PharmaSUG 2021 - Paper AD-179

Macro to Compare Titles and Footnotes in Produced TLF and Corresponding Shells

Igor Goldfarb, Accenture Life Sciences;
Ella Zelichonok, Naxion

ABSTRACT

The goal of this work is to develop a macro that automates an important part of the final review process of TLF (tables, listings and figures). Specifically, the proposed macro compares titles and footnotes which are residing in the approved shells with titles and footnotes in the corresponding actual outputs and identifies the differences. The proposed tool can significantly simplify review work for biostatisticians and lead programmers who have to verify that TLF were generated correctly according to the shells.

Final review of the produced TLF represents an important task in a process of statistical programming. Comparing and making sure that the actual outputs were created strictly in accordance with the approved shells' document (typically MS Word file) is a tedious process requiring scrupulous work that is subject to human errors.

The proposed macro (developed in Excel VBA) automates this process. It reads the shell document (Word) and creates a SAS®-readable ordered table of content (TOC, Excel) in a matter of seconds. As a second step the developed macro reads the generated outputs (single Word file or set of them) that are reviewed and identifies their titles and footnotes. During the third stage the macro compares the two sets of information and marks the distinctions. Any further updates in the shell document and, in turn, in the created TLF, can be produced for review another time simply by rerunning this macro. Generally speaking, this macro can act as Version Control tool for titles and footnotes in created TLF and their Shells.

INTRODUCTION

It is well known that a journey of a new invented biologic product or chemical compound from research laboratory to the authority's approval as an effective and safe medicine takes many years. Clinical trials take significant part of this time.

Conduction of clinical trial and analysis of the obtained results represent in our days well established and regulated (both on national and international levels) environment. It is well known that the collected data are expected to be analyzed strictly in accordance with Statistical Analysis Plan (SAP), which is developed as a result of close collaboration of biostatisticians, lead programmers, data managers, clinicians, medical writers, etc. The results of these analyses are typically presented in the form of tables, listings and figures (TLF). Normally TLF are produced according to the document that is called shells (or mocks). The shell document is a necessary part of any SAP and contains 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 statistical programmers start to produce tables, listings and figures according to the SAP and the instructions in the shells. Double programming (when two persons – production programmer and validation programmer – independently work on one output) represents a standard approach in the pharmaceutical industry. Nevertheless, a typical statistician or a lead programmer faces a necessity to review the produced outputs before delivering them. The review process includes many different components and one of them – verification that the actual outputs were created strictly in accordance with the specifications (mocks' document).

The shells (Word document) cannot be used in SAS® directly as they created. A popular approach is that 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.) is transferred into 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 special 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 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).

The developed macro mSHELLvsTLF allows to automate the process of the reviewing of titles and footnotes in the actual outputs against their shells. 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) and generates an Excel sheet (named Shells) containing ordered TOC (Goldfarb and Zelichonok, 2020). After that the macro reads the actual TLF that require the review (it can be a single output or multiple files) and collects their titles and footnotes in the similar way in another Excel sheet. Then the proposed macro takes the set of titles and footnotes of the tables under review and compare them with ones that were generated while reading specification document. The uncovered distinctions are analyzed and marked using the color code. The task of a statistician or lead programmer is to determine if a found difference represents a real issue and the output under review (or, alternatively, appropriate shell) should be fixed or it is just a result of imperfectness of the proposed macro.

OVERVIEW – HOW MACRO WORKS

Macro mSHELLvsTLF was developed in a framework of internal project of systemic automation conducted by the Department of Clinical Programming and Statistics, Accenture Life Science. The macro represents a natural continuation of the previous work of the authors (Goldfarb and Zelichonok, 2020), devoted to development of another macro, mSHELL2TOC. Macro mSHELL2TOC allowed to automate a process of initial creation and numerous late updates of the Table of Content (TOC) used to output tables, listings and figures according to the shells provided by the project statistician. Macro mSHELLvsTLF utilizes the same paradigm and provides the statistician or lead programmer with an automated way of comparing titles and footnotes as they were placed in the shell document with actual titles and footnotes as they appeared in the final outputs (tables, listings and figures produced by statistical programmers).

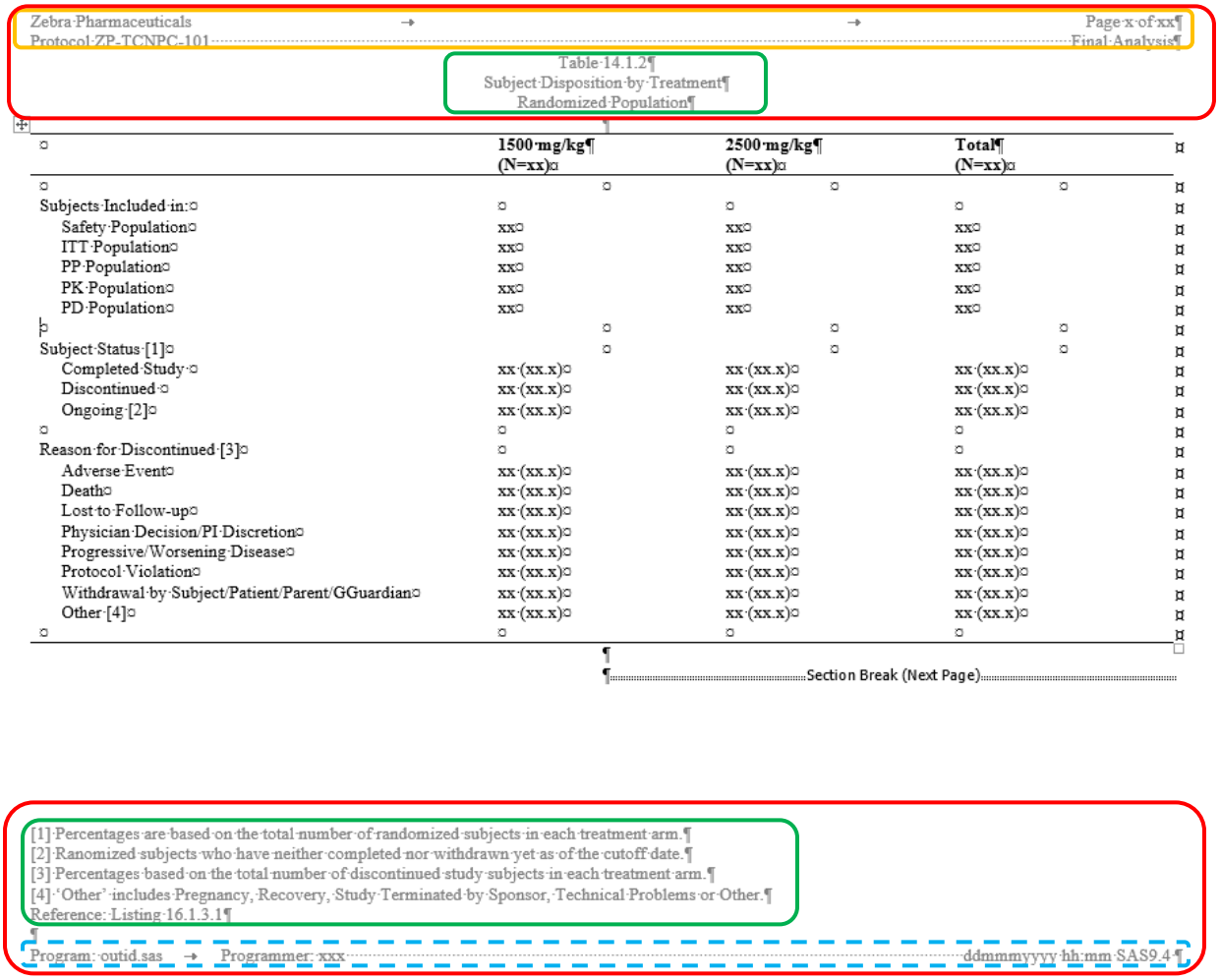
The macro mSHELLvsTLF was created using EXCEL Visual Basic for Applications (VBA) and requires running the appropriate module within VBA environment. The macro is residing in the regular EXCEL file, it reads the shell document (MS WORD) and actual outputs (typically .rtf files, MS WORD). As the first step macro mSHELLvsTLF reads the shell document and creates TOC within the original Excel file. TOC displays every document mentioned in the shells in the way that it appears in the ascending order within its group (tables, listings, figures). As the second step the macro reads titles and footnotes from the actual outputs that were produced for this project and are expected to be reviewed. The third and the final stage of the process contains a line by line comparison between titles and footnotes from the mocks document and actual outputs. In case of a difference the macro produces a color-coded message informing the statistician about an uncovered distinction that would require additional review.

