PharmaSUG China 2023 - Paper AP-128

Interaction with Protocol and SAP – VBA simplifies programmer's day

Jingxin Fu, Shuting Lou, Yun Lu, Pfizer (China) Research and Development Co.,Ltd. China

ABSTRACT

Statistical programmers are tasked with analyzing large amounts of information from Protocols and SAPs. However, navigating through these documents can be time-consuming and error-prone. This paper presents a VBA tool that simplifies the programmer's day by scanning and extracting subtitles and tables from protocols and SAPs. Additionally, the tool can read tables containing information such as visit windows and criteria and transform them into a programming-friendly format. These features streamline data analysis processes, simplify daily workflow, and save time that would have been spent on manual data entry.

The tool automates repetitive tasks and improves efficiency, allowing programmers to spend more time on data analysis and less time on document processing. With this tool, users can avoid direct copy and paste from Protocols and SAPs, reducing the risk of human error. The tool also integrates seamlessly with Excel's built-in functions, such as filtering, sorting, and plotting, which enhances data analysis capabilities. Moreover, the use of VBA within Excel negates the necessity of installing additional software. Additionally, this tool creates CDISC-compliant output, ensuring that the extracted information is compatible with industry standards. The tool produces outputs that are readable to multiple programming languages and are adaptable to different usage purposes, making it easier and more flexible for programmers to integrate the outputs into their analysis workflows. Programmers with any level of expertise can utilize the tool, thanks to its accessible and user-friendly interface.

After testing on several Protocols and SAPs, this tool shows the ability to accurately extract relevant information, reducing the potential for errors and improving programming quality. Overall, the VBA tool presented in this paper offers a practical solution for improving productivity and reducing errors in statistical programming while ensuring compliance with industry standards.

INTRODUCTION

VBA, which is short for Visual Basic for Applications, is a programming language that is used to automate tasks in Microsoft Office applications, such as Excel, PowerPoint, and Word. It provides users with additional customizable features beyond those typically included in Office apps. A few examples of VBA can automate are data cleaning and formatting, capitalizing text, and merging multiple worksheets. In addition, VBA allows for the transfer of data between different applications and edit information based on the original versions. It can be used to streamline the process of copying data from Excel and pasting it into a Word or PowerPoint slide or create a dynamic link between applications so that changes made in Excel are automatically updated in Word or PowerPoint. Compared to the manual copying and pasting technique, data transfer using VBA can be simpler, faster, and more accurate. Moreover, there's no need to purchase VBA software separately, as it is included with Microsoft Office. With these advantages in mind, this paper explores the potential and possibilities of VBA application to automate repetitive tasks for statistical programmers.

On a day-to-day basis, SAP and Protocol are critically important documents in clinical studies providing the statistical programmer with relevant information and detail on the clinical trial design, the scope of planned analyses, and methodology. Programmers must navigate through them repeatedly to have an accurate understanding of important information and construct datasets and outputs. There's no room for error. The obvious antidote to repetitive and error-prone tasks is automation. This paper presents a VBA tool that simplifies the programmer's day by scanning and extracting subtitles and tables from protocols and SAPs. Additionally, the tool can read tables containing information such as visit windows and criteria and transform them into a programming-friendly format.

With the use of this tool, users can increase efficiency and accuracy by eliminating the need to open SAP and Protocol files frequently and copy and paste manually. As the output generated by the tool can be

read by programming languages, users can integrate the output into their analysis workflow more easily and flexibly as well.

These features streamline data analysis processes, simplify daily workflow, and save time that would have been spent on manual data entry. This paper will provide a detailed description of the tool to illustrate how to simplify the programmer's work.

SAP&PROTOCOL TOOL

The SAP&Protocol Tool is an add-in for Microsoft Excel that implements the interaction between Excel and Word documents (Statistical Analysis Plan and Protocol) by using VBA. It has a user-friendly interface and complete the extraction within seconds. The following flowchart (Figure 1) depicts the overall structure and functionality of this tool. The below sections will introduce the User interface design, SAP Tool, and Protocol Tool respectively.

