PharmaSUG China 2018 – Paper CC-23

Make your program tracking sheet more powerful - Using VBA

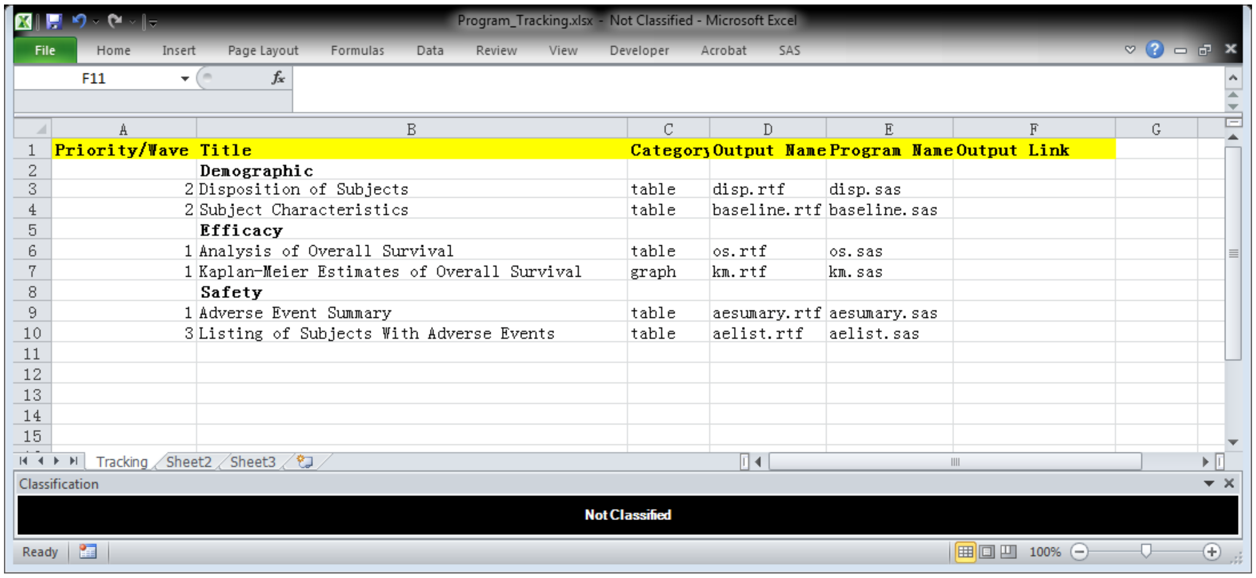
Pengfei Guo, MSD R&D (China) Co., Ltd., Shanghai, China
Jing Zhang, MSD R&D (China) Co., Ltd., Shanghai, China

ABSTRACT

Most of programmers use programming tracking sheet mainly as a format of Excel sheet to record programs names, output names, categories and other useful information in their daily work. In this paper, we will share some VBA macros to make the tracking sheet more powerful. Not only a tracking tool but also can help programmer automatically finish repeating and time-consuming tasks such as adding outputs hyperlinks and move/copy outputs to specified folders.

INTRODUCTION

Although in various names and formats, the program tracking sheet with some common nature is widely used in SAS programmer's daily work. The most common format is Excel sheet. Please see blew Display 1as an example of program tracking sheet.



Priority/Wave	Title	Category	Output Name	Program Name	Output Link
	Demographic				
2	Disposition of Subjects	table	disp.rtf	disp.sas	
2	Subject Characteristics	table	baseline.rtf	baseline.sas	
	Efficacy				
1	Analysis of Overall Survival	table	os.rtf	os.sas	
1	Kaplan-Meier Estimates of Overall Survival	graph	km.rtf	km.sas	
	Safety				
1	Adverse Event Summary	table	aesumary.rtf	aesumary.sas	
3	Listing of Subjects With Adverse Events	table	aelist.rtf	aelist.sas	

Display 1 Screen Capture of an example of program tracking sheet

Since it is excel based document, we can use VBA to help us complete some repeatable and time-consuming tasks such as adding outputs hyperlinks and move/copy outputs to specified folders.

