SESUG 2022 Paper 177

One Click Excel Cleanup - Generate High Quality SDTM/ADAM Specification

Suresh Acharya, Merck & Co., Inc., Rahway, NJ, USA

ABSTRACT

The focus of this paper is to generate a high-quality Excel specification document in one click with the help of a Visual Basic for Applications (VBA) macro. In the pharmaceutical SAS programming industry, documents such as SDTM and ADaM specifications are required to create SDTM and ADaM datasets, and most times these are in Excel format. Statisticians and SAS programmers typically update these documents multiple times in-life as per the study requirements and analysis needs. These updates are frequently tracked by using various types of markings such as different colored text, highlighted cells, and overstrikes. These updates are often not formatted consistently leaving highly inconsistent text formats and fonts in the document. Removal of formatting is a manual process which is tedious and time consuming. It takes 20-30 hours per protocol deliverable with increased likelihood of human error and inconsistencies. As the number of worksheets increases, the time spent to clean up the markings also increases significantly. The VBA tool presented in this paper generates a high-quality Excel specification document that is consistently formatted, as per the formatting standards, and saves 20-30 hours during each delivery cycle. Final output is clean of any strikeouts, empty rows, empty columns, out of context borders and colors. In addition, it also updates font style, size and color to a consistent standard format while removing any additional cell fill color. With an option to clean a single sheet or multiple sheets, this tool assists in creating high quality Excel SDTM or ADAM specifications.