The macro was developed within an existing environment of the MS Office 2013 that is currently installed on the SAS server (Accenture Life Sciences). It is possible, therefore, that the next upgrade of the MS

Office will require customized macro tuning to address the changes in VBA that might be introduced by that upgrade.

SHELLS - WORD DOCUMENT

In order to allow the developed macro to run smoothly the WORD file containing the shells for TLF should be properly prepared in specific way. The full set of detailed instructions can be found in the previous publication of the authors (Goldfarb and Zelichonok, 2020). For reader's convenience we added to the Appendix B the slightly shortened version of the working instructions for the macro mSHELL2TOC. Here we present the abbreviated version of the full guidance document.



Zebra Pharmaceuticals → → Page x of xx
Protocol ZP-TCNPC-101 Final Analysis

Table 14.1.2
Subject Disposition by Treatment
Randomized Population

	1500 mg/kg (N=xx)	2500 mg/kg (N=xx)	Total (N=xx)
Subjects Included in:			
Safety Population	xx	xx	xx
ITT Population	xx	xx	xx
PP Population	xx	xx	xx
PK Population	xx	xx	xx
PD Population	xx	xx	xx
Subject Status [1]			
Completed Study	xx (xx.x)	xx (xx.x)	xx (xx.x)
Discontinued	xx (xx.x)	xx (xx.x)	xx (xx.x)
Ongoing [2]	xx (xx.x)	xx (xx.x)	xx (xx.x)
Reason for Discontinued [3]			
Adverse Event	xx (xx.x)	xx (xx.x)	xx (xx.x)
Death	xx (xx.x)	xx (xx.x)	xx (xx.x)
Lost to Follow-up	xx (xx.x)	xx (xx.x)	xx (xx.x)
Physician Decision/PI Discretion	xx (xx.x)	xx (xx.x)	xx (xx.x)
Progressive/Worsening Disease	xx (xx.x)	xx (xx.x)	xx (xx.x)
Protocol Violation	xx (xx.x)	xx (xx.x)	xx (xx.x)
Withdrawal by Subject/Patient/Parent/GGuardian	xx (xx.x)	xx (xx.x)	xx (xx.x)
Other [4]	xx (xx.x)	xx (xx.x)	xx (xx.x)

Section Break (Next Page)

[1] Percentages are based on the total number of randomized subjects in each treatment arm.
[2] Ranomized subjects who have neither completed nor withdrawn yet as of the cutoff date.
[3] Percentages based on the total number of discontinued study subjects in each treatment arm.
[4] 'Other' includes Pregnancy, Recovery, Study Terminated by Sponsor, Technical Problems or Other.
Reference: Listing 16.1.3.1

Program: outid.sas → Programmer: xxx ddmmmyyyy hh:mm SAS9.4

Figure 1. Typical shell for disposition table. Red color marks title (header in Word document) and footnotes (footer in Word document), green color denotes part of the header and footer that are unique for this table. Yellow color marks part of the title that is repeated (identically) in every output (protocol number, legal name of the client, etc.). Light blue dotted line depicts part of the footnotes that is automatic and do not require comparison with the original shell document.

Title and footnotes of the projected TLF should be entered as header and footer of the WORD document, correspondingly (refer to Figure 1 below).
Every shell must be confined within its own Section and is separated from other shells by Section Break.

1. The general information appearing in the header of every one of the outputs (e.g., company name, protocol number, etc.) must be the same in every one of the shells.
2. Title of the output is presumed to contain two, three or more different parts located on the separate lines - output type and number (e.g., Table 14.1.1, Listing 16.2.1, Figure 14.2.1.1), title of the output (e.g., “Subject Disposition and Analysis Population by Treatment Group”), and population definition (e.g., “Safety Population”).
3. 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).
4. Instructions to use the same layout for another output (so called repeated TLF) are normally presented in the part of the shell called “Notes to programmers” and must start with the words “Repeat this Table”.

