns – A, B and C.
        i. Column A of the TOC file contains standardized identifying descriptor (ID) for every output that is described in the LiveMOCK. A conversion is run using a standard approach based on two digits representing every level of hierarchy. For example, for table 14.1.1 a descriptor T140101 will be created, for listing 16.2.15 – ID L160215, for figure 14.3.7.11 – ID F14030711 and so on. Groups of rows containing the same TLF ID are gathered together and are separated from each other by one or two empty lines to make a visibility of the information more effective.
        ii. Column C of the TOC file displays the unique content of the headers and footnotes for every one of the outputs that were mentioned in the shells (regardless of the actual status – real mock or a shell that was determined by a request to build an output by repeating some real mock). Title of the output normally takes 3 lines (common lines of all tiles are NOT counted here). Deviations from this number are permissible and not restricted, but at least one line of a title is expected. The number of lines for footnotes can vary from 0 (ne meaningful footnotes) to 9.
        iii. Column B of the TOC file contains standardized description of the titles (i.e., TITLE*i*) or footnotes (i.e, FOOTNOTE*i*) depending on the content of the Column C. In other words, it says us if text in column C is a footnote or a title and on what line this part of the text should be placed in the actual output (TLF). Value of *i* (running number) depends on the actual number of lines in the title and footnotes of the original shell.
    c. The first lines of the TOC (one, two or more – according to the shells) will display information that is common for all outputs and is required to be displayed in every TLF. Column A for these lines will contain record “_ALL_”, column B – “TITLE1”, “TITLE2”, “TITLE3” (according to the LiveMOCK), column C – text that is expected to appear in all outputs.

DISCUSSION

The accumulated experience of the practical application of the developed macro mSHELL2TOC taught us some lessons. Some of the lessons leant are worth to be shared with the prospective users of the macro.

The MS Word shell document should be properly and carefully prepared strictly in accordance with the rules and instructions described above. The matter is that the macro is very sensitive to the issues in the mock Word file. As an example from our practice we could refer to an actual case when the common part of the titles (company name, protocol number, etc.) that is supposed to be exactly identical across the mock document appeared to be different in two parts of the shell file. The macro is built in the way that it tries to identify the identical lines in every title and when it didn't find them an error message appeared. An elementary debugging allowed to understand the reason of the problem and to fix the shells document in appropriate way, i.e., to make all common lines of the title absolutely identical.

Another outcome that can be derived from out relatively small experience of running the macro is practical one. We concluded that even if one already has mocks Word file that he started work one it is always time- and efforts-saving to open another Word document than to rewrite the existing one trying to follow the rules. It is always beneficial to start it from the very beginning and revise it (using Ctrl-C – Ctrl-V process as much as possible) ensuing the stickiness to the instructions described above.

VIEW TO THE FUTURE

It is well known that a way of a synthesized compound from research laboratory to the FDA approval as an effective and safe drug takes many years. Clinical trials take significant part of this time. Data collected during a clinical trial are cleaned, reviewed, verified, reconciled, fixed (if necessary), and, finally, analyzed, processed, summarized and displayed in the form of TLF. Every one of these steps takes its own time, no error is permitted in this sequence and all team members share the common task of reducing the total time required for drug approval.

Post-text TLF produced for clinical trial report for submission (CSR) continue to become more standardized. A number of cross-functional teams work tirelessly to achieve this goal and to bring harmonized industry reporting standards to daily practice in the pharmaceutical industry. The authors believe that the future will bring these standards that could be supported by a library of standardized programs. However, at present we witness a wide spectrum of various approaches to the presentation of the TLF and to formulation how their titles should sound and what information their footnotes should include. On the other hand, there is a shared understanding across the pharmaceutical industry that the intricacies of reporting results for complex clinical trials will always require some level of shell customization. Finally, it is evident that to evaluate treatments for drastically different indications the statisticians and medical writers have to develop various shells to display the final results in an adequate way. The charge of every clinical specialist, principal investigator, statistician, statistical programmer, medical writer who are analyzing and reporting clinical study data should be to approach these intricacies with a mind toward innovation and efficient use of resources. The goal of this paper is to suggest both specific tool (macro) and general methods that can be helpful in development and implementation of time-saving approaches and can spark elegant and innovative solutions in the future.

