Paper 228-2010

Automate PowerPoint Slide Generation with ODS and VBScript

Ya Huang, Amylin Pharmaceuticals, Inc., San Diego, CA

ABSTRACT

SAS® can generate nice figures and tables in Microsoft Word and Microsoft Excel format, but it hasn't been easy for SAS to generate Microsoft PowerPoint slides. This paper introduces a method that combines the power of Windows Scripting technology with SAS ODS and SAS/GRAPH® to automate the process of PowerPoint slides generation. Using this method, a PowerPoint presentation with dozens of slides made of figures and tables can be generated by a single SAS run in just a couple of minutes! Better yet, the figures and tables are editable, almost like native PowerPoint tables and figures

INTRODUCTION

Often Clinical Scientists present clinical study results with the following three types of PowerPoint slides:

1. Bullet point slide to make statements. Usually made by typing the statement directly into PowerPoint.
2. Table slides to present tabulated efficacy and safety results. Usually hand made by creating an empty table in PowerPoint, then manually filling in the numbers from Table/Figure/Listing (TFL).
3. Figure slides (line plot, bar chart etc.) to deliver efficacy results. Usually made by entering data into GraphPad Prism®, and editing the figure in Prism.

Among the three types of slide, Bullet point slide can hardly be automated, nor is it necessary. For the other two, there are compelling reasons for automation: Manual process is time consuming, error prone; Manual process may have to start all over again if the underlying data is updated.

To automate the process, we need to find a way to transfer SAS tables and figures into PowerPoint. With ODS, we can create tables and figures in many formats, such as RTF, PDF and EXCEL etc. Unfortunately, there is no PowerPoint destination built into ODS yet. Before ODS, we used DDE to transfer figures and tables into MS Word. So naturally, we would try to do the same thing for PowerPoint. However Microsoft's support of DDE in PowerPoint is very weak. Many people have tried to go that route to no avail. Fortunately Microsoft has another technology called Windows Scripting which can be used for this kind of job.

Windows Scripting is conceptually similar to DOS batch programming. In DOS batch programming, we use DOS commands as programming language to write the script as a BAT file. Windows Script Host (WHS) is a newer and more advanced environment than DOS. Scripts running in WHS can do much more complex jobs than DOS batch program. WHS allows several different languages to run on it. Most popular one is VBscript (Visual Basic, Scripting Edition).

Similar to DDE, we run scripts to mimic the manual process. For example, we want to insert figures into PowerPoint slide. The first thing we do is open up PowerPoint application, then click "create new presentation", then select "blank presentation", then we go to "Insert" -> "Picture" -> "From file" and select the figure file we need to insert. When we write a script, we basically try to put together some codes, so that each line corresponds to one of the steps we do manually.

Windows Scripting alone can be a topic for many papers. It is definitely beyond the scope of this paper. The best place to learn Scripting is MicrosoftTechNet, Scriptcenter.

In this paper, we will focus on how Windows scripting (VBscript to be specific) is used to transfer figures and tables from SAS into PowerPoint. The discussion will be based on examples. For each example, we will discuss what the key statements and/or parameters are about. Since the methods of transferring figures and tables to PowerPoint are different, we will discuss them separately.

SAS FIGURE TO POWERPOINT

PowerPoint accepts many types of graph file formats, such as JPG, GIF, PNG and EMF. All the formats do not have the same presentation quality though. Bitmap type format, such as JPG, GIF and PNG have rough edges for font and other objects. When it's shown on the projector screen, it is blurred, or not 'Crystal Clear' in Clinical Scientist's word.

Another drawback of bitmap format is that it cannot be edited. To get the best quality and editable figures, we have to use vector format, which for Windows is EMF. To generate EMF format figure, we use SASEMF driver. The old EMF driver works too, but SASEMF seems to have better font control.

Manually, there are two main methods to add an external figure to PowerPoint slide: **Insert** and **Copy/Paste**. **Insert** is a simple process that can be done by PowerPoint alone. **Copy/paste** is a bit complicated though. Depends on where the source figure is from, a Windows application is needed to open the figure file first. For example, we want to move a figure from EXCEL to PowerPoint, we have to use Excel to open that file first, then we can copy it and paste to PowerPoint. As we mentioned earlier, script mimics the manual process. So a simple manual process like **Insert** usually means script is simple too. For that, we will use **Insert** instead of **Copy/Paste** method to move figure into PowerPoint.

