

Developing SAS Application Using Microsoft Excel

Zhuan (John) Xu, 1ST Consulting LLC, West Des Moines, Iowa

ABSTRACT

The SAS system is a powerful tool for data and statistical analysis. SAS macro variables are introduced as parameters to a SAS program. This allows users familiar with the SAS system to perform various analyses. Many times, a user friendly graphic user interface is desirable for such SAS programs. Additionally, as analysts often load the result from SAS program into spreadsheet such as MS Excel for further analysis and presentation, it is helpful to automate the process.

This paper describes details of using MS Excel to create a GUI. The GUI will allow users to enter or select parameters and generate corresponding SAS macro variables. It will then run the SAS program and produce SAS output in the same Excel file. In this way, an SAS application is completely embedded inside a spreadsheet.

INTRODUCTION

This paper is a tutorial on integrating SAS system with MS Excel. We develop a SAS program and create some macro variables as parameters. On the Excel side, we create a graphic user interface and use VBA to generate SAS macro variables. We use OLE automation to submit SAS program from Excel. And we use Dynamic Data Exchange (DDE) to export SAS dataset to Excel.

Combining the powerful of the SAS system and the easiness of MS Excel will significantly enhance the productivity.

SAS MACRO VARIABLES AND ITS APPLICATION

SAS Macro variable is a parameter for a SAS program. It holds some text as its values. We use %LET, or CALL SYMPUT to create a macro variable. We apply the macro variable with an ampersand (&) in front of its name. With macro variable, one can just change a single value no matter how many times it is used in the program. Note that parameters can be set at the beginning of the program and thus outside the programming logic. It provides a simple way for other tools to interface with SAS program.

VISUAL BASIC FOR APPLICATION (VBA) IN MS EXCEL

VBA is a script language used in many applications. It is best thought of as Microsoft's common macro language. VBA is easy to learn. Its syntax is easy for SAS programmers. Excel in macro mode will generate VBA code automatically. In order to generate the VBA code, just turn on the macro record in Excel and then perform

some steps. Finally turn off the macro record. This will generate a macro program in VBA, which can be cleaned and modified.

VBA is powerful enough to perform many common tasks, such as starting another application, reading and writing files, query system etc.

NEEDS FOR INTEGRATION

Analysts often move data and results of analysis from SAS into Excel file for further ad hoc analysis or format the results for presentation. They may also use Excel to create charts and other graphics or integrate with other analysis.

The end users are often feel more comfortable to work within Excel environment. They are not training for SAS programming. But in many cases they have needs to revise SAS programs developed by professional SAS programmers. Although we SAS professionals often use macro variables to parameterize the SAS programs, some users may still make mistakes in revising the code.

We can use SAS/AF to providing a graphic user interface to a SAS program. But this requires SAS/AF skill and it also take more resources. Furthermore, there are distribution and user training issue for the SAS/AF applications.

For such users, it is ideal to provide a user interface in Excel, an environment they are familiar with, to run the SAS program. And have the result automatically loaded into the same Excel file.

INTEGRATION OF SAS WITH EXCEL

So we want to use SAS as a 'server' to work with data and perform analysis, and use MS Excel as front end. We can use VBA to link MS Excel to SAS together.

SAS support many methods to work with other PC applications. For example, we can use Dynamic Data Exchange (DDE) to export SAS dataset to Excel or import data from Excel. SAS can also act as a OLE automation server which allows some PC application to control SAS. Excel is one of the applications that can be used as a OLE client.

In the next part of the paper, we will use an example to present the details for developing such application. We will use a step by step approach.

