PharmaSUG 2023 - Paper AP-309

Macro to Automate Crossover Review in Produced Outputs

Igor Goldfarb, Accenture Life Sciences;
Ella Zelichonok, Naxion

ABSTRACT

The goal of this work is to develop a macro that automates an important and time-consuming part of a final review process of produced tables, listings, and figures (TLF) – crossover review. Performing this type of validation is a well-accepted practice typically performed manually by biostatisticians as the final stage in the multi-step quality check (QC) process. The proposed tool can significantly simplify review work for biostatisticians who have to verify that TLF were generated correctly, and they are consistent across the study and its different sections.

Final review of the produced TLF represents an important task in a flow of raw data to final outputs ready for submission. Comparison, analysis and making sure that the actual outputs are all consistent across the study is a tedious procedure requiring scrupulous manual work that is subject to human errors.

The proposed macro (developed in Excel VBA) automates this process. It reads the titles and corresponding content of the produced outputs (e.g., big N - values of safety population by study treatment group, by subgroup, etc.) and in a matter of seconds creates an ordered table of content (TOC, Excel) that includes also data of interest. At the next step the macro analyzes the read data and verifies that all produced outputs are consistent across the study (e.g., numbers of safety population in all outputs by age subgroup add up correctly to the corresponding values in study treatment groups). In case of any inconsistency found the macro marks the distinctions. Any further updates in the created TLF, can be easily reviewed another time by rerunning this macro.

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.

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 represents a standard approach in the pharmaceutical industry. Double programming presumes that two persons – production programmer and validation programmer – independently work on one output, compare the outcomes, analyze distinctions, investigate them, and eliminate the differences by uncovering an error and updating the original programs. This methodology allows to essentially reduce a number of human errors, but cannot eradicate them completely. For example, one group of programmers works on the safety outputs and uses term "Sex" in displaying results by this subgroup, while another group works on efficacy outputs,

and they use term “Gender” for the same purpose. Normal communication between the two groups rarely allows to uncover differences of this type. Therefore, 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 – so called cross over review when one has to verify that the delivery does not contain contradictory information (e.g., gender subdivision in one table is 50 males and 50 females, while in the other - 51 males and 49 females) and all terms are displayed in consistent way (e.g., gender in one set of tables and sex – in others).

The proposed initial version 1.0 of the macro mXREVIEW 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, 2021), devoted to development of other macros, namely mSHELL2TOC and mSHELLvsTLF. Macro mSHELL2TOC allows 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 developed macro mXREVIEW allows to automate cross-over review of the produced TLF across the whole study and to verify that all interconnected outputs are presented in the same way and all “children” outputs are in an absolute synchronization with their “parent” tables. It makes the final delivery more consistent and robust because it essentially reduces a need in manual tedious review of hundreds of outputs.

LITERATURE REVIEW

The idea to automate the cross-over review process is not new. Statisticians and lead programmers constantly face a need to perform this type of review across the study to make sure that delivery is consistent across all TLF and results presented in one portion of the outputs do not contradict displayed data in another section of the same set of TLF. Nevertheless, the authors found very limited number of papers devoted to the automation of the cross-over review of the produced outputs.

To address the challenge of automation, programmers made number of attempts in the past to develop a system that can allow to automate to some extent the review of the outputs. 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.

The most recent example of attempts in this direction is “Verify”, a machine learning (ML) based commercial tool developed by a company Beaconcure and presented in numerous publications during PharmaSUG-2022 (Donovan and Mayorov, 2022, Carmeli, 2022, Carmeli, Mayorov, and Donovan, 2022, Carmeli and Bar, 2022). “Verify” is able to perform both within-table and cross-table checks quickly and consistently for all deliverables. This can be achieved by running a set of standard cross-table checks defined by statisticians. The key to success in implementing automation is the combination of the human factor and ML. According to the developers “Verify” can greatly reduce the time and efforts to perform cross-over review of the tables.

Earlier Busa (2019) describes how they have used the power of SAS and TIBCO Spotfire®, to build "Interactive TLFs" using SDTM datasets to meet the predefined demands. The author demonstrated through a case-study how a clinical team can use their platform to review safety statistical outputs/TLFs (e.g., demographics, disposition, AEs, concomitant medications, laboratory and vital signs) more interactively and thereby to avoid flipping through hundreds if not thousands of static pages.

Malcolm (Malcolm, 2019) presented a paper where he outlines an approach to TFL automation that involved creation of the CDISC Analysis Results Metadata at the start of the process, not the end, and uses this metadata to generate the TFL. A SAS program structure is described that allows standard TFL to be created while also providing flexibility to easily incorporate study-specific analyses. The proposed approach allows to fully automate the generation of TLF and potentially reduce a need in cross-over review of the final outputs.

Authors realize the limited character of conducted literature research and understand that some valuable works could remain outside of it. Though the detailed literature overview is way outside of the scope of this paper, the authors will be thankful for any reference to publication/blog devoted to the work in similar direction.

OVERVIEW – HOW MACRO WORKS

The macro mXREVIEW was created using EXCEL Visual Basic for Applications (VBA), it resides in the regular Excel file and requires running the appropriate module within VBA environment. The macro reads the actual outputs (typically .rtf files, MS WORD) and performs a comparative analysis of the related or inter-connected results displayed in different tables. As the first step, the present version 1.0 of the macro mXREVIEW reads the table where baseline and demographic characteristics are normally displayed and creates a tab named “Baseline” in the output Excel file where all potential subgroups (e.g., age group, race group, gender, ECOG status, etc.) are combined as they are displayed in the source table. On the next step the macro runs all over the produced tables, finds the outputs that represent the results of the statistical analysis by the subgroups and compares corresponding numbers of subjects in subgroups and ways how these subgroups are displayed. Finally, any distinction between the values of populations appearing in the tables displaying the outcome of the data analysis by subgroups and the source (demographic and baseline characteristics) is reported (color-coded) in the second tab of the output Excel file (“Errors"). The files (displaying analysis by subgroup) where no errors were detected are collected in the third tab of the output Excel file ("Success”).