CREATION OF TOC FROM SPECIFICATIONS FOR SHELLS

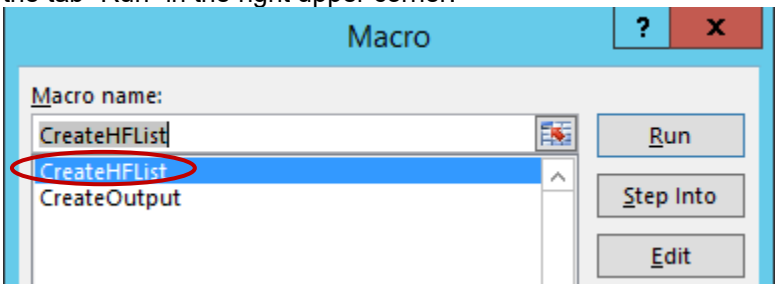
As it was shortly described above during the first step of the automated process the specifications for shells are read and corresponding Table of Contents is created in the form of Excel spreadsheet. Remind, the created TOC is SAS®-readable – refer to the previous publication of the authors (Goldfarb and Zelichonok, 2020).

Here is a step by step instruction how this part of the macro should be run and how its results should be interpreted.

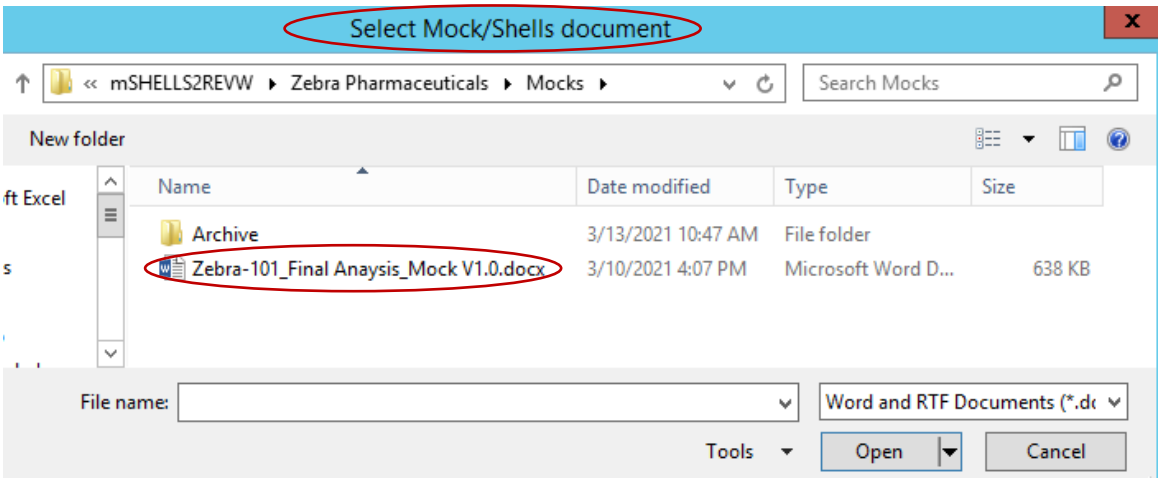
1. Open EXCEL file where the macro is residing. One might want to clean up the Sheet 1, i.e. erase everything that is probably remaining there from previous exercises (macro will do it anyway).

2. Press the Tab “Developer”, go the left side of the bar and press the tab “Macros”.

3. When a small window opens please make sure that you see names of two internal macros in the Macro Name field - “CreateHFList” and “CreateOutput”. Choose “CreateHFList” first, then press the tab “Run” in the right upper corner.



4. The file dialog window (entitled “Select Mock/Shells Document”) opens where only Word files are shown. Select an appropriate file and press the tab “Open”



5. Macro CreateHFList 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. Note, that the macro changes the names of the sheet from “Sheet 1” to “Shells”.

6. Check the content of the sheet named Shells – it now contains TOC that was created based on the shells that were developed in appropriate way and combined all together in the LiveMOCK document (practice of Accenture Life Sciences). Make sure that a content of TOC is corresponding to the source file – LiveMOCK.

7. Structure of TOC. TOC represents an Excel file with meaningful information in the first 3 columns – A, B and C (see Fig 2 below as an example).
    a. 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 the information more visible and clear.
    b. 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. 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 (no meaningful footnotes) to 9.
    d. 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.

	A	B	C
1	_ALL_	TITLE1	Zebra Pharmaceuticals
2	_ALL_	TITLE2	Protocol ZP-TCNPC-101
3	_ALL_	TITLE3	Final Analysis
4			
5	T140101	TITLE4	Table 14.1.1
6	T140101	TITLE5	All Subjecs Screened and Reasons for Screen-Failure
7	T140101	TITLE6	Screened Population
8	T140101	FOOTNOTE1	Reference: Listing 16.1.3.1
9			
10			
11	T140102	TITLE4	Table 14.1.2
12	T140102	TITLE5	Subject Disposition by Treatment
13	T140102	TITLE6	Randomized Population
14	T140102	FOOTNOTE1	[1] Percentages are based on the total number of randomized subjects in each treatment arm.
15	T140102	FOOTNOTE2	[2] Ranomized subjects who have neither completed nor withdrawn yet as of the cutoff date.
16	T140102	FOOTNOTE3	[3] Percentages based on the total number of discontinued study subjects in each treatment arm.
17	T140102	FOOTNOTE4	[4] 'Other' includes Pregnancy, Recovery, Study Terminated by Sponsor, Technical Problems or Other.
18	T140102	FOOTNOTE5	Reference: Listing 16.1.3.1
19			
20			
21	T140105	TITLE4	Table 14.1.5
22	T140105	TITLE5	Demographics Characteristics by Treatment
23	T140105	TITLE6	Safety Population
24	T140105	FOOTNOTE1	Note: Percentages are based on the total number of subjects at safety population in each treatment arm.
25	T140105	FOOTNOTE2	Reference: Listing 16.1.1
26			
27			
28	T14010501	TITLE4	Table 14.1.5.1
29	T14010501	TITLE5	Baseline Demographics Characteristics by Treatment
30	T14010501	TITLE6	Safety Population
31	T14010501	FOOTNOTE1	Note: Percentages are based on the total number of subjects at safety population in each treatment arm, otherwise specified. PBMCs = Peripheral Blood
32	T14010501	FOOTNOTE2	Mononuclear Cells.
33	T14010501	FOOTNOTE3	Reference: Listing 16.1.1
34			

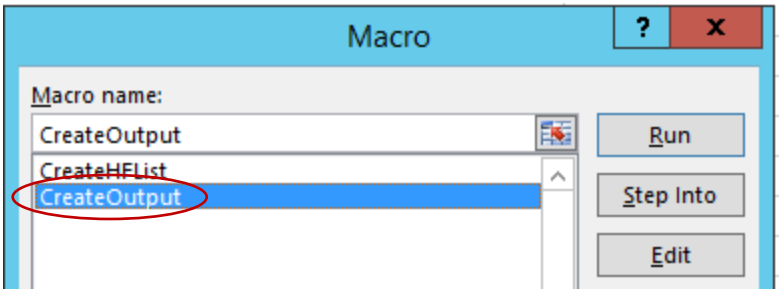
Figure 2. Typical SAS© ready TOC created from the shells document.

