Paper 3241-2015

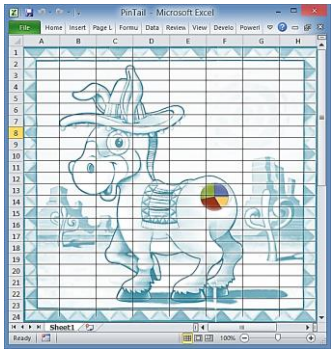
Pin the SAS® Tail on the Microsoft Excel Donkey: Automatically Sizing and Positioning SAS Graphics for Excel

Ted Conway, Chicago, IL

ABSTRACT

Okay, you've read all the books, manuals, and papers and can produce graphics with SAS/GRAPH® and Output Delivery System (ODS) Graphics with the best of them. But how do you handle the Final Mile problem—getting your images generated in SAS® sized just right and positioned just so in Microsoft Excel? This paper presents a method of doing so that employs SAS Integration Technologies and Excel Visual Basic for Applications (VBA) to produce SAS graphics and automatically embed them in Excel worksheets. This technique might be of interest to all skill levels. It uses Base SAS®, SAS/GRAPH, ODS Graphics, the SAS macro facility, SAS® Integration Technologies, Microsoft Excel, and VBA.

INTRODUCTION

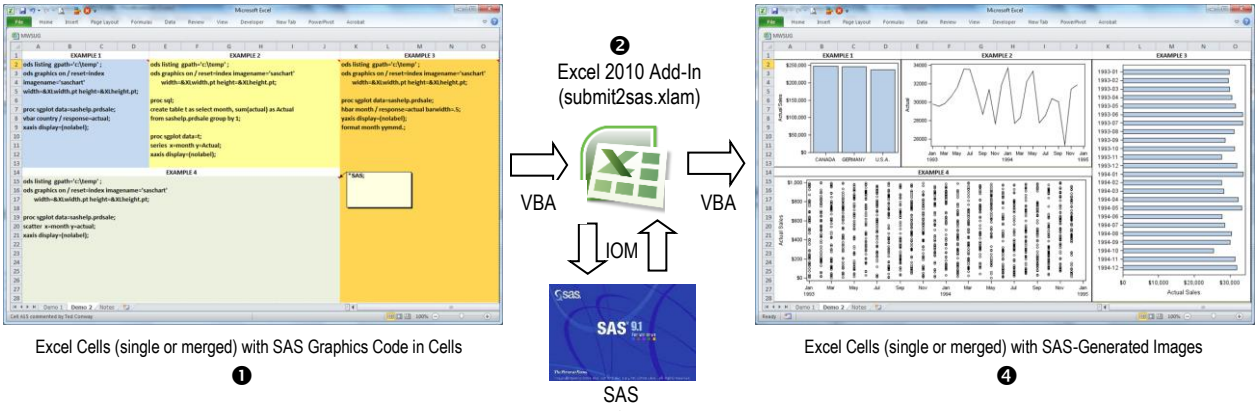


For better or worse, Excel workbooks are a staple of corporate communication. So, like it or not, you'll most likely find yourself having to present your SAS results in Microsoft Office documents from time to time. Fortunately, there are many options available to you for getting SAS output into Excel – from simple brute force cut-and-pasting to packaged solutions like the SAS Add-In for Microsoft Office.

In this paper, we'll explore a technique that uses a combination of SAS code and Excel VBA to fill-in-the-holes of your Excel spreadsheets by automatically creating properly-sized SAS charts and graphs and inserting them into their proper places in the gridded Excel layout.

So, let's play Pin the SAS Tail on the Excel Donkey!

WHAT'S THE BIG IDEA?