The authors believe that the developed macro mSEHLL2TOC can be used widely to automate the process of generating a SAS®-readable Table of Content from the properly prepared mocks document and to save time and efforts for numerous statisticians, lead programmer and those who face the similar task in their professional routine.

CONCLUSIONS

To recap the discussion of the developed macro mSHELL2TOC it would be worthwhile to summarize macro's capabilities and emphasize its main advantages:

1. The developed macro mSHELL2TOC allows to create (in seconds!) SAS®-readable ordered TOC (Excel) from the properly prepared TLF shells and to eliminate once manual and very time-consuming process.
2. The code of the macro resides in the standard MS Excel file (Excel VBA). While running it reads headers, footnotes and key instructions from the properly prepared mock document (standard MS Word file).
3. The proposed macro identifies common part of the titles (e.g., company's name, protocol number, etc.) and footnotes for all TLF as well as detects differences for specific outputs.
4. The suggested macro analyzes all the requests for repeating outputs, updates their sequential numbers and titles and adds them to the TOC (in the ordered way).
5. The created macro is capable to change population if required for repeating tables/figures (appropriate request should be placed in the description of the repeating output).
6. The developed macro generates an Excel file containing ordered TOC that is immediately ready to be used for the final run of SAS® programs to output planned TLF.
7. Any further updates in the shell document can be incorporated in TOC simply by rerunning this macro. And it takes literally seconds!

REFERENCES

Bentley, J.E. (2013), "Reading Data from Microsoft® Word Documents: It's Easier Than You Think". *Proceedings of SAS Global Forum*, Paper 121-2013.

Huang, D., and Lynn, L., Paper (2008), "Read Titles and Footnotes from a Table Shell into a SAS Dataset" Author, Daniel Huang, *Proceedings of PharmaSUG 2008*, Paper PO07.

Burmenko, P., and Cardozo, T., (2014) "Come Out of Your Shell: A Dynamic Approach to Shell Implementation in Table and Listing Programs", *Proceedings of PharmaSUG 2014*, Paper BB17.

Chen, H., (2012), "Prove QC Quality – Create SAS ® Dataset from RTF file." *Proceedings of NESUG 2012*, Paper PH03.

Gupta, A., (2011), "Reading Title and Footnote from RTF Output into SAS® utilizing Microsoft® Excel", *Proceedings of PharmaSUG 2011*, Paper CC11.

Hagendoorn, M., Squire, J., and Johnny Tai, J., (2006), ""Save Those Eyes: A Quality-Control Utility for Checking RTF Output Immediately and Accurately", *Proceedings of the Thirty-first SAS Users Group International Conference*, Paper 066-31.

Tran, D., (2008), "%RTFparser", *Proceedings of Pharmaceutical Users Software Exchange Conference - 2008*, Paper PO03.

Xu, M., and Zhou. J., (2007), "%DIFF: A SAS Macro to Compare Documents in Word or ASCII Format", *Proceedings of the PharmaSUG 2007*, Paper CC09.

Zhou, J., (2009), "Importing Data from Microsoft Word into SAS®", *Proceedings of PharmaSUG 2009*, Paper CC18

ACKNOWLEDGMENTS

The authors are very thankful to Jack Jacobsen, Senior Director of Clinical Programming and Statistics (Accenture ALSS), for his constant support of this work.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please feel free to contact anyone of the authors at:

Igor Goldfarb
Accenture ALSS
Igor.goldfarb@accenture.com

Ella Zelichonok
Naxion
ezelichonok@naxionthinking.com

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

APPENDIX – TEXT OF THE MACRO

```
Sub mSHELL2TOC()
'
' Macro1 Macro
' This macro reads the shells for project TLF (WORD file) and creates TOC in the form of EXCEL spreadsheet
' It is presumed that shells are created in a way where header/footer of the WORD file contains title/footnotes of the actual output to be created according to this shell
' The macro reads headers and footers of every section of the WORD file and converts them into text for TITLEi and FOOTNOTEj cells of TOC for corresponding output
'

Dim dc As Document
Dim OBJSection As Section
Dim OBJHF As HeaderFooter
Dim W As Word.Application
Dim File_Name As String



'Allow only single select from file dialog
  Application.FileDialog(msoFileDialogOpen).AllowMultiSelect = False
'Filter WORD documents only to be shown for user to choose from
  Application.FileDialog(msoFileDialogOpen).Filters.Add "Word Documents", "*.doc*", 1
'Show file dialog
  If Application.FileDialog(msoFileDialogOpen).Show Then
    'Select the file from the dialog
```

```
   File_Name = Application.FileDialog(msoFileDialogOpen).SelectedItems(1)
  Else
    'exit macro if no file is selected
    Exit Sub
  End If

 'prepare Excel sheet by clearing out the content
  Cells.ClearContents

'Creating instance of Word application (MS Word must be installed)
Set W = CreateObject("Word.Application")
'Open seleced file into a word document
Set dc = W.Documents.Open(File_Name, , True)


'Declaring needed variables
Dim HdrRange As Word.Range

Dim r As Integer

r = 0
Dim arrFirstHeader, arrFirstFooter
Dim nStart, rFirstHeaderRow, rFirstFooterRow

Dim nFooterEnd
Dim SectionText
Dim iRepeat
Dim Repeats As New Collection

Dim k0 As Integer
Dim SecText As String
'Identify first non-empty section that also doesn't have "general notes for programmers" instructions
(usually at the beginning)
For k = 1 To dc.Sections.Count
    k0 = k
    SecText = dc.Sections(k).Range.Text
    If InStr(1, SecText, "general notes", vbTextCompare) = 0 And SecText <> "" Then

        Exit For
    End If
Next k


For k = k0 To dc.Sections.Count
' Variable k represents an ordinal number of a section in WORD document (it is
' assumed that every shell represents separate section within WORD file)


'SectionText variable will contin full text of the section
'Need it to check if this section need to be repeated for other table/figure
SectionText = dc.Sections(k).Range.Text


'empty out repeat lines collection
For I = Repeats.Count To 1 Step -1
    Repeats.Remove (I)
```

```
Next

'identifying sections to repeat
'it will always start with "Repeat this layout"
iStartCopyRow = 0 'this will hold beginning of the excel row where section repeats
iRepeat = 0
iRepeat = InStr(iRepeat + 1, SectionText, "repeat this layout ", vbTextCompare) 'index of the phrase
Do While iRepeat > 0
   'keep on looking for the phrase

   'will read up to the end of the line
   iNextCr = InStr(iRepeat, SectionText, vbCr)

   iFor = InStr(iRepeat, SectionText, " for ", vbTextCompare)
   sRepeat = Mid(SectionText, iFor + 5, iNextCr - iFor - 6)
   If InStr(1, sRepeat, "the ") = 1 Then sRepeat = Mid(sRepeat, 5)

   'add the line to the collection of the repeat lines for the section
   Repeats.Add (sRepeat)
   iRepeat = InStr(iRepeat + 1, SectionText, "repeat this layout ", vbTextCompare)
Loop

'1=wdHeaderFooterPrimary
'2=wdHeaderFooterFirstPage

Set HdrRange = dc.Sections.Item(k).Headers(1).Range

'In word sometimes headers erouneously are repeated in sections - in this case compare to previous
header and if equal - replace with wdHeaderFooterFirstPage(=2) element of the header
If HdrRange.Text = prevHeaderText Then
   'Set HdrRange = dc.Sections.Item(k).Headers(2).Range
End If

prevHeaderText = HdrRange.Text


Dim arr, arr2