The authors will demonstrate an application of the proposed macro using a set of outputs developed for hypothetical company “Zebra Pharmaceutical, Inc.”. The output directory includes tables, listings and figures produced for this imaginary client.

The macro was developed within an existing environment of the MS Office 2016 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.

DEMOGRAPHIC AND BASELINE CHARACTERISTICS – THE SOURCE

To start running the macro one needs to direct it to the table where the demographic and baseline characteristics are tabulated. Typically, this table contains all basic subgroups (e.g., sex, race, age, etc.) and corresponding counts that are used later for planned subgroup analysis (e.g., ECOG status at baseline). As it was shortly described above, at the first step the macro mXREVIEW reads this table and extracts all the data that can define the subgroups and creates a tab named “Baseline” in the output Excel file. Figure 1 displays Table 14.1.2.1 “Demographic and Baseline Characteristics by Treatment Group” (1a) and corresponding tab “Baseline” in the resulting Excel file (1b).

Table 14.1.2.1
Demographics and Baseline Characteristics by Treatment Group
ITT Population

	Statistics	DL1 (N=6)	DL2 (N=3)	DL3 (N=6)	DL4 (N=10)	Overall (N=25)
Age at Baseline (years)	n	6	3	6	10	25
	Mean (SD)	63.2 (10.87)	60.0 (9.00)	64.5 (12.21)	57.9 (8.94)	61.0 (10.04)
	Median	65.5	60.0	69.5	57.5	61.0
	Min, Max	50, 75	51, 69	45, 76	43, 70	43, 76
Age Group at Baseline						
< 65 years	n (%)	3 (50.0)	2 (66.7)	2 (33.3)	7 (70.0)	14 (56.0)
≥ 65 years	n (%)	3 (50.0)	1 (33.3)	4 (66.7)	3 (30.0)	11 (44.0)
Sex						
Male	n (%)	2 (33.3)	1 (33.3)	5 (83.3)	7 (70.0)	15 (60.0)
Female	n (%)	4 (66.7)	2 (66.7)	1 (16.7)	3 (30.0)	10 (40.0)
Ethnicity						
Hispanic or Latino	n (%)	1 (16.7)	0	1 (16.7)	0	2 (8.0)
Not Hispanic or Latino	n (%)	0	0	2 (33.3)	7 (70.0)	9 (36.0)
Missing	n (%)	5 (83.3)	3 (100)	3 (50.0)	3 (30.0)	14 (56.0)

3	**Overall Count: 25**		
4			
5	Attribute	Category	Count
6	Age Group at Baseline		
7		< 65 years	14
8		≥ 65 years	11
9	Sex		
10		Male	15
11		Female	10
12	Ethnicity		
13		Hispanic or Latino	2
14		Not Hispanic or Latino	9
15		Missing	14
16	Race		
17		White	16
18		Black or African American	4
19		American Indian or Alaska Native	0
20		Asian	3
21		Hawaiian or Other Pacific Islander	0
22		Missing	2
23	Race Group		
24		White	16
25		Non-White	7
26		Missing	2
27	Smoke History		
28		Yes	15
29		No	6
30		Missing	4
31	ECOG Status at Baseline		
32		0	6
33		1	16
34		Missing	3

Figure 1. 1a (upper part) - typical table (first page) containing demographic and baseline characteristics displayed by treatment group. 1b (lower part) – tab "Baseline" where all subgroups and their corresponding number of subjects (total/overall) extracted from the source/baseline table are combined.

The tab “Baseline” in the resulting Excel file (Fig 1b) is self-explaining. The macro mXREVIEW extracts all parameters from the source table (Fig.1a) that can serve as subgroups and displays them in this tab. The first column contains appropriate attribute, the second one – legitimate categories. The last column represents subpopulations – Big “N” corresponding to every one of the categories. One can see that all continuous variables from the Table 14.1.2.1 were excluded, and only potential subgroups are included in this tab.

For example, the gender subgroup is displayed as “Sex” and it contains 15 male and 10 female subjects in total population (overall columns). It means that any table displaying the results of the analysis by gender will consist of two sections – one will start with “Sex: Male” (N=15 in Overall column), another – with “Sex: Female” (N=10 in Overall column). The same way of presentation should be followed in all outputs where the results are tabulated by gender. On the other hand, parameter “Race" most likely will not be used for subgroup analysis because of a non-balanced number of subjects (small number of subjects in all race categories, except “White”). Instead of this parameter “Race Group” can be used.

COMPARATIVE ANALYSIS OF THE OUTPUTS – “ERRORS”

On the second stage of the macro execution, it reads all existing outputs in the folder under analysis and compares their presentation with the baseline parameters and corresponding values of populations that were extracted from the source table (Demographic and Baseline Characteristics) and saved in the tab “Baseline” (Fig.1) of the resulting Excel file. The structure of the tab “Errors” is very intuitive and purpose of every column in this tab is crystal clear. The first two columns display output number and the attribute where a distinction between the baseline way of presentation and this output was detected. The next two columns depict baseline categories and their corresponding populations (Big “N”). Finally, the last two columns present the categories and their corresponding populations (Big “N”) as they were tabulated in the analyzed output. The color code allows to point to the category/value that are different from what was extracted from the source table.

To illustrate how the results of the review and comparison are displayed and analyzed let’s consider a set of generic outputs that were produced for the hypothetical client Zebra Pharmaceuticals, Inc. To make the process of the analysis and presentation of the results more visible some outcomes were altered, and various types of artificial errors were inserted manually.