VBA stands for Visual Basic for Applications an event-driven programming language from Microsoft that is now predominantly used with Microsoft office applications such as M-Excel, MS-Word, and MS-Access. It is called glue code and have similar natures compared with SAS which makes it is easy to learn for SAS programmer.

VBA OVERVIEW

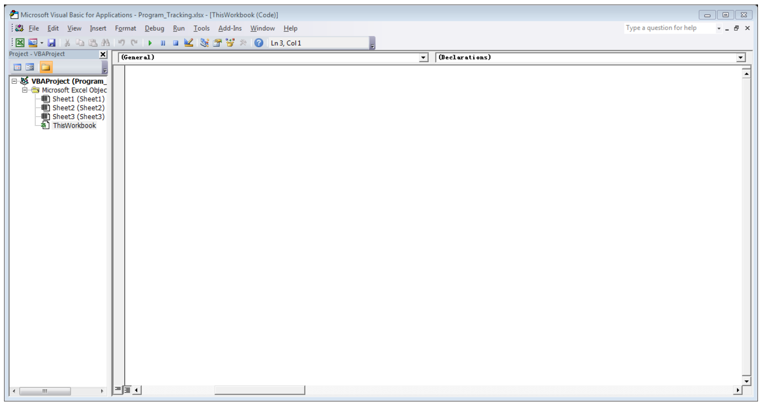
Blew are some basic concepts in VBA which make the programmer understand VBA code quickly and clearly.

ACCESSING VBA EDITOR

There are several methods to open a VBA editor, blew are 2 usually used ways:

1. In Excel 2010 and 2013 click the “File” menu then select “Options”. From the dialogue box, click on “Customize Ribbon” on the left side. From the right hand side you'll then see an area called "Customize the Ribbon". Under "Main Tabs" check the box for “Developer”. Then you can find the Developer tab next to the View tab. Click “Visual Basic” button under that tab.
2. In Excel window, press "ALT+F11".

Whichever method you choose you should see a screen like this Display 2.



Display 2 Screen Capture of VBA Editor

SUB PROCEDURES

A Sub is a small chunk of code to do a specific job. Sub procedures are always enclosed within Sub and End Sub statements. You can also assign sub variable by placing variable names between the round brackets. Each variable is separated by a comma.

```
Sub SecondCode(VarName1, VarName2)
    statement …
End Sub
```

As well as specifying the variable names, you can specify a variable type, as well:

```
Sub SecondCode(VarName1 As Boolean, VarName2 As String)
    statement …
End Sub
```

VARIABLES

Variable is a named memory location used to hold a value that can be changed during the script execution. In VBA, you need to declare the variables before using them. There are many VBA data types, which can be divided into two main categories, namely numeric and non-numeric data types.

```
Dim Variable_name As Variable_type
```

Following are the basic rules for naming a variable.

- You can't start a variable name with a number.
- You can't use Visual Basic reserved keywords as variable name.
- You can't use a space, period (.), exclamation mark (!), or the characters @, &, $, # in the name.
- Name can't exceed 255 characters in length.

LOOPS

VBA also use Loops to execute a block of code multiple times. VBA provides the several types of loops to handle looping requirements. The usually used can be divided into 2 categories: For Loops and Do Loops. Please see blew Table 1for syntax summaries.

Table 1 Syntax for Loops

Categories		Syntax

For Loops	for loop	`For counter = start to end <Step stepcount>`
		`   statement …`
		`next`

	for … each loop	`For each element in group`
		`   statement …`
		`next`

Do Loops	do … while loop	`Do while condition`
		`   statement …`
		`loop`

		`Do`
		`    statement …`
		`loop while condition`

	do … until loop	`Do until condition`
		`   statement …`
		`loop`

		`Do`
		`    statement …`
		`loop until condition`

IF THEN STATEMENT

Similar to SAS program, the VBA has if-then conditional statement to control the execution flow of a script. The statement will be closed with “End If”. The syntax is as blew:

```
If Condition_To_Test1 Then
    Executed code 1
<ElseIf Condition_To_Test2 Then
    Executed code 2>
<Else
    Executed code else>
End If
```

OBJECT MODEL

The VBA Object Model is the hierarchy of all of the programming objects in Excel, each of which has its own set of characteristics. Now take a book in real world as an example.

