PhUSE US Connect 2018

Paper CT14

Push a Button in Excel to run all Your SAS Programs

Taylor Markway, Sarah Cannon Development Innovations, Chapel Hill, NC USA

ABSTRACT

This paper will show you how to use Visual Basic for Applications (VBA) to run multiple programs listed in Excel. We will first look at an example which includes various validation reports.. Through this technique, audiences familiar with VBA will acquire the tools to create the main framework and to add personalized features based on individual need.

INTRODUCTION

This paper presents a SAS program batching technique using Excel and VBA.The technique allows users to set up VBA and generate a .bat file to run multiple Excel programs. Modifications can be made to accommodate various business rules and computing environments.

CREATING A BASIC VERSION

1. ADD THE VBA CODE

Add the code in Appendix 1 (2 subroutines and 1 function) to an Excel workbook. Detailed steps included below:

1) With Excel open press ALT+F11, this launches the VBA Editor
2) Double click on “ThisWorkbook” under the workbook you have open, in the example this is “PhuseButton.xlsx”
3) Copy and paste the code from appendix 1 to the text field where it says “Option Explicit” in Figure 1.

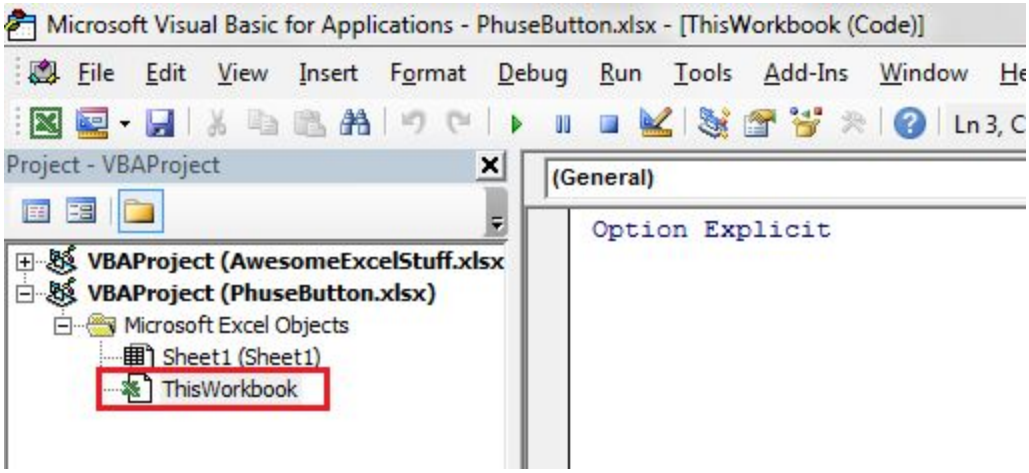

Figure 1. Excel VBA Editor

2. ADD A BUTTON

Add a button to your spreadsheet with the top left hand corner above your list of programs. Assign the button to ThisWorkbook.CreateBatch.

If you are having trouble, try these steps and resources below:

1. With your spreadsheet open, you first need to make sure you have the Developer tab turned on. See MS Devloper Tab Support.
2. Insert a button from the developer controls part of the ribbon, see Figure 2.

3. Assign the button to ThisWorkbook.CreateBatch macro. See MS Assign a macro to a Form or a Control Button
4. The button will have the standard name “Button 1”, which can be renamed “CreateBatch”

You should now have a button in place looking something like the workbook in Figure 3.