arr = Split(HdrRange.Text, vbCr)
If k = k0 Then arrFirstHeader = arr: rFirstHeaderRow = r

If k = k0 + 1 Then

   For i = 0 To UBound(arr)
      If arr(i) <> arrFirstHeader(i) Then
         nStart = i
         Exit For
      End If
   Next i
End If

Dim sTableTag As String
For i = nStart To UBound(arr)
   If arr(i) <> "" Then
      arr2 = Split(arr(i), vbTab)
      If UBound(arr2) = 0 Then
         'sometimes tab is not identified
```

```
            'in that case arr2 will be only one element - ubound of 0
            'we noted that 2 tabs sometimes is read as 00
            'so we 'll split by 00
            arr2 = Split(arr(i), "00Page")
        End If


        r = r + 1

        'the very first row in the sections
        If iStartCopyRow = 0 Then iStartCopyRow = r


        Cells(r, 2).Value = "TITLE" & i + 1
        For j = 0 To UBound(arr2)
            If arr2(j) <> "" Then
                'getting first non empty - in case of empty tabs
                Cells(r, 2 + 0 + 1).Value = arr2(j)
                Exit For
            End If
        Next j
    End If
Next i


'Footnotes are read
Dim FtrRange As Word.Range

Set FtrRange = dc.Sections.Item(k).Footers(wdHeaderFooterPrimary).Range

'r = r + 1
'Cells(r, 1).Value = "Footers"
If FtrRange.Text <> "" Then
arr = Split(FtrRange.Text, vbCr)

For i = UBound(arr) To 0 Step -1
        'Neutralize footnote that is generated automatically - Program: outid.sas....
        If InStr(1, arr(i), "Program: outid.sas") > 0 Then arr(i) = ""

    If arr(i) = "" And UBound(arr) > 0 Then

        ReDim Preserve arr(UBound(arr) - 1)
    Else
        Exit For
    End If
Next i
Else
Exit For
End If


If k = k0 Then arrFirstFooter = arr: rFirstFooterRow = r

If k = k0 + 1 Then
    Dim p As Integer
    p = UBound(arrFirstFooter)
```

```
    nMatch = 0

    For i = UBound(arr) To 0 Step -1

        If arr(i) = arrFirstFooter(p) Then

            nMatch = nMatch + 1
            p = p - 1

        Else

            Exit For
        End If
    Next i
End If

For i = 0 To UBound(arr) - nMatch
    If arr(i) <> "" Then
        arr2 = Split(arr(i), vbTab)
        r = r + 1

        Cells(r, 2).Value = "FOOTNOTE" & i + 1
        For j = 0 To UBound(arr2)


            Cells(r, 2 + j + 1).Value = arr2(j)
        Next j
    End If
Next i

iEndCopyRow = r

r = r + 1
Cells(r, j + 1).Value = ""

r = r + 1
Cells(r, j + 1).Value = ""

'iterate through each repeat statement
For l = 1 To Repeats.Count
    sRepeat = Repeats(l)
    'replace quotes from word with regular quotes
    sRepeat = Replace(sRepeat, Chr(147), Chr(34))
    sRepeat = Replace(sRepeat, Chr(148), Chr(34))

    'Parse out the name of the Table/Figure/Listing
    sName = Mid(sRepeat, 1, InStr(sRepeat, """") - 2)

    'determine the quotes - where title will be
    iQ1 = InStr(1, sRepeat, """")
    iQ2 = InStr(iQ1 + 1, sRepeat, """")

    'Parse out the title
    sTitle = Mid(sRepeat, iQ1 + 1, iQ2 - iQ1 - 1)

    'check if change population was requested
```

```
    iPop = InStr(iQ2, sRepeat, "change population", vbTextCompare)
    sPopulation = ""
    If iPop > 0 Then
       'determine the quotes - where new population will be
       iQ1 = InStr(iPop, sRepeat, """")
       If iQ1 > 0 Then
          iQ2 = InStr(iQ1 + 1, sRepeat, """")

          'Parse out the new population
          If iQ2 > 0 Then sPopulation = Mid(sRepeat, iQ1 + 1, iQ2 - iQ1 - 1)
       End If
    End If

    mCounter = 0
    'copy the rows from the section that repeats
    For m = iStartCopyRow To iEndCopyRow
       mCounter = mCounter + 1
       r = r + 1
       Cells(r, 2).Value = Cells(m, 2).Value
       Select Case mCounter
       Case 1
          Cells(r, 3).Value = sName 'replace with the new name
       Case 2
          Cells(r, 3).Value = sTitle 'replace with the new title
       Case 3
          'replace with new Population if needed
          If sPopulation <> "" Then
             Cells(r, 3).Value = sPopulation
          Else
             Cells(r, 3).Value = Cells(m, 3).Value
          End If
       Case Else
          Cells(r, 3).Value = Cells(m, 3).Value 'the rest stays the same
       End Select

    Next m
    r = r + 1
    Cells(r, j + 1).Value = ""

    r = r + 1
    Cells(r, j + 1).Value = ""
Next l

Next k


For k = 1 + rFirstFooterRow + (UBound(arrFirstFooter) + 1) - nMatch To 1 + rFirstFooterRow +
(UBound(arrFirstFooter) + 1) - 1
Rows(k).Delete
Next k

Rows(1 + rFirstHeaderRow + nStart).EntireRow.Insert

'assigning tags in a special format
AssignTags nStart, r
```

```
're-sorting in case it wasn't properly sorted in the word document, especially with repeated sections
SortThis 1, r

dc.Close
W.Quit
End Sub

Sub SortThis(r1, r2)
'insert 2 columns
'first copumn will have the tags, including empty spaces
'2nd column will have the counters
Columns(1).Insert
Columns(1).Insert
rCounter = 0
For r = r1 To r2
   rCounter = rCounter + 1
   If Cells(r, 3).Value <> "" Then
      sTag = Cells(r, 3).Value

      Select Case Mid(sTag, 1, 1)
      Case "T"
         sTag = "1" & sTag 'want to have Tables fist, then Listings, then Figures, therefore attaching prefix
1,2,3 to ensure it
      Case "L"
         sTag = "2" & sTag
      Case "F"
         sTag = "3" & sTag
      End Select
      Cells(r, 1).Value = sTag
   Else
      Cells(r, 1).Value = sTag
   End If

   Cells(r, 2).Value = rCounter
Next r


'sort by first 2 columns
   ActiveSheet.Sort.SortFields.Clear
   ActiveSheet.Sort.SortFields.Add Key:=Range(Cells(r1, 1), Cells(r2, 1)) _
      , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
   ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Add Key:=Range(Cells(r1, 2), Cells(r2, 2)) _
      , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
   With ActiveWorkbook.Worksheets("Sheet1").Sort
      .SetRange Range(Cells(r1, 1), Cells(r2, 5))
      .Header = xlGuess
      .MatchCase = False
      .Orientation = xlTopToBottom
      .SortMethod = xlPinYin
      .Apply
   End With

   'delete working columns used for sorting
   Columns(1).Delete
   Columns(1).Delete
End Sub
```

```
Sub AssignTags(nStart, nEnd)
'This sub assigns tags in the first column of TOC
Dim sTag As String

'Assigns values to parts of titles that are common for all TLF and will be repeated in all outputs
'Runs from 1 to nStart, where nStart represents a number of repeating lines in the TLF titles
For r = 1 To nStart
   Cells(r, 1) = "_ALL_"
Next r

'Assigns values to parts of titles that are unique for every output
'Runs from (nStart+1) to nEnd, where nENd represents a total number of lines in the TLF titles
For r = nStart + 1 To nEnd
   If sTag = "" Then
      'Uses function that will parse the table/figure/listing title
      sTag = ProcessTag(Cells(r, 3).Value)

   End If
   If Cells(r, 3).Value = "" Then sTag = ""
   If sTag <> "" Then Cells(r, 1) = sTag

Next

End Sub

Function ProcessTag(sTag As String)
   'This function will parse the table/figure/listing title
   'First letter will be taken from the name (i.r., T for Table, F - for Figure, L - for Listing)

   If sTag = "" Then ProcessTag = "": Exit Function
   Dim sResult As String
   Dim arr
   arr = Split(sTag, " ")
   sResult = Mid(arr(0), 1, 1)
   If UBound(arr) >= 1 Then

      sNumbers = arr(1)

      For j = 1 To Len(sNumbers)
         If Asc(Mid(sNumbers, j, 1)) < 32 Then
            sNumbers = Mid(sNumbers, 1, j - 1)
            Exit For
         End If
      Next j
      arrNumbers = Split(sNumbers, ".")

   'Each output number will be formatted based on 2-digit representation of every level of granularity
   '(e.g., Table 14.1.1 will become T140101, Figure 14.2.1.3 will become F14020103, etc.)
      For j = 0 To UBound(arrNumbers)
         sResult = sResult & Format(arrNumbers(j), "00")
      Next j
```

```
    End If
    ProcessTag = sResult
End Function
```