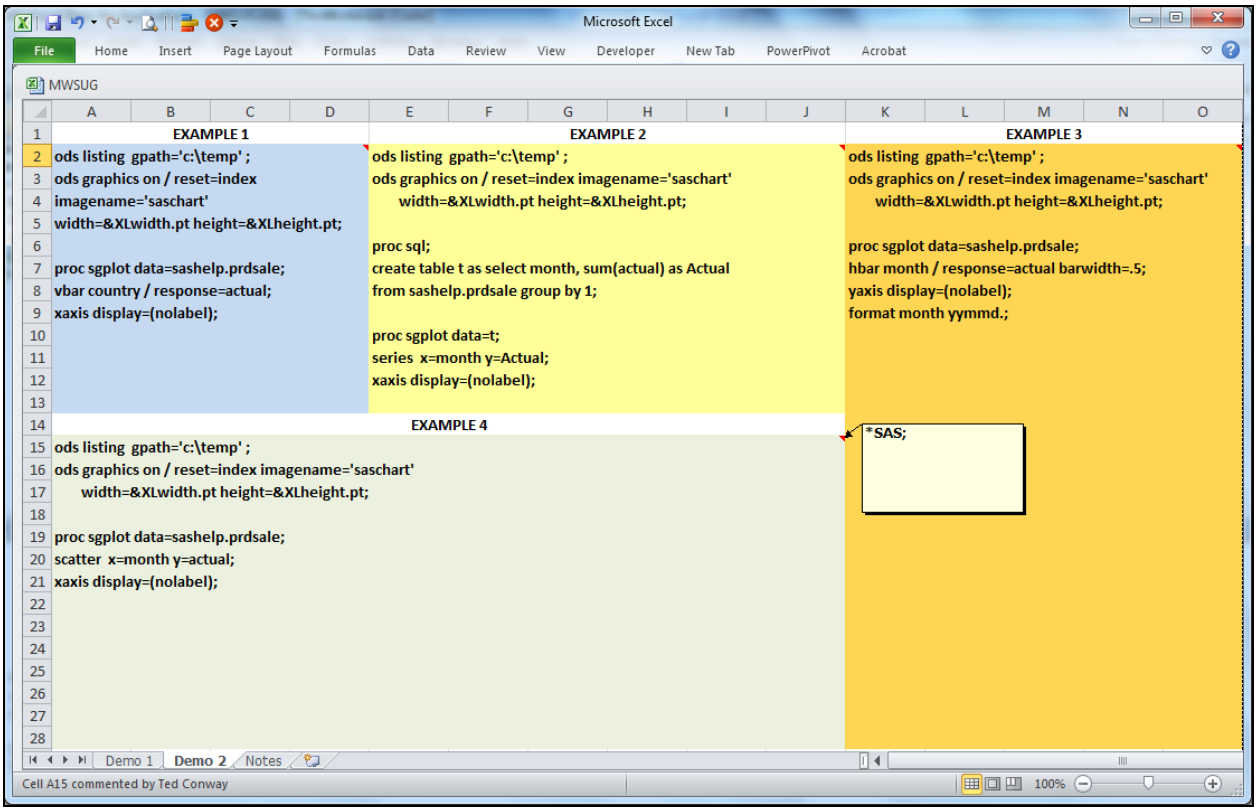


PROCESSING OVERVIEW

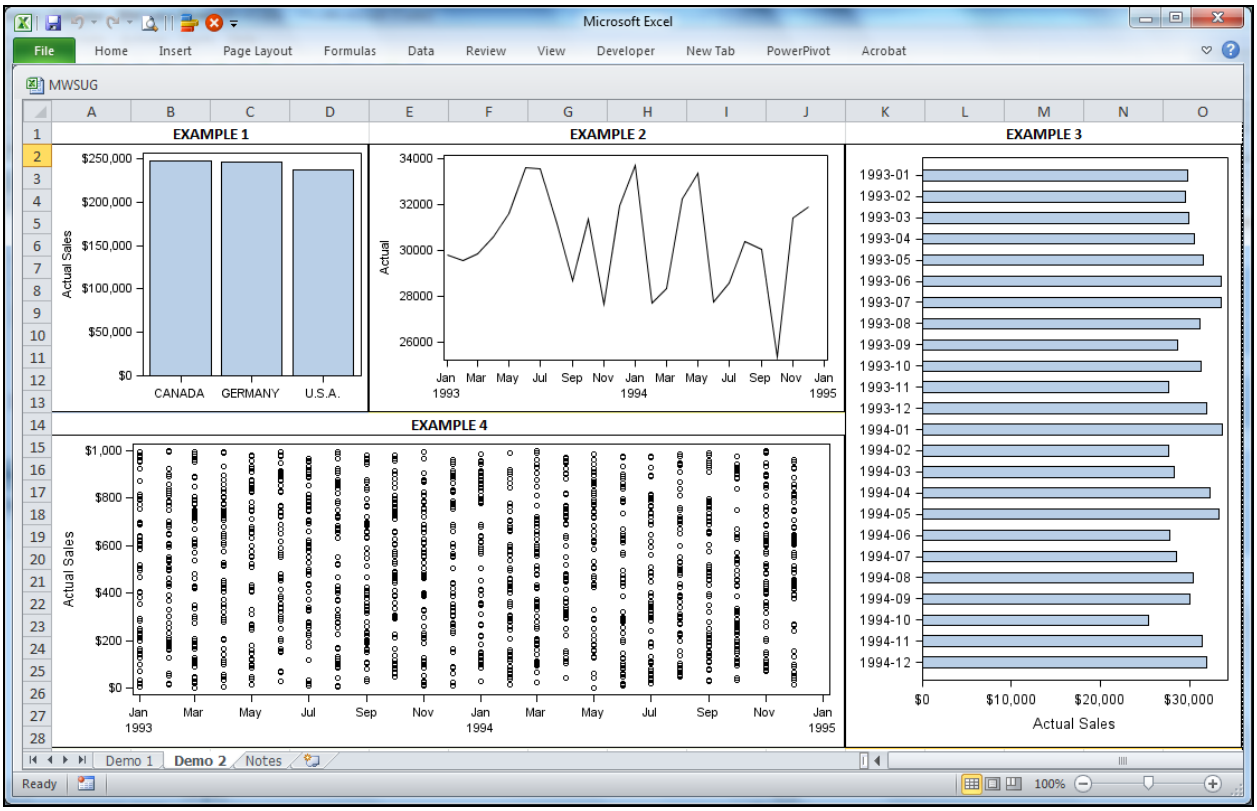
We start with a worksheet ❶ and specify SAS code in cells tagged with "special" Excel comments, i.e., starting with "***SAS;***". When a toolbar button is clicked, VBA code in ***submit2sas.xlam***, an Excel 2010 add-in workbook ❷, is run to pass the code in the tagged cells and comments to SAS via IOM ❸ for execution. SAS creates images (.png format), which VBA inserts into the proper position, leaving us with an updated worksheet containing the requested images ❹. If the process is rerun, any old images in the worksheet are first deleted by the VBA code.

In the following pages, we'll take a look at an example of how SAS and Excel VBA code can be used to automate the process of producing SAS graphics and embedding them in Excel worksheets (full VBA code is presented).

LET'S ZOOM IN FOR A CLOSER LOOK!



BEFORE: GRID LAYOUT WITH SAS CODE



AFTER: SAS-GENERATED CHARTS

SO, HOW'D YOU DO THAT? EXCEL VBA CODE (SUBMIT2SAS.XLAM)

```
'==> Create charts by calling SAS

Public obWSM As New SASWorkspaceManager.WorkspaceManager            ' IOM-related objects, variables
Public obws As New SAS.Workspace, errmsg As String, onetime As Boolean

Sub DrawCharts()
                                                                    ' Establish SAS IOM connection?
If onetime = False Then
  Set obws = obWSM.Workspaces.CreateWorkspaceByServer("Local", VisibilityProcess, Nothing, "", "", errmsg)
  onetime = True
  End If