Objects: Chapter, Page, etc.

Properties: Price, Weight, etc.

Methods: Open, Closed, etc.

Similarly we have the same type of characteristics for Excel Objects. You can observe the following Worksheet characteristics. A Worksheet can have the following characteristics:

Objects: Range, Cell, etc.

Properties: Sheet Name, Sheet Color, etc.

Methods: Select, Activate, Copy, Paste, etc.

They are connected with dot (period) and a property of one subject may also be an object for next level. Please see blew an example:

```
   Application.ActiveSheet.Range("A1").Select
```

The ActiveSheet is a property of Application and also an object for next level.

Table 2 is summary of comparisons of SAS and VBA.

Table 2 Comparisons of SAS and VBA

	Common	**Difference**	
		SAS	**VBA**
MACRO/SUB	1.Enclosed within key words; 2.Can be nested; 3.Revoked by keywords	1.Both positional parameters and keyword parameters	1.No keyword parameters
VARIABLES	1.Naming Convention	1.Data type and format separately	1. Combined type; 2. Variant (Numeric or String)
LOOPS	1. Enclosed in block; 2. expressions :Start to stop|While|Until	1.Only have Do loop	1. Do & For loop; 2. Condition can be end of block
IF THEN STATEMENT	1.Can be multiple conditions	1.Space in “Else If”	1.No Space in “ElseIf”; 2.End with “End If”

VBA CODE

Now let us turn back to the initial objectives. Please see blew Figure 1 the program flow chart.