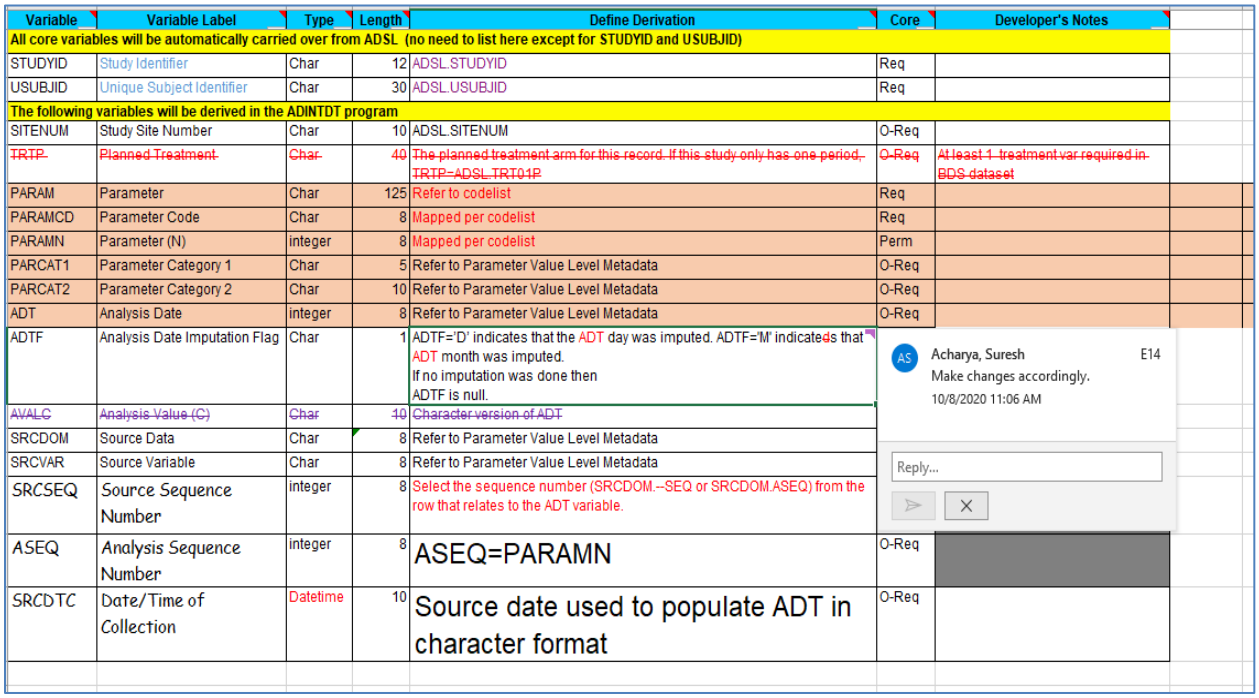
INTRODUCTION

Visual Basic for Applications (VBA) is an event driven programming language available in Microsoft products such as Microsoft Office Word, Excel, Access, Power Point, Microsoft Project, and Outlook. This allows developers to add new features and tools in Excel file to automate the process and add user defined functions. Such programs are saved within the user interface of Excel and can be executed anytime when needed. One such example is the use of VBA macro to remove strikethrough texts in Excel document (Liu, 2020).

In the clinical trial SAS programming domain, specifications are one of the starting points to develop datasets further used in analysis. Based on the level it could be in the form of SDTM or ADaM specifications which are used to develop SDTM and ADaM datasets respectively. These datasets are further used downstream to generate tables, graphs, and listings to be submitted to regulatory bodies for approval. In addition, the specification document is also used as an input for define.xml document which is submitted along with datasets. Therefore, a clean specification document with uniform formatting makes it easy to read while meeting the quality standards.

During the cycle, the specification goes a through number of edits, both from programmers and statisticians. As everyone has their own unique way of tracking these changes, the results lead to inconsistent formatting. This could be anything from strikethroughs, font color, background color, font size, empty rows/columns, filter, comments, and inconsistent borders. One such example of inconsistent formatting is displayed in Figure 1. In both SDTM and ADaM specifications there could be any number of worksheets in one workbook. Cleaning each one of these worksheets while trying to maintain consistency takes a significant amount of time. Despite the effort there could always be some human error. Our

objective is to automate the removal of all these formatting mark-ups throughout the workbook and generate a consistently formatted Excel document using Visual Basic for Applications (VBA) tool. This paper will explain details on how to create a VBA tool and show an example which can be used to clean any type of Excel documents.



Variable	Variable Label	Type	Length	Define Derivation	Core	Developer's Notes
All core variables will be automatically carried over from ADSL (no need to list here except for STUDYID and USUBJID)						
STUDYID	Study Identifier	Char	12	ADSL.STUDYID	Req	
USUBJID	Unique Subject Identifier	Char	30	ADSL.USUBJID	Req	
The following variables will be derived in the ADINTDT program						
SITENUM	Study Site Number	Char	10	ADSL.SITENUM	O-Req	
~~TRTP~~	~~Planned Treatment~~	~~Char~~	~~40~~	~~The planned treatment arm for this record. If this study only has one period, TRTP=ADSL.TRT01P~~	~~O-Req~~	~~At least 1 treatment var required in BDS dataset~~
PARAM	Parameter	Char	125	Refer to codelist	Req	
PARAMCD	Parameter Code	Char	8	Mapped per codelist	Req	
PARAMN	Parameter (N)	integer	8	Mapped per codelist	Perm	
PARCAT1	Parameter Category 1	Char	5	Refer to Parameter Value Level Metadata	O-Req	
PARCAT2	Parameter Category 2	Char	10	Refer to Parameter Value Level Metadata	O-Req	
ADT	Analysis Date	integer	8	Refer to Parameter Value Level Metadata	O-Req	
ADTF	Analysis Date Imputation Flag	Char	1	ADTF='D' indicates that the ADT day was imputed. ADTF='M' indicates that ADT month was imputed. If no imputation was done then ADTF is null.		
~~AVALC~~	~~Analysis Value (C)~~	~~Char~~	~~10~~	~~Character version of ADT~~		
SRCDOM	Source Data	Char	8	Refer to Parameter Value Level Metadata		
SRCVAR	Source Variable	Char	8	Refer to Parameter Value Level Metadata		
SRCSEQ	Source Sequence Number	integer	8	Select the sequence number (SRCDOM.--SEQ or SRCDOM.ASEQ) from the row that relates to the ADT variable.		
ASEQ	Analysis Sequence Number	integer	8	ASEQ=PARAMN	O-Req	
SRCDTC	Date/Time of Collection	Datetime	10	Source date used to populate ADT in character format	O-Req	

Figure 1. Heavily formatted Excel worksheet

GUIDELINES ON CREATING AND USING VBA TOOL

Visual Basic for Applications (VBA) macro to clean the Excel document can be invoked by following the steps below.

Step 1: Evoke VBA Macro Window.

As described above, VBA Macro should be run within Excel application as it provides the host environment. Thus, you open Excel document first and then evoke VBA Macro window to initialize VBA Macro built. Follow below to do so:

Select ***View*** in Excel Menu bar

Click on ***Macros***, select ***View Macros***, Macro window pops up

Type the Macro name "DeleteAllFormats" in ***Macro*** window (Figure 2)

Click on ***Create***, VBA Editor window pops up

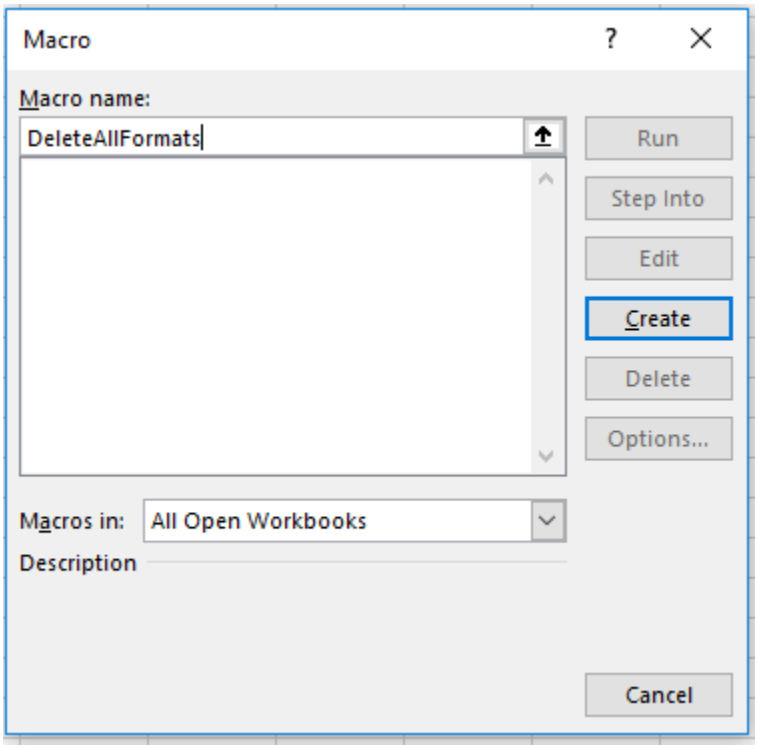


Figure 2: VBA Macro window

Step 2: Write VBA Macro in VBA Editor.

VBA Editor displayed as Figure 3

VBA Editor always has two pre-written lines

- Macro starts with "***Sub***" followed by Macro Name in the first line
- Macro ends with "***End Sub***" (Figure 3)