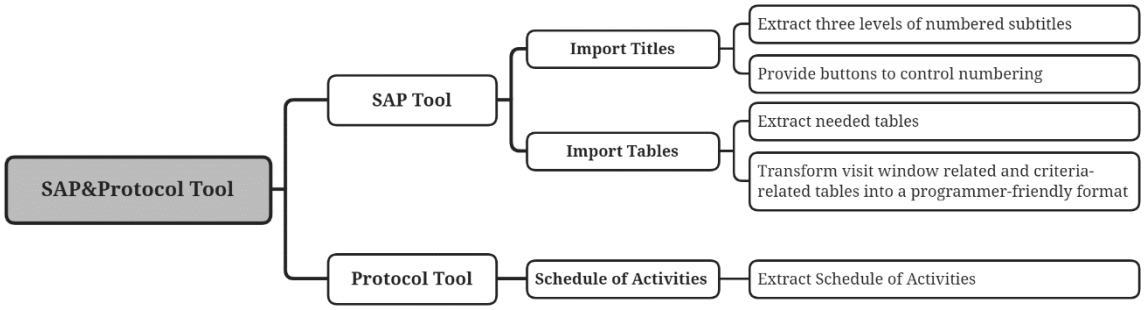


Figure 1. Flowchart of SAP&Protocol Tool

PART 1 USER INTERFACE DESIGN

As a matter of convenience, the SAP&Protocol Tool is saved as a macro-enabled add-in (XLAM). This makes it convenient for user to share the tool to others and the tool can be loaded into Excel by navigating to the “Add-ins” section in the Excel options and selecting the desired add-in file. Once loaded, the functionality of the tool is always available in Excel and users can invoke the SAP&Protocol Tool in any Excel file.

In addition, the tool uses the Office RibbonX Editor (Figure 2) to customize Excel Ribbon to make the interface clear and easy to operate.

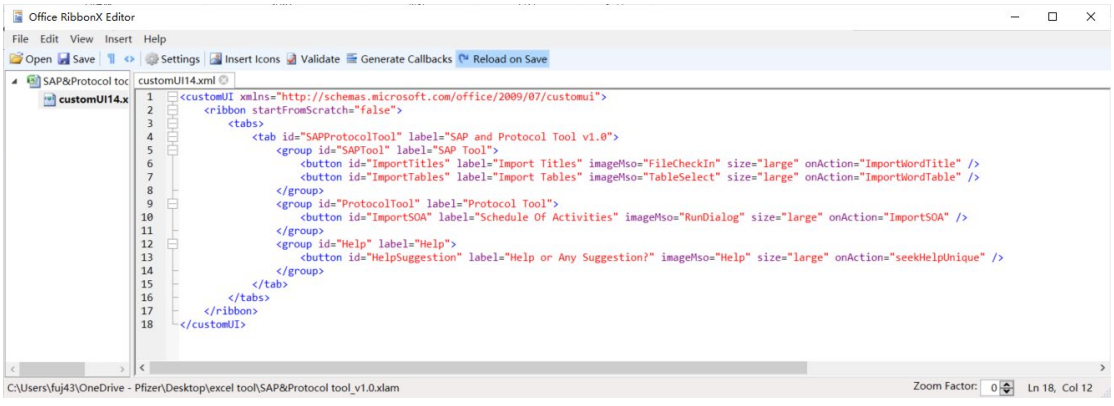


Figure 2. xml fragment to customize Excel Ribbon

When loading SAP&Protocol Tool.xlam file, “SAP and Protocol Tool v1.0” should appear as tab on the Ribbon in Excel (Figure 3). This tab includes all the functions in the tool and allows the user to use it as soon as excel is opened.

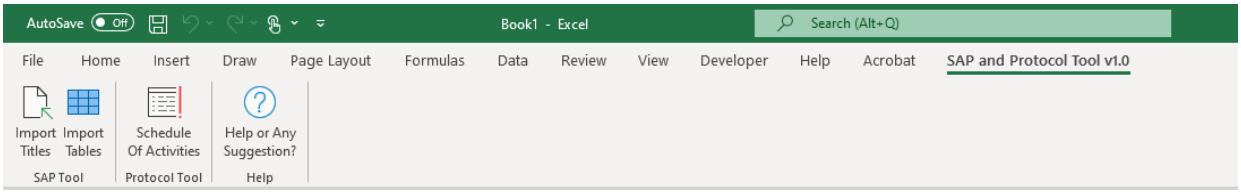


Figure 3. the customized SAP and Protocol Tool v1.0 Tab

According to different source files, The SAP and Protocol Tool tab consists of two groups: SAP Tool and Protocol Tool. The SAP Tool contains two buttons with labels: Import Titles and Import Tables; the Protocol Tool contains one: Schedule of Activities. Each button in the group has different settings.

PART 2 SAP TOOL

SAP Tool group is mainly used to extract subtitles and tables from SAP to Excel. With the execution of the following VBA code (Figure 4), User can open a Word document and check the existence of tables in the document. The purpose of “GetOpenFilename” is to display the standard Open dialog box, with the file filter set to Word files. When the “GetObject” function is used, and the object in the specified Word file is activated.

```
Dim wdFileName As Variant
wdFileName = Application.GetOpenFilename("Word files (*.docx),*.docx", , _
"Browse for file containing table to be imported")

If wdFileName = False Then Exit Sub 'user cancelled import file browser)
Set wdDoc = GetObject(wdFileName) 'open Word file
With wdDoc
If .Tables.Count = 0 Then
    MsgBox "This document contains no Tables."
    End If
If .Tables.Count > 0 Then
    MsgBox "This document contains " & .Tables.Count & " Tables."
    End If
```

Figure 4. Code to open a word document and check the existence of tables

When user clicks the “Import Titles” or “Import Tables” command button, a file dialog appears, and SAP file user need should be selected. Please refer to the following figure (Figure 5). Here, Users can read the content in a word document without opening Word manually.

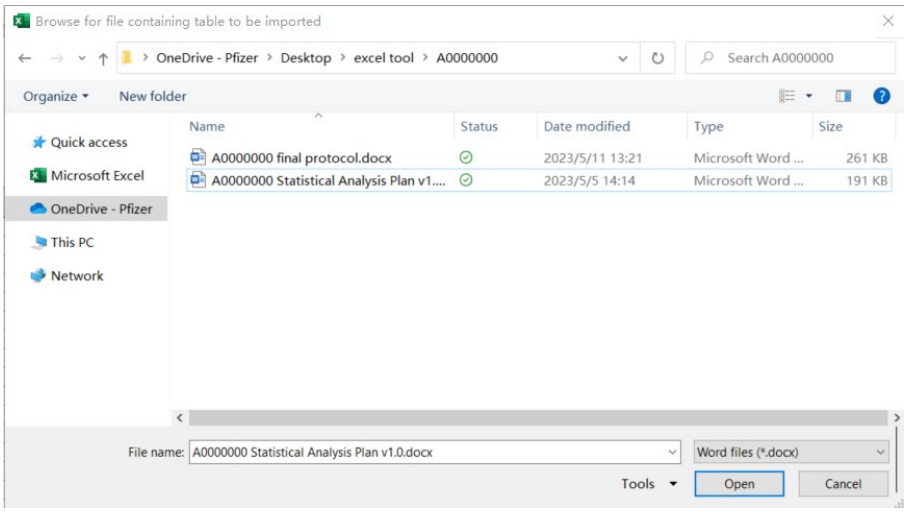
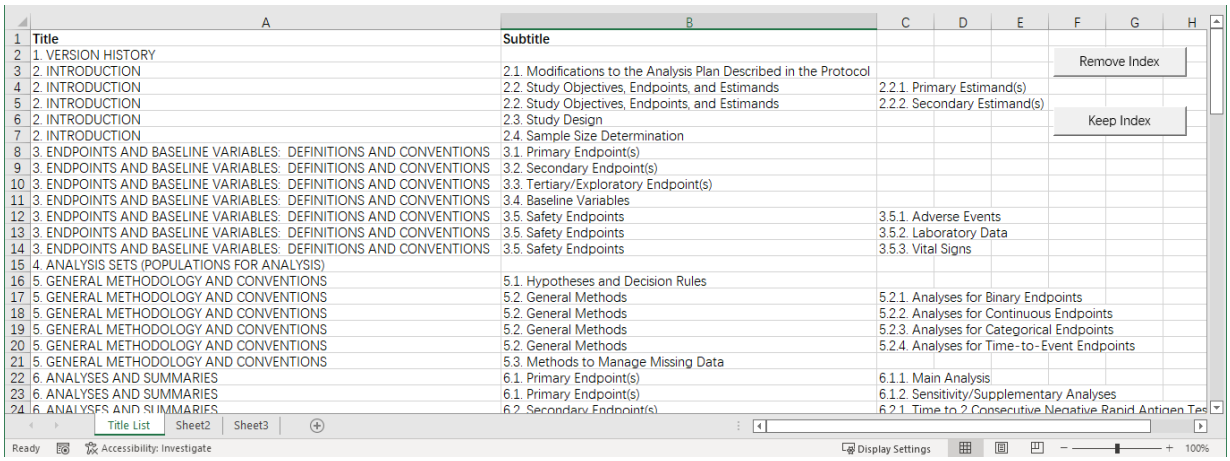


Figure 5. Open File dialog

If user clicked “Import Titles” button, three levels of numbered subtitles could be imported into “Title List” sheet in Excel (Figure 6). This facilitates programmers to quickly understand the structure of SAP and create LoT file according to endpoints. Two buttons “Remove Index” and “Keep Index” are also provided to control numbering making it easier for users to copy the desired content and use it directly



	Title	Subtitle	
1	1. VERSION HISTORY		
2	2. INTRODUCTION	2.1. Modifications to the Analysis Plan Described in the Protocol	
3	2. INTRODUCTION	2.2. Study Objectives, Endpoints, and Estimands	2.2.1. Primary Estimand(s)
4	2. INTRODUCTION	2.2. Study Objectives, Endpoints, and Estimands	2.2.2. Secondary Estimand(s)
5	2. INTRODUCTION	2.3. Study Design	
6	2. INTRODUCTION	2.4. Sample Size Determination	
7	3. ENDPOINTS AND BASELINE VARIABLES: DEFINITIONS AND CONVENTIONS	3.1. Primary Endpoint(s)	
8	3. ENDPOINTS AND BASELINE VARIABLES: DEFINITIONS AND CONVENTIONS	3.2. Secondary Endpoint(s)	
9	3. ENDPOINTS AND BASELINE VARIABLES: DEFINITIONS AND CONVENTIONS	3.3. Tertiary/Exploratory Endpoint(s)	
10	3. ENDPOINTS AND BASELINE VARIABLES: DEFINITIONS AND CONVENTIONS	3.4. Baseline Variables	
11	3. ENDPOINTS AND BASELINE VARIABLES: DEFINITIONS AND CONVENTIONS	3.5. Safety Endpoints	3.5.1. Adverse Events
12	3. ENDPOINTS AND BASELINE VARIABLES: DEFINITIONS AND CONVENTIONS	3.5. Safety Endpoints	3.5.2. Laboratory Data
13	3. ENDPOINTS AND BASELINE VARIABLES: DEFINITIONS AND CONVENTIONS	3.5. Safety Endpoints	3.5.3. Vital Signs
14	4. ANALYSIS SETS (POPULATIONS FOR ANALYSIS)		
15	5. GENERAL METHODOLOGY AND CONVENTIONS	5.1. Hypotheses and Decision Rules	
16	5. GENERAL METHODOLOGY AND CONVENTIONS	5.2. General Methods	5.2.1. Analyses for Binary Endpoints
17	5. GENERAL METHODOLOGY AND CONVENTIONS	5.2. General Methods	5.2.2. Analyses for Continuous Endpoints
18	5. GENERAL METHODOLOGY AND CONVENTIONS	5.2. General Methods	5.2.3. Analyses for Categorical Endpoints
19	5. GENERAL METHODOLOGY AND CONVENTIONS	5.2. General Methods	5.2.4. Analyses for Time-to-Event Endpoints
20	5. GENERAL METHODOLOGY AND CONVENTIONS	5.3. Methods to Manage Missing Data	
21	6. ANALYSES AND SUMMARIES	6.1. Primary Endpoint(s)	6.1.1. Main Analysis
22	6. ANALYSES AND SUMMARIES	6.1. Primary Endpoint(s)	6.1.2. Sensitivity/Supplementary Analyses
23	6. ANALYSES AND SUMMARIES	6.2. Secondary Endpoint(s)	6.2.1. Time to 2 Consecutive Negative Rapid Antigen Tes

Figure 6. Display three levels of numbered subtitles of the SAP

If user clicked “Import Tables” button, a table list would pop up (Figure 7). The user can select one or more tables to be imported from SAP in order to clearly read the information of needed table clearly and analyze it. Moreover, it also eliminates the need for users to go through SAP file repeatedly to find tables and their related information.

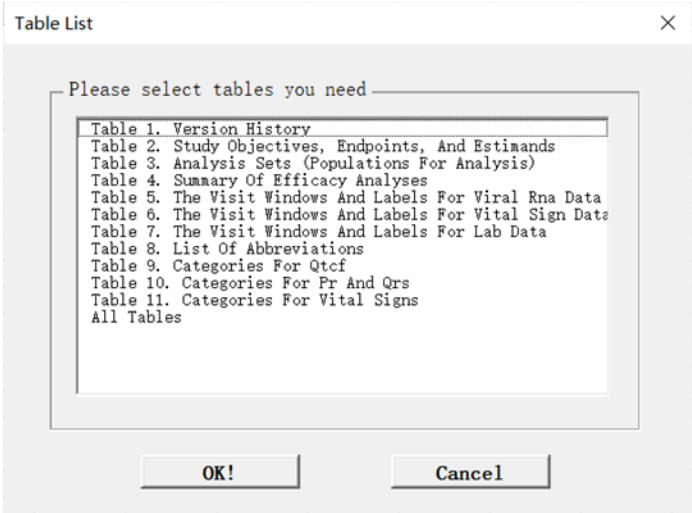
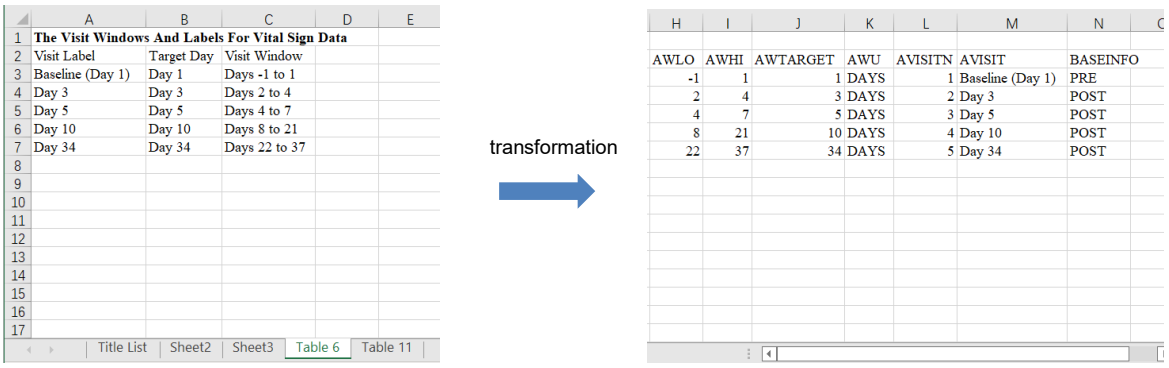


Figure 7. Table List of SAP

It is worth noting that if the selected table contains information about visit windows and criteria, for example, Table 5, Table 6, Table 7, Table 9, Table 10, and Table 11 in the table list (Figure 7), the table will be transformed into a programming-friendly format.

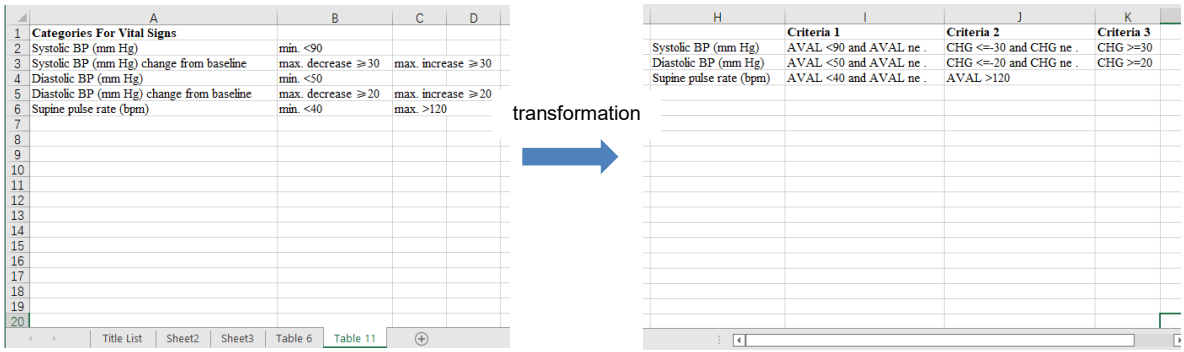
The following Figure 8 and Figure 9 give two examples of how to transform visit window and criteria related tables. These transformations make it easier for programmers to write code, integrate the outputs and make the information compatible with industry standards.



Original visit window related table

Transformed visit window related table

Figure 8. Visit Windows and Labels for Vital Sign Data and its transformed table



Original criteria-related table

Transformed criteria-related table

Figure 9. Categories For vital Signs and its transformed table

The following code (Figure 10 and Figure 11) demonstrates the rules for transforming visit window related and criteria-related tables, respectively.

Transformation of visit windows related tables is handled by the following code (Figure 10).

```
If InStr(UCase(ActiveSheet.Cells(2, n).Value), "LABEL") > 0 Then
For m = 3 To lr
ActiveSheet.Cells(m, lc + 9) = m - 2
ActiveSheet.Cells(m, lc + 10) = ActiveSheet.Cells(m, n).Value
Next
ElseIf InStr(UCase(ActiveSheet.Cells(2, n).Value), "TARGET") > 0 Then
For m = 3 To lr
ActiveSheet.Cells(m, lc + 7) = Split(ActiveSheet.Cells(m, n), Chr(32))(1)
Next
ElseIf InStr(UCase(ActiveSheet.Cells(2, n).Value), "WINDOW") > 0 Then
For m = 3 To lr
ActiveSheet.Cells(m, lc + 5) = Split(ActiveSheet.Cells(m, n), Chr(32))(1)
ActiveSheet.Cells(m, lc + 6) = Split(ActiveSheet.Cells(m, n), Chr(32))(3)
ActiveSheet.Cells(m, lc + 8) = UCase(Split(ActiveSheet.Cells(m, n), Chr(32))(0))
If Split(ActiveSheet.Cells(m, n), Chr(32))(1) < 0 Then
ActiveSheet.Cells(m, lc + 11) = "PRE"
Else
ActiveSheet.Cells(m, lc + 11) = "POST"
End If
Next
End If
```