1. Example #1 – Table 14.3.1.2.2 (Figure 2). One can easily see a distinction in a way how subgroup is displayed in this table – instead of “Sex: Xxxx” it reads as “Gender: Xxxx”, what is incorrect. Note that the number of males and females (Big N) are correctly displayed here.

1			Baseline		Reported	
2	File Name	Attribute	Categories	Overall N	Categories	Overall N
37	Table 14.3.1.2.2.RTF	Sex	Sex: Male	15	Gender : Male	15
38			Sex: Female	10	Gender : Female	10
39						

Zebra Pharmaceutical, Inc. Page 1 of 18
Zebra-301

Table 14.3.1.2.2
TEAE by System Organ Class and Preferred Term, by Treatment Group and Sex
Safety Population

Gender : Male

System Organ Class Preferred Term	DL1 (N=2) n (%)	DL2 (N=1) n (%)	DL3 (N=5) n (%)	DL4 (N=7) n (%)	Overall (N=15) n (%)

Figure 2. Outcome of the analysis of the Table 14.3.1.2.2. Upper part shows a record in the tab “Errors”, the lower part – screenshot of the actual table.

2. Example #2 - Table 14.2.6.1.5 (Figure 3). The red color clearly demonstrates differences between this output and values that were presented in the table for demographic and baseline characteristics. First, there are 4 categories reported in this output – in addition to three legitimate values of ECOG Status at baseline (“0”, “1”, “Missing”) one can see the fourth one – “2” and it is an error. Secondly, the values of Big N do not match – there are 6 and 16 subjects in subgroups with ECOG Status at baseline “0” and “1” correspondingly, but several subjects with missing ECOG Status at baseline is presented incorrectly in this table (3 is a correct value, while 1 is wrong).

			Baseline		Reported	
2	File Name	Attribute	Categories	Overall N	Categories	Overall N
20	Table 14.2.6.1.5.RTF	ECOG Status at Baseline	ECOG Status at Baseline: 0	6	ECOG Status at Baseline: 0	6
21			ECOG Status at Baseline: 1	16	ECOG Status at Baseline: 1	16
22			ECOG Status at Baseline: Missing	3	ECOG Status at Baseline: Missing	1
23					ECOG Status at Baseline: 2	3
24						

Zebra Pharmaceutical, Inc. Page 4 of 4
Zebra-301

Table 14.2.6.1.5
Kaplan-Meier Estimates of Progression-Free Survival (PFS) Based on Investigator Assessment by Treatment Group and ECOG Status at Baseline
ITT Population

ECOG Status at Baseline: 2

Statistics	DL1 (N=1)	DL2 (N=1)	DL3 (N=1)	DL4 (N=0)	Overall (N=3)

Figure 3. Outcome of the analysis of the Table 14.2.6.1.5. Upper part shows a record in the tab “Errors”, the lower part – screenshot of the actual table.

3. Example #3 - Table 14.3.1.1.1 (Figure 4). At the first glance the presentation by the Age Group in this table is different from what was displayed in the table for demographic and baseline characteristics – “Age Group” in the analyzed table and “Age Group at Baseline” in the Table 14.1.2.1 (Demographic and Baseline Characteristics). But one is not allowed to miss another distinction – two symbols “>=” are used instead of one “≥” and it adds up to general inconsistence of the delivery. In the cases like this one it is up to the statistician or the lead programmer of the project – to fix it and make a delivery consistent across all outputs or to leave it as is and to concentrate on more important issues. No need to say that the consistent delivery of hundreds of outputs looks much better than one having multiple differences across the study (even if they are not major).

			Baseline		Reported	
2	File Name	Attribute	Categories	Overall N	Categories	Overall N
25	Table 14.3.1.1.1.RTF	Age Group at Baseline	Age Group at Baseline: < 65 years	14	Age Group: < 65 years	14
26			Age Group at Baseline: ≥ 65 years	11	Age Group: >= 65 years	11
27						

Zebra Pharmaceutical, Inc. Page 3 of 4
Zebra-301

Table 14.3.1.1.1
Overview of TEAE by Treatment Group and Age Group
Safety Population

Age Group: >= 65 years

Parameter	DL1 (N=3) n (%)	DL2 (N=1) n (%)	DL3 (N=4) n (%)	DL4 (N=3) n (%)	Overall (N=11) n (%)

Figure 4. Outcome of the analysis of the Table 14.3.1.1.1. Upper part shows a record in the tab “Errors”, the lower part – screenshot of the actual table.

4. Example #4 - Table 14.2.6.1.1 (Fig. 5). The situation with this table is essentially different from what we observed in the previous example (#3). In addition to two distinctions in the way of presentation ("Age Group" vs "Age Group at Baseline" and ">=" vs "≥") one can easily see that the second age subgroup is displayed incorrectly – it reads as "greater or equal to 80 years old", which is evidently wrong.

			Baseline		Reported	
2	File Name	Attribute	Categories	Overall N	Categories	Overall N
9	Table 14.2.6.1.1.RTF	Age Group at Baseline	Age Group at Baseline: < 65 years	14	Age Group: <65 years	14
10			Age Group at Baseline: ≥ 65 years	11	Age Group: >= 80 years	11

Zebra Pharmaceutical, Inc. Page 2 of 3
Zebra-301

Table 14.2.6.1.1
Kaplan-Meier Estimates of Progression-Free Survival (PFS) Based on Investigator Assessment by Treatment Group and Age Group
ITT Population

Age Group: >= 80 years

Statistics	DL1 (N=3)	DL2 (N=1)	DL3 (N=4)	DL4 (N=3)	Overall (N=11)

Figure 5. Outcome of the analysis of the Table 14.2.6.1.1. Upper part shows a record in the tab "Errors", the lower part – screenshot of the actual table.

5. Example #5 – Table 14.2.2.1.2 (Figure 6). One can easily see a distinction in a way how subgroup is displayed in this table – instead of "Sex: Male" it reads as "Sex Group: Male", what is incorrect.