SAS TABLE TO POWERPOINT

One cannot insert an existing table into PowerPoint. One way to transfer SAS table into PowerPoint is to make the table a "figure", and then insert it into PowerPoint. The only ODS destination that can convert a table into a "figure" is ODS/EPS. EPS based table has good quality, it can even be edited.

Can we get a real nice table just like the native PowerPoint one? A simple test shows that we can copy/paste a Word table into PowerPoint and it is just like a native PowerPoint table! SAS ODS/RTF can easily create a nice Word table and has full control of the font, color, border etc. If we properly control all these attributes in SAS ODS, we can avoid manual editing of the table.

Since table cannot be inserted into PowerPoint, we need to use Copy/Paste method, and Word will be used to open the RTF file.

EXAMPLE 1, OUR VERY FIRST VBSCRIPT

To write our very first VBscript, let's open Notepad and type in the following line. We then save the file as "test.vbs" under c:\temp.

```
Wscript.echo "Hello World!"
```

Now to run the script, we open the command line window and type in this:

```
C:\temp>cscript test.vbs
```

If everything goes smoothly, we should see "Hello World!" on the screen!

As you can see, VBscript is basically a plain text file. It can be created/edited with any available text editor. To run it, we need command-based script host CScript.exe. Obviously, test.vbs is too simple to do anything meaningful for us. But we can expect that with more complex scripts, we will be able to do much more.

EXAMPLE 2, INSERT A SINGLE FIGURE INTO POWERPOINT

The following script opens up PowerPoint application, and inserts a figure from file c:\temp\test1.emf. It then closes PowerPoint application and saves the document as figure.ppt.

```
1. Set objPPT = CreateObject("PowerPoint.Application")
2. Set objPresentation = objPPT.Presentations.Add()
3. Set objSlide = objPresentation.Slides.Add(1,12)
4. Set pic = objSlide.shapes.AddPicture("C:\temp\test1.emf",-1,-1,72,144)
5. objPresentation.SaveAs("C:\temp\figure.ppt")
6. objPresentation.Close
7. objPPT.Quit
```

EXPLANATION OF THE SCRIPT:

1. To start the PowerPoint application
2. To open up a presentation
3. To add a new slide with blank layout (no title, nosubtitle). There are two arguments in Add() method, first one is to set the slide number. Since we only have one slide, it is set to 1 here. The second one is to set slide

layout. “12” is for blank layout, we will use 12 throughout the paper. For the curious minded, one can either play around to see what other number brings to you, or search MSDN library for the list of possible values.

4. To insert a figure c:\temp\test1.emf into slide 1. There are eight arguments in AddPictures() method. The first one is the file name of the figure. Second and third arguments tell PowerPoint t to make a physical copy of the source figure or a link. We want a physical copy so that slide can be used elsewhere. The value for physical copy is “-1,-1”. The fourth one is the X coordinate of the starting position of the figure from left border, measured in points (72 points=1 inch). The fifth one is the Y coordinate of the starting position of the figure from top border. For this example, test1.emf will be inserted one inch from the left and two inches from the top. The sixth and seventh arguments are to set the size of figure, since we decide to control the size of figure in SAS, we don’t have to use them here.
5. To save the presentation as figure.ppt into c:\temp.
6. To close the presentation.
7. To quit PowerPoint application.

EXAMPLE 3, CREATE MULTIPLE SLIDES PRESENTATION

To create multiple slides in a presentation, we need to repeat the lines 2 and 3, and set the first argument in Add() method to the slide number. For example, if we replace line 2 and 3 with the following, we will get a two-slide PPT file, with first slide figure from test1.emf, second slide figure from test2.emf.

```
Set objSlide = objPresentation.Slides.Add(1,12)
Set pic = objSlide.shapes.AddPicture("C:\temp\test1.emf",-1,-1,72,144)
Set objSlide = objPresentation.Slides.Add(2,12)
Set pic = objSlide.shapes.AddPicture("C:\temp\test2.emf",-1,-1,72,144)
```

EXAMPLE 4, PANELED FIGURES WITHOUT USING PROC GREPLAY

In traditional SAS graph, Proc greplay is used to create paneled figures, such as two figures side by side etc. However, Proc greplay is quite cumbersome. Even an experienced programmer may have hard time making really nice figures out of it, mostly because it’s difficult to get a good control on the font. Surprisingly, with VBscript, we can easily create paneled figure without any distortion.