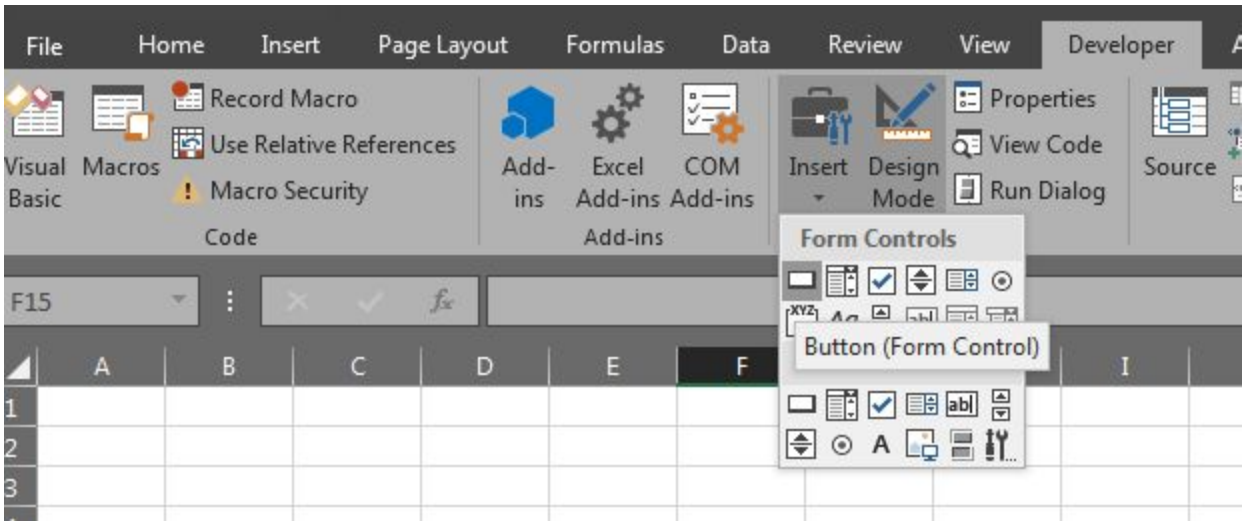

Figure 2. Developer Tab Add Button

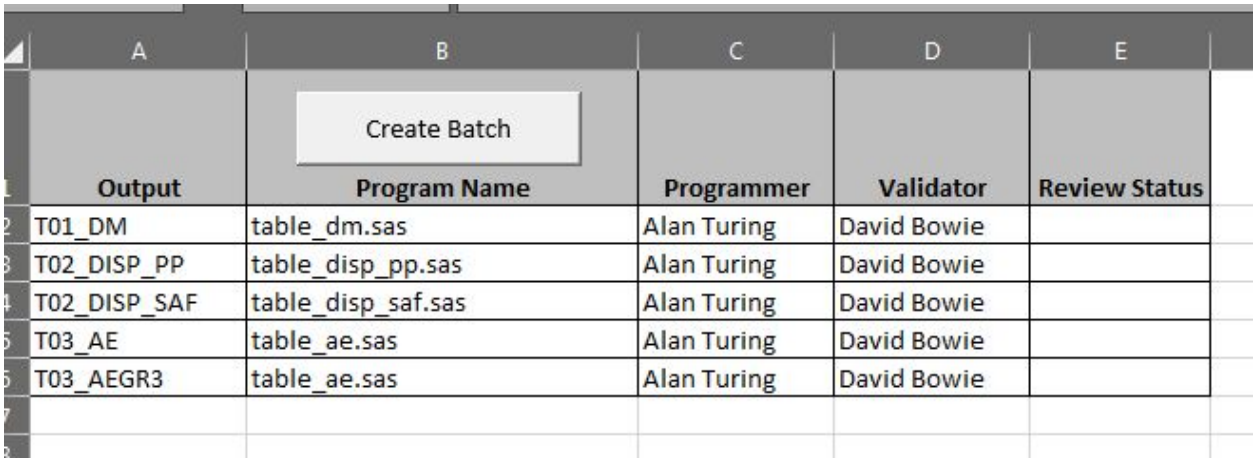

Figure 3. Button Added to Program Tracker

3. CREATE THE USER FORM

The setup requires a user form which collects the path of SAS programs and the path of the VBA generated batch file including the filename.

You will need to

1. Insert a UserForm (Figure 4)
2. Name the UserForm BatchForm (Figure 5)
3. Add two text fields named: BatchPath and ProgPath. (Note: VBA is case sensitive).

This UserForm is where you can add additional user settings.