READING TITLES AND FOOTNOTES FROM ACTUAL OUTPUTS

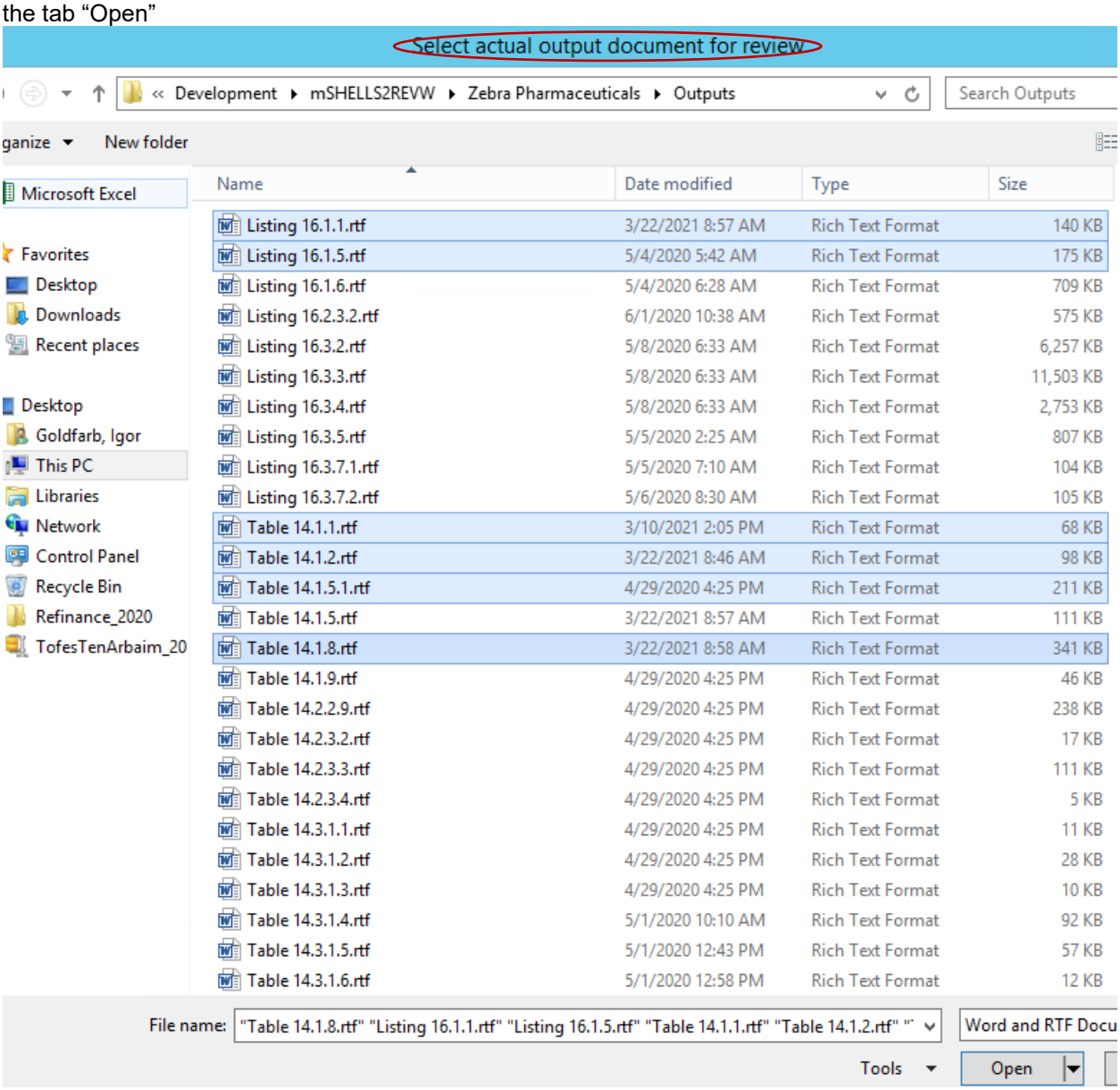
Time for the second stage of the process comes when internal macro "CreateHFList" finishes its work and creation of TOC is completed. During this step the macro mSHELLvsTLF reads titles and footnotes from the actual tables, listings and figures that were produced by the statistical programmers and should be reviewed to verify that all titles and footnotes exactly correspond to ones specified in the shell document LiveMOCK.

A short guidance below can help potential users of the macro

1. Move cursor to the second line in the Macro window to highlight internal macro "CreateOutput". Press the tab "Run" in the right upper corner.



2. The file dialog window (entitled “Select actual output document for review”) opens where only Word files are shown. Select all appropriate files that are going to be reviewed this time and press the tab “Open”



3. Make sure that a new sheet was opened and all TLF that were chosen for review are present in the list and columns D and E in the new sheet are filled in with the text that was copied from the titles and footnotes of the actual TLF (Table 14.1.1 in the screenshot below).

	A	B	C	D	E
28	P:\ClinStats\Igor Goldfarb\Development\mSHELLS2REVW\Zebra Pharmaceuticals\Outputs\Table 14.1.1.rtf				
29					
30	_ALL_	TITLE1	Zebra Pharmaceuticals	TITLE1	Zebra Pharmaceuticals
31	_ALL_	TITLE2	Protocol ZP-TCNPC-101	TITLE2	Protocol ZP-TCNPC-101
32	_ALL_	TITLE3	Final Analysis	TITLE3	Final Analysis
33	T140101	TITLE4	Table 14.1.1	TITLE4	Table 14.1.1
34	T140101	TITLE5	All Subjecs Screened and Reasons for Screen-Failure	TITLE5	All Subjecs Screened and Reasons for Screen-Failure
35	T140101	TITLE6	Screened Population	TITLE6	Screened Population
36	T140101	FOOTNOTE1	Reference: Listing 16.1.3.1	FOOTNOTE1	Reference: Listing 16.1.3.1

ANALYSIS AND REVIEW – ACTUAL VS SPECS

The final, third stage of the process starts when one has both internal macros completed. It means that the EXCEL spreadsheet contains full TOC built strictly in accordance with the provided specifications (shell document – LiveMOCK) and all titles and footnotes that were extracted from the actual TLF (selected for review) produced by the programmers. The differences are marked in a number of ways and are self-explaining. For example, if a footnote is present in specs, but is missing in actual output – corresponding cell will contain red error statement “missing in output”. If vice versa – “missing in shell” of the same red color will appear.