Figure 10. Code to transform visit windows related tables

Figure 11 shows the specific code of transformation of Criteria related tables.

```
If InStr(UCase(Range("A" & j)), "CHANGE") <= 0 And InStr(UCase(Range("A" & j)), "INCREASE") <= 0 And InStr(UCase(Range("A" & j)), "DECREASE") <= 0 Then
Cells(i + 1, 1c + 4 + n) = Replace(Cells(j, n), dict.Keys, dict.Items)
If Cells(i + 1, 1c + 4 + n) Like ">=*AVAL <*" Then
Cells(i + 1, 1c + 4 + n) = Mid(Split(Cells(i + 1, 1c + 4 + n), " ", 2)(0), 3) & "<= " & Split(Cells(i + 1, 1c + 4 + n), " ", 2)(1)
ElseIf Cells(i + 1, 1c + 4 + n) Like ">*AVAL <*" Then
Cells(i + 1, 1c + 4 + n) = Mid(Split(Cells(i + 1, 1c + 4 + n), " ", 2)(0), 2) & "< " & Split(Cells(i + 1, 1c + 4 + n), " ", 2)(1)
ElseIf (Cells(i + 1, 1c + 4 + n) Like ">*" Or Cells(i + 1, 1c + 4 + n) Like "<*") Then
Cells(i + 1, 1c + 4 + n) = "AVAL " & Cells(i + 1, 1c + 4 + n)
End If
If Cells(i + 1, 1c + 4 + n) Like "AVAL <*" Then
Cells(i + 1, 1c + 4 + n) = Cells(i + 1, 1c + 4 + n) & " and AVAL ne ."
End If
End If
```

Figure 11. Code to transform criteria related tables

PART 3 PROTOCOL TOOL

Reviewing the Schedule of Activities (SoA) in a Protocol is an integral part of a programmer's daily work. The SoA forms a crucial part of a Clinical Trial Protocol. It is usually presented as a table listing the study visits as a timeline, and the procedures and events associated with each visit, which are keys to understanding how the objectives of the study are to be implemented. However, Schedule of Activities (SoA) is usually stored in a word File so that operations such as filtering cannot be performed. In this case, Protocol Tool realizes the import of SoA into Excel as a table, enabling seamless integration of SoA with Excel built-in functions.

Once the "Schedule of Activities" command button is clicked, an open file dialog will appear as Figure 5 User needs to select the required Protocol file and a pop-up message box can remind user if the Protocol file contains SoA. This function can also be used in other scenarios such as review protocol.

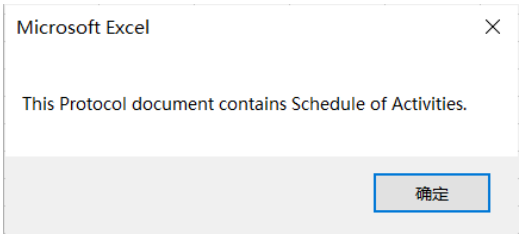
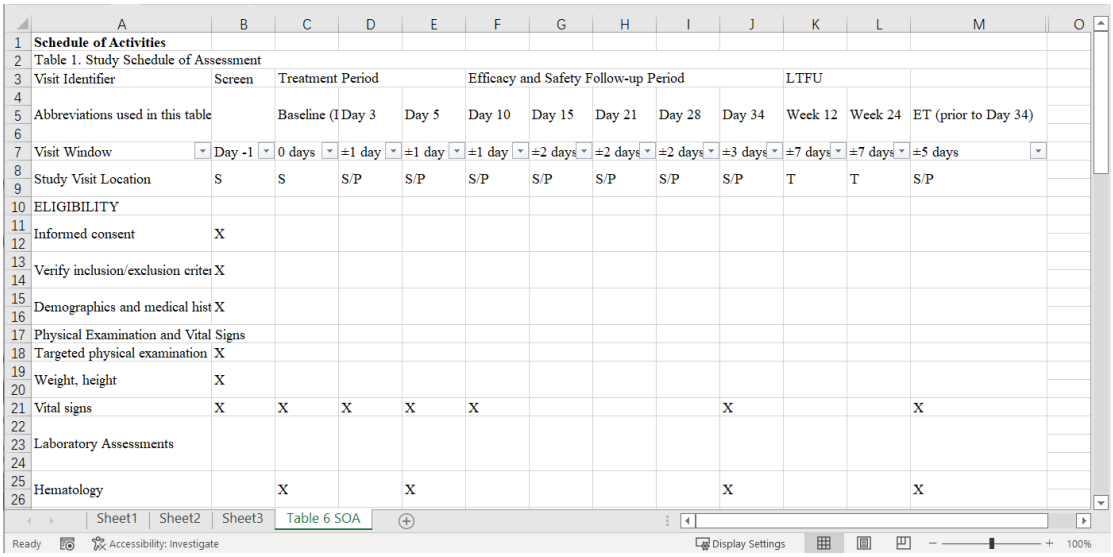