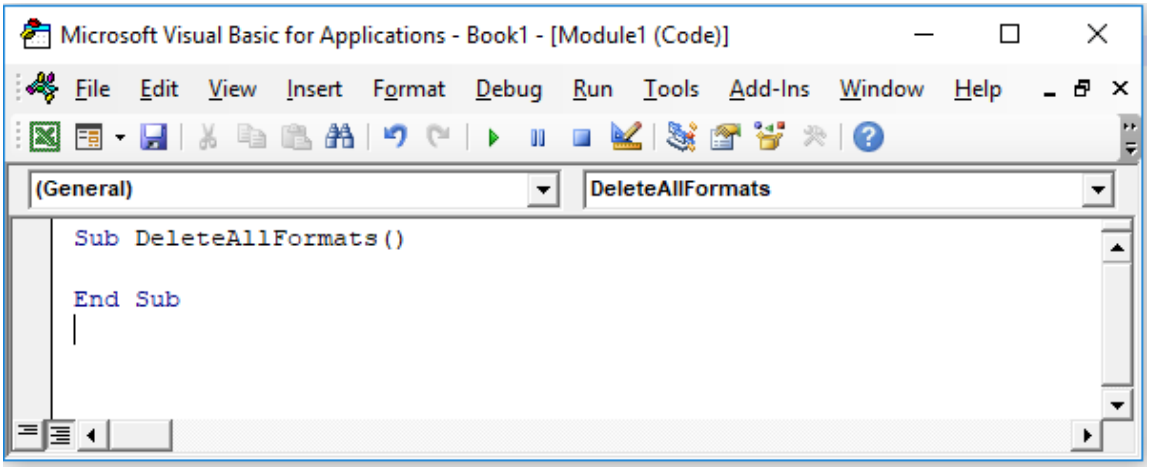


Figure 3: VBA Editor window

Write Macro in VBA Editor (Figure 4)

- Delete the original two lines.
- Develop code inside the text box (Figure 4) as explained in the section below.

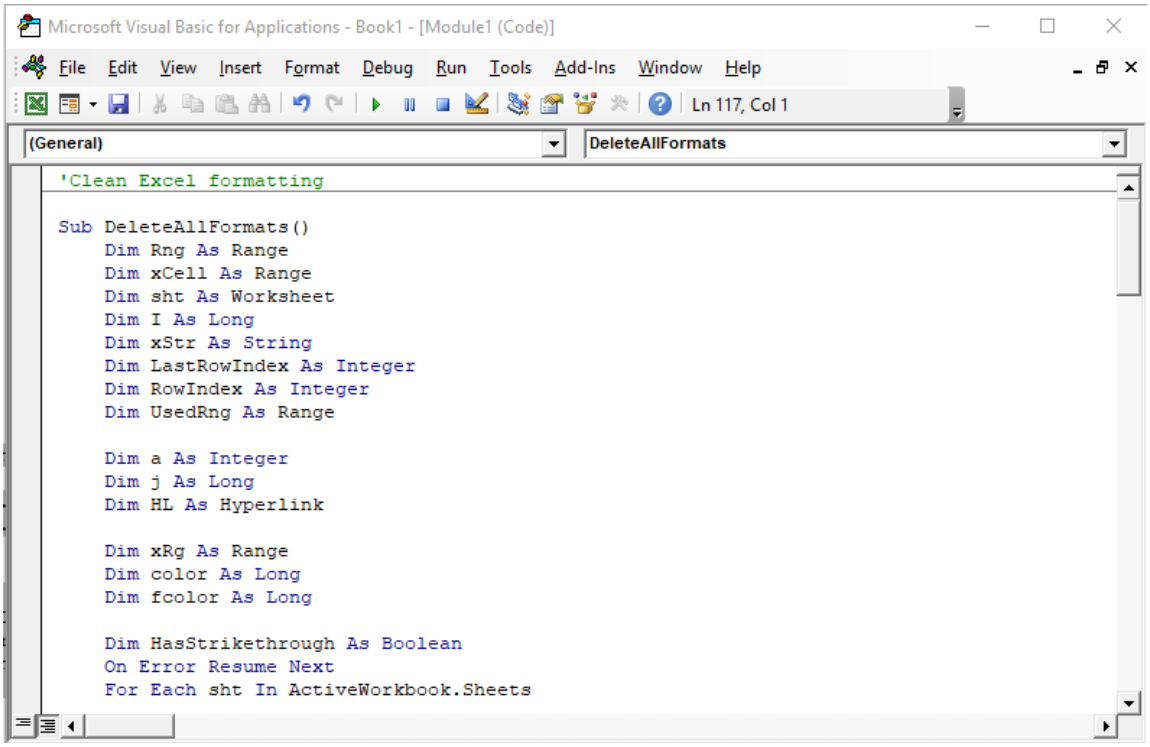


Figure 4: VBA Macro on "DeleteAllFormats"

Step 3: Run VBA Macro.