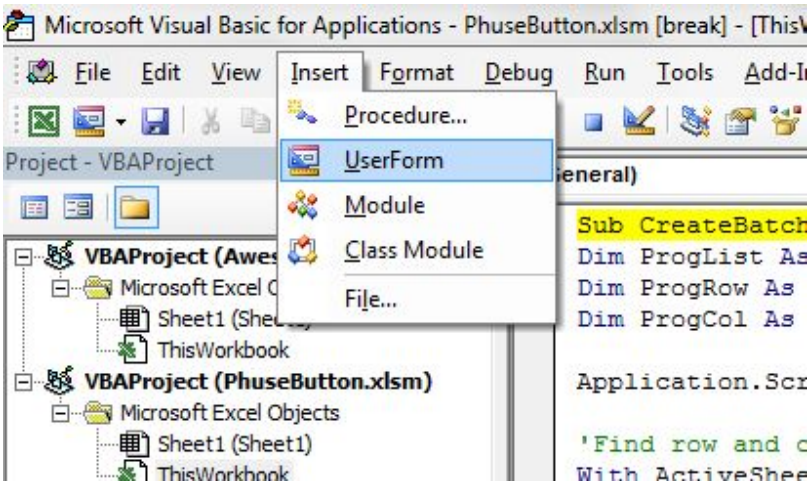

Figure 4. Inserting a UserForm

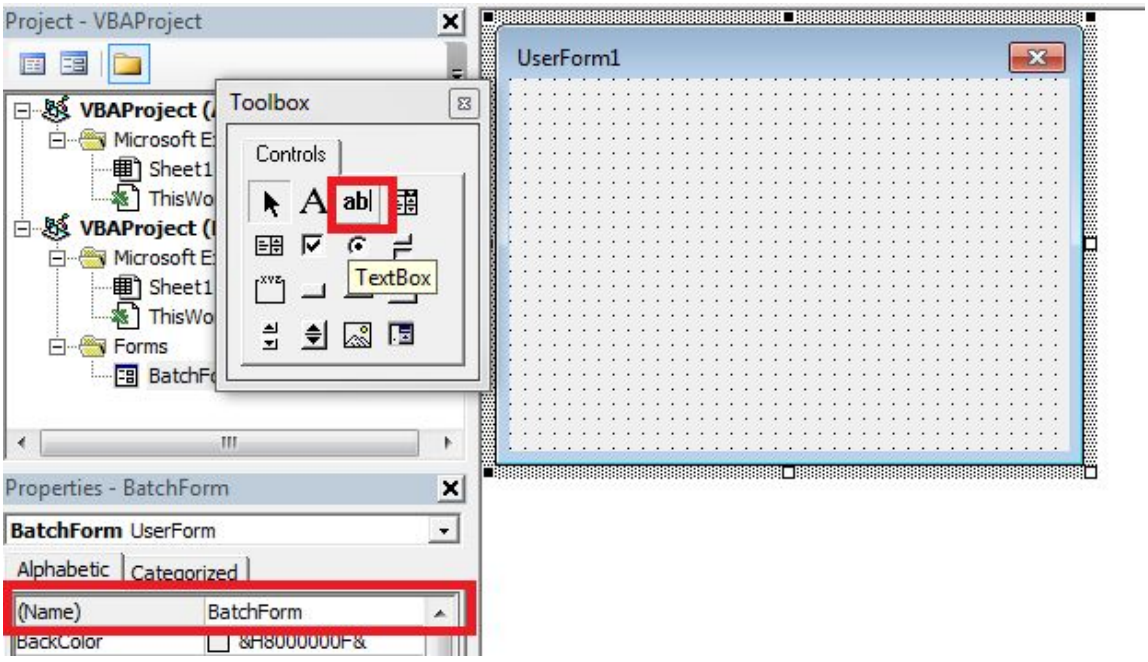

Figure 5. Adding TextBox to UserForms to Collect Input

The last step is to label the fields accordingly, and to add in a button to close the form and continue processing the VBA.

In the UserForm editor click on the CommandButton in the Toolbox to add it to the form, Figure 6. The name here does not matter for the VBA code provided. Update the caption to something meaningful and then double click on the button. It should bring up an editor where you can add the text “me.Hide” in the subroutine as in Figure 7.