			Baseline		Reported	
2	File Name	Attribute	Categories	Overall N	Categories	Overall N
6	Table 14.2.2.1.2.RTF	Sex	Sex: Male	15	Sex Group: Male	15
7			Sex: Female	10	Sex Group: Female	10

Zebra Pharmaceutical, Inc. Page 1 of 2
Zebra-301

Table 14.2.2.1.2
Objective Response Rate as Assessed per Investigator by Treatment Group and Sex
Safety Population

Sex Group: Male

Statistics	DL1 (N=2)	DL2 (N=1)	DL3 (N=5)	DL4 (N=7)	Overall (N=15)

Figure 6. Outcome of the analysis of the Table 14.2.2.1.2. Upper part shows a record in the tab "Errors", the lower part – screenshot of the actual table.

6. Example #6 – Table 14.3.1.2.5.1 (Figure 7). The Figure 7 illustrating this example depicts only screenshot from the tab “Errors”. The reason is quite obvious – it is impossible to display something that is absent. The matter is that the baseline shows a programmer that there are three categories for the attribute “ECOG Status at Baseline” – “0”, “1” and “Missing”. The reviewed output however does not contain pages responsible for presentation the processed data for subjects with “Missing” value in this parameter and this line remains empty in the “Reported Section”. As an additional remark one can note that numbers of subjects (6 and 16) in the produced output do not add up to the correct value of overall population (25).

1			Baseline		Reported	
2	File Name	Attribute	Categories	Overall N	Categories	Overall N
40	Table 14.3.1.2.5.1.RTF	ECOG Status at Baseline	ECOG Status at Baseline: 0	6	ECOG Status at Baseline : 0	6
41			ECOG Status at Baseline: 1	16	ECOG Status at Baseline : 1	16
42			ECOG Status at Baseline: Missing	3		
43						

Figure 7. Outcome of the analysis of the Table 14.3.1.2.5.1. A record in the tab “Errors” shows that one of the baseline categories (Missing) is missing in the produced table.

COMPARATIVE ANALYSIS OF THE OUTPUTS – “SUCCESS”

The third tab of the output Excel file is titled “Success” and it contains a list of the produced outputs that were analyzed by the macro and no issues were uncovered (within the scope of the macro’s abilities) during this part of the QC process. It means that a lead programmer or statistician reviewing the final delivery can be sure that these tables are presented correctly from this point of view and no additional manual review is required. Figure 8 demonstrates in the column “File Name” a list of produced outputs that passed the cross-over review successfully. The second column, “Attribute”, shows the name of the subgroup that was used to develop this table.

Note, that this tab displays only the outputs that reviewed by the macro, not all of the outputs in the directory. For example, Tables similar to 14.2.1.1 (typically displaying the primary end point in the efficacy section) or 14.3.1.1 (typically presenting an overview of TEAE in the safety section) will never appear in this list because they do not present the results of the analysis by subgroups.

1	File Name	Attribute
23	Table 14.3.1.3.3.RTF	Smoke History
24	Table 14.3.1.3.4.RTF	Race Group
25	Table 14.3.1.3.5.RTF	ECOG Status at Baseline
26	Table 14.3.1.4.2.RTF	Sex
27	Table 14.3.1.4.3.RTF	Smoke History
28	Table 14.3.1.4.4.RTF	Race Group
29	Table 14.3.1.4.5.RTF	ECOG Status at Baseline
30	Table 14.3.1.5.2.RTF	Sex
31	Table 14.3.1.5.3.RTF	Smoke History
32	Table 14.3.1.5.4.RTF	Race Group
33	Table 14.3.1.5.5.RTF	ECOG Status at Baseline

Figure 8. A screen shot of the tab “Success” shows a few of produced outputs which passed the cross-over review successfully.

DISCUSSION

Accenture Life Sciences accrued some experience of practical application of the developed macro mXREVIEW. 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 proposed macro essentially reduces amount of efforts required to verify that the final outputs (at least part of them displaying the results of the statistical analysis by subgroups) are produced in a correct way. The developed macro mXREVIEW allows in seconds to verify that all interconnected outputs are presented in the same way and all “children” outputs are synchronized with their “parent” tables. It makes the final delivery more consistent and robust because it essentially reduces a need in manual review of hundreds of outputs.

As every professional working in the pharmaceutical industry knows the subgroup analysis plays very important role in the process of the agency’s review and a requirement to perform this type of investigation appears in numerous guidance documents issued by regulating authorities. Final review of the produced TLF represents an important task in a process of statistical programming and consists of a number of components. Cross-over review is one of these components and the proposed macro mXREVIEW allows to replace a tedious process requiring scrupulous work (subject to human errors!) with its automated analogue.

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 mXREVIEW 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.

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.

First, the present version of the proposed macro considers and examines the value of overall/total populations, while the investigation of the similar values for specific treatment groups (refer to Figure 1 – groups DL1, DL2, DL3, DL4) remain out of scope. This addition would allow to verify that all values of Big N are synchronized across the study and not only ones appearing in Overall/Total column.

Second, the next version of the proposed macro can examine the values from the main body of the tables. One can mention parameters like “Number of Total TEAE” or “Number of Subjects with at least one TEAE” as an example. It is obvious that being displayed by subgroup in the “child” table the values for these parameters must add up to the corresponding number in the “parent” table.

Another idea that is awaiting to be implemented in one of the future versions of the proposed macro relates to an ability to verify the values of Big N for different population across the whole study. It is well known that typical study is analyzing the collected data using different kinds of populations – full analysis set (FAS), safety population (SAF), intent-to-treat population (ITT), per protocol population (PP), etc. Every population

is used for the very specific purpose as it is described in the corresponding SAP for the study. Values of Big N and appropriate numbers in the body of the tables and figures are differing from each other for various populations. The future version of the macro is supposed to be able to read the title of the output under consideration, to identify the population (e.g., ITT population) and to perform cross-over review using tables and figures based on this population only.