To illustrate how the results of the review and comparison are displayed let’s consider a couple of examples that were copied from the actual projects and used in the TLF produced for the hypothetical client Zebra Pharmaceuticals.

1. Example #1 – Listing 16.1.1. One can easily see several of distinctions that were uncovered as a result of macro run.
    a. There is a line (Title7) with the text “Treatment = 1500 mg/kg (n=6)” that was not presented in the shell document (dark red dashed line). It can be easily explained by the fact that it was displayed as a part of the regular body text of the WORD document in the SPEC subdivision by treatment groups, not its header. When the programmers produced the listing the standard macro put “by statement” into the title of the output .rtf document and it was read by the macro in this capacity. Reviewer’s conclusion – it is OK, no updates in the program required.
    b. Footnotes 1, 2 and 3 do not match each other, but it is the same issue. An elementary analysis shows that the footnote [2] was not copied in the proper way from the shell and it caused the issue. The long text is split in both versions, but it is done differently. following the specifications, where the footnote [2] is split into two lines with the text in the second one – “M=Multiple, NC=Not Collectable (Local Regulatory Req)”). Reviewer’s conclusion – it is OK, no updates in the program required, but it is recommended to use macro mSHELL2TOC (Goldfarb and Zelichonok, 2020) to guarantee the proper correspondence between specifications and TOC used for TLF production.

	A	B	C	D	E
1	P:\ClinStats\Igor Goldfarb\Development\mSHELLS2REVW\Zebra Pharmaceuticals\Outputs\Lis			From Shells:	
2					
3	_ALL_	TITLE1	Zebra Pharmaceuticals	TITLE1	Zebra Pharmaceuticals
4	_ALL_	TITLE2	Protocol ZP-TCNPC-101	TITLE2	Protocol ZP-TCNPC-101
5	_ALL_	TITLE3	Final Analysis	TITLE3	Final Analysis
6	L160101	TITLE4	Listing 16.1.1	TITLE4	Listing 16.1.1
7	L160101	TITLE5	Demographics Characteristics	TITLE5	Demographics Characteristics
8	L160101	TITLE6	Safety Population	TITLE6	Safety Population
9	L160101	TITLE7	Treatment = 1500 mg/kg (N=6)	TITLE7	Missing in Shell
10	L160101	FOOTNOTE1	[1] M=Male. F= Female.	FOOTNOTE1	[1] M=Male. F= Female.
11	L160101	FOOTNOTE2	[2] W=White. B=Black or African American. A=Asian. AI=American Indian or Alaska Native. NH=Native Hawaiian or Other Pacific Islander. O=Other. M=Multiple. NC=Not collectable (Local Regulatory Req).	FOOTNOTE2	[2] W=White. B=Black or African American. A=Asian. AI=American Indian or Alaska Native. NH=Native Hawaiian or Other Pacific Islander. O=Other.
12			Missing in Output	FOOTNOTE3	M=Multiple. NC=Not collectable (Local Regulatory Req).
13					

2. Example #2 – Table 14.1.8. One easily notices that the footnotes 1 and 2 are marked by red here. From the first glance it looks as two pairs of footnotes are identical on both specs and productions sides. More attentive view, however, allows us to discover the conjunction word “and” that is residing in different places – in the shell document it belongs to FOOTNOTE1, while in the production version this word sits in FOOTNOTE2. Reviewer’s conclusion – it is OK, no updates in the program required, but it is recommended to use macro mSHELL2TOC (Goldfarb and Zelichonok, 2020) to guarantee the proper correspondence between specifications and TOC used for TLF production.

	A	B	C	D	E
64	P:\ClinStats\Igor Goldfarb\Development\mSHELLS2REVW\Zebra Pharmaceuticals\Outputs\Table 14.1.8.rtf				
65					
66	_ALL_	TITLE1	Zebra Pharmaceuticals	TITLE1	Zebra Pharmaceuticals
67	_ALL_	TITLE2	Protocol ZP-TCNPC-101	TITLE2	Protocol ZP-TCNPC-101
68	_ALL_	TITLE3	Final Analysis	TITLE3	Final Analysis
69	T140108	TITLE4	Table 14.1.8	TITLE4	Table 14.1.8
70	T140108	TITLE5	Concomitant Medications by Treatment	TITLE5	Concomitant Medications by Treatment
71	T140108	TITLE6	Safety Population	TITLE6	Safety Population
72	T140108	FOOTNOTE1	Note: The World Health Organization Drug Dictionary (WHO-DD) Version in December 2016 was used. At each level (WHO-DD ATC Class Level 3	FOOTNOTE1	Note: The World Health Organization Drug Dictionary (WHO-DD) Version in December 2016 was used. At each level (WHO-DD ATC Class Level 3 and
73	T140108	FOOTNOTE2	and Preferred Term) of summarization, a patient was counted once if the patient reported one or more medications.	FOOTNOTE2	Preferred Term) of summarization, a patient was counted once if the patient reported one or more medications.
74	T140108	FOOTNOTE3	Reference: Listing 16.1.6	FOOTNOTE3	Reference: Listing 16.1.6
75					

3. Example #3 – Table 14.3.4.3. It is one of the simplest cases – the comparison clearly states that the title of the table is missing in the specifications. It is a reflection of the situation (very familiar to the statisticians and lead programmers working in the industry) when the requirement for this table was added at the later stage (*ad hoc* - when most of the outputs and shell document were already ready), the output was programed and produced, but the shell document was not updated appropriately. Reviewer’s conclusion – the shell document must be updated to keep 1-to-1 correspondence between LiveMOCK and actual list of produced TLF.

	A	B	C	D	
469	P:\ClinStats\Igor Goldfarb\Development\mSHELLS2REVW\Zebra Pharmaceuticals\Outputs\Table 14.3.4.3.rtf				
470					
471	_ALL_	TITLE1	Zebra Pharmaceuticals	TITLE1	Zebra Pharmaceuticals
472	_ALL_	TITLE2	Protocol ZP-TCNPC-101	TITLE2	Protocol ZP-TCNPC-101
473	_ALL_	TITLE3	Final Analysis	TITLE3	Final Analysis
474	T14030403	TITLE4	Table 14.3.4.3	TITLE4	Missing in Shell
475	T14030403	TITLE5	Incidence of Post-baseline Urinalysis Parameters by Treatment	TITLE5	Missing in Shell
476	T14030403	TITLE6	Safety Population	TITLE6	Missing in Shell
477	T14030403	FOOTNOTE1	Reference: Listing 16.3.4	FOOTNOTE1	Missing in Shell
478					

DISCUSSION

Accenture Life Sciences accrued some experience of practical application of the developed macro mSHELLvsTLF. The accumulated practice taught us some lessons. Some of the lessons leant are worth to be shared with the prospective users of the macro.