CREAT SAS PROGRAM

Step 1: Review the Raw Data

We will work with data in a text file. The file contains the following data fields:

1. Company
2. # of Employee
3. Industry
4. Nation
5. Profits
6. Sales

The first three are for classification and last three can be used for analysis. Here is some sample data:

```
Lucas Industries 46 Automobiles Britain 39 3864
GKN              27 Automobiles Britain 58 3037
```

Step 2: Read data

Suppose the data file is located at c:\MWSUG\bdata.txt, we can use the following SAS program to create a SAS dataset BUSINESS in the same folder.

```
libname alib 'c:\sugi25e';
data alib.business;
  infile 'c:\sugi25e\bdata.txt';
  input
    @1 COMPANY      $20.
    @21 EMPLOYS     5.
    @27 INDUSTRY    $15.
    @43 NATION      $10.
    @54 PROFITS     10.
    @65 SALES       10.;
run;
```

Step 3: Create Summary

In this example, we will just summarize the data.

```
proc sort;
  by industry;
run;
proc means;
  by industry;
  var profits;
  title 'Profit by Industry';
run;
```

Step 4: Move the Output to a SAS Dataset

We can easily modify the above code to move the analysis result into a SAS dataset named SUMS.

```
proc means noprint data=alib.business;
  by industry;
  var profits;
  output out=sums
    n=n mean=mean sum=sum min=min max=max;
run;
```

Step 5: Refine the Output

In order to make the output looks better, we can add label and format and use PROC PRINT to generate a report.

```
proc print data=sums noobs label;
  title "Profits by industry";
  label industry = 'Industry'
    n = 'Count'
    mean = 'Average Profits'
    sum = 'Total Profits'
    min = 'Minimum'
    max = 'Maximum';
  format mean dollar10.2
    sum min max dollar10.0;
```

```
var industry n mean sum min max;
run;
```

Step 6: Use DDE to Export the Result to Excel

Using DDE to move export a SAS data set to Excel is pretty simple. We define a filename with DDE option and specify that we are going to use Excel. We also need to list the first cell and last cell. In the following code, we start at the second row so first row can be used for header. One minor problem is that we need to specify the last row, which will change for different dataset. Here we used a macro variable &LastRow which can be generated.

```
filename OutF dde
  "excel|Sheet1!r2c1:r&LastRow.c6";
data _null_;
  set sums;
  file OutF;
  put industry n mean sum min max;
run;
```

Step 7: Add Parameters

The users may want to change group variable and analysis variable. And we can also foresee that the name and the location of the data file may change. So we create the following macro variables:

```
/* Group Variable selection */
%let group=Nation;
/* Analysis variable selection */
%let anavar=Profits;
/* Directory for data file */
%let dir=c:\sugi25e;
/* Data file name */
%let fname=bdata.txt;
```

Step 8: Final SAS Program

With everything we need on the SAS side, we can construct the final SAS program. Note that the macro variables have been commented out. They will be generated by the Excel User Interface. The final program prog2.sas uses CALL SYMPUT in a DATA _NULL_ to generate the macro variable &Lastrow.

```
/* prog2.sas */
/*
  %let group=Nation;
  %let anavar=Profits;
  %let dir=c:\sugi25e;
  %let fname=bdata.txt;
*/
libname alib "&dir";
data alib.business;
  infile "&dir.\&fname";
  input
    @1 COMPANY      $20.
    @21 EMPLOYS     5.
    @27 INDUSTRY    $15.
    @43 NATION      $10.
    @54 PROFITS     10.
    @65 SALES       10.;
run;

proc sort;
  by &group;
run;
proc means noprint;
  by &group;
```

```

var &anavar;
output out=sums
      n=n mean=mean sum=sum min=min max=max;
run;

proc print noobs label;
  title "&anavar by &group";
  label
    &group="&group"
    n='Count'
    mean="Average &anavar"
    sum="Total &anavar"
    min='Minimum'
    max='Maximum';
  format mean dollar10.2
    sum
    min
    max dollar10.0;
  var &group n mean sum min max;
run;

/* Get last row. Then transfer data to excel */
data _null_;
  set sums nobs=count;
  call symput('LastRow',compress(
    put(count+1,8.)));
  stop;
run;

filename OutF dde
      "excel|Sheet1!r2c1:r&LastRow.c6";
data _null_;
  set sums;
  file OutF;
  put &group n mean sum min max;
run;

```

CREATE EXCEL USER INTERFACE

Step 1: Start SAS Server in Excel

We use SAS as a OLE automation server. VBA function CreateObject will start the server. In general, you can use SAS.Application in Create Object. You add .612 at the end if you want to use version 6.12. In this example, we make SAS visible so we can actually see SAS processing the program. In a real application, you can set it to False so SAS is completely invisible. We set Wait to True so Excel will not start the next step until SAS finished request.

```

Dim SAS As Object
Set SAS = CreateObject("SAS.Application.612")
SAS.Visible = True
SAS.Wait = True