Figure 12. Pop-up message box to confirm the existence of SoA in the Protocol

After confirmation, the Schedule of Activities can be displayed in Excel as shown in Figure 13. Excel built-in functions such as filtering, sorting, plotting, etc. can now be performed for SoA to help programmers to get an accurate view of what each visit needs and enhance data analysis capabilities. For instance, check out the schedule to Vital Signs and ECG.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Schedule of Activities**												
2	Table 1. Study Schedule of Assessment												
3	Visit Identifier	Screen	Treatment Period			Efficacy and Safety Follow-up Period					LTFU		
5	Abbreviations used in this table		Baseline (I	Day 3	Day 5	Day 10	Day 15	Day 21	Day 28	Day 34	Week 12	Week 24	ET (prior to Day 34)
7	Visit Window	Day -1	0 days	±1 day	±1 day	±1 day	±2 days	±2 days	±2 days	±3 days	±7 days	±7 days	±5 days
9	Study Visit Location	S	S	S/P	S/P	S/P	S/P	S/P	S/P	S/P	T	T	S/P
10	ELIGIBILITY												
12	Informed consent	X											
14	Verify inclusion/exclusion crite	X											
16	Demographics and medical hist	X											
17	Physical Examination and Vital Signs												
18	Targeted physical examination	X											
20	Weight, height	X											
21	Vital signs	X	X	X	X	X				X			X
23	Laboratory Assessments												
25	Hematology		X		X					X			X

Sheet1 | Sheet2 | Sheet3 | Table 6 SOA

Ready | Accessibility: Investigate | Display Settings | 100%

Figure 13. Example of SoA in Excel

CONCLUSION

After testing on several Protocols and SAPs, this tool shows the ability to accurately extract relevant information, reducing the potential for errors and improving programming quality. Overall, the VBA tool presented in this paper offers a practical solution for improving productivity and reducing errors in statistical programming while ensuring compliance with industry standards.

Using VBA is a good starting point for automation. By automating repetitive or time-consuming tasks, programmer can save time and minimize the risks of errors or inconsistencies. There are more to explore how to integrate VBA or other programming languages into a programmer's daily workflow to unlock more advanced automation capabilities.

REFERENCES

"Word VBA reference"

https://learn.microsoft.com/en-us/office/vba/api/overview/word

Ben, Chinowsky. 2014. "Getting Started with VBA in Excel 2010"

https://msdn.microsoft.com/en-us/library/office/ee814737(v=office.14).aspx.

Andreu, Fernando. 2022. "Office RibbonX Editor"

https://github.com/fernandreu/office-ribbonx-editor

CONTACT INFORMATION

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

Jingxin Fu

Pfizer (China) Research and Development Co.,Ltd. China

Jingxin.fu@Pfizer.com

Shuting Lou

Pfizer (China) Research and Development Co.,Ltd. China

shuting.lou@pfizer.com

Yun Lu

Pfizer (China) Research and Development Co.,Ltd. China

yun.lu@pfizer.com