The main conclusion is that using the proposed macro essentially reduces amount of efforts required to verify that the final outputs are produced strictly in accordance with the shells. As every professional working in the pharmaceutical industry knows the approved shells are undergoing numerous updates according to the comments of all participants of the CSR development. In many cases footnotes are revised and/or updated to explain specific details of a calculation and to make potential reviewers' work easier. In some cases the titles are altered too to reflect the content of the output more correctly or in more details or to ensure following some company's standards. The macro allows to perform such verification in seconds and to immediately provide appropriate comments to statistical programmers.

Another outcome from our experience in application of the proposed macro coincides with one that was made earlier regarding macro mSHELL2TOC (Goldfarb and Zelichonok, 2020). There is no surprise in this "accidental" coincidence – both macros are using the same type of the shells document, macros' internal logic heavily relies on the presumed structure of the document and they are very sensitive to the issues in the mock file. Therefore, LiveMOCK (the MS Word shell document) should be properly and carefully prepared strictly in accordance with the rules and instructions mentioned above. Development of the shell document while strictly following the guidance's rules can be little more time-consuming than free-written document, but this small investment in the project in its very beginning pays itself off tens times when a biostatistician or lead programmer has to review the produced outputs before delivery. Any review of this type must include verification of exact correspondence of the titles and footnotes of the actual TLF to what was prescribed in the shells.

Authors will be thankful for any reference to publication/blog devoted to the work in similar direction.

FURTHER PERSPECTIVE

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. 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 mSHELLvsTLF can be further improved and to be used widely to automate the review process and to save time and efforts for numerous statisticians, lead programmer and those who face the similar task in their professional routine.

SHORT TERM TASKS

An experience accumulated by Accenture allows the authors to formulate the list of problems that are simply technical by nature and can be considered as short-term tasks. As an example one can mention appearance in the titles or footnotes Greek letters, subscripts, superscripts, special symbols, symbols of weak inequalities, etc. The developed macro does not distinguish regular ASCII symbols and superscript, for example. Therefore, when the units of Body Mass Index (BMI) are correctly defined in the shells as

"kg/m^2", the macro reads them as "kg/m2" (which is obviously incorrect) and adds them in this way to the sheet Shells. If the produced table displays the BMI units correctly then the macro will compare "kg/m^2" vs "kg/m2", identify the distinction between shell and the output and mark the corresponding lines by red color. This false negative response will require manual review by the lead programmer or statistician.

These types of problems are mostly technical and they do not require to alter main logic standing behind the structure of the macro and their main algorithms. Expected resolution of these issues require addition to the developed macro specific technical updates that will treat the special symbols in separate way, more diligent than how regular ASCII symbols are processed by VBA. Their resolution will make the proposed macro more universal and flexible.

FURTHER PERSPECTIVE - INTIENT

As the biopharma landscape grows and evolves, life sciences industry leaders are looking for and finding new opportunities to raise the standard and personalization of patient care and to make every stage of pharmaceutical research more effective.

Accenture Life Sciences (and many of our clients) are switching to INTIENT – a platform that can connect lab results, clinical trial data, patient safety reports, and regulatory submissions to patient information. INTIENT is a global platform powered by Google Cloud that, along with innovative technologies, helps customers to effectively apply automation, advanced analytics, artificial intelligence and machine learning, as well as manage their data.

INTIENT Clinical is one of the INTIENT product suites, which was developed with the purpose of helping the world's leading biopharmaceutical companies run faster clinical trials with better transparency and easier access to trial information. It is planned that all conversions of the raw data into SDTM domains, development of ADaM datasets, production of TLF and other typical steps will be conducted within this new environment.

The authors expect that MS Office installed in INTIENT Clinical will have some peculiarities that will differ from the version in use now. These distinctions will be analyzed and the macro will be re-written or updated accordingly (depending of a scale of the differences).

CONCLUSIONS

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

1. The developed macro mSHELLvsTLF automates a review of titles and footnotes in the produced TLF versus their shells.
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 developed macro generates an Excel sheet (named Shells) containing ordered TOC.
4. The suggested macro reads the actual TLF that require the review (it can be a single output or multiple files). Titles and footnotes are generated in the similar way in another Excel sheet.
5. The proposed macro takes the set of titles and footnotes of the tables under review (that were read from the actual TLF) and compare them (using on universal numbering) with ones that were

generated while reading specification document.

6. The distinctions uncovered during the analysis are analyzed and marked using the color code. Colored parts require additional manual review, unmarked parts mean coincidence between the shells and the actual outputs.
7. Any further TLF for review can be read (separate Excel sheet will be opened) without necessity to run through the specification document and to generate the sheet Shells again. And it takes literally seconds!

REFERENCES

Goldfarb, I., and Zelichonok, E., (2020), Macro To Produce SAS®-Readable Table of Content From TLF Shells, *Proceedings of the PharmaSUG 2020*, Paper AD-106.

ACKNOWLEDGMENTS

The authors are very thankful to upper management of Accenture Life Sciences and Naxion, correspondingly, for their 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 Life Sciences
Igor.goldfarb@accenture.com