- Click on ***Run button*** (green triangle shape, Figure 5). OR,
- Click on ***Run*** in Menu bar, then select ***Run button*** (green triangle shape, Figure 6).
- Message box appears "Clean all Sheets in this Workbook" (Figure 7). If you want to clean all sheets at one click "Yes", else, click "No".
- If you click "No", message box appears that prompts you to provide name of individual sheet that you want to clean (Figure 8a). Provide sheet name (case doesn't matter) as in the example (Figure 8b).
- VBA Macro running completes successfully with ***notification*** window pops up (Figure 9a) for all sheets cleaned at once, or, single sheet cleaned (Figure 9b).
- Click Ok and check Excel document to make sure VBA Macro works as expected.

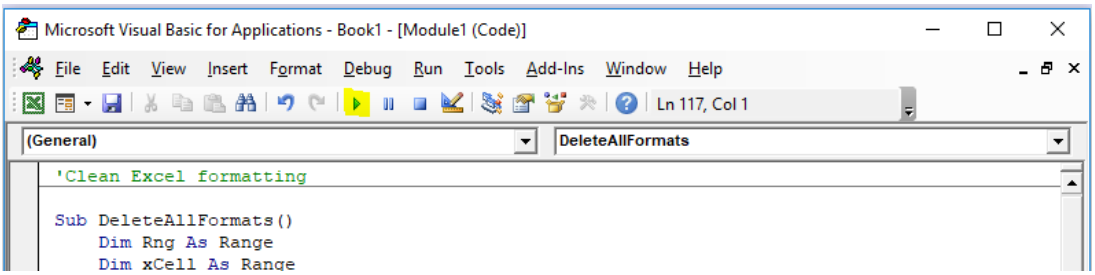


Figure 5: VBA Macro Run Button

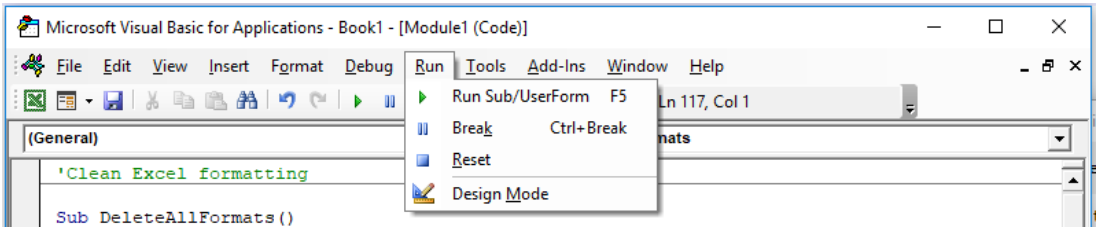


Figure 6: VBA Macro Run Menu

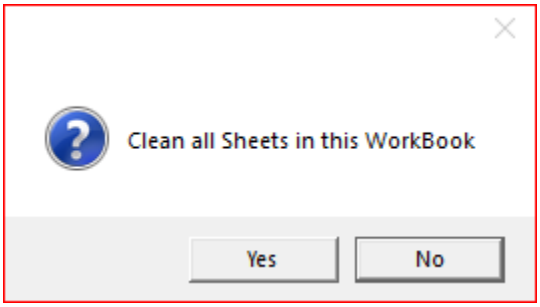


Figure 7: Option to Clean all Sheets

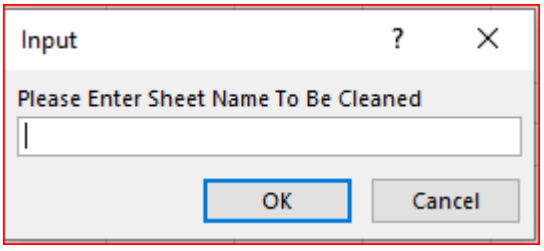


Figure 8a: Clean single sheet

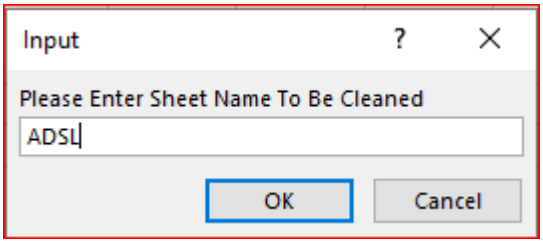


Figure 8b: Option B example.