DeleteCharts                                                        ' Get rid of any existing charts

'==> Call SAS with cell width/height & code from each tagged comment/cell, position SAS-generated image

For Each cmt In ActiveSheet.Comments                                ' Get SAS code from comments/cells
  If UCase(Left(cmt.Text, 5)) = "*SAS;" Then
    obws.LanguageService.Submit "%let XLwidth=" & cmt.Parent.MergeArea.Width & _
                               "; %let XLheight=" & cmt.Parent.MergeArea.Height & "; " & _
                                cmt.Parent.Value & cmt.Text & "; run;"
    Debug.Print obws.LanguageService.FlushLog(1000000)              ' Send SAS log to debugging window
    ActiveSheet.Shapes.AddPicture "c:\temp\saschart.png", False, True, _
                                  cmt.Parent.Left, cmt.Parent.Top, _
                                  cmt.Parent.MergeArea.Width, cmt.Parent.MergeArea.Height
  End If
Next cmt

End Sub

'==> Delete any existing charts

Sub DeleteCharts()
For Each s In ActiveSheet.Shapes
  If s.Type = 13 Then s.Delete
Next
End Sub

'--> Close SAS IOM Workspace Connection

Private Sub Workbook_BeforeClose(Cancel As Boolean)
On Error Resume Next
obws.Close
End Sub
```

Note: See http://support.sas.com/rnd/itech/doc/dist-obj/winclnt/winvbpro.html for SAS IOM reference requirements. **SUBMIT2SAS.XLAM – VBA**

VBA PROCESSING NOTES

As you can see from the above, thanks to the SAS IOM interface, there's not much VBA code required to get Excel and SAS to communicate with each other.

