Paper #006-30

How to get SAS® data into PowerPoint™ using SAS9

David Bosak, Comsys IT Partners, Kalamazoo, MI

ABSTRACT

The purpose of this paper is to describe three techniques of creating PowerPoint presentations from SAS. Two of these techniques are independent of SAS version. One of the methods is dependant on new features in SAS9. The paper will describe these techniques, and comment on the relative merits of each technique.

INTRODUCTION

SAS interacts well with Microsoft Word and Excel. ODS will output Word and Excel. You may also use the new SAS Microsoft Addins to shuttle data back and forth from SAS to Word and Excel. However, SAS provides no tools for creating or otherwise interacting with another popular Office format: PowerPoint.

PowerPoint is a frequently requested report format. It is especially popular among executives. These executives are often the audience for the analytics and forecasting that SAS provides. Therefore - all too frequently -it is the SAS analysts who are burdened with the task of creating PowerPoint presentations.

The typical way of getting SAS data into PowerPoint is to output to Excel from ODS, and then cut and paste into PowerPoint. At this moment there are hundreds of analysts across the country cutting and pasting data into PowerPoint. It is a tedious task, and no one likes to do it.

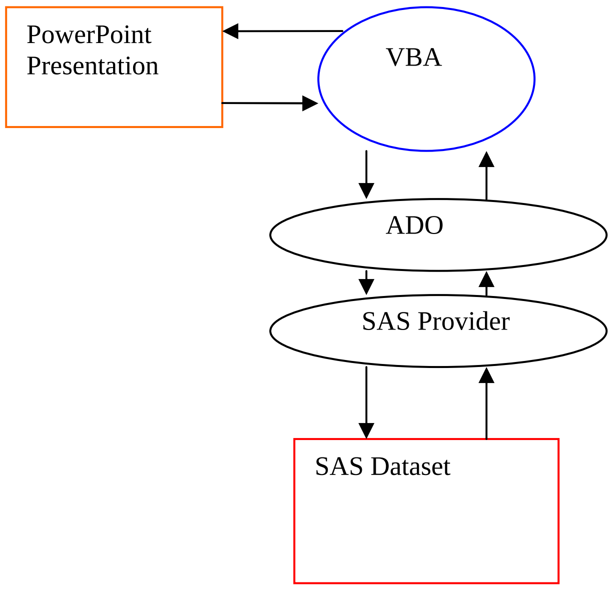
The purpose of this paper is to describe 3 techniques of creating PowerPoint presentations from SAS. Two of these techniques are independent of SAS version. One of the methods is dependant on new features in SAS9. The paper will describe these techniques, and comment on the relative merits of each technique.

The three techniques presented by this paper are:

- PowerPoint Automation
- Linking to Excel
- ODS PowerPoint Tagset

POWERPOINT AUTOMATION

Powerpoint automation is a powerful technique that has been available for many years. Like all Microsoft Office products, PowerPoint has VBA scripting capability. The PowerPoint Automation technique leverages VBA script to pull data from SAS and stick it into PowerPoint. Here is how it works:



VBA is able to interact both with the presentation and with the SAS Dataset. VBA is often described as a ‘glue’ that can paste together disparate components to create a workable application. In the case at hand, that is what VBA is doing. Base SAS cannot interact directly with PowerPoint. PowerPoint will not allow you to embed SAS data. The solution is to glue them together with VBA.

VBA does not accesses SAS data directly. It uses two tools: ADO and a SAS Provider.

ADO stands for ActiveX Data Objects. ADO is an object model used to manipulate data. Usually, ADO is used to manipulate data stored in an RDBMS like Oracle or SQL Server. But ADO can also be used with other kinds of data sources, as long as that data source has a provider.

SAS created two providers that can be used with ADO: The SAS ODBC Driver and the SAS Local Provider.

The SAS Local provider is very simple. It does not allow you to sort, filter, or run SQL statements against the data. The Local Provider merely gives you access to the entire dataset. You select variables by referencing a column name, and select observations by looping through the entire dataset until you find the observation you want.

While the Local Provider has many limitations, it is adequate for most PowerPoint Automation projects. In most cases, all you need to do is get the data out of the dataset. More importantly, the SAS Local Provider is free. It is also distributable to any workstation – even if that workstation does not have SAS installed. **Appendix A: PowerPoint Automation - Local Provider** contains an example of how to populate a graph in PowerPoint with SAS data using VBA and the SAS Local Provider.