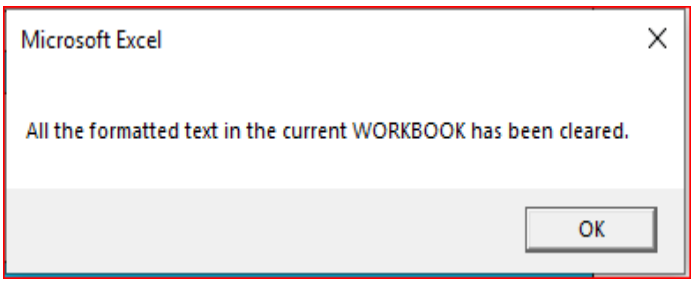


Figure 9a: All Sheets cleaned

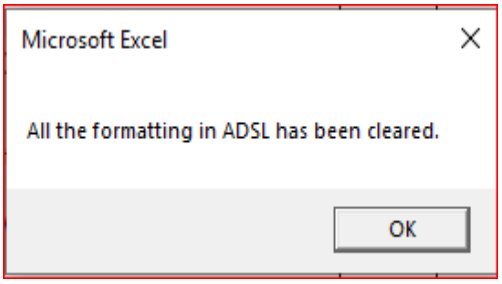


Figure 9b: Single Sheet cleaned

EXAMPLE CODES FOR THE CLEAN EXCEL VBA MACRO

Cleaning of worksheet begins with removal of all the strikeouts (Figure 10).

```
For Each xCell In Rng
    HasStrikethrough = False
    For i = 1 To Len(xCell)
        With xCell.Characters(i, 1)
            If Not .Font.Strikethrough Then xStr = xStr & .Text
            Else
                If HasStrikethrough = False Then xStr = Trim(xStr)
                    xStr = xStr & " "
                Else
                    xStr = xStr & ""
                End If
                HasStrikethrough = True
            End If
        End With
    Next
    If HasStrikethrough = True Then
        xStr = Replace(xStr, ChrW(&HA0), " ")
        xCell.Value = WorksheetFunction.Trim(xStr)
    End If
    xStr = ""
Next xCell
```

Figure 10

Removal of strikeouts sometimes will result is empty rows and columns. These empty rows and columns don't serve any purpose and hence, needs to be removed (Figure 11).

```
If ActiveSheet.Name <> " " Then
   For K = Cells.SpecialCells(xlCellTypeLastCell).Row To 1 Step -1
   If WorksheetFunction.CountA(Rows(K)) = 0 Then
       ActiveSheet.Rows(K).Delete
   End If
   Next
   For C = ActiveSheet.Cells.SpecialCells(xlLastCell).Column To 1 Step -1
   If WorksheetFunction.CountA(Columns(C)) = 0 Then
       Columns(C).Delete
   End If
   Next
End If
```

Figure 11

Next is to make texts uniform for font size, color, and type along with removal of unwanted borders outside the range. During this process all comments associated with cells will be removed as well (Figure 12).

```
If ActiveSheet.Name <> "" Then
    ThisWorkbook.Sheets(J).Range(ThisWorkbook.Sheets(J).Cells(1, 1), ThisWorkbook.Sheets(J).Cells).Font.Name = "Arial"
    ThisWorkbook.Sheets(J).Range(ThisWorkbook.Sheets(J).Cells(1, 1), ThisWorkbook.Sheets(J).Cells).Font.Size = "9"
End If
    ThisWorkbook.Sheets(J).Range(ThisWorkbook.Sheets(J).Cells(1, 1), ThisWorkbook.Sheets(J).Cells).Borders.LineStyle = xlNone
    ThisWorkbook.Sheets(J).Range(ThisWorkbook.Sheets(J).Cells(1, 1), ThisWorkbook.Sheets(J).Cells).Borders.LineStyle = xlContinuous
    ThisWorkbook.Sheets(J).Range(ThisWorkbook.Sheets(J).Cells(2, 1), ThisWorkbook.Sheets(J).Cells).ClearComments

    For Each xRg In ActiveSheet.Range(ActiveSheet.Cells(1, 1), ActiveSheet.Cells(350, 50))
        color = xRg.Interior.color
        If color <> RGB(0, 204, 255) Then
            xRg.Interior.color = RGB(255, 255, 255)
        End If
        fcolor = xRg.Font.color
        If fcolor <> RGB(13, 69, 199) And fcolor <> RGB(251, 253, 254) Then
            xRg.Font.color = vbBlack
        End If
    Next xRg
    For Each HL In ActiveSheet.Hyperlinks
        HL.Range.Font.color = vbBlue
Next
```

Figure 12

In addition, a single button can be added anywhere in the Excel document to invoke the VBA code. This creates a one click cleanup tool for future use.