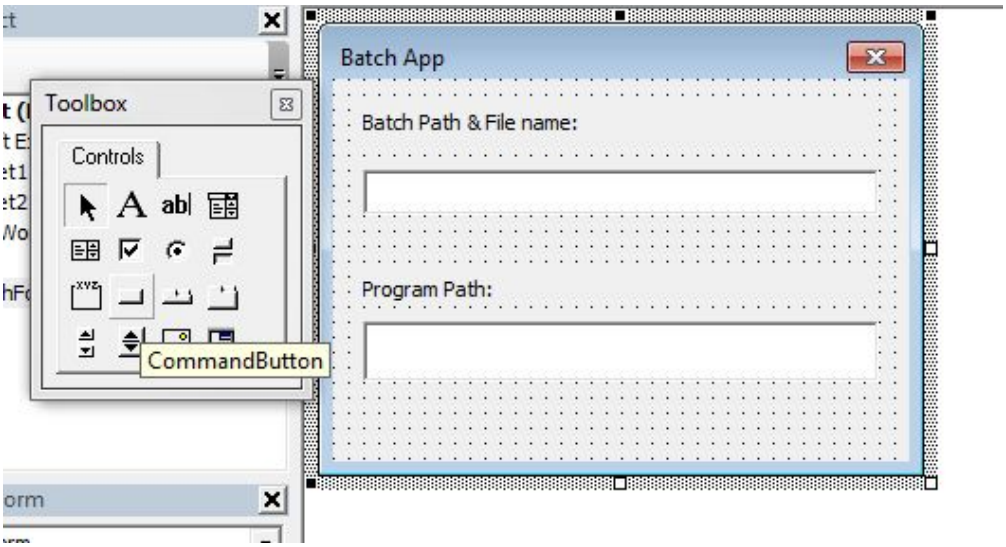

Figure 6. Adding CommandButton to the UserForm

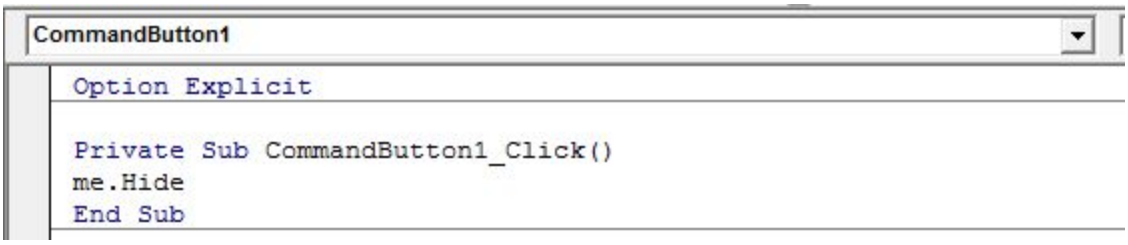

Figure 7. Adding Sub to CommandButton

4. UPDATE THE SAS COMMAND

In the writeBatch subroutine at the end of the appendix there is a line of VBA

```
Print #FileNum, "C:\sas.exe - sysin %ProgPath%\" & ProgName.Value & ".sas"
```

This line will need to be updated based on your installation of SAS and computing environment.

All the setup is now complete and you should now have a working version. When you click the button the UserForm you created should popup and you should be able to input the appropriate paths and file names.

EXAMPLE WITH ADDITIONAL FEATURES

Leveraging standard directory structures and SAS validation macros we were able to add features to generate standard reports as part of the batch process whenever we use the application to write our batch files for us.

Figure 8 is a screenshot from an Excel spreadsheet used to track programs. In this example, we are tracking the SAS programs to create SDTM datasets for particular client deliveries. It includes the SDTM output name, the program name, and columns to indicate if an output is required for a delivery. Above each of the delivery columns is a “Run Programs” button. This button launches the VBA macro similar to the one built earlier in the paper.