The successful implementation of the prospective additions – both mentioned above and those still in the process of formulation - would definitely improve the robustness of the final delivery and save valuable time of lead programmers and statisticians by automation of cross-over review of the produced outputs. Further development of the macro can transform its current version in a valuable and powerful tool for streamlining the review process and reducing the risk of errors. The authors' intention is to develop a user-friendly and effective macro that can help to improve the efficiency and accuracy of clinical trial data analysis.

CONCLUSIONS

To recap the discussion of the current version (1.0) of the developed macro mXREVIEW it would be worthwhile to summarize macro's capabilities and emphasize its main advantages:

1. The developed macro mXREVIEW automates a cross-over review of the produced outputs across the whole study.
2. The code of the macro resides in the standard MS Excel file (Excel VBA). While running it generates three different tabs – "Baseline", "Errors" and "Success" – in the same Excel file.
3. The proposed macro reads the basic information from the table (standard MS Word file) where all demographic and baseline characteristics of the study population are presented and creates a list of potential subgroups in the tab "Baseline".
4. Suggested macro reads the actual tables, finds the ones using subgroups for analysis and examines them. The outputs where any distinction is uncovered are added to the tab Errors using color code to clearly mark the differences.
5. The distinctions uncovered during the comparative analysis of inter-related outputs require additional manual review and following decision of lead programmer or statistician.
6. The outputs where no errors were detected are displayed in the third tab – "Success". And all three steps of the macro run take literally seconds!

REFERENCES

Busa, B., (2019), Interactive TLFs - A Smarter Way to Review your Statistical Outputs, *Proceedings of the PharmaSUG 2019,* Paper AD-326

Carmeli, I., (2022), Data Mining of Tables: The Barrier for Automation, *Proceedings of the PharmaSUG 2022*, Paper AI-010.

Carmeli, I. and Bar, Y., (2022), Validation of Statistical Outputs Using Automation, *Proceedings of the PharmaSUG 2022*, Paper SA-011.

Carmeli, I., Mayorov, K., and Donovan, H., (2022), The Emerging Use of Automation to Address the Challenge of Cross-Table Consistency Checking of Output Used in the Reporting of Clinical Trial Data, *Proceedings of the PharmaSUG 2022*, Paper SA-009.

Donovan, H. and Mayorov, K., (2022), The Emerging Use of Automation to Address the Challenge of Cross-Table Consistency Checking of Output Used in the Reporting of Clinical Trial Data, *Proceedings of the PharmaSUG 2022*, Paper SA-009.

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.

Goldfarb, I., and Zelichonok, E., (2021), Macro to Compare Titles and Footnotes in Produced TLF and Corresponding Shells, *Proceedings of the PharmaSUG 2021*, Paper AD-179.

Malcolm, S., (2019), Large-scale TFL Automation for regulated Pharmaceutical trials using CDISC Analysis Results Metatadata (ARM), *Proceedings of the PharmaSUG 2022*, Paper AD-203.

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 mXREVIEW
'ReadMe text - below
'Attention - Make sure to add reference to Microsoft Word in Tools->References menu
Dim W As Word.Application



 Dim attributes As New Collection  'will contain all the found attributes
 Dim categories As New Collection  'will contain all the found categories of corresponding attributes
 Dim catCounts As New Collection 'will contain all the counts of corresponding categories of attributes
 Dim OverallCount As String


'This macro will create 3 spreadsheets. First one - Baseline - will contain read info from the baseline file
'Then the program will read all the files in the directory where baseline file is and report:
'   errors found in the second spreadsheets (named Errors)
'   list of all the successfully read files (in spreadsheet named Success)

Sub xReview10()
'
' Macro1 Macro
' This macro reads the baseline word document

    Dim File_Name As String
    Dim sOutputFile As Variant
    Dim ShellSheet As Worksheet


   'temporarily assign the name and skip open file dialog to choose the file
'    File_Name = "F:\Internal Use\Stats\Macro ParmaSUG2023\Zebra Pharmaceutical\Output\Table
14.1.2.1.RTF"
'    GoTo DoWork


    '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 Baseline document"



    'Show file dialog
    If Application.FileDialog(msoFileDialogOpen).Show Then

        frmProcessing.Show
        frmProcessing.MousePointer = fmMousePointerHourGlass

        frmProcessing.lblProcessing.Caption = "Starting Microsofft Word"

        Application.DisplayAlerts = False
```

```
        Set W = CreateObject("Word.Application")



        'disable word and application alerts
        W.DisplayAlerts = wdAlertsNone
        Application.DisplayAlerts = False



        File_Name = Application.FileDialog(msoFileDialogOpen).SelectedItems(1)

DoWork:

        'remove all the sheets except the first one which will be used for baseline info
        For i = Sheets.Count To 2 Step -1
            Sheets(i).Delete
        Next
        Sheets(1).Name = "Sheet1"
        Worksheets.Add After:=Sheets(1)   'adding 2nd sheet that will contain information about processed
files where errors found
        Worksheets.Add After:=Sheets(2) 'adding 3rd sheet that will contain information about processed
files with no problems

        Sheets(1).Name = "Baseline"
        Sheets(2).Name = "Errors"
        Sheets(3).Name = "Success"


        'read baseline file and fills first spreadsheet with found attributes, categories and counts
        frmProcessing.lblProcessing.Caption = "Reading Baseline Information"
        ReadBaseLineInfo File_Name


        ExamineTables File_Name

        frmProcessing.lblProcessing.Caption = "Formatting the Output"
        FormatOutput