Ella Zelichonok
Naxion
ezelichonok@naxionthinking.com

APPENDIX A – TEXT OF THE MACRO

```
  'Macro mSHELLvsTLF
'This macro reads the shells for project TLF (MS Word file), creates ordered TOC in the form of EXCEL spreadsheet, reads the titles and footnotes from the actual outputs that were chosen for review (another Excel sheet), compare the titles and footnotes from the actual TLF with their counterparts from the shells, analyzes the differences and marks the distinctions using color code.

Dim W As Word.Application

Sub CreateHFList()
'
    Dim File_Name As String
    Dim sOutputFile As Variant
    Dim ShellSheet As Worksheet

    Set W = CreateObject("Word.Application")

    W.DisplayAlerts = wdAlertsNone

    '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 and RTF Documents", "*.doc*;*.rtf", 1
    Application.FileDialog(msoFileDialogOpen).Title = "Select Mock/Shells document"

   'Supress Alerts
   Application.DisplayAlerts = False



    'Show file dialog
    If Application.FileDialog(msoFileDialogOpen).Show Then
        'look through the sheets to see if sheet "shells" exist
        For i = 1 To Sheets.Count
            If LCase(Sheets(i).Name) = "shells" Then
                Set ShellSheet = Sheets(i)
                ShellSheet.Activate
                Exit For
            End If
        Next
        'Create Shells sheet if not found
        If ShellSheet Is Nothing Then
            Sheets.Add.Select
            ActiveSheet.Name = "Shells"
        End If

        'Select the file from the dialog
        File_Name = Application.FileDialog(msoFileDialogOpen).SelectedItems(1)

        'Activate Shells sheet
        Cells.ClearContents
        ActiveSheet.Cells(1, 1).Select
```

```
        'Call function that creates Shells spreadsheet
        CreateHFListFromDoc File_Name, True

        'Calls function that examines and reports changes in output
        CreateOutput



    End If
    'Quit Word application
    W.Quit
    Application.DisplayAlerts = True
End Sub

Sub CreateOutput()
        Dim ShellSheet As Worksheet
        Dim OutputSheet As Worksheet
        Dim NeedToCloseWord As Boolean

        'Supress erroneous errors
        On Error Resume Next
        Err = 0

        'Supress alerts
        W.DisplayAlerts = False

        'Initialize Word Application
        If W Is Nothing Or Err > 0 Then
            Set W = CreateObject("Word.Application")
            NeedToCloseWord = True
        End If

        'Activate error handling
        On Error GoTo 0

        'Find Shells spreadsheet
        For i = 1 To Sheets.Count
            If LCase(Sheets(i).Name) = "shells" Then
                Set ShellSheet = Sheets(i)
                Exit For
            End If
        Next

        'if shells spreadsheet is not found, display message and call CreateHFList function that will create
Shells spreadsheet
        If ShellSheet Is Nothing Then
            If MsgBox("Need to create Shells sheet first by running CreateHFList macro! Do you want to run
it?", vbQuestion Or vbYesNo) = vbYes Then
                CreateHFList
            End If
            Exit Sub
        End If

        'THIS PORTION IS TO SELECT output FILES - Multi select
         Application.FileDialog(msoFileDialogOpen).AllowMultiSelect = True
        'Filter WORD and RTF documents only to be shown for user to choose from
```

```
    Application.FileDialog(msoFileDialogOpen).Filters.Add "Word and RTF Documents", "*.doc*;*.rtf", 1
    Application.FileDialog(msoFileDialogOpen).Title = "Select actual output document for review"
    'Show file dialog
    If Application.FileDialog(msoFileDialogOpen).Show Then
      'Create new spreadsheet that will contain output listing
      ActiveWorkbook.Sheets.Add.Select
      Set OutputSheet = ActiveSheet
      'loop through all the selected files
      For Each sOutputFile In Application.FileDialog(msoFileDialogOpen).SelectedItems
        Selection.Value = sOutputFile
        'create list of Tables, Listings and Figures for a specific output file
        CreateHFListFromDoc CStr(sOutputFile), False
      Next

      'Merge results of Shells and Output listings
      mergeResults ShellSheet, OutputSheet
    End If

    'Close Word
    If NeedToCloseWord Then W.Quit

End Sub

'This function merges results found in Shells spreadsheet and Output spreadsheet identifying the
differences in Red
Sub mergeResults(ShellSheet As Worksheet, OutputSheet As Worksheet)
  Dim collKeys As New Collection
  Dim collRows As New Collection
  Dim collValues As New Collection
  Dim sKey As String
  Dim sValue As String

  'Manually handle errors
  On Error Resume Next

  'looping through all the rows
  For i = 1 To ShellSheet.Cells.SpecialCells(xlLastCell).Row
    If ShellSheet.Cells(i, 2).Value <> "" Then
      'Identifying key and value for the collection
      sKey = ShellSheet.Cells(i, 1).Value & "_" & ShellSheet.Cells(i, 2).Value
      sValue = ShellSheet.Cells(i, 3).Value

      Err = 0
      sKey = collKeys(sKey)
      If Err > 0 Then
        'appending to the collection of Keys, rows and vlaues
        collKeys.Add sKey, sKey
        collRows.Add i, sKey
        collValues.Add sValue, sKey
      End If
      Err = 0
    End If

  Next i

  'Identifying the last row
```

```
LastRow = OutputSheet.Cells.SpecialCells(xlLastCell).Row
i = 1
Do

    If OutputSheet.Cells(i, 2).Value <> "" Then
        'Table name is in the 1st column
        sTable = OutputSheet.Cells(i, 1).Value

        If OutputSheet.Cells(i, 2).Value Like "TITLE*" Then
            nLastTitle = Val(Replace(OutputSheet.Cells(i, 2).Value, "TITLE", ""))
            rowLastTitle = i
        ElseIf OutputSheet.Cells(i, 2).Value Like "FOOTNOTE*" Then
            nLastFN = Val(Replace(OutputSheet.Cells(i, 2).Value, "FOOTNOTE", ""))
            rowLastFN = i
        End If

        'form key as concatinated values in clumn 1 and 2
        sKey = OutputSheet.Cells(i, 1).Value & "_" & OutputSheet.Cells(i, 2).Value
        'value is in column 3
        sValue = OutputSheet.Cells(i, 3).Value

        mockValue = ""
        mockValue = collValues(sKey)
        'If mockValue is not found in collection then it's missing in Shell
        If mockValue = "" Then mockValue = "Missing in Shell"
        OutputSheet.Cells(i, 4).Value = OutputSheet.Cells(i, 2).Value
        OutputSheet.Cells(i, 5).Value = mockValue
        'If values are not matching, color code in Red
        If mockValue <> sValue Then
                OutputSheet.Cells(i, 4).Font.Color = -16776961
                OutputSheet.Cells(i, 5).Font.Color = -16776961
        End If
    ElseIf i > 1 And sTable <> "" And sTable <> "_ALL_" Then

        Do
            mockValue = ""
            nLastTitle = nLastTitle + 1
            rowLastTitle = rowLastTitle + 1
            sKey = sTable & "_TITLE" & nLastTitle
            mockValue = collValues(sKey)

            If mockValue = "" Then Exit Do