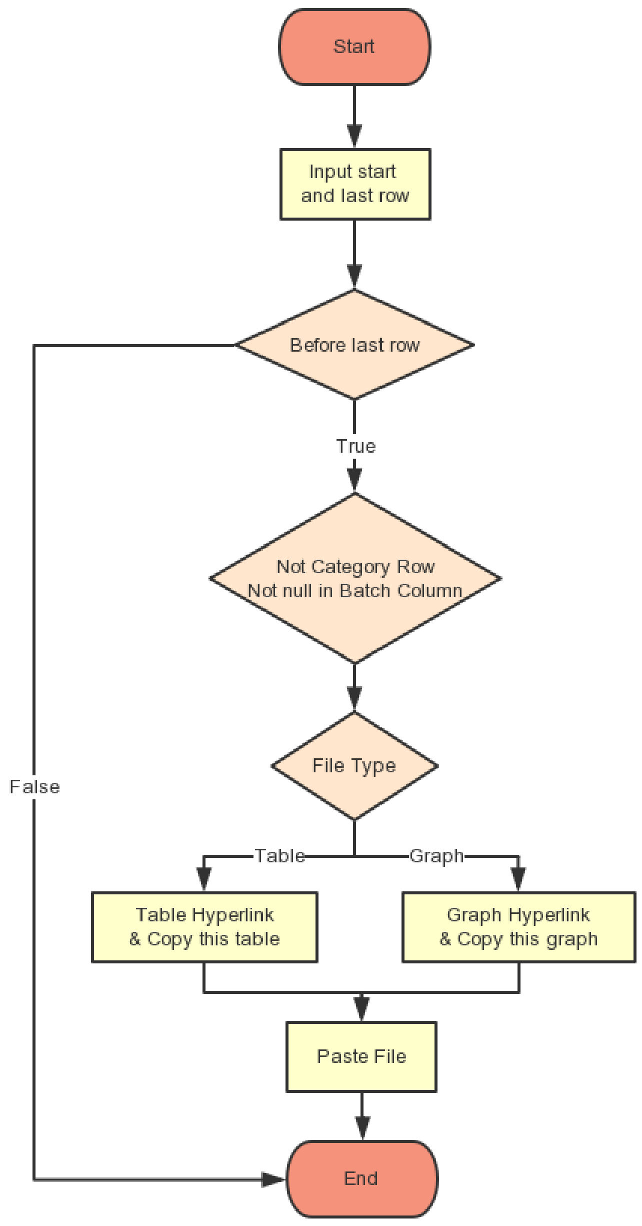

Start
Input start and last row
Before last row
True
False
Not Category Row Not null in Batch Column
File Type
Table
Graph
Table Hyperlink & Copy this table
Graph Hyperlink & Copy this graph
Paste File
End