Replace Example 2, line 2 and 3 with the following we will get a slide with test1 and test2 side by side.

```
Set objSlide = objPresentation.Slides.Add(1,12)
Set pic = objSlide.shapes.AddPicture("C:\temp\test1.emf",-1,-1,72,144)
Set pic = objSlide.shapes.AddPicture("C:\temp\test2.emf",-1,-1,360,144)
```

Note that first figure is placed one inch from left, and second figure is placed next to it at five inches from left. If we want, we can even create partially overlapped figures if the starting position is properly controlled.

EXAMPLE 5, USE DESIGN TEMPLATE FOR THE PRESENTATION

Often it is desired that all slides are on a standard company-wise template. This can be easily accomplished by adding one line after line 2 in Example 2:

```
objPresentation.ApplyTemplate("C:\temp\tmp3.pot")
```

PowerPoint built-in templates can be found at C:\Program Files\Microsoft Office\Templates\Presentation Designs. If you have your own template file, you can specify here (tmp3.pot for example). To learn how to create your own template, you can check this link: http://office.microsoft.com/en-us/powerpoint/HP011474521033.aspx.

EXAMPLE 6, THE COMPLETE CODE FOR SAS FIGURES TO POWERPOINT

In this example, one Proc Gplot is used to create a scatter plot as test1.emf. One Proc Gchart is used to create a bar chart as test2.emf. They are both saved in temporary WORK folder, so that once SAS session ends, they are discarded automatically. Note that we use SASEMF driver as we discussed earlier. We also control many attributes of figure by setting the proper goptions. For example, we mentioned earlier that test1.emf is four inches wide. It is controlled by hsize=4in goption. Since SASEMF driver has better font control, we are able to use font 'Wingdings 2' for symbol. Visually, it looks much better than SAS software font symbol.

Next to the SAS/Graph procedures is a data _null_ step to write the VBscript. This script is basically the same as in

Example 6. The only difference is that file name is replaced with WORK folder path. The script itself, namely "emf2ppt.vbs" is also saved in temporary WORK folder.

There are many ways to run a script from within SAS. For example, X command, CALL SYSTEM() routine etc. Here we use PIPE method. We prefer PIPE method because it can read in messages from WHS, which may give clues why a script run fails. This message is not available in SAS log.

```
filename fout "%sysfunc(pathname(work))\test1.emf";
goptions reset=all device=sasemf hsize=4in vsize=4in
         gsfname=fout gsfmode=replace ftext='Arial/bo';

axis1 label=(a=90);
symbol1 color=red  i=none font='Wingdings 2' v='98'x h=11pt;
symbol2 color=blue i=none font='Wingdings 3' v='71'x h=11pt;
legend1 label=none position=(bottom right inside) down=2;

proc gplot data=sashelp.class;
plot weight*height=sex /vaxis=axis1 legend=legend1 noframe;
run;

filename fout "%sysfunc(pathname(work))\test2.emf";
goptions reset=all device=sasemf hsize=4in vsize=4in
         gsfname=fout gsfmode=replace ftext='Arial/bo';

axis1 label=(a=90);
axis2 label=('Age');
axis3 label=none;
pattern1 color=red value=s;
pattern2 color=blue value=s;
legend1 label=none position=(top right inside) down=2;

proc gchart data=sashelp.class;
vbar age / subgroup=sex axis=axis1 maxis=axis2 gaxis=axis3
           legend=legend1 patternid=subgroup noframe;
run;

filename script "%sysfunc(pathname(work))\emf2ppt.vbs";
data _null_;
file script;
put 'Set objPPT = CreateObject("PowerPoint.Application")';
put 'Set objPresentation = objPPT.Presentations.Add()';
put 'objPresentation.ApplyTemplate("C:\temp\tmp3.pot")';
put 'Set objSlide = objPresentation.Slides.Add(1,12)';
put 'Set pic = objSlide.shapes.AddPicture("' "%sysfunc(pathname(work))\test1.emf" '",-
1,-1,72,144)';
put 'Set pic = objSlide.shapes.AddPicture("' "%sysfunc(pathname(work))\test2.emf" '",-
1,-1,360,144)';
put 'objPresentation.SaveAs("C:\temp\figure.ppt")';
put 'objPresentation.Close';
put 'objPPT.Quit';
run;

filename rs pipe "cscript //nologo ""%sysfunc(pathname(work))\emf2ppt.vbs""";
data _null_;
infile rs;
input;
put _infile_;
run;
```

The code above will generate a PowerPoint slide as shown in figure 1. As we can see, we've got a scatter plot side by side to a bar chart, each has nice font, color etc. We also made the slide on a nice template.