The SAS ODBC Driver comes with Base SAS, and requires Base SAS to be installed on the machine is it running on. The ODBC Driver actually launches a Base SAS session in the background every time you request data from the dataset. The benefit of the ODBC Driver is that it allows you to have full sort and filter capabilities, and even allows you to run SQL statements on the data.

Here are the advantages and disadvantages of the PowerPoint Automation technique:

Advantages

- Unlimited programmatic control of PowerPoint
- No additional license costs
- Allows you to create very sophisticated systems
- Does not require SAS

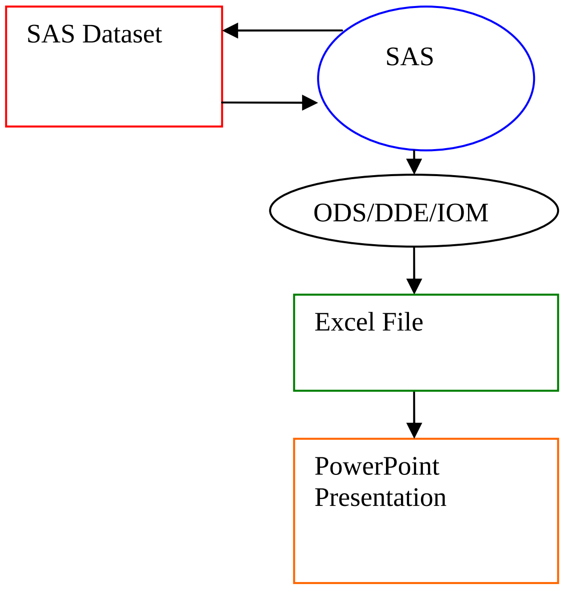
Disadvantages

- You have to be an experienced VBA Programmer, or hire one
- Can be difficult to implement, depending on what you are doing
- Can take a lot of time

LINKING FROM EXCEL

If you have a simple project, don't know VBA, and don't have much time, then linking data from Microsoft Excel is perhaps the best approach.

Let's see how this technique works.



Since SAS cannot manipulate PowerPoint directly, this technique uses Excel as a proxy. The PowerPoint presentation is linked to the Excel workbook, such that any changes made to the Excel file are automatically propagated to PowerPoint. If desired, you may then unlink the PowerPoint from the Excel to have a freestanding presentation.

Try this demonstration:

1. Open up both Excel and PowerPoint.
2. Create some sample data and simple chart in Excel
3. Highlight the chart and copy it to your clipboard
4. Go to PowerPoint and click "Paste Special" on the Edit menu
5. In the Paste Special dialog, click the "Paste Link" option and select the Microsoft Excel Chart Object in the list.
6. Click OK, and the chart will paste into the presentation.

Now try changing the data back in the Excel workbook that the chart is based on. You will notice that the chart will change both in the Excel workbook and in the PowerPoint presentation.

You can imagine a more sophisticated report, with many charts, graphs, and tables all linked to the same or multiple SAS workbooks. Every time you update the workbook – by whatever means desired – the data in the presentation will also be updated.

SAS can manipulate Excel through several means: DDE, ODS, or the Microsoft Addin for Excel. Which method you select depends on the requirements of the system you are building and the resources available.

Here are the advantages and disadvantages of the linking technique:

Advantages

- Very simple
- No additional licenses
- SAS can drive the system

Disadvantages

- No programmatic control over PowerPoint
- Must create charts in Excel
- Links are somewhat flaky, and may not work the way you want

ODS POWERPOINT TAGSET

You may have noticed that there few if any tools that can create PowerPoint presentations. SAS cannot create them: but neither can Crystal Reports, or Microstrategy, or Brio. What is the problem with PowerPoint?

The problem with PowerPoint is that Microsoft has never published the specification. Microsoft has published the specification for Excel. Therefore, many tools are able to create Excel files. The PowerPoint specification, however, remains a mystery to all but a few developers inside the Office development team at Microsoft.

There is a way to figure out the specification. You could look at PowerPoint files in a hex editor and try to figure out the patterns. However, this is a very difficult task, and would take a long time.

There is a somewhat easier task that is within the reach of a clever SAS programmer who has some time: Use ODS to create a PowerPoint Web file.

Try this demonstration:

1. Open up PowerPoint.
2. Create a simple presentation.
3. Go to the File menu and click “Save as Web Page.”
4. Navigate to the desired directory and click Save.