In a nutshell, here's what happens when the  toolbar button linked to the **DrawCharts** macro is pressed:

1. A **local SAS Workspace** (on the PC) is created, if one doesn't already exist, to run SAS code
2. Any existing charts on the worksheet are deleted by the **DeleteCharts** macro (which is also linked to the  toolbar button)
3. For each **Excel Comment** on the worksheet that tags SAS code to be run (i.e., Comments starting with ***SAS;**):
   - ✓ SAS code is generated that creates two **macro variables** containing the width (***XLwidth***) and height (***XLheight***) of the Cell that's tied to the Comment
   - ✓ The generated SAS code, as well as the code in the Cell and Comment, is **submitted to SAS via IOM**
   - ✓ SAS generates an image file with a standard file name (***c:\temp\saschart.png***)
   - ✓ The contents of the **SAS Log** are sent to Excel's Immediate Window for debugging purposes
   - ✓ Excel **inserts the SAS-generated image** into the worksheet Cell

When the Excel Add-In is closed, the local SAS Workspace is also closed.

CONCLUSION

With just a few dozen lines of VBA code, it's possible to cobble together a serviceable, general-purpose, interactive SAS chart generator that'll allow you to enhance your Excel worksheets with embedded SAS graphics without much effort at all. The formatting flexibility afforded by Excel even enables the creation of stand-alone dashboards of sorts. And since you have the full power of the SAS language at your disposal, you can easily transform data into a suitable format for charting.

Some error handling enhancements would be in order for deployment to a wider audience, but SAS-savvy users could make do with the available SAS log output.

Other possible enhancements that come to mind include tapping Excel's enhanced image features (e.g., transparency, picture effects), providing a server-based version for SAS Enterprise Guide users, employing Microsoft Office shapes as SAS code containers to allow this technique to be used with PowerPoint/Word macros, and perhaps even extending the concepts to other programming languages that may be used from time-to-time alongside SAS.

CREDITS/REFERENCES

- ✓ SAS. *SAS Product Documentation*. http://support.sas.com/documentation/index.html
- ✓ Contextures. *Excel Comments VBA*. http://www.contextures.on.ca/xlcomments03.html
- ✓ Delwiche, Lora D. and Slaughter, Susan J. *SAS Graphing Made Easy with ODS Graphics Procedures*. http://support.sas.com/resources/papers/proceedings14/1267-2014.pdf
- ✓ Conway, Ted. *%SPARKY: A SAS® Macro for Creating Excel Sparklines*. http://support.sas.com/resources/papers/proceedings12/084-2012.pdf

CONTACT INFORMATION

Ted Conway resides in Chicago, Illinois. Spam filters notwithstanding, he can be reached at tedconway@aol.com.

TRADEMARKS

APPENDIX – SAS CODE USED TO GENERATE CHARTS

EXAMPLE 1

```
ods listing  gpath='c:\temp' ;
ods graphics on / reset=index imagename='saschart' width=&XLwidth.pt height=&XLheight.pt;

proc sgplot data=sashelp.prdsale;
vbar country / response=actual;
xaxis display=(nolabel);
```

EXAMPLE 2

```
ods listing  gpath='c:\temp' ;
ods graphics on / reset=index imagename='saschart' width=&XLwidth.pt height=&XLheight.pt;

proc sql;
create table t as select month, sum(actual) as Actual
from sashelp.prdsale group by 1;

proc sgplot data=t;
series  x=month y=Actual;
xaxis display=(nolabel);
```

EXAMPLE 3

```
ods listing  gpath='c:\temp' ;
ods graphics on / reset=index imagename='saschart' width=&XLwidth.pt height=&XLheight.pt;

proc sgplot data=sashelp.prdsale;
hbar month / response=actual barwidth=.5;
yaxis display=(nolabel);
format month yymmd.;
```

EXAMPLE 4

```
ods listing  gpath='c:\temp' ;
ods graphics on / reset=index imagename='saschart' width=&XLwidth.pt height=&XLheight.pt;

proc sgplot data=sashelp.prdsale;
scatter  x=month y=actual;
xaxis display=(nolabel);
```