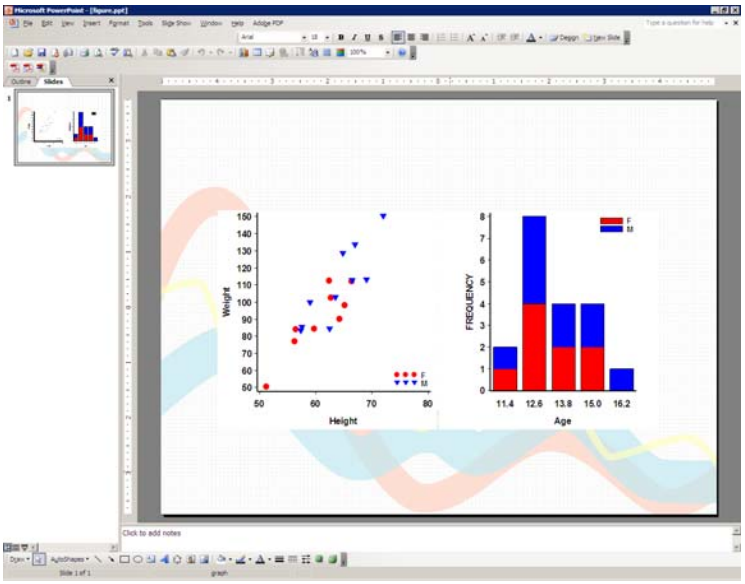

Figure 1. Paneled Figures on Design Template

EXAMPLE 7, THE COMPLETE CODE FOR SAS TABLE TO POWERPOINT

In this example, we use ODS/RTF along with Proc Tabulate to create a Word table. We control the table attribute by using different styles. As figure files in Example 6, the RTF file is saved in temporary WORK folder too, so that it can be discarded after the SAS session ends.

The VBscript part again is written in a data _null_ step. It is similar to the figure script, except some lines are added to start up Word, which in turn open the newly created RTF file, so that copy/paste method can be used to transfer the table.

```
ods rtf file="%sysfunc(pathname(work))\tb1.rtf" style=rtf;

proc tabulate data=sashelp.class;
var weight / style=[font_size=40pt];
var height / style=[font_size=40pt];
class sex;
classlev sex / style=[cellwidth=1in font_size=40pt];
table sex='', (weight height)*sum=''*[style=[cellwidth=3in font_size=40pt]];
run;

ods rtf close;

filename script "%sysfunc(pathname(work))\wdtb2ppt.vbs";

data _null_;
file script;
1.  put 'Set objWord = CreateObject("Word.Application")';
2.  put 'objWord.Visible = False';
3.  put 'Set objDoc = objWord.Documents.Open("' "%sysfunc(pathname(work))\tb1.rtf"
    '")';
4.  put 'Set objSelection = objWord.Selection';
5.  put 'Set objPPT = CreateObject("PowerPoint.Application")';
6.  put 'objPPT.Visible = True';
7.  put 'Set objPresentation = objPPT.Presentations.Add';
8.  put 'objPresentation.ApplyTemplate("C:\temp\tmp3.pot")';
9.  put 'objDoc.Activate';
10. put 'objDoc.Tables(1).Range.Select';
11. put 'objSelection.Copy';
12. put 'Set objSlide = objPresentation.Slides.Add(1,12)';
13. put 'objPPT.ActiveWindow.View.Paste';
14. put 'objPresentation.SaveAs("c:\temp\table.ppt")';
```

```
15. put 'objPresentation.Close';
16. put 'objPPT.Quit';
17. put 'objDoc.Close';
18. put 'objWord.Quit';
run;

filename xx pipe "cscript //nologo ""%sysfunc(pathname(work))\wdtb2ppt.vbs""";

data _null_;
infile xx;
input;
put _infile_;
run;
```

EXPLANATION OF THE SCRIPT:

1. Start Word application
2. Set Word to invisible mode
3. Open tb1.rtf file with Word
4. Select the document
5. Start PowerPoint application
6. Set PowerPoint to visible mode
7. Open a new presentation
8. Add template
9. Make Word document Active
10. From the Word document, find the first table
11. Copy into clipboard
12. Switch to PowerPoint and add slide 1 with blank layout
13. Paste the table from clipboard to PowerPoint window
14. Save the presentation
15. Close the presentation
16. Quit PowerPoint
17. Close Word document
18. Quit Word

Line 10 tells Word to pick the first table in the RTF file. If multiple tables are in the file, small change to the code is needed, so that it can loop through all the tables and copy/paste to PowerPoint slide.