Now go to the directory you selected. You will find a single HTM file with the same name as your presentation. You will also find a folder with the same name as the presentation name plus “_files” appended to the end. The folder will contain a bunch of HTM, GIF, and XML files, plus a style sheet and a script file.

When you run the Web PowerPoint presentation, you will notice that it opens in Internet Explorer – not PowerPoint. However, it will act just like a regular PowerPoint presentation, and contain the most commonly used features. For our purposes, this functionality is sufficient.

The crux of this technique is that while that you cannot manipulate a regular PowerPoint file with SAS, you can manipulate these Web PowerPoint files. The Web PowerPoint files are all file types that can be output from SAS ODS.

PROC TEMPLATE

To generate a Web PowerPoint presentation from SAS, we will create our own Tagset. A Tagset is a set of named events that output custom markup. SAS uses the markup to create a specific type of output from an ODS command.

For instance, HTML is a type of markup. RTF is another kind of markup. SAS creates an output object independent of markup, and then applies the markup when you call a proc that generates a file: such as proc print, or proc report. The markup is like a wrapper around the data. This wrapper helps applications interpret the document. SAS uses the word Tagset to describe the ODS definition for a particular markup.

The new SAS template procedure allows you to modify an existing tagset or create your own tagset. In the case at hand, we will create a new tagset based on an existing one. We will create a version of HTML that has the capabilities of PowerPoint. The new tagset will be called PPTHTML. PPTHTML will inherit from the HTML4 tagset.

Before going further, let’s review how the tagset definitions work.

Tagset definitions are collections of events triggered by ODS. The events fire in a particular sequence. When an event fires, it puts out the code you have defined for that event. Let’s look at a simple definition:

```
ods path sasuser.templat (update)
   sashelp.tmplmst (read);

proc template;
   define tagset tagsets.test /store=sasuser.templat;
    notes "This is a test definition";
      parent=tagsets.html4;
      define event doc_head;
         start:
            put "<head>" NL;
            put VALUE NL;
         finish:
            put "</head>" NL;
      end;
      define event doc_meta;
            put "<meta name=""Generator""" NL;
            put " content=""SAS Software, see www.sas.com"">" NL;
      end;
      define event doc_body;
           start:
            put "<body lang=EN-US ";
            put " onload=""alert('Hello World');"">";
           finish:
            put "</body>" NL;
      end;
   end;
run;
```

The above code shows a tagset definition with three events. The events are *doc_head*, *doc_meta* and *doc_body*. The *doc_head* and *doc_body* events are separated into two sections: start and finish. The *doc_meta* event is called "stateless" because it has no start and finish sections. These three events generate HEAD tags, a META tag, and BODY tags for the document. By altering the markup, you can change the way the HTML is generated for all documents that use this tagset.

The parent of this sample tagset is identified as HTML4. That means this tagset is inherited from HTML4. Required events that are not defined in this tagset will be defined in HTML4, or tagsets that HTML4 inherits from. Any event defined in this tagset will override any like-named event in HTML4 or any of its parents.

Variables are another important aspect of tagset definitions. In the above code, the word VALUE in the doc_head event is a variable. Most variables are populated by SAS, and allow you to anticipate and control aspects of the markup based on their values.

SAS also allows you to create your own variables with the SET statement. For example:

```
set $SECTION_NAME "head";
```

These variables can be used later to make decisions, as in the following code:

```
put "<script src=script.js></script>" NL / if cmp("head",$SECTION_NAME);
```

This code will output the script tag only if the $SECTION_NAME variable contains the value "head".

Having understood the basic elements of a tagset definition, let's return to the PTHTML tagset. The PPTHTML tagset is similar to the HTML4 tagset, with the addition of some extra tags and tag attributes. The PPTHTML definition can be found in **Appendix B: Creating a PowerPoint HTML Tagset**
.

Here are the key changes made in the PPTHTML definition:

- Added a META tag to identify the files as a PowerPoint slide.
- Added a script reference to "script.js". This script file is generated by PowerPoint, and contains JavaScript functions that mimic regular PowerPoint behavior.

- Added some events to the BODY tag.
- Added a DIV tag to identify the body as a Slide, and further define some style attributes.