        W.Quit
        Application.DisplayAlerts = True

        Unload frmProcessing
        MsgBox "Done", vbInformation
    End If
End Sub

'This function examines all the documents in the same directory as FileName
Sub ExamineTables(FileName As String)
Dim fn As String
Dim fn0 As String
Dim dirName As String
Dim r As Integer 'will hold Excel row number to print information in errors sheet
Dim r3 As Integer 'will hold Excel row number to print information in success sheet

Dim i1 As Integer
```

```
i1 = InStr(1, FileName, Dir(FileName), vbTextCompare)
dirName = Mid(FileName, 1, i1 - 1)

fn0 = Dir(FileName)
fn = Dir(dirName & "Table *.*")
Sheets(2).Cells.ClearContents
Sheets(3).Cells.ClearContents


'forming header in the second sheet


Sheets(2).Activate
r = 1
Sheets(2).Cells(r, 3) = "Baseline"
Sheets(2).Cells(r, 5) = "Reported"

    With Range(Cells(r, 3), Cells(r, 4))
        .HorizontalAlignment = xlCenter
        .Merge
    End With
    With Range(Cells(r, 5), Cells(r, 6))
        .HorizontalAlignment = xlCenter
        .Merge
    End With

r = r + 1
Sheets(2).Cells(r, 1) = "File Name"
Sheets(2).Cells(r, 2) = "Attribute"
Sheets(2).Cells(r, 3) = "Categories"
Sheets(2).Cells(r, 4) = "Overall N"
Sheets(2).Cells(r, 5) = "Categories"
Sheets(2).Cells(r, 6) = "Overall N"

Range(Cells(r, 1), Cells(r, 6)).HorizontalAlignment = xlCenter



With ActiveWindow
    .SplitColumn = 0
    .SplitRow = 2
End With
ActiveWindow.FreezePanes = True

r3 = 1
Sheets(3).Cells(r3, 1) = "File Name"
Sheets(3).Cells(r3, 2) = "Attribute"

Sheets(3).Activate

With ActiveWindow
    .SplitColumn = 0
    .SplitRow = 1
End With
ActiveWindow.FreezePanes = True
```

```
Sheets(2).Activate

Do While fn <> ""

    If fn <> fn0 Then

        frmProcessing.lblProcessing.Caption = "Processing file " & fn
        ProcessFile dirName, fn, r, r3

    End If

    fn = Dir()
Loop


'autofit columns
For i = 1 To 6
    Sheets(2).Columns(i).EntireColumn.AutoFit

Next
For i = 1 To 2
    Sheets(3).Columns(i).EntireColumn.AutoFit

Next

End Sub

'This function will process each file and examine tables in this file to crosscheck with counts for found
subcategories
Sub ProcessFile(dirName As String, fn As String, ByRef r As Integer, ByRef r3 As Integer)
    Dim FileName As String

    FileName = dirName & fn

    Dim dc As Document
    Dim HdrRange 'will contaionheaders of the sections
    Set dc = W.Documents.Open(FileName, , True)
    Dim arr
    Dim FoundAttName As String

    'Examine first section - all other sections will have the same header

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

    DoEvents


    arr = Split(HdrRange.text, vbCr)
    arr = CleanLines(arr)
    Dim foundAttr As New Collection
     'clear foundAttr collection - in case it sticks in memory
    For i = foundAttr.Count To 1 Step -1
       foundAttr.Remove (i)
    Next
```

```
    FoundAttName = ""
    For i = 0 To UBound(arr)
        If InStr(1, arr(i), " by ", vbTextCompare) > 0 Then
            For Each att In attributes
                att0 = Trim(Replace(att, "at baseline", "", , , vbTextCompare))
                If InStr(1, arr(i) & " ", " by " & att & " ", vbTextCompare) > 0 Or InStr(1, arr(i) & " ", " and " & att &
" ", vbTextCompare) > 0 Then
                    foundAttr.Add Trim(att)
                ElseIf InStr(1, arr(i) & " ", " by " & att0 & " ", vbTextCompare) > 0 Or InStr(1, arr(i) & " ", " and " &
att0 & " ", vbTextCompare) > 0 Then
                    foundAttr.Add Trim(att)
                End If
            Next
        End If

    Next i

    'examine all possible candidates - race vs race group - need to find the longest
    FoundAttName = ""
    For Each att In foundAttr

        If Len(att) > Len(FoundAttName) Then FoundAttName = att
    Next


    If FoundAttName <> "" Then
    'Attribute name found
    'need to get overall count and examine the sections

    'get overall count in the last cell of the table


    'examine all the sections
    Dim docCategories As New Collection
    Dim reportedCatCounts As New Collection
     For i = reportedCatCounts.Count To 1 Step -1
        reportedCatCounts.Remove (i)
    Next
    ReDim catChecks(UBound(categories(FoundAttName)))
        For k = 1 To dc.Sections.Count
            DoEvents
            Set HdrRange = dc.Sections.Item(k).Headers(wdHeaderFooterPrimary).Range

            arr = Split(HdrRange.text, vbCr)
            arr = CleanLines(arr)
            Dim docCat
            docCat = ""
            For i = 0 To UBound(arr)



                If InStr(1, arr(i), ":", vbTextCompare) > 0 Then

                    'allowing error will let handle duplicate entries
                    On Error Resume Next
                    docCategories.Add arr(i), arr(i)
```

```
                docCat = arr(i)
                On Error GoTo 0

                For j = 1 To UBound(categories(FoundAttName))
                    'compress before comparing - to avoid space misalignment
                    If Replace(arr(i), " ", "") = Replace(FoundAttName, " ", "") & ":" &
Replace(categories(FoundAttName)(j), " ", "") Or Replace(arr(i), " ", "") = Replace(FoundAttName0, " ", "")
& ":" & Replace(categories(FoundAttName)(j), " ", "") Then