The code above will generate a PowerPoint slide as shown in figure 2. If we click the table, we can even edit it just like a native PowerPoint table.

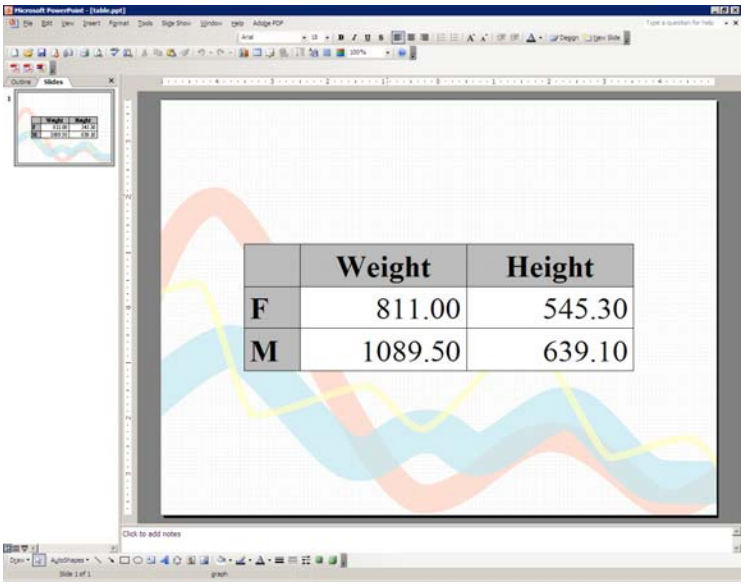

Figure 2. Table on Design Template

EXAMPLE 9, TRANSPARENT FIGURES TO POWERPOINT

In example 6, background is partially blocked by the figures. Sometimes, we want figure on a watermark background. This requires transparent figures, which can be generated with CGM driver. The drawback of CGM driver is that font control is not as good as SASEMF. Figure 3 shows a "DRAFT" watermarked slide.

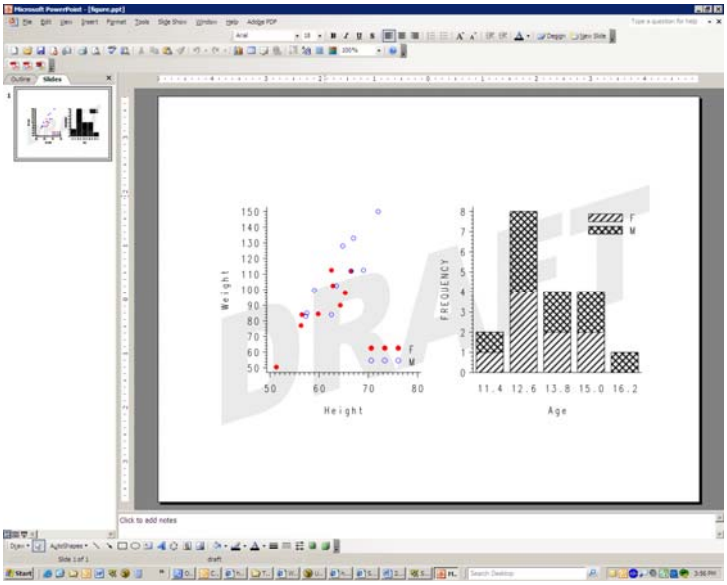

Figure 3. Transparent Figures on Watermark Template

CONCLUSION

Windows scripting technology combined with SAS ODS and SAS/Graph to automate the generation of PowerPoint presentation has proven successful. In doing so, we also found a pleasant surprise, i.e., we can make paneled figure much easier than using SAS/Graph alone.

REFERENCES

MicrosoftTechnet, Script Center. http://www.microsoft.com/technet/scriptcenter.

Microsoft MSDN library. http://msdn.microsoft.com/en-us/library.

ACKNOWLEDGMENTS

Author would like to thanks Tamra Roddy for reviewing this paper.

CONTACT INFORMATION

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

Name	Ya Huang
Enterprise	Amylin Pharmaceuticals, Inc.
Address	9360 Towne Centre Drive
City, State ZIP	San Diego, CA 92121
Work Phone:	(858) 642-7288
E-mail:	ya.huang@amylin.com