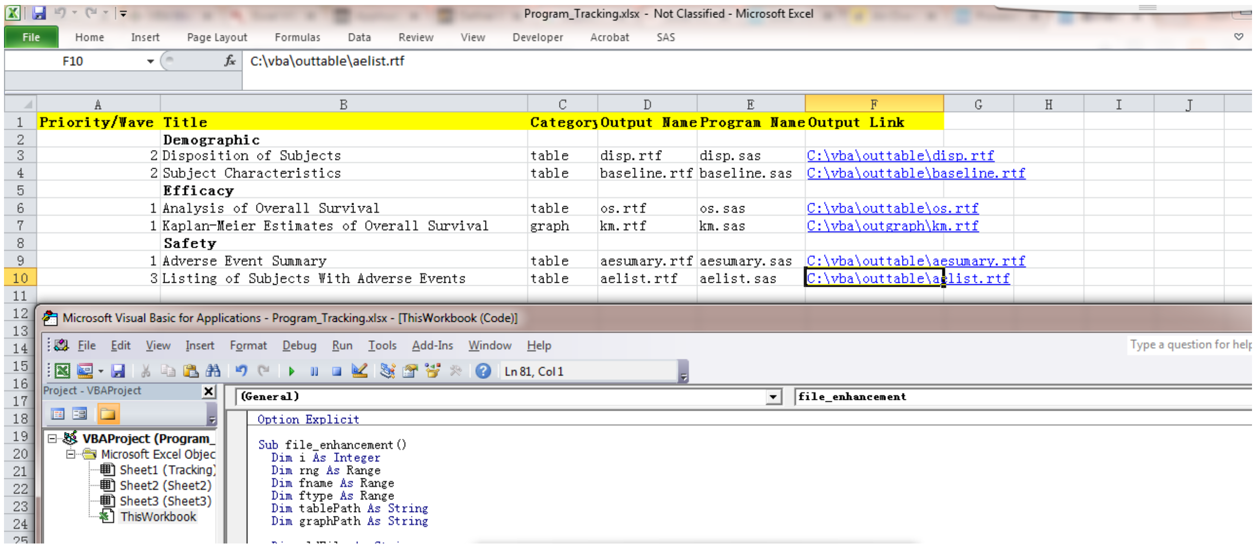
Figure 1 Program Flow Chart

The source code is also provided.

```
Sub file_enhancement()
  Dim i As Integer
  Dim rng As Range
  Dim fname As Range
  Dim ftype As Range
  Dim tablePath As String
  Dim graphPath As String

  Dim oldFile As String
  Dim newFile As String

  '=================================================
  'Set the paths to the folders you are processing
  'REMEMBER END BACKSLASH
  '=================================================
  tablePath = "C:\vba\outtable\"
  graphPath = "C:\vba\outgraph\"


  '===================================================
  'loop through column B to the last row you filled
  '===================================================
  i = 1

  Do While i < 15

  '==========================================================
  'set up the start of the range you want to loop through
  'EDIT SHEET NAME AND FIRST CELL ADDRESS IF NECESSARY
  '=========================================================
  Set rng = Sheets("Tracking").Range("A" & i)
  Set fname = Sheets("Tracking").Range("D" & i)
  Set ftype = Sheets("Tracking").Range("C" & i)



      If rng.Value = 1 Or rng.Value = 2 Or rng.Value = 3 Then

        'Output name ne null
        If fname.Value <> "" Then

            'Grahp
            If ftype.Value = "graph" Then

                'build up the full path to the old file
                 oldFile = graphPath & fname.Value

                 Range("F" & i).Select

                 ActiveSheet.Hyperlinks.Add anchor:=Selection, Address:=graphPath &
                  fname.Value


            'Table
            ElseIf ftype.Value = "table" Then

                'build up the full path to the old file
                 oldFile = tablePath & fname.Value

                 Range("F" & i).Select

                 ActiveSheet.Hyperlinks.Add anchor:=Selection, Address:=tablePath &
                  fname.Value
            End If

            'build up the full path to the new file you want to create
             newFile = "C:\vba\By_batch\Batch" & rng.Value & "\" & fname.Value
            'copy the old file to the new new folder
             FileCopy oldFile, newFile

        End If

      End If

    'Move to next row
    i = i + 1
  Loop
End Sub
```

RESULT

After the code is run, the program tracking sheet is filled with outputs hyperlinks in Display 3 and the outputs are copies into related subfolders.


Program_Tracking.xlsx - Not Classified - Microsoft Excel
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Developer
Acrobat
SAS
F10
C:\vba\outtable\aelist.rtf

Priority/Wave	Title	Category	Output Name	Program Name	Output Link
	Demographic				
2	Disposition of Subjects	table	disp.rtf	disp.sas	C:\vba\outtable\disp.rtf
2	Subject Characteristics	table	baseline.rtf	baseline.sas	C:\vba\outtable\baseline.rtf
	Efficacy				
1	Analysis of Overall Survival	table	os.rtf	os.sas	C:\vba\outtable\os.rtf
1	Kaplan-Meier Estimates of Overall Survival	graph	km.rtf	km.sas	C:\vba\outgraph\km.rtf
	Safety				
1	Adverse Event Summary	table	aesumary.rtf	aesumary.sas	C:\vba\outtable\aesumary.rtf
3	Listing of Subjects With Adverse Events	table	aelist.rtf	aelist.sas	C:\vba\outtable\aelist.rtf

```
Option Explicit

Sub file_enhancement()
  Dim i As Integer
  Dim rng As Range
  Dim fname As Range
  Dim ftype As Range
  Dim tablePath As String
  Dim graphPath As String
```

Display 3 Program Tracking Sheet updated

CONCLUSION

The VBA script has some common natures compared with SAS and is easy-to-learn language for SAS programmer. It can help you complete repeatable and time-consuming tasks combining with your MS-EXCEL/WORD/ACCESS documents. This paper is just a simple example. You can adapt you own VBA code based on your business need.

REFERENCES

"Getting Started with VBA in Office". Available at https://msdn.microsoft.com/en-us/vba/office-shared-vba/articles/getting-started-with-vba-in-office

"Excel VBA Programming". Available at http://www.homeandlearn.org/index.html

CONTACT INFORMATION

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

Name: Pengfei Guo
Enterprise: MSD R&D (China) Co., Ltd.
Address: Building A, Headquarters Park, Phase 2, 1582 Gumei Road, Xuhui District
City, State ZIP: Shanghai, 200233
Work Phone: +86 21 2211 8543
Fax: +86 21 2211 8899
E-mail: peng.fei.guo1@merck.com

Name: Jing Zhang
Enterprise: MSD R&D (China) Co., Ltd.
Address: Building A, Headquarters Park, Phase 2, 1582 Gumei Road, Xuhui District
City, State ZIP: Shanghai, 200233
Work Phone: +86 21 2211 8736
Fax: +86 21 2211 8899
E-mail: jing.zhang18@merck.com