                        catChecks(j) = 1
                        Exit For
                    End If

                Next
            End If


        Next i

        'now look at the table in this section and get last cell of the first non empty row
        Dim secText, valOverall
        secText = dc.Sections.Item(k).Range.text
        Dim i1, i2, i3
        i1 = InStr(1, secText, "overall", vbTextCompare)
        If i1 > 0 Then
            i2 = InStr(i1, secText, "=")
            If i2 > 0 Then
                i3 = InStr(i2, secText, ")")

                'remove error handling because sometimes table spans for 2 sections and there will be
duplicate entries
                On Error Resume Next
                reportedCatCounts.Add Mid(secText, i2 + 1, i3 - i2 - 1), docCat
                On Error GoTo 0
            End If


        End If

    Next k

    nMatched = 0
    For j = 1 To UBound(categories(FoundAttName))
        If reportedCatCounts.Count < UBound(categories(FoundAttName)) Then Exit For
        If catChecks(j) = 1 And Val(catCounts(FoundAttName)(j)) =
Val(reportedCatCounts(docCategories(j))) Then nMatched = nMatched + 1
    Next j

    If nMatched = UBound(categories(FoundAttName)) And nMatched = docCategories.Count Then
        r3 = r3 + 1
        Sheets(3).Cells(r3, 1) = fn
        Sheets(3).Cells(r3, 2) = FoundAttName
    Else
        r = r + 1
        Sheets(2).Cells(r, 1) = fn
```

```
            Sheets(2).Cells(r, 2) = FoundAttName

            For j = 1 To UBound(categories(FoundAttName))
               Sheets(2).Cells(r, 3) = FoundAttName & ": " & categories(FoundAttName)(j)
               Sheets(2).Cells(r, 4) = catCounts(FoundAttName)(j)

               If j <= docCategories.Count Then
                  Sheets(2).Cells(r, 5) = docCategories(j)
                  Sheets(2).Cells(r, 6) = reportedCatCounts(docCategories(j))

               Else
                  'this will happen if there are more baseline categories than reported - will mark it red
                  Sheets(2).Cells(r, 3).Font.Color = 192
                  Sheets(2).Cells(r, 4).Font.Color = 192

               End If

               'mark red any mismatches
               If Replace(Sheets(2).Cells(r, 3), " ", "") <> Replace(Sheets(2).Cells(r, 5), " ", "") Then
                  Sheets(2).Cells(r, 5).Font.Color = 192
               End If
               If Sheets(2).Cells(r, 4) <> Sheets(2).Cells(r, 6) Then
                  Sheets(2).Cells(r, 6).Font.Color = 192
               End If
               r = r + 1
            Next j

            For j = UBound(categories(FoundAttName)) + 1 To docCategories.Count
               If j <= docCategories.Count Then
                  Sheets(2).Cells(r, 5) = docCategories(j)
                  Sheets(2).Cells(r, 6) = reportedCatCounts(docCategories(j))

                  'this will happen if there are more reported categories than baseline - will mark it red
                  Sheets(2).Cells(r, 5).Font.Color = 192
                  Sheets(2).Cells(r, 6).Font.Color = 192
               End If
                r = r + 1
            Next j

         End If

      Else
      'not reporting when file has no categories


      End If


      dc.Close 0
   End Sub

   'This function will read all the baseline information and fill corresonding collections (attributes, categories
   and catCounts ) as well as finding OverallCount
   Sub ReadBaseLineInfo(FileName As String)
   Dim dc As Document
```

```
Set dc = W.Documents.Open(FileName, , True)

Dim firstCellText As String
Dim lastCellText As String

Dim AttName As String

 'clear all the collections
 For i = attributes.Count To 1 Step -1
   attributes.Remove (i)
 Next
  For i = categories.Count To 1 Step -1
   categories.Remove (i)
 Next
 For i = catCounts.Count To 1 Step -1
   catCounts.Remove (i)
 Next


ReDim cats(0)
ReDim counts(0)

'iterating through the tables of the word document
For t = 1 To dc.Tables.Count
   'iterating through all the rows of a given table
   For r = 1 To dc.Tables(t).Rows.Count
      DoEvents
      'getting the first cell that will contain a label and clean it removing all the non-ascii characters
      firstCellText = CleanText(dc.Tables(t).Rows(r).Cells(1).Range.text)

      'last cell will contain counts
      lastCellText = CleanText(dc.Tables(t).Rows(r).Cells(dc.Tables(t).Rows(r).Cells.Count).Range.text)

      If OverallCount = "" Then
         'overallcount will need to be read only once
         If InStr(1, lastCellText, "overall", vbTextCompare) > 0 Then
            OverallCount = lastCellText

            'getting a number out of string that is of the form: Overall(n=XX)
            OverallCount = Replace(OverallCount, "overall", "", , , vbTextCompare)
            OverallCount = Replace(OverallCount, "N=", "", , , vbTextCompare)
            OverallCount = Replace(OverallCount, "(", "")
            OverallCount = Replace(OverallCount, ")", "")
            OverallCount = Replace(OverallCount, " ", "")
         End If
      End If

      If firstCellText <> "" And AttName = "" Then
         'first label appearing after empty labels
         AttName = Trim(firstCellText)

         ReDim cats(0)
         ReDim counts(0)
      ElseIf firstCellText <> "" And AttName <> "" Then
         'if attName is filled, next non-empty rows will contain categories
         ReDim Preserve cats(UBound(cats) + 1)
```

```
            cats(UBound(cats)) = Trim(firstCellText)


            ReDim Preserve counts(UBound(counts) + 1)

            'count might be in a form of 5(6%) - need to get a number that is before paren
            Dim theCount As String
            theCount = lastCellText
            If InStr(theCount, "(") > 0 Then theCount = Mid(theCount, 1, InStr(theCount, "(") - 1)
            counts(UBound(cats)) = theCount