Dataset Name	Run Programs / Program Name	Program Version	Deliverables					
			Safety Run Programs	IB/FDA Run Programs	Cohort Review Run Programs	DSUR Run Programs	ASCO Run Programs	CSR Run Programs
DM	d0_dm		d0_dm	d0_dm	d0_dm	d0_dm	d0_dm	d0_dm
SUPPDM	d0_dm		d0_dm	d0_dm	d0_dm	d0_dm	d0_dm	d0_dm
TA	d0_ta		d0_ta	d0_ta	d0_ta	d0_ta	d0_ta	d0_ta
TE	d0_te		d0_te	d0_te	d0_te	d0_te	d0_te	d0_te
TI	d0_ti		d0_ti	d0_ti	d0_ti	d0_ti	d0_ti	d0_ti
TS	d0_ts		d0_ts	d0_ts	d0_ts	d0_ts	d0_ts	d0_ts
TV	d0_tv		d0_tv	d0_tv	d0_tv	d0_tv	d0_tv	d0_tv
AE	d1_ae		d1_ae	d1_ae	d1_ae	d1_ae	d1_ae	d1_ae
SUPPAE	d1_ae		d1_ae	d1_ae	d1_ae	d1_ae	d1_ae	d1_ae
CE	d1_ce						d1_ce	d1_ce

Figure 8. Run Programs Button in Excel

When a user clicks the “Run Programs” button the form in Figure 9 pops up. In our version this form is dynamic and changes for different types of outputs, be it SDTM, ADaM or TFL outputs. The form in Figure 9 is for SDTM programs. This is where users set options such as the program directory and delivery name. They can also check that all the correct programs are going to be included in the batch file. The “See Programs List” button brings up a list of all the programs that are included in the batch file.

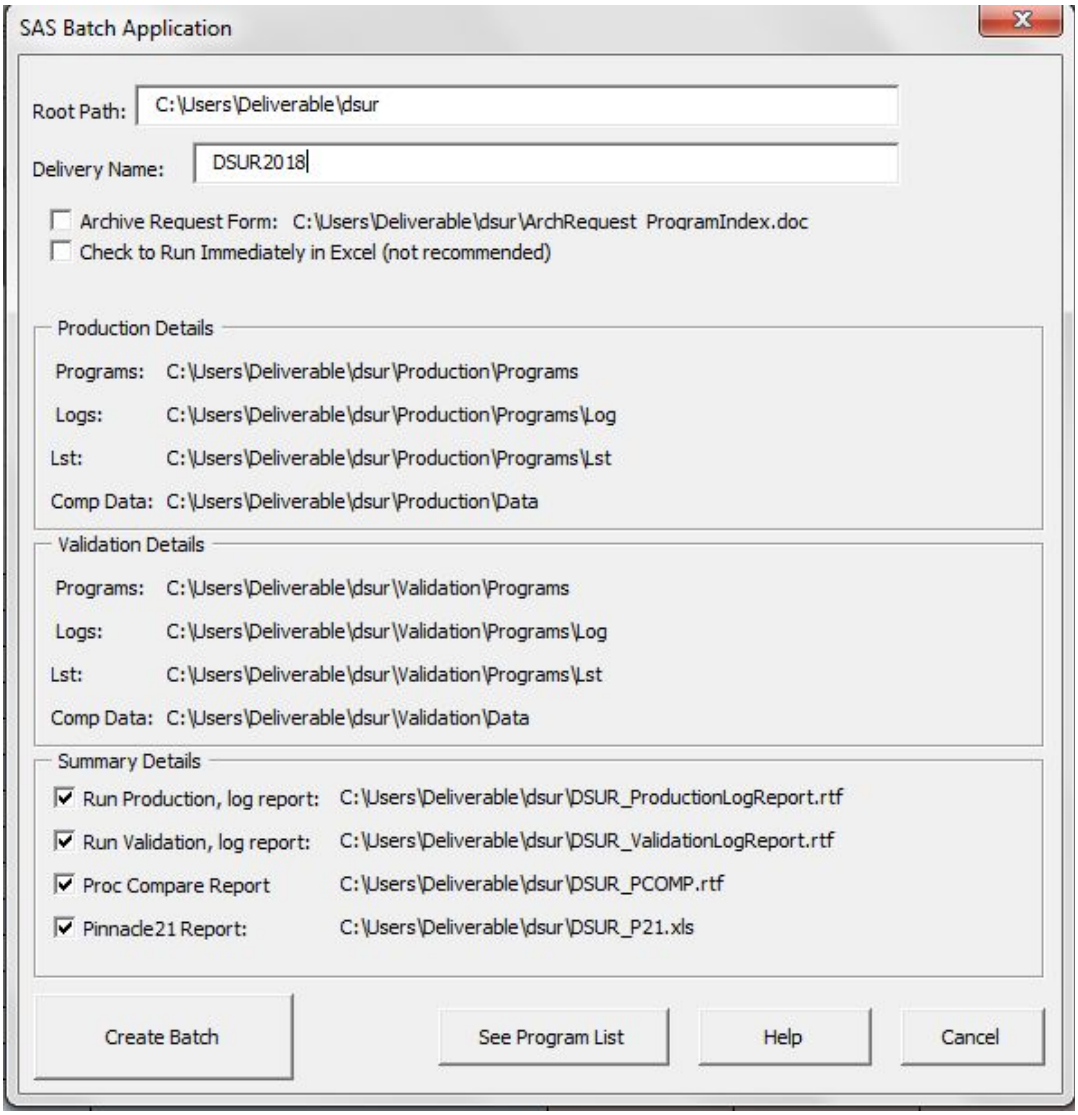

Figure 9. Batch Run Menu

Above the “See Programs List’ button is the Summary Details section. This is where a user can elect to include our standard log and/or PROC COMPARE summary reports. This paper will not cover the code to create the summary reports. The system paths on the right of the user form cannot be edited since these are determined by our business rules. Instead, we allow users to edit a “Root Path” for the delivery and the file paths in the details section will update automatically.

In this example, clicking on the “Create Batch” button will create a .bat file named DSUR2018.bat. If the “Run Immediately in Excel” option was selected it would also immediately launch the .bat file. This can tie up resources in Excel for the duration of the batch process and is not recommended.

We now have the batch file in Figure 9, DSUR2018.bat. Clicking or scheduling the batch will execute our SAS programs that were in the Excel file, then check our logs, compare our datasets, and create a Pinnacle 21 report for all SDTM datasets included in this delivery.

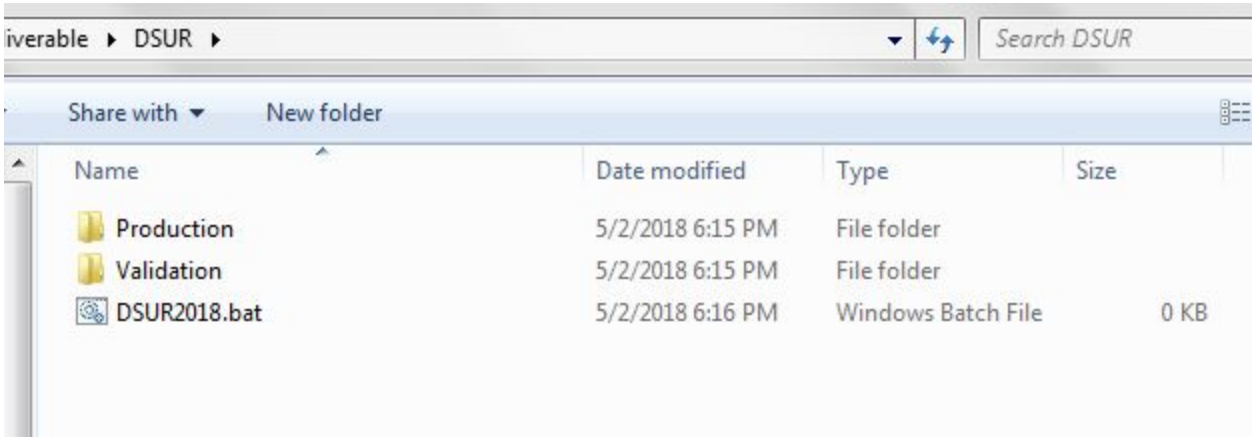

Figure 10. Batch File Output

ADDING FEATURES

If this was your first time working with VBA I’d recommend checking out a additional resources in order to include all the features you find necessary. Getting Started With VBA in Excel 2010 is a good place to start. To include Pinnacle 21, check out How to Automate Validation with Pinnacle 21 Command Line Interface and SAS by Garrett and Vinokurov.