```

Step 2: Control SAS Inside Excel

We can use submit method to run SAS program. Just like in SAS editor, we can use %INCLUDE to work with a SAS program in a text file. When done, we can use quit method to close SAS.

```

SAS.Submit("options ls=80 mprint;")
SAS.Submit(%include 'prog2.sas;')
SAS.Quit

```

Step 3: Create User Interface

Follow the procedure listed below to create user interface.

- Start MS Excel.
- Click on menu View -> Toolbars and check Forms and Visual Basic.
- Select Sheet2 and click on Drop-Down on Toolbar Forms. Draw a rectangular on Sheet2. This will create a drop-down. Name the drop-down as Group. In a cell on the left hand side of drop-down, type in 'Group Variable' for label.
- Repeat (3). Name the new drop down as Analysis. Type in the left cell 'Analysis Variable' as its label.
- In another cell, type in c:\sugi25e. And name the cell as Dir.
- In another cell, type in bdata.txt. And name the cell as FileN.
- Select the button from the Toolbar Forms. Then draw a rectangular on Sheet2 to create a button. Assign Macro dialog will popup. Just click on New to create a new macro calculate():

```

Sub calculate()
  Dim SAS As Object

  'Clear existing result
  Worksheets("Sheet1").Activate
  Rows("2:17").Select
  Selection.Delete Shift:=xlUp
  Range("A1").Select
  Worksheets("Sheet2").Activate

  'Start SAS
  Set SAS = CreateObject("SAS.Application")

  SAS.Visible = True
  SAS.Wait = True

  'Pick user selection and convert them to
  'SAS macro statement
  SAS.Submit ("options ls=80 mprint;")
  With ActiveSheet.DropDowns("Group")
    F = "%let group=" & .List(.Value) & ";";
    SAS.Submit (F)
  End With

  With ActiveSheet.DropDowns("Analysis")
    F = "%let anavar=" & .List(.Value) & ";";
    SAS.Submit (F)
  End With

  SAS.Submit ("%let Dir=" & Range("Dir").Value &
    ";")
  SAS.Submit ("%let FName=" &
    Range("FileN").Value & ";")

  'Submit the external SAS program
  F = "%include '" & Range("Dir").Value &
    "\prog2.sas';"
  SAS.Submit (F)

  'Show the sheet with result
  Worksheets("Sheet1").Activate

  If SAS.Visible = False Then
    SAS.Quit
    Set SAS = Nothing
  End If
End Sub

```

- In Sheet3, Fill B1 with 'Nation', B2 with 'Industry', B4 with 'Profits', B5 with 'Sales'.
- Right click on Drop-Down 'Group' and select Format

Object on the pop up menu. Then fill the Input Range with 'Sheet3!B1:B2'.

- Do the same for Drop-Down 'Analysis', but fill the Input Range with 'Sheet3!B4:B5'.
- Create a directory c:\MWSUG2000 and copy bdata.txt and prog2.sas to the directory.
- On Sheet2, Click the button to run the program.

We can also make some cosmetic modification to make the screen looks better. The appendix

CONCLUSION

Excel can be used for developing user interface for SAS application. Such application is user friendly and will reduce potential human error introduced in revising the SAS program.

CONTACT INFORMATION

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

Zhuan (John) Xu



APPEDIX: SCREEN SHOT OF EASY EXTRACTOR

400 53rd Place
West Des Moines, IA 50226
Phone: (515) 778-4093
Email: johnxu@home.com

Company info:

1ST Consulting, L.L.C.
400 53rd Place
West Des Moines, IA 50266
Phone: (515) 778-4093
Fax: (559) 677-8739



SAS® is a registered trademark of SAS Institute, Inc. in the USA and other countries.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Microsoft Excel
ExcelApp.xls

MWSUG 2000 Demo

Calculate

Group Variable:

Analysis Variable:

File Directory:

File Name: