Paper 23

AUTOMATING THE PRODUCTION OF CUSTOMIZED POWERPOINT PRESENTATION GRAPHS BY INTEGRATING THE FUNCTIONALITY OF SAS® SOFTWARE WITH MICROSOFT VISUAL BASIC FOR APPLICATIONS

Aileen L. Yam, PharmaNet, Inc., Princeton, NJ

ABSTRACT

With the advent of Visual Basic for Applications (VBA) and Object Linking and Embedding (OLE), macros can be written to bring wonders to programming. This paper gives simple sample code to exemplify the integration of SAS® software and Microsoft Office software to automate the production of customized graphs. During the presentation, a more fully developed application complete with dialog boxes for generating a variety of graphs via option and command buttons will be demonstrated.

I. PROGRAMMING DESIGN

There was a request to deliver more than 70 high-resolution line and bar graphs in PowerPoint application within a very short time frame.

I pondered on the following considerations in designing the programming approaches:

1. PowerPoint was required by clients as the final media for the graphs, so that they could access the graphs and do some last-minute changes easily.

2. PowerPoint 97 has VBA macros for controlling slide objects, but not for creating graphs. In other words, the current version of PowerPoint lacks the mechanism for supercharging the process.

3. Excel 97 has VBA macros for creating graphs and is capable of automation.

4. The data are in SAS data sets. Transferring data from SAS to Excel can be automated.

5. All Microsoft Office application objects can be connected with OLE.

6. Therefore, the integrated solution is:

   a. transfer SAS data to Excel;

   b. create graphs in Excel with VBA macros;

   c. use OLE to copy Excel graphs into PowerPoint.

II. SAS TO EXCEL

SAS software was used:

1. to create all the summary information such as percents, means and mean changes from baseline;

2. to arrange the order of the data;

3. to transfer the SAS data to Excel.

The data consisted of:

1. a sequence number to indicate the rows of data that go into a graph. One sequence number was assigned to each graph;

2. a formatted variable for visit number that appeared in the graph as x-axis labels, such as, 'Baseline' for visit 0, 'Week 2' for visit 2, and 'Endpoint' for visit 991;

3. the values of each data point in the graphs;

4. the minimum and the maximum values of the y-axis for fixing the scale of the y-axis;

5. the y-axis label;

6. the legend labels;

7. the titles for the graphs.

The data were organized in the above order to make it easy for writing the Excel VBA macro program. For example, the sequence number was the first variable so that the graph number could be readily identified. The x-axis labels and the data points were arranged together in order that they could be selected as data range for the graphs. The titles were put last, because both the title numbers and the description of titles were subject to frequent changes and programming logic would remain intact regardless of the changes in titles.

The following is a SAS program that makes use of DDE to automate the transfer of SAS data to Excel.

```
/*********************************************************
* sas_xls.sas - Converts SAS data sets to Excel files
                with DDE
*  NOTE: Must be used with Excel file open
*********************************************************/

options ls=100 pageno=1 noxwait noxsync mprint;

libname sasdat 'C:\projname\data';
%let outdir=%str(C:\projname\graphs);


%macro sas_xls(inlib=,inds=,outds=,sheetnam=
  Master);

*-------------------------------------------------------------------------;
***  Retrieve variable names, variable position and
     total number of rows and columns;
*-------------------------------------------------------------------------;

proc contents data=&inlib..&inds out=cont(keep=name
     npos nobs);
run;

proc sort data=cont;
     by npos;

data _null_;
     set cont end=eof;
     by npos;
call symput('col'||left(_n_),trim(NAME));

if eof then do;
     call symput('columns',trim(left(_n_)));
     call symput('rows',trim(left(nobs)));
end;
run;

*-------------------------------------------------------------------------;
***  Invoke Excel and open existing Excel file;
*-------------------------------------------------------------------------;

x  "c:\progra~1\micros~1\office\excel.exe  &outdir\
     &outds";
data _null_;
     x = sleep(5);
run;

*-------------------------------------------------------------------------;
***  Put column headings in the first row;
*-------------------------------------------------------------------------;

filename  excdata  dde  "EXCEL|&outdir\[&outds]
     &sheetnam!r1c1:r1c&columns";
run;

data _null_;
     file excdata notab lrecl=600;
put %do i=1 %to &columns;
     "%trim(&&col&i)" '09'x
     %end;
     ;
run;

*-------------------------------------------------------------------------;
***  Put data starting from the second row;
*-------------------------------------------------------------------------;

filename  excdata  dde  "EXCEL|&outdir\[&outds]
     &sheetnam!r2c1:r%eval(&rows+1)c&columns.";

data _null_;
     set &inlib..&inds;
file excdata notab lrecl=600;
put %do i=1 %to &columns;
     &&col&i '09'x
     %end;
     ;
run;

*-------------------------------------------------------------------------;
***  Save Excel data file and quit;
*-------------------------------------------------------------------------;

filename cmdde dde 'EXCEL|SYSTEM';

data _null_;
     file cmdde;
     put '[save]';
     put '[quit]';
run;

%mend sas_xls;

%sas_xls(inlib=sasdat,inds=gline3,outds=lindat3.xls);
```

III. EXCEL VISUAL BASIC MACRO

The first worksheet in Excel was named Master, and contained data for many graphs. The Excel macro program used the sequence numbers to generate one extra worksheet for each graph. A DO LOOP was created to copy data from the Master worksheet into individual worksheets and one graph was created in each worksheet following the Master worksheet. The

sequence numbers also served as names of the extra worksheets created.

The macro recorder in Excel can help generate some of the VBA code. However, in order for the macro-recorded code to be reusable for many graphs, it needs to be modified in the following manner:

1. The recorded code only performs the specific actions recorded. The references to worksheets, charts, data ranges, etc. are specific. Variables need to be used in place of the hard-coded references.

2. The recorded code lists many arguments and options that are not needed. It is a good practice to clean up the extra code.

3. Throughout the Excel macro program, all the AutoScaleFont conditions need to be set to false, so that the fonts are not re-scaled when they are copied to PowerPoint.

4. Even though it is convenient to have the data dynamically linked to the graphs, it is better not to do so, because only the data are linked, while other parts of the graphs such as the titles, the scales of the axes, and the legends can only be updated when the entire program is re-run.

5. Chart size and the size of the plot area need to be pre-determined and defined so that all the graphs will have the same dimensions.

```
Sub Line3()
'
' A THREE-LINE GRAPH MACRO
' MACRO WRITTEN 8/16/98 by AILEEN YAM
'
' KEYBOARD SHORTCUT: Ctrl+q
'
'**************************************************
'VARIABLES CREATED (IN UPPER CASE)
'
'SEQNO = Chart Sequence Number
'SEQNON = Chart Sequence Number (Next)
'FIRSTSEQ =  Column location of the first SEQNO
             of a chart
'NEXTSEQ =   Column location of the next SEQNO
             of a chart
'LASTSEQ =   Column location of the last SEQNO
             of a chart
'NUMGRP = Number of groups in x-axis
'COLUMNSTR =  A temporary variable for getting
              the last column
'LASTCOL = The last column number in data
'NCHART = Total number of charts
'
'**************************************************
'
'CREATE DATE VARIABLE IN A SPECIAL FORMAT
'
Dim NEWDATE As String
NEWDATE = Format(Date, "dddd, d mmmm yyyy")
'
'**************************************************
'
'FIND OUT THE LAST COLUMN NUMBER
'
Dim COLUMNSTR As String
Dim LASTCOL As Integer
'
Sheets("Master").Select
'
COLUMNSTR = Cells(1, 1).Text
'
Do While COLUMNSTR <> ""
    LASTCOL = LASTCOL + 1
    COLUMNSTR = Cells(1, LASTCOL).Text
Loop
'
LASTCOL = LASTCOL - 1
'
'**************************************************
'
BEGINNING OF DO LOOP
'
Dim NCHART As Integer
Dim FIRSTSEQ, NEXTSEQ, LASTSEQ, NUMGRP _
    As Integer
Dim SEQNO As String
Dim SEQNON As String
'
NCHART = 0
FIRSTSEQ = 2     'DATA START FROM ROW 2
NEXTSEQ = FIRSTSEQ + 1
SEQNO = Cells(FIRSTSEQ, 1).Value
SEQNON = Cells(NEXTSEQ, 1).Value
'
Do While SEQNO <> ""
'
    NCHART = NCHART + 1
'
    Do While SEQNO = SEQNON
        NEXTSEQ = NEXTSEQ + 1
        SEQNON = Cells(NEXTSEQ, 1).Value
    Loop
'
    LASTSEQ = NEXTSEQ - 1
    NUMGRP = LASTSEQ - FIRSTSEQ + 2
'
'**************************************************
'
'COPY DATA
'
    Range(Cells(FIRSTSEQ, 1), Cells(LASTSEQ, _
LASTCOL)).Select
    Selection.Copy
'
    Sheets.Add
    Range("A2").Select
    ActiveSheet.Paste
    With ActiveSheet
        .Move after:=Worksheets(Worksheets.Count)
        .Name = SEQNO
    End With
```

```
'
'*****************************************************
'
'COPY COLUMN HEADINGS
'
    Sheets("Master").Select
    Range(Cells(1, 1), Cells(1, LASTCOL)).Select
    Selection.Copy
    Sheets(SEQNO).Select
    Range(Cells(1, 1), Cells(1, LASTCOL)).Select
    ActiveSheet.Paste
'
'*****************************************************
                    'CREATE CHART
'*****************************************************
'
'***********
'SELECT DATA FOR CHART AND LEGEND
'
    Sheets(SEQNO).Select
    ActiveSheet.Range("C2:F" & NUMGRP).Select
'
    Charts.Add
    ActiveChart.ChartType = xlLineMarkers
    ActiveChart.SetSourceData Source:= _
Sheets(SEQNO).Range("C2:F" & NUMGRP),  _
PlotBy:=xlColumns
    ActiveChart.SeriesCollection(1).Name = _
Sheets(SEQNO).Cells(2, 10).Value
    ActiveChart.SeriesCollection(2).Name = _
Sheets(SEQNO).Cells(2, 11).Value
    ActiveChart.SeriesCollection(3).Name = _
Sheets(SEQNO).Cells(2, 12).Value
'
'***********
'DEFINE ACTIVE CHART LOCATION
'
    ActiveChart.Location Where:=xlLocationAs _
Object, Name:=SEQNO
'
'***********
'INCREASE LINE MARKER SIZE
'
    ActiveChart.SeriesCollection(1).MarkerSize = 7
    ActiveChart.SeriesCollection(2).MarkerSize = 7
    ActiveChart.SeriesCollection(3).MarkerSize = 7
'
'***********
'DEFINE CHART SIZE
'
    With ActiveSheet.ChartObjects(1)
        .Top = 30
        .Left = 30
        .Height = 480
        .Width = 680
    End With
'
'***********
'ASSOCIATE DATA VALUES WITH CHART TITLES
'AND Y-AXIS
'
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = NEWDATE & _
Chr(10) & Chr(10) & Range("O2") & Chr(10) & _
Range("P2") & Chr(10) & Range("Q2")
        .ChartTitle.Font.Size = 18
        .Axes(xlCategory, xlPrimary).HasTitle = False
        .Axes(xlValue, xlPrimary).HasTitle = True
        .Axes(xlValue,xlPrimary).AxisTitle. _
Character.Text = Sheets(SEQNO).Range("I2")
    End With
'
'***********
'DEFINE X-AXIS RANGE
'
    With ActiveChart
        .HasAxis(xlCategory, xlPrimary) = True
        .HasAxis(xlValue, xlPrimary) = True
    End With
    ActiveChart.Axes(xlCategory,xlPrimary). _
CategoryType = xlCategoryScale
    With ActiveChart
        .Axes(xlCategory)
        .HasMajorGridlines = False
        .HasMinorGridlines = False
    End With
'
'***********
'DEFINE Y-AXIS RANGE
'
    With ActiveChart.Axes(xlValue)
        .MinimumScale = Range("G2")
        .MaximumScale = Range("H2")
        .HasMajorGridlines = False
        .HasMinorGridlines = False
    End With
'
'***********
'SELECT LEGEND LOCATION AND SIZE
'
    ActiveChart.HasLegend = True
    ActiveChart.Legend.Select
    Selection.Position = xlBottom
    Selection.AutoScaleFont = False
    With Selection.Font
        .Name = "Arial"
        .FontStyle = "Bold"
        .Size = 12
    End With
'
'***********
'DEFINE Y-AXIS SIZE
'
    ActiveChart.Axes(xlValue).AxisTitle.Select
    Selection.AutoScaleFont = False
    With Selection.Font
        .Name = "Arial"
        .FontStyle = "Bold"
        .Size = 12
    End With
'
    ActiveChart.Axes(xlValue).Select
    Selection.TickLabels.AutoScaleFont = False
    With Selection.TickLabels.Font
        .Name = "Arial"
        .FontStyle = "Bold"
        .Size = 12
    End With
'
'***********
'DEFINE X-AXIS SIZE
'
    ActiveChart.Axes(xlCategory).Select
```

```
        Selection.TickLabels.AutoScaleFont = False
        With Selection.TickLabels.Font
            .Name = "Arial"
            .FontStyle = "Bold"
            .Size = 12
        End With
        With Selection.Border
            .Weight = xlHairline
        End With
        With Selection
            .MajorTickMark = xlOutside
            .MinorTickMark = xlNone
            .TickLabelPosition = xlLow
        End With
'
'***********
'ADD DOTTED LINES TO JOIN DATA POINTS
'
        ActiveChart.LineGroups(1).HasHiLoLines = True
        ActiveChart.ChartGroups(1).HiLoLines.Select
        With Selection.Border
            .ColorIndex = 1
            .Weight = xlThin
            .LineStyle = xlDot
        End With
'
'***********
'DELETE CHART BORDER
'
        ActiveChart.ChartArea.Select
        With Selection.Border
            .Weight = 2
            .LineStyle = 0
        End With
'
'***********
'
'DEFINE SIZE OF THE PLOT AREA
'
        ActiveChart.PlotArea.Select
        With Selection
            .Left = 110
            .Top = 120
            .Height = 270
            .Width = 550
        End With
'
'*******************************************************
                'END CREATE CHART
'*******************************************************
'
'***********
'RESUME DO LOOP
'
        Sheets("Master").Select
'SET FIRSTSEQ OF NEXT GROUP
        FIRSTSEQ = LASTSEQ + 1
'SET NEXTSEQ OF NEXT GROUP
        NEXTSEQ = FIRSTSEQ + 1
        SEQNO = Cells(FIRSTSEQ, 1).Value
        SEQNON = Cells(NEXTSEQ, 1).Value
'
Loop
'*******************************************************
'
End Sub
```

IV. OLE AUTOMATION TO COPY GRAPHS FROM EXCEL TO POWERPOINT

```
Sub task()

'***********
'WITH THE FOR...NEXT LOOP, COUNT THE NUMBER
'OF GRAPHS TO BE PUT INTO POWERPOINT
'
Dim ppt As New PowerPoint.Application
ppt.Presentations.Add msoTrue
For Each s In Worksheets
    For Each c In s.ChartObjects
      c.Copy
      counter =ppt.ActivePresentation.Slides.Count+1
      ppt.ActivePresentation.Slides.Add Index:= _
counter, Layout:=ppLayoutBlank
      ppt.ActiveWindow.View.GotoSlide Index:=counter
      ppt.ActiveWindow.View.Paste
    Next c
Next s
ppt.Visible = msoCTrue

'***********
'APPLY TEMPLATE WITH A COLOR BACKGROUND
'
ppt.ActivePresentation.ApplyTemplate "C:\projname
\color.pot"
'
'***********
'
End Sub
```

V. SUMMARY

This paper shows how to extend SAS programming expertise into new areas, and build integrated solutions with SAS software and Microsoft Office software.

For additional information, contact:

Aileen L. Yam
PharmaNet, Inc.
504 Carnegie Center
Princeton, NJ 08540-6242