CONCLUSION

As studies progress and the scope of deliveries expand, programs may be included or excluded for each delivery. If Excel is already being used to keep track of this information then this VBA batching technique is a way to quickly create batch files to execute multiple SAS programs.

REFERENCES

Getting Started With VBA in Excel 2010, https://msdn.microsoft.com/en-us/library/office/ee814737(v=office.14).aspx

MS Support, Developer Tab Support, https://support.office.com/en-us/article/show-the-developer-tab-e1192344-5e56-4d45-931b-e5fd9bea2d45

MS Support, Assign a macro to a Form or a Control Button https://support.office.com/en-us/article/assign-a-macro-to-a-form-or-a-control-button-d58edd7d-cb04-4964-bead-9c72c843a283

MS Support, Automate tasks with the Macro Recorder https://support.office.com/en-us/article/Automate-tasks-with-the-Macro-Recorder-974ef220-f716-4e01-b015-3ea70e64937b

Lisa Sanbonmatsu (NESUG 00), Batch Processing with SAS®: Beyond Running Programs Overnight, http://www.lexjansen.com/nesug/nesug00/cc/cc4017.pdf

Helen Sun and Cindy Wong (SAS Global Forum 2010), A Macro to Create a Batch Submit SAS Program, http://support.sas.com/resources/papers/proceedings10/092-2010.pdf

Chen and Gilbert (SUGI 27), Run All Your SAS Programs in One Program: Automatically, http://www2.sas.com/proceedings/sugi27/p105-27.pdf

Lisa Mendez (SAS Global Forum 2011), Create a Main Program to Run Multiple SAS Programs: Utilizig the %INCLUDE Statment. http://support.sas.com/resources/papers/proceedings11/100-2011.pdf

Amy Garrett and Aleksey Vinokurov (PharmaSUG 2018), How to Automate Validation with Pinnacle 21 Command Line Interface and SAS, https://www.pharmasug.org/proceedings/2018/DS/PharmaSUG-2018-DS20.pdf

RECOMMENDED READING

Jennie McGuirk and Rachna Mishra (PhUSE 2017), Programming Development and Validation Tracking Application, http://www.lexjansen.com/phuse/2017/pp/PP11.pdf

CONTACT INFORMATION

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

Taylor Markway
Sarah Cannon
1100 Charlotte Ave
Nashville, TN 37203
Work Phone: 615-329-7274
Email: Taylor.Markway@SarahCannon.com