            'Inserting Missing in Output values and color-code it in red
            Rows(rowLastTitle).Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
            Cells(rowLastTitle, 4).Value = "TITLE" & nLastTitle
            Cells(rowLastTitle, 5).Value = mockValue
            Cells(rowLastTitle, 3).Value = "Missing in Output"
            Cells(rowLastTitle, 3).Font.Color = -16776961

            rowLastFN = rowLastFN + 1
            LastRow = LastRow + 1
        Loop
        Do
            mockValue = ""
            nLastFN = nLastFN + 1
            rowLastFN = rowLastFN + 1
```

```
            sKey = sTable & "_FOOTNOTE" & nLastFN
            mockValue = collValues(sKey)

            If mockValue = "" Then Exit Do
            'Inserting Missing in Output values and color-code it in red
            Rows(rowLastFN).Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
            Cells(rowLastFN, 4).Value = "FOOTNOTE" & nLastFN
            Cells(rowLastFN, 5).Value = mockValue
            Cells(rowLastFN, 3).Value = "Missing in Output"
            Cells(rowLastFN, 3).Font.Color = -16776961
            LastRow = LastRow + 1
          Loop
        End If
        i = i + 1
        'looping to the last row
    Loop Until i > LastRow


    'Adjusting Column widths
    Columns(4).ColumnWidth = 16
    Columns(5).ColumnWidth = 60
    'Allowing text wrap for better viewing
    Columns(5).WrapText = True
    Cells(1, 4).Value = "From Shells:"
End Sub


'This function creates listing of tables/listings/figures from specified File (File_Name)
Sub CreateHFListFromDoc(File_Name As String, AllSecs As Boolean)


Dim dc As Document
Dim OBJSection As Section
Dim OBJHF As HeaderFooter


Set dc = W.Documents.Open(File_Name, , True)



Dim HdrRange As Word.Range

Dim r0 As Integer
Dim r As Integer

Dim arrFirstHeader
Dim nStart, rFirstHeaderRow

Dim nFooterEnd
Dim SectionText
Dim iRepeat
Dim Repeats As New Collection

Dim k0 As Integer
Dim SecText As String
nSections = k0
```

```
r0 = Selection.Row

r = r0

    If AllSecs Then
    'Do it only for Mock Shell
    '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 for programmers", vbTextCompare) = 0 And SecText <> "" Then
        If InStr(1, SecText, "general notes", vbTextCompare) = 0 And SecText <> "" Then

            Exit For
        End If
    Next k
    nSections = dc.Sections.Count
Else
    k0 = 1
    nSections = 1
End If

For k = k0 To nSections
    ' 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


    If AllSecs Then
        'Do it only for Mock Shell
        'empty out repeat lines collection
        For l = Repeats.Count To 1 Step -1
            Repeats.Remove (l)
        Next

        'identifying sections to repeat
        'it will always start with "Repeat this"
        iStartCopyRow = 0 'this will hold beginning of the excel row where section repeats
        iRepeat = 0
        iRepeat = InStr(iRepeat + 1, SectionText, "repeat this ", 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 ", vbTextCompare)
    Loop
End If

'header Range
Set HdrRange = dc.Sections.Item(k).Headers(wdHeaderFooterPrimary).Range


'Identifying elements of the header
Dim arr, arr2
arr = Split(HdrRange.Text, vbCr)
arr = CleanLines(arr)
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)
        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 0 'UBound(arr2)

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



'Footer Range
Dim FtrRange As Word.Range

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


If FtrRange.Text <> "" Then
    arr = Split(FtrRange.Text, vbCr)
```

```
    arr = CleanLines(arr)

Else
    Exit For
End If


'Looping through Footer items
For i = 0 To UBound(arr)
    If arr(i) <> "" Then
        arr2 = Split(arr(i), vbTab)
        r = r + 1

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



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

'assigning tags in a special format
If AllSecs Then r0 = 1
AssignTags r0, r

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

'formatting
Columns(1).ColumnWidth = 16
Columns(2).ColumnWidth = 16
Columns(3).ColumnWidth = 60
Columns(3).WrapText = True
```

```
If AllSecs = False Then ActiveSheet.Cells(r + 1, 1).Select
dc.Close 0

End Sub


Function CleanLines(arr)
'This function will remove Empty lines and those containing outid.sas from array and return array of all
non-empty lines
   Dim coll As New Collection
   For i = 0 To UBound(arr)
      arr(i) = Replace(arr(i), "", vbTab) 'remove bullet
      arr(i) = Replace(arr(i), Chr(7), vbTab) 'remove bullet
      arr(i) = Trim(arr(i))
      arr(i) = Replace(arr(i), "     ", vbTab)
      For Each s In Split(arr(i), vbTab)
         If s <> "" And InStr(LCase(s), "outid.sas") = 0 And _
            InStr(LCase(s), "program:") = 0 And _
            InStr(LCase(s), "programmer:") = 0 And _
            InStr(LCase(s), "sas 9.4") = 0 And _
            InStr(LCase(s), "sas9.4") = 0 And _
            InStr(s, "Page ") <> 1 Then

               If s <> "" Then coll.Add Trim(s)

         'coll.Add (arr(i))
         End If
       Next
   Next i

   If coll.Count = 0 Then
      ReDim arrReturn(0)
   Else
      ReDim arrReturn(coll.Count - 1)
      For i = 0 To UBound(arrReturn)
         arrReturn(i) = coll(i + 1)
      Next
   End If

   CleanLines = arrReturn
End Function

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 List, 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
    ActiveSheet.Sort.SortFields.Add Key:=Range(Cells(r1, 2), Cells(r2, 2)) _
        , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveSheet.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
sTag = "_ALL_"

'Assigns values to parts of titles that are unique for every output
For i = nStart To nEnd
    If InStr(Cells(i, 3).Value, "Table ") = 1 Or _
        InStr(Cells(i, 3).Value, "Listing ") = 1 Or _
        InStr(Cells(i, 3).Value, "Figure ") = 1 Then
            'Uses function that will parse the table/figure/listing title
            sTag = ProcessTag(Cells(i, 3).Value)
    End If
    If Cells(i, 3).Value <> "" Then Cells(i, 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
```

APPENDIX B – REUIREMENTS FOR PREPARATION OF SHELL DOCUMENT

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. Description of the full set of details can be found in the previous publication by the authors (Goldfarb and Zelichonok, 2020). For reader's convenience we added to the Appendix B the shortened version of the working instructions for the macro.

The guidance below briefly describes the first step of the macro progress - how the shell document is transformed into the Table of Contents. In other words, how the titles and footnotes from the mock file are transferred to the appropriate lines in EXCEL spreadsheet where TOC is residing.

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

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: ADXX 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".