        ElseIf firstCellText = "" And AttName <> "" And UBound(cats) > 0 Then
            'prepare for next attribute
            attributes.Add AttName
            categories.Add cats, AttName
            catCounts.Add counts, AttName

            AttName = ""
            ReDim cats(0)
            ReDim counts(0)
        ElseIf firstCellText = "" And AttName <> "" And UBound(cats) = 0 Then
            AttName = ""
        End If
    Next r

    'last iteration - before end of the table - need to fill attributes and categories if non-empty
    If AttName <> "" And UBound(cats) > 0 Then

        attributes.Add AttName
        categories.Add cats, AttName
        catCounts.Add counts, AttName

        AttName = ""
        ReDim cats(0)
        ReDim counts(0)
    End If


Next t

'Displayiong results from reading baseline on the first sheet of curent excel document
Sheets(1).Activate
Sheets(1).Cells.ClearContents
Sheets(1).Cells.Style = "Normal"

Sheets(1).Cells(3, 1) = "Overall Count: " & OverallCount

Sheets(1).Cells(5, 1) = "Attribute"
Sheets(1).Cells(5, 2) = "Category"
Sheets(1).Cells(5, 3) = "Count"

r = 6
For a = 1 To attributes.Count
    Sheets(1).Cells(r, 1) = attributes(a): r = r + 1
    For c = 1 To UBound(categories(a))
        Sheets(1).Cells(r, 2) = categories(a)(c)
```

```
        Sheets(1).Cells(r, 3) = catCounts(a)(c)
        r = r + 1
    Next c
Next a

'autofit columns
Sheets(1).Columns(1).EntireColumn.AutoFit
Sheets(1).Columns(2).EntireColumn.AutoFit
Sheets(1).Columns(3).EntireColumn.AutoFit
Sheets(1).Columns(4).EntireColumn.AutoFit
Sheets(1).Columns(5).EntireColumn.AutoFit

Sheets(1).Cells(1, 1) = FileName

dc.Close 0
End Sub

Function CleanText(text)
    Dim newText As String


    For i = 1 To Len(text)
        If Asc(Mid(text, i, 1)) >= 32 Then newText = newText & Mid(text, i, 1)
    Next i
    newText = Trim(newText)
    newText = Replace(newText, "     ", " ")
    'newText = Replace(newText, "at baseline", "", , , vbTextCompare)

    CleanText = newText
End Function

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
", 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


Private Sub FormatOutput()
    Dim rLast
    Dim cLast

    '*********************************
    ' FORMATTING Success Spreadsheet
    '*********************************
    Sheets(3).Activate
    With Cells.SpecialCells(xlCellTypeLastCell)
        rLast = .Row
        cLast = .Column
    End With
    cLast = 2

    Range(Cells(1, 1), Cells(1, cLast)).Select
    FormatAsHeader
    Range(Cells(2, 1), Cells(rLast, cLast)).Select
    FormatAsTable True

    Rows(2).Select
    ActiveWindow.FreezePanes = True
    Range("A2").Select

    '*********************************
    ' FORMATTING Errors Spreadsheet
    '*********************************
    Sheets(2).Activate
    With Cells.SpecialCells(xlCellTypeLastCell)
        rLast = .Row
        cLast = .Column
    End With
    cLast = 6

    Range(Cells(1, 3), Cells(1, cLast)).Select
    FormatAsHeader
    Range(Cells(2, 1), Cells(2, cLast)).Select
    FormatAsHeader

    Range(Cells(3, 1), Cells(rLast, cLast)).Select
    FormatAsTable False

    For i = 3 To rLast
        If Cells(i, 1) <> "" Then
```

```
            Range(Cells(i, 1), Cells(i, cLast)).Select
            DrawTopLine
        End If

    Next i
    Rows(3).Select
    ActiveWindow.FreezePanes = True

    Range("A3").Select

    '*********************************
    ' FORMATTING Baseline Spreadsheet
    '*********************************
    Sheets(1).Activate


    Columns(2).NumberFormat = "@"

    With Cells.SpecialCells(xlCellTypeLastCell)
        rLast = .Row
        cLast = .Column
    End With
    cLast = 3

    Range(Cells(5, 1), Cells(5, cLast)).Select
    FormatAsHeader

    Range(Cells(6, 1), Cells(rLast, cLast)).Select
    FormatAsTable False

    For i = 6 To rLast
        If Cells(i, 1) <> "" Then
            Range(Cells(i, 1), Cells(i, cLast)).Select
            DrawTopLine
        End If

    Next i

    Range("A1").Font.Bold = True
    Range("A3").Font.Bold = True

    Rows(6).Select
    ActiveWindow.FreezePanes = True

    Range("A6").Select

    'Activate Errors sheet at the end
    Sheets(2).Activate
End Sub
Private Sub DrawTopLine()
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
```

```
End Sub
Private Sub FormatAsHeader()
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Range(Cells(1, 3), Cells(1, 6)).Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .ColorIndex = xlAutomatic
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .ColorIndex = xlAutomatic
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .ColorIndex = xlAutomatic
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .ColorIndex = xlAutomatic
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .ColorIndex = xlAutomatic
        .TintAndShade = 0
        .Weight = xlThin
    End With
    Selection.Borders(xlInsideHorizontal).LineStyle = xlNone
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorLight2
        .TintAndShade = 0.799981688894314
        .PatternTintAndShade = 0
    End With
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
        .WrapText = True
    End With
End Sub
Sub FormatAsTable(IsHorizontal As Boolean)
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
```

```
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    If IsHorizontal Then
        With Selection.Borders(xlInsideHorizontal)
            .LineStyle = xlContinuous
            .ColorIndex = 0
            .TintAndShade = 0
            .Weight = xlThin
        End With
    Else
        Selection.Borders(xlInsideHorizontal).LineStyle = xlNone

    End If
End Sub
```