APPENDIX 1

```
'==============================================================================
' Module Desc.: Main sub for creating the batch file
' Purpose     : Write a batch file to execute SAS programs via a button
' Note        :
'==============================================================================
Option Explicit


Sub CreateBatch()
Dim ProgList As Range
Dim ProgRow As Integer
Dim ProgCol As Integer

Application.ScreenUpdating = False

'Find row and column of cell beneath top left hand corner
With ActiveSheet.Buttons(Application.Caller).TopLeftCell
    ProgRow = .Row + 1
    ProgCol = .Column
End With

'Turn row and column data into list of programs
Set ProgList = Programs(ProgRow, ProgCol)

'Get user input
BatchForm.Show

'Write commands to run SAS
Call WriteBatch(BatchForm.ProgPath.Value, BatchForm.BatchPath.Value, ProgList)

Application.ScreenUpdating = True
End Sub

'==============================================================================
' Function Desc.: Find all programs to run
' Purpose       : Given a Row & Column, return all unique values below the given row
' Note          : returns a range
'==============================================================================
Function Programs(ProgRow As Integer, ProgCol As Integer, Optional AsRange As Boolean)
As Range
Dim lRow, i As Integer
Dim ProgSource As Range
Dim ProgSheet As Object

Set ProgSheet = ActiveSheet

'Check that the Row and Col input have cells with program names
If IsEmpty(ActiveSheet.Range(Cells(ProgRow, ProgCol), Cells(ProgRow + 20, ProgCol)))
Then
    MsgBox "Value: " & ActiveSheet.Cells(ProgRow, ProgCol).Value _
            & vbNewLine & "IsEmpty: " & IsEmpty(ActiveSheet.Cells(ProgRow, ProgCol)) _
            & vbNewLine & "Cell(r,c): cell(" & ProgRow & " , " & ProgCol & ")"
    MsgBox ("Top-Left corner of button must be above program row. Move Button and try
again.")
```

```
End If

'From user input find the last cell with program names
lRow = Cells.Find(What:="*", _
                  After:=ActiveSheet.Cells(1, ProgCol), _
                  LookAt:=xlPart, _
                  LookIn:=xlValues, _
                  SearchOrder:=xlByRows, _
                  SearchDirection:=xlPrevious, _
                  MatchCase:=False).Row

'Create Program Name Source Range
Set Programs = Cells.Range(Cells(ProgRow, ProgCol), Cells(lRow, ProgCol))

'Create new sheet to manipulate data on
'Check if it already exists, if so delete to clear out old programs
With ActiveWorkbook
    For i = 1 To .Sheets.count
        If .Sheets(i).Name = "_DataSheet_" Then
            'Delete old data
            Application.DisplayAlerts = False
            .Sheets("_DataSheet_").Visible = xlSheetVisible
            .Sheets("_DataSheet_").Delete
            Application.DisplayAlerts = True
            Exit For
        End If
    Next i
End With

'Add new sheet to manipulate data on
With ActiveWorkbook
        .Sheets.Add
        With ActiveSheet
            .Name = "_DataSheet_"
            .Visible = xlVeryHidden
        End With
End With

'Copy the program range and remove duplicate program names
Programs.Copy

With ActiveWorkbook.Sheets("_DataSheet_")
    .Visible = xlSheetVisible
    .Activate
    .Range("A1").PasteSpecial _
            Paste:=xlPasteValues, _
            operation:=xlNone, _
            SkipBlanks:=False, _
            Transpose:=False
    .Range("A1:A" & lRow).RemoveDuplicates Columns:=1, Header:=xlNo
End With

'Update last row for unique program names
lRow = Cells.Find(What:="*", _
                  After:=ActiveWorkbook.Sheets("_DataSheet_").Cells(1, 1), _
                  LookAt:=xlPart, _
```

```
                    LookIn:=xlValues, _
                    SearchOrder:=xlByRows, _
                    SearchDirection:=xlPrevious, _
                    MatchCase:=False).Row


'Set output range containing program list
Set Programs = Cells.Range("A1:A" & lRow)

'Return focus on source sheet
ProgSheet.Select

End Function


'============================================================================
' Sub Desc.   : WriteBatch
' Purpose     : To write a batch file given a range containing all program names
' Note        :
'============================================================================

Sub WriteBatch(ProgPath As String, BatchFile As String, ProgList As Range)
Dim FileNum As Integer
Dim ProgName As Object

'Determine the next file number available for use by the FileOpen Function
FileNum = FreeFile

Open BatchFile For Output As FileNum

'Write lines of text for the BatchFile file
Print #FileNum, "@Echo off"
Print #FileNum, "echo ----------------------------------------------------------------"
Print #FileNum, "echo Batch Generated From File: " & ThisWorkbook.Name
Print #FileNum, "echo Batch Generated From Sheet: " & ActiveSheet.Name
Print #FileNum, "echo Batch Created By: " & Application.UserName
Print #FileNum, "echo Batch Create Date and Time: " & Now
Print #FileNum, "echo ----------------------------------------------------------------"
Print #FileNum, " "
Print #FileNum, "set ProgPath=" & ProgPath
Print #FileNum, " "

'Write command to run SAS and the specific program
For Each ProgName In ProgList
    MsgBox "ProgName Value: " & ProgName.Value & vbNewLine _
    & "ProgName.Address= " & ProgName.Address

    If ProgName.Value <> "" Then
        Print #FileNum, "echo Running " & ProgName.Value
        Print #FileNum, "C:\sas.exe - sysin %ProgPath%\" & ProgName.Value & ".sas"
    End If
Next ProgName

Print #FileNum, "Rem"
Print #FileNum, "Rem"
```

```
'Pause the batch program
Print #FileNum, "echo The batch has finished running, press any key to exit"
Print #FileNum, "pause > nul"
Print #FileNum, "cls"
Print #FileNum, "exit"

'Save and close batch file
Close FileNum

End Sub
```