OUTPUT AFTER INVOKING THE VBA MACRO

Once the macro is invoked, all the formatting is cleared in the worksheet with consistent font size and type throughout the document (Figure 13). To make it more readable, colors of certain rows can be retained based on the need. A clean workbook is easy to read and creates less issues when used as an input document by other programming tools.

Variable Name	Variable Label	Type	Length	Define Derivation	Core	Developer's Notes
All core variables will be automatically carried over from ADSL (no need to list here except for STUDYID and USUBJID)						
STUDYID	Study Identifier	Char	12	ADSL.STUDYID	Req	
USUBJID	Unique Subject Identifier	Char	30	ADSL.USUBJID	Req	
The following variables will be derived in the ADINTDT program						
SITENUM	Study Site Number	Char	10	ADSL.SITENUM	O-Req	
PARAM	Parameter	Char	125	Refer to codelist	Req	
PARAMCD	Parameter Code	Char	8	Mapped per codelist	Req	
PARAMN	Parameter (N)	integer	8	Mapped per codelist	Perm	
PARCAT1	Parameter Category 1	Char	5	Refer to Parameter Value Level Metadata	O-Req	
PARCAT2	Parameter Category 2	Char	10	Refer to Parameter Value Level Metadata	O-Req	
ADT	Analysis Date	integer	8	Refer to Parameter Value Level Metadata	O-Req	
ADTF	Analysis Date Imputation Flag	Char	1	ADTF='D' indicates that the ADT day was imputed. ADTF='M' indicate s that ADT month was imputed. If no imputation was done then ADTF is null.	Cond	Required if ADT is derived
SRCDOM	Source Data	Char	8	Refer to Parameter Value Level Metadata	Perm	
SRCVAR	Source Variable	Char	8	Refer to Parameter Value Level Metadata	O-Req	
SRCSEQ	Source Sequence Number	integer	8	Select the sequence number (SRCDOM.--SEQ or SRCDOM.ASEQ) from the row that relates to the ADT variable.	O-Req	
ASEQ	Analysis Sequence Number	integer	8	ASEQ=PARAMN	O-Req	
SRCDTC	Date/Time of Collection	Datetime	10	Source date used to populate ADT in character format	O-Req	

Figure 13. Clean Excel worksheet

CONCLUSION

Manual cleaning of Excel files with several worksheets is a tedious and time-consuming process. Every time new updates are made, we need to scan through all worksheets to find any updates and determine whether those have been properly formatted. Changes on the document also demand proper communication and time therein adding more resources. Automating the formatting process significantly reduces the resources with little to no chance of error. In our experience, this tool saved 20-30 hours in one protocol deliverable. When utilized across number of studies the resources saved are significant.

Another major advantage is we can create a uniformly formatted specification document throughout all studies. With a click of a button, we can generate a clean specification document with:

1. No Strikeouts.
2. No Empty Rows.
3. Uniform Font Color (Exclusions apply).
4. Clean Background.
5. No Extra Border.
6. No Fill Outside Range.
7. Border in Range All Filled.
8. Uniform Background color (Exclusions apply).
9. Intact Hyperlink/s Color.
10. Filters removed to show all texts.
11. No Comments.

REFERENCES

Liu, Li. PharmaSug 2020. Remove Strikethrough Texts from Excel Documents by VBA Macro Available at https://www.lexjansen.com/pharmasug/2020/QT/PharmaSUG-2020-QT-127.pdf

CONTACT INFORMATION

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

Suresh Acharya
suresh.acharya@merck.com