Once the tagset has been defined, you can use it to create output. To use the tagset, simply reference the library and tagset name in the ODS statement, and then output your data using whatever means you prefer:

```
ods tagsets.ppthtml path="c:\sassy\Population_files" body='slide0002.htm';

proc print data=midwestpop;
title "Midwest Population Counts by State";
title2 "1980 - 1990";
run;

ods tagsets.ppthtml close;
```

The code above generates a PowerPoint HTML slide with a title and a table. As you can see, once the tagset is created, using it to generate presentations is trivial. **Appendix C: Creating a PowerPoint Presentation with custom Tagset** contains the ODS code to generate a simple graph.

Note that this tagset will not create an entire Web PowerPoint presentation. It will only create slide pages to update or expand an existing presentation. Several other files are necessary to create the proper infrastructure for the presentation. This version of PPTHTML does not create the infrastructure due to the complexity involved. With additional work, the technique could be expanded to generate the entire presentation.

Here are the advantages and disadvantages of the ODS Template technique:

Advantages

- Don’t have to learn VBA
- No additional licenses
- SAS can drive the system
- Potential for full control over the presentation

Disadvantages

- Hard to understand PowerPoint HTML/XML without a specification
- Takes a lot of time

CONCLUSION

This paper described three techniques for getting SAS Data into Microsoft PowerPoint. The first technique described how to pull data from SAS using PowerPoint automation. The second technique showed how to link PowerPoint to Excel, and then populate the Excel workbook from SAS. The third technique used the new Proc Template procedure in SAS9 to create a PowerPoint-readable HTML slide. Each technique has advantages and disadvantages depending on the requirements of the project and the resources available.

For any questions regarding these techniques, please contact the author using the information provided below.

ACKNOWLEDGMENTS

Thanks to Kevin Kramer from Comsys IT Partners for help with ODS.

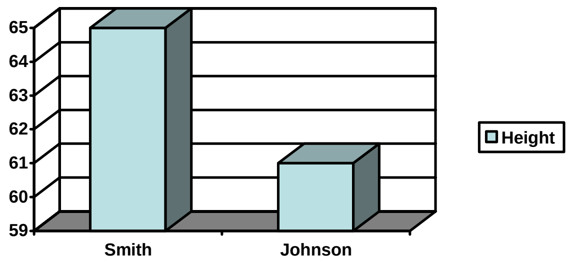
CONTACT INFORMATION

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

David Bosak
Principal Consultant
Comsys IT Partners
5278 Lovers Lane
Kalamazoo, MI 49002

Email: dbosak@comsys.com
Phone: 269-344-4100 ext 536

Appendix A: PowerPoint Automation - Local Provider



```
Sub ExtractSASData()
  Dim conn As Object
  Dim rst As Object

  On Error GoTo ErrorHandler

    ' Create the Connection and Recordset objects
    Set conn = CreateObject("ADODB.Connection")
    Set rst = CreateObject("ADODB.Recordset")

    ' Set the provider to the SAS local Provider
    conn.Provider = "sas.LocalProvider.1"

    ' Set the Data Source to point to the DataSet library like a libref
    conn.Properties("Data Source") = "c:\sassy"
    conn.Open

    'Open the patients dataset
    rst.CursorLocation = 3 'adUseClient
    rst.Open "patients", conn, 0, 1, 512

    'Reference the selected Graph object and Datasheet
    Set oGraph = ActiveWindow.Selection.ShapeRange.OLEFormat.Object
    Set oApp = oGraph.Application
    Set oSheet = oApp.DataSheet

    'Get the name and height for the first two patients and
    'put them in the datasheet for the graph
    oSheet.Cells(1, 2) = rst.Fields("lname") 'First patient is Smith
    oSheet.Cells(2, 2) = rst.Fields("Height")
    rst.MoveNext
    oSheet.Cells(1, 3) = rst.Fields("lname") 'Second patient is Johnson
    oSheet.Cells(2, 3) = rst.Fields("Height")

    conn.Close
    Set rst = Nothing
    Set conn = Nothing

    Exit Sub
ErrorHandler:
      MsgBox Err.Description, vbInformation
End Sub
```

Appendix B: Creating a PowerPoint HTML Tagset

```
ods path sasuser.templat (update)
   sashelp.tmplmst (read);

proc template;
   define tagset tagsets.ppthtml /store=sasuser.templat;
    notes "This is the PowerPoint HTML definition";
      parent=tagsets.html4;
        define event doc_head;
         start:
                set $SECTION_NAME "head";
            put "<head>" NL;
            put VALUE NL;
         finish:
                put "<p:slidetransition advancetime=""0"" effect=""push""
direction=""left"" flag=""1""/>" NL;
            put "</head>" NL;
      end;
              define event doc_meta;
         put "<meta name=""Generator"" content=""SAS Software, see
www.sas.com""";
         set $generic getoption("generic");
         putq " sasversion=" SASVERSION /if cmp( $generic, "NOGENERIC");
         put $empty_tag_suffix;
         put ">" NL;
         trigger show_charset;
         put "<meta " VALUE $empty_tag_suffix ">" NL /if exists( value);
         break /if ^any( htmlcontenttype, encoding);
         break /if ^exists( htmlcontenttype) and  ^ exists ( $show_charset);
         put "<meta";
         put " http-equiv=""Content-type"" content=""";
         put HTMLCONTENTTYPE;
         put "; " /if exists( HTMLCONTENTTYPE, encoding, $show_charset);
         put "charset=" encoding /if exists( $show_charset);
         put """";
         put $empty_tag_suffix;
         put ">" NL;
         put "<meta name=ProgId content=PowerPoint.Slide>";
      end;


        define event javascript;
         start:
            put "<script src=script.js></script>" NL / if cmp("head",
$SECTION_NAME);
            put "<script language=""javascript"" type=""text/javascript"">"
NL;
            put "<!-- " NL;
         finish:
            put NL "//-->" NL;
            put "</script>" NL;
            put "<noscript></noscript>" NL;
      end;

      define event doc_body;
           start:
                set $SECTION_NAME "body";
            put "<body lang=EN-US ";
            put " onload=""LoadSld()""";
            put " onclick=""DocumentOnClick()""";
                  put " onresize=""_RSW()""";
                  put " onkeypress=""_KPH()""";
            put " bgproperties=""fixed""" / WATERMARK;
            putq " class=" HTMLCLASS;
            putq " background=" BACKGROUNDIMAGE;
            trigger style_inline;
            put ">" NL;
            trigger slideobj;
            trigger pre_post;
            put NL;
            trigger ie_check;
         finish:
            trigger pre_post;
            trigger slideobj;
            put "</body>" NL;
      end;

        define event slideobj;
          start:
                put "<div id=SlideObj
style='position:absolute;top:0px;left:0px;" NL;
            put "width:534px;height:400px;font-size:16px;background-
color:white;clip:rect(0%, 101%, 101%, 0%);" NL;
            put "visibility:hidden;filter:revealtrans(Duration=1,
Transition=7)'>" NL;
              finish:
            put "</div>" NL;
        end;


     notes "Clear out startup and shutdown functions";
       define event startup_function;
     end;
     define event shutdown_function;
     end;


  end;
run;
```

Appendix C: Creating a PowerPoint Presentation with custom Tagset

```
/* Create data */
data midwestpop;
   input State $ City80 City90
         Rural80 Rural90;

  datalines;
OH   8.791   8.826  2.007  2.021
IN   3.885   3.962  1.605  1.582
IL   9.461   9.574  1.967  1.857
MI   7.719   7.698  1.543  1.598
WI   3.176   3.331  1.530  1.561
MN   2.674   3.011  1.402  1.364
IA   1.198   1.200  1.716  1.577
MO   3.314   3.491  1.603  1.626
ND    .234    .257   .418   .381
SD    .194    .221   .497   .475
NE    .728    .787   .842   .791
KS   1.184   1.333  1.180  1.145
;

/* Set graphics options */
options nofmterr orientation=landscape linesize=256;
goptions reset=global gunit=pct border ftext=swissb htitle=6 htext=3;

/* turn off listing */
ods listing close;

/* Turn on ODS for Slide 2, referencing our custom tagset */
ods tagsets.ppthtml path="c:\sassy\Population_files" body='slide0002.htm'
style=sasweb;
ods results=on;

proc print data=midwestpop;
title "Midwest Population Counts by State";
title2 "1980 - 1990";
run;

/* Configure ODS for Slide 3 */
ods tagsets.ppthtml path="c:\sassy\Population_files" body='slide0003.htm'
style=sasweb;

title "1980 Population Chart";
title2;

goptions device=gif gsfname="c:\sassy\Population_files" gsfmode=replace;
/* Create chart */
proc gchart data=midwestpop();
vbar State
  / sumvar=City80
    coutline=gray
      caxis=blue
    noheading
    legend
    name="popchart"
;
run;


/* close ODS */
ods tagsets.ppthtml close;
/* turn listing back on */
```

```
ods listing;
```