Paper FS-04-2013

SAS Automation - From Password Protected Excel Raw Data To Professional-Looking PowerPoint Report

Jeff Hao, PhD, JPMorgan Chase & Co., Columbus, OH

ABSTRACT

In banking and finance industry, the most common report format presented to the executives is Microsoft PowerPoint (PPT). How to realize the SAS automation process from Excel raw data to PPT report is a tremendous challenge.

The first step of the process is to directly import password protected Excel raw data into SAS data sets. The second step is to convert the tables and graphs to be used on PowerPoint slides into image formats such as PNG or JPG. The last step deals with exporting the tables and figures images into customized professional-looking PowerPoint report. Actually, the first and third steps are the most difficult parts of the whole automation process.

By combining SAS with DDE, Proc Template, SAS/Graph Statistical Graphics (SG) procedures, Graph Template Language (GTL), Proc Report, ODS Printer, and VBScript, the whole automation process comes true.

This paper presents a detailed step by step approach to the seamless automation process from password protected Excel raw data to PowerPoint report by a single SAS run click.

INTRODUCTION

Microsoft® Excel is an excellent tool for formatting and presenting data. For security sake, especially for sensitive materials, Excel files are actually password protected to restrict access to the information.

SAS® has become a powerful business solution tool for clinical trial reporting, data management, statistical analysis, econometrics, data mining, business planning, forecasting, and decision supports, etc. due to its versatile functions, especially its automation capability.

Without password protection, MS Excel files can be imported into SAS data sets in one of the following ways: 1) Proc Import; 2) Proc Access; 3) Using Import Wizard (Manually); 4) Using DDE routine syntax; 5) Using SAS/Access to ODBC (Proc SQL); 6) Using JMP (Manually or programmatically); or 7) Using Data step (saving the Excel file into CSV first, then using FILENAME and INFILE statements). Since this content is beyond scope of the paper, we will not discuss it in detail here.

However, if we use PROC IMPORT to read a password protected Excel file, it will return the following error message:

```
ERROR: Connect: External table is not in the expected format.
ERROR: Error in the LIBNAME statement.
```

Using SAS/ACCESS to ODBC (Open Database Connectivity) will return the same error message as PROC IMPORT. In fact, none of the methods mentioned above will enable us to read a password protected Excel file into SAS data set.

To solve this issue, one simple way is to open the file with password and then to save it into an Excel file without password, and finally using PROC IMPORT to read the file into SAS data set. If there is only one such files, it won't be an issue.

However, if there are more than 5, 10, 20, 50, even more than 100 such excel files, it will be time-consuming and error–prone to do so. Meanwhile, from the standpoint of security, it's not a recommended practice for us to save a high sensitive file into a none password protection file first, then destroy it after SAS read it. Unfortunately, we are facing this challenge everyday. It's necessary to find a way to read high sensitivity files into SAS data sets without sacrificing security concern.

Can we find an easy way to deal with this challenge? The answer is 'Yes'. We can use Dynamic Data Exchange (DDE) to set up the communication bridge between SAS and Microsoft Excel so that SAS can use X4ML commands

to import the password protected Excel files. This is the first step of the SAS automation process.

DDE is a method to dynamically exchange data between MS-Windows applications. DDE uses a client-server relationship to enable client applications to request data from a server application. In the context of SAS and DDE, SAS is always the client, regardless of which application is receiving the data. SAS request data from server applications, sends data to server applications, or sends commands to server applications (Smith, 2010).

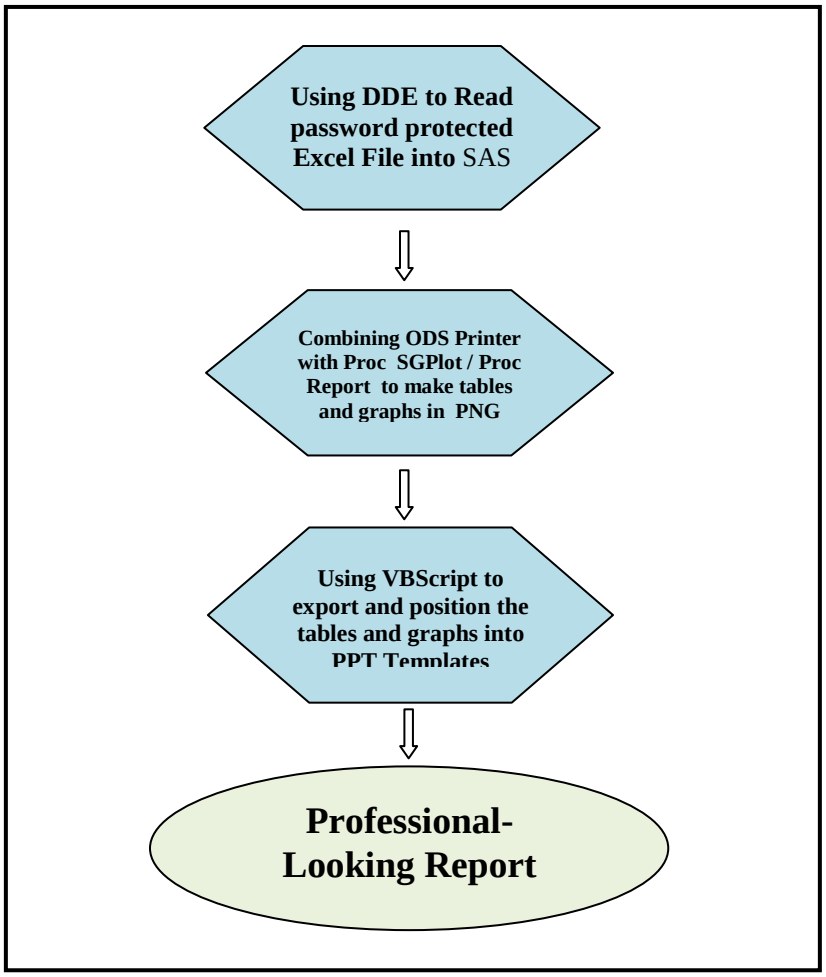


Figure 1. The Flow Chart of SAS Automation Process

It's worthy noting that the SAS code presented in this paper for opening password protected Excel file is to increase the security of the sensitive information contained in the Excel file, not decrease the security. The password should be stored in different path than the SAS code. In the SAS code, the password is in macro variables, not the true value. It's the user's responsibility to conform to the compliance issues for security.

Usually, in banking industry, the PPT reports contain both of tables and graphs and their positions on the slide are strictly pre-determined and formatted.

To positioning the tables and figures on a PPT file, the tables and figures must be in image formats such as PNG, JPEG, etc. Once we finish the first step, the next challenge is how to export SAS graphs and tables into image files.

However, up to now, SAS does not have ODS PPT yet although it has ODS RTF, ODS Excel XP, ODS document, ODS HTML, ODS Markup, ODS package, and ODS PDF, etc. We have to find an alternative way to output PPT file.

Graph (line plot, bar chart etc.) images can be easily made by Combining SAS/Graph SG procedures such as PROC

SGPLOT, PROC SGPANEL, PROC SGRENDER, PROC SGSCATTER, or PROC SGDESIGN, etc. with ODS printer etc. Table images can be made by combining PROC REPORT, PROC TABULATE, PROC FREQ, or PROC PRINT with ODS printer etc. The key thing is how to define paper size and DPI resolution to control the image size since we prefer achieving the best quality of the images without increasing the file size.

In the last step of the automation process, we will combine SAS with VBScript to export the tables and graphs into the PPT template. In VBScript code, we define the positions of the tables and graphs (X and Y axis coordinators), the size of each image on the slide, and the path of the output PPT file.

Figure 1 describes the flow chart of the whole process of SAS automation from the password protected Excel raw data to professional-looking PowerPoint report. Without losing generality, in this paper, we will use SASHelp.USEcon data set as an example to discuss each step in detail respectively.

DIRECTLY READING PASSWORD PROTECTED EXCEL FILE INTO SAS

With password protection, MS Excel files can be imported in following three ways: 1) manually saving the Excel file as a non-password protection file first, then using the methods listed above to import it into SAS data set; 2) Using DDE Macro to read the excel file with the password, saving a copy of the Excel file without the password to a separate directory, reading the non- password protected copy via PROC IMPORT procedure, finally, destroying the non- password protected Excel copy after importing (For detailed information, please refer to SAS-L with subject 'Import password protected .xlsx file in SAS'); and finally 3) directly reading password protected Excel file into SAS via DDE. In this paper, we will mainly discuss the third method.

In MS Excel, there are two level password protections. One level password protections is "password to open" (Read and Write), which only has one password set for the file. The two level password protections has both of "password to open" (Read Only) and "password to modify" (Read and Write), which has two different passwords set to open the file. We will discuss the two scenarios in detail respectively.

In this paper, for demonstration purpose, we will export the SASHelp.USEcon data set into password protected MS Excel to local drive first, then use DDE statements to demonstrate how to directly read the password protected MS Excel file into SAS under the two scenarios.

Before we start to discuss the first step of SAS automation process, let's export the sample data set into one and two level password protected MS Excel files via DDE in local drive:

OUTPUT ONE LEVEL PASSWORD PROTECTED EXCEL FILE

The following code is to save a Excel file with one level password:

```
%let sasdt=SASHelp.USEcon;
%let deleteit=yes;
%let xls=C:\temp\USEcon.xlsx;
%let xls1=C:\temp\USEcon_one_level.xls;
%let xls2=C:\temp\USEcon_two_level.xls;

PROC EXPORT outfile="&xls" data=&sasdt dbms=excel replace;
RUN;

/*First, SAS writes an Excel file here without the password. */
/*starts the Excel application.  */
OPTIONS NOXWAIT NOXSYNC;
DATA _null_;
rc = system('start excel');
rc = sleep(3);
RUN;

/*SAS opens the Excel worksheet and save it with one level password.*/
FILENAME xls DDE 'excel|system';
DATA _null_;
```

```
   FILE xls;
   PUT '[open("'"&xls"'")]';
RUN;

DATA _null_;
   FILE xls;
   PUT '[error("false")]';
   PUT '[save.as("'"&xls1"'",1,"'"&pswd_1"'")]';
   rc = sleep(3);
   PUT '[file.close(false)]';
   PUT '[quit]';
RUN;
```

After saving the Excel file with one level password protected, we finally delete the one without password protected.

```
/* Finally SAS destroys the Excel file without password protected. */
%if &deleteit=yes %then %do;
   systask command "del ""&xls"" ";
%end;
```

OUTPUT TWO LEVEL PASSWORD PROTECTED EXCEL FILE

The code for saving two level password protected Excel file is similar to that with one level password protected file mentioned above except the following difference. Before we save the file, we need to add the following code to ensure SAS has 5 seconds sleep before it uses DDE to save the fiel.

```
   data _null_;
    rc = sleep(5);
   run;
```

The 'Put' statement is as follows:

```
   PUT '[save.as("'"&xls2"'",1,"'"&pswd_1"'",,"'"&pswd_2"'" )]';
```

Now we will use the newly created one level and two level password Excel files as examples of SAS automation process.

DIRECTLY READING ONE LEVEL PASSWORD PROTECTION EXCEL FILE INTO SAS

The arguments for DDE to open Excel file is as follows. For detailed information, please refer to Microsoft Support (2005) for detailed information.

```
/*DDE to Excel has the 'OPEN' function with the following arguments:*/
  OPEN(file_text, update_links, read_only, format, prot_pwd, write_res_pwd,
       ignore_rorec, file_origin, custom_delimit, add_logical, editable, file_access,
       notify_logical, converter)
```

Through DDE, SAS can easily read one level password protected Excel file. Please note, in the Put '[OPEN]' statement, 'TRUE'/'FALSE' is for On/Off of 'read only'. 'TRUE' is default for 'read only'. In 'INFILE' statement, the options of MISSOVER and DSD must be applied to deal with missing value. The SAS code is as follows:

```
%let path= C:\temp;
%let file= USEcon_one_level.xls;
OPTIONS NOXWAIT NOXSYNC;
DATA _null_;
rc=system('start excel');
rc = sleep(5);
RUN;
```

```
OPTIONS NOXWAIT NOXSYNC;
FILENAME data DDE 'excel|system';
DATA _null_;
FILE data;
PUT '[error(false)]';
PUT '[OPEN("'"&path"'\'"&file"'",,TRUE,,"'"&pswd_1"'")]';
rc =sleep(15);
RUN;

FILENAME dt1 DDE "excel|&path.\[&file.]USECON!r2c1:r253c12" ;
RUN;

DATA USEcon_1;
infile dt1 missover dsd dlm='';
INFORMAT DATE mmddyy10. AIRRPMD 12.  AIRRPMT 12. CHEMICAL 12. COAL 12.  DURABLES 12.
  HS1FAM 12. HSTOTAL 12.  NONDUR 12. PETROL 12. TOBACCO 12. VEHICLES 12.;
INPUT DATE AIRRPMD AIRRPMT CHEMICAL COAL DURABLES HS1FAM HSTOTAL NONDUR PETROL TOBACCO
  VEHICLES;
FORMAT _all_;
FORMAT date mmddyy10.;
RUN;

DATA _null_;
   FILE data;
   PUT '[error(false)]';
   PUT '[file.close(false)]';
   PUT '[quit()]';
RUN;
```

Please note, the DDE triplet 'FILENAME' statement is to set up a DDE communication as follows:

```
FILENAME dt1 DDE "excel|&path.\[&file.]USECON!r2c1:r253c12" ;
```

The 'NOXWAIT' and 'NOXSYNC' options ensure the x window statement executes independently from SAS session.

The 'rc =sleep(15)' statement is very important since it puts SAS session to sleep for 15 seconds to avoid the DDE communication error before Excel file is fully opened for use.

DIRECTLY READING TWO LEVEL PASSWORD PROTECTION EXCEL FILE INTO SAS

The DDE code for SAS to read the two level password protected Excel file (read and write) is similar to the one for reading one level password protected Excel file except the following changes:

```
%let file= USEcon_two_level.xls;
PUT '[OPEN("'"&path"'\'"&file"'",,FALSE,,"'"&pswd_1"'","'"&pswd_2"'")]';
```

It will come out the Usecon_2 data set, which is exactly same as the data set Usecon_1. Thereafter, we will use the data set Usecon_1 to output tables and graphs images as the second step of the automation process.

ODS PRINTER PROCESS

In this step, we will first use Proc Template to customize all of style fonts, graph fonts, colors, margins, background and front ground colors, etc., and then export the template into SASuser.Templat. By adjusting the padding size in template and DPI in ODS PRINTER, we can control the tables and graphs image size to achieve the best image quality as needed. We will discuss procedures for the table and graphs separately.

The following is part of the code for Proc template to define style "Jeffhao". The parent style is styles.printer. By modifying the styles of fonts, graph fonts, table lines, table fonts, tables background /front ground color, and the

attributes of graphs such as histogram, ellipse, band, contour, and even the gridline attributes, we can use the style to make any customized tables and graphs as demanded. We can also use GTL to customize any styles as needed.

```
PROC TEMPLATE;
   DEFINE style Styles.Jeffhao / store = SASUSER.TEMPLAT;
      parent = styles.printer;
      style fonts /
        'docFont' = ("Trebuchet MS",7pt)
        'headingFont' = ("Trebuchet MS",6pt,bold)
        'headingEmphasisFont' = ("Trebuchet MS",6pt)
        'EmphasisFont' = ("Trebuchet MS",8pt,italic)
        'StrongFont' = ("Trebuchet MS",8pt,bold)
        'TitleFont2' = ("Trebuchet MS",8pt,bold italic)
        'TitleFont' = ("Trebuchet MS",9.5pt,bold italic);
      style GraphFonts /
        'GraphTitleFont' = ("<MTserif>, <serif>",7.0pt,bold)
        'GraphFootnoteFont' = ("<MTserif>, <serif>",7.0pt)
        'GraphLabelFont' = ("<MTserif>, <serif>",6.0pt,bold)
        'GraphValueFont' = ("<MTserif>, <serif>",6pt,bold)
        'GraphUnicodeFont' = ("<MTserif-unicode>",6.0pt,bold)
        'GraphDataFont' = ("<MTserif>, <serif>",6.5pt,bold);
      style Table from Output /
        borderwidth = 0.65pt
        borderspacing = 0.2pt
        padding = 1.2pt
        rules = ALL
        bordercollapse = separate;
      style Graph from Graph /
        borderspacing = 0.005pt
        borderwidth = 0.005pt;
      style color_list /
        'bg'  = white
        'fg'  = black
        'bgH' = CX4E7AA7
        'link' = blue;
      style Body from Document/
        pagebreakhtml = html('PageBreakLine')
        marginleft = 0cm
        marginright = 0cm
        margintop = 0cm
        marginbottom = 0cm;
     style colors
        ...
      style GraphColors /
        'gaxis' = cx000000
        'ggrid' = cx808080
        'goutlines' =cx000000
      class GraphGridLines /
        contrastcolor = GraphColors('ggrid')
        linestyle = 2
        linethickness = 1px
        displayopts = "auto";
   end;
RUN;
```

ODS printer is a Universal Printing, which is a printing system providing both interactive and batch printing capabilities to SAS applications and procedures on all the operating environments supported by SAS.

By default, Universal Printing is turned ON in all operating environments except Windows system. Therefore, before

use it, the UNIVERALPRINT system option must be set for Universal Printing to be ON. If Universal Printing is enabled in Windows, SAS overrides the use of the Windows system printer and causes ODS to use the Universal Printing (SAS Institute, 2010).

We can use Universal Printing to produce the following commonly recognized file types: GIF, PCL, PDF, PNG, PS, and SVG. The following ODS destinations that use the Universal Printing interface: ODS Printer, ODS PDF, ODS PS, and ODS PCL. In this paper, we use ODS Printer and output PNG images by defining PRINTERPATH=PNG.

OUTPUT TABLES INTO PNG IMAGE

The code for outputting table in image (PNG) format is as follows. Please keep in mind that we define the paper size as we need so that the resolution can be enhanced without increasing the image file's size. The printerpath can be any print format such as PNG, SVG, or JPG. Here we take PRINTERPATH=PNG.

```
TITLE;
FOOTNOTE;
OPTIONS NODATE NONUMBER NOCENTER  ORIENTATION=landscape SPOOL topmargin = 0.02 cm
bottommargin = 0.02 cm leftmargin = 0.02 cm rightmargin = 0.02 cm;
ODS NOPROCTITLE;
ODS ESCAPECHAR="^";
OPTIONS  NODATE PAPERSIZE=('xx cm','xx cm')    PRINTERPATH=png;
ODS _ALL_ CLOSE;
ODS PRINTER file="&path.&table " DPI=150 STYLE=Jeffhao;
```

	Resources		Products		
DATE	COAL	PETROL	CHEMICAL	VEHICLES	TOBACCO
MAY89	82486	12712	23244	21025	2,351
JUN89	78544	13077	24264	19346	2,581
JUL89	66269	12384	21819	11786	1,399
AUG89	90824	12350	22661	19082	2,028
SEP89	84618	12625	23602	20127	2,554
OCT89	89574	12777	22187	20217	2,230
NOV89	86965	12357	21805	20385	2,543
DEC89	72554	12738	22353	16653	2,827
JAN90	90276	13268	22131	13065	1,417
FEB90	81756	12571	23190	20275	1,829
MAR90	91292	12442	24548	21776	2,617
APR90	82880	12687	24252	20260	2,222
MAY90	86200	12995	23506	22523	2,663
JUN90	84276	13039	24736	23033	2,557
JUL90	79535	13035	22049	14133	1,726
AUG90	91515	16683	24493	20110	2,468
SEP90	82813	18752	25487	19682	2,744
OCT90	93078	19604	24260	22197	2,393
NOV90	86461	18201	23929	17212	3,179
DEC90	75487	16080	23031	11784	2,346
JAN91	85810	14935	23701	15467	1,548
FEB91	82592	13261	24205	17002	1,826
MAR91	85012	12838	24200	15952	2,944
APR91	79324	13509	24971	18767	1,817
MAY91	79917	14352	24560	20605	2,906
JUN91	76896	14136	24992	19809	3,130
JUL91	79720	13672	22566	14233	1,792
AUG91	88818	14394	24037	19311	2,456
SEP91	81504	14406	25047	20827	2,954
OCT91	90230	14587	24115	23388	2,390
NOV91	81644	14271	23034	20181	3,454
DEC91	79244	12981	22590	14344	2,825

Figure 2. Table in PNG image format

```
PROC REPORT data=&dt. NOWD NOFS SPLIT='\' HEADLINE HEADSKIP SPANROWS PS=50 LS=134
style(report) =[background=white bordercolor=black  just=left];
.
.
.
RUN;
ODS PRINTER CLOSE;
ODS LISTING;
```

In this procedure, for demonstration purpose, we tabulate the sales into 2 parts: one is resource sales such as coal (Bituminous Coal Production) and Petrol (Petroleum and Coal Products) from May, 1989 to Dec., 1991. The other one is product sales such as vehicle (Motor Vehicles and Parts), tobacco (Tobacco Products), and chemical (Chemicals and Allied Products) (See Figure 2).

By defining papersize, ODS printer DPI, and the padding in proc template '*STYLE TABLES FROM OUTPUT*' statement, we can control the PNG image size of the table. Therefore, we can finally control the size of the PPT report. That's another major concern for SAS automation process.

OUTPUT GRAPHS INTO PNG IMAGE

The code fro outputting figures in image (PNG) format is as follows. Similarly, we define the paper size as needed on PPT slides. Again, we take PRINTERPATH = PNG.

```
TITLE;
FOOTNOTE;
OPTIONS NODATE NONUMBER NOCENTER  ORIENTATION=landscape SPOOL topmargin = 0.02 cm
bottommargin = 0.02cm leftmargin = 0.02 cm rightmargin = 0.02 cm;
ODS NOPROCTITLE;
ODS ESCAPECHAR="^";
OPTIONS  NODATE PAPERSIZE=('xx cm','xx cm') PRINTERPATH=png;
ODS _ALL_ CLOSE;
ODS PRINTER file="&path.&fig1." DPI=150 STYLE=Jeffhao;
ODS GRAPHICS on /noborder scale=off width=yy cm height=zz cm;
PROC SGPLOT data=&dt.;
.
.
RUN;
ODS GRAPHICS OFF;
ODS PRINTER CLOSE;
ODS LISTING;
```

Figure 3 shows the eleven years sales trends of vehicle, petrol, and chemical from Jan 1981 to Dec 1991. The code for Figure 4 is similar to the one for Figure 3, except the following change:

```
ODS PRINTER file="&path.&fig2." DPI=150 STYLE=Jeffhao;
```

In Figure4, we compare durable goods sales (Durable Goods Industries, Total) versus non-Durable goods (Non-durable Goods Industries, Total).

In PROC SGPLOT procedures, we did not use the syntax that manually define the maximum, minimum, and interval of the Y axis values for figure 3 and Figure 4 as below:

```
YAxis display=(nolabel) grid  values=(6000 to 28000 by 2000) offsetmin=0 offsetmax=0;

YAxis display=(nolabel) grid  values=(60000 to 150000 by 10000) offsetmin=0
offsetmax=0;
```

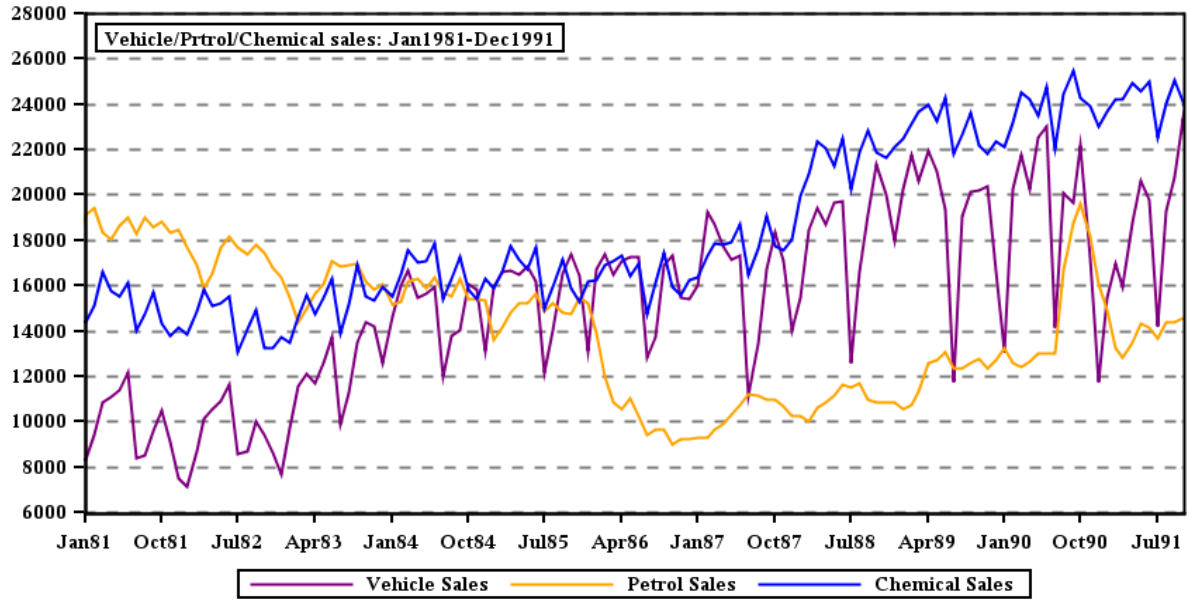


Figure 3. Graph in PNG image format

Instead, we use a macro to determine the maximum, minimum, and interval values automatically, then adopt and modify the methodology recommended by Li (2003) to determine the optimized interval values, it works very well. Since it's beyond the scope e of the paper, we will not discuss the issue in detail here.

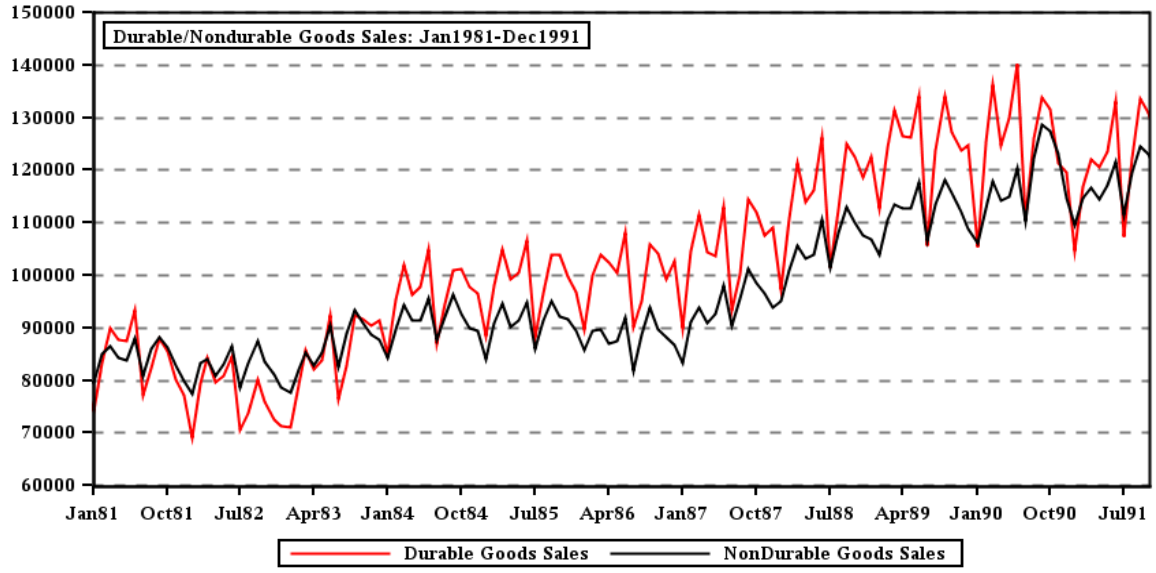


Figure 4. Graph in PNG image format

EXPORTING MULTI-SLIDE POWERPOINT REPORT AUTOMATICALLY

The last step for SAS automation process is to use Microsoft VBScript (Visual Basic Script) to output the table and figures images into the pre-designed professional-looking PPT template and output the PPT report into designated path.

VBScript is similar to VBA (Visual Basic or Visual Basic for Applications). It talks to host applications using Windows Script. By combining with SAS, we can realize the automation of reporting process smoothly (Microsoft Developer Network, MSDN).

The follows are the SAS code for generating VBScript used for outputting the table and graph images into PPT template:

```
%let ppt=C:\Temp\final_samplePPT.pptx;
libname temp "C:\temp";
%let workpath=%sysfunc(pathname(temp));

filename vbs "&workpath\xls2PPT.vbs";
data _null_;
file vbs;
length saveas $1024;

put 'Set objPPT = CreateObject("PowerPoint.Application")';
put 'objPPT.Visible = True';
put 'Set objPresentation = objPPT.Presentations.Add()';
put 'objPresentation.ApplyTemplate("C:\Temp\MWSUG_PPT_Template.potx")';

put 'Set objSlide = objPresentation.Slides.Add(1,12)';
put 'Set pic = objSlide.shapes.AddPicture("C:\temp\fig1.png",0,-1,374,119,335,198)';
put 'Set pic = objSlide.shapes.AddPicture("C:\temp\fig2.png",0,-1,374,330,335,198)';
put 'Set pic = objSlide.shapes.AddPicture("C:\temp\tables.png",0,-1,20,119,340,411)';
put 'Set pic = objSlide.shapes.AddPicture("C:\temp\MWSUG_Conf.png",0,-1,130,36,468,77)';

saveas='objPresentation.SaveAs("'||"&ppt."||'")';
put saveas;
put 'objPresentation.Close';
put 'objPPT.Quit';
run;
```

In this slide, we add one table, two graphs, and one Midwest SAS User Group Conference logo with pre-determined position. By controlling the padding distance, the DPI resolution, and the papersize option, we achieve the PPT slide report with desired quality and file size.

If we want to output 15 slides, we can just simply add the ‘PUT’ statement as follows:

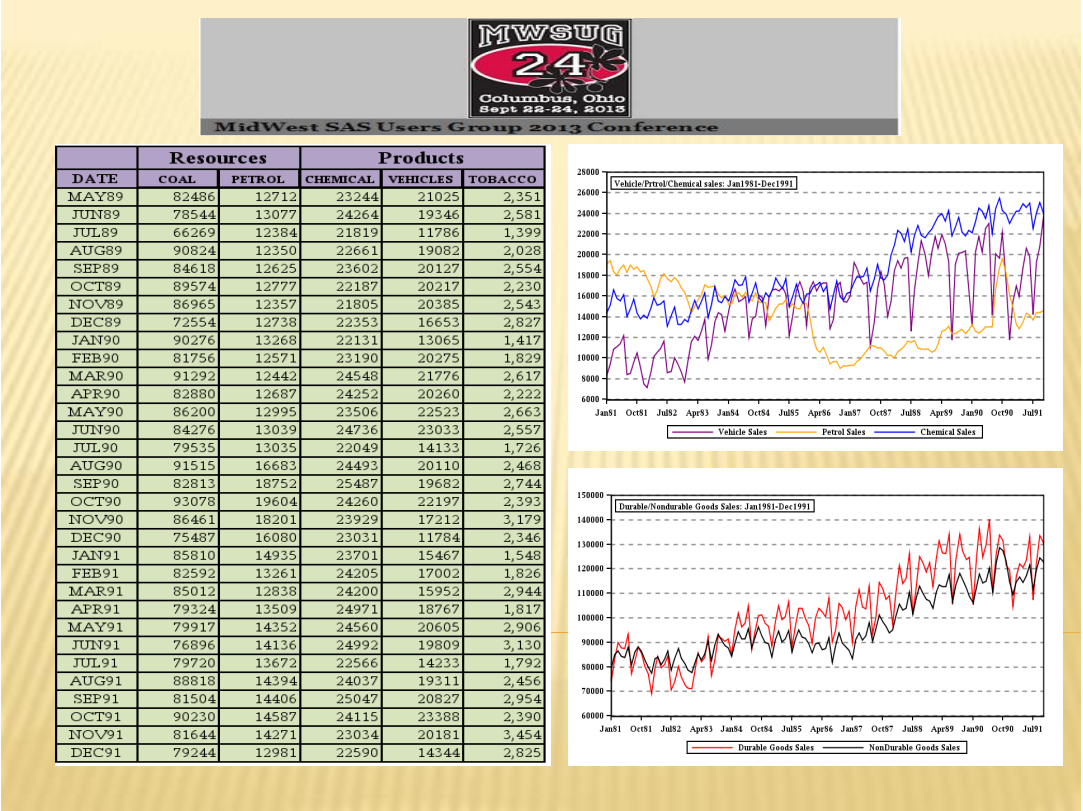
```
PUT 'Set objSlide = objPresentation.Slides.Add(1,12)';
PUT 'Set objSlide = objPresentation.Slides.Add(2,12)';
.
.
.
PUT 'Set objSlide = objPresentation.Slides.Add(15,12)';
```

After the VBScript file is created, we use filename pipe to output the PPT report into the designated path. The code is as follows:

```
FILENAME ppt pipe "cscript //nologo ""&workpath\xls2PPT.vbs""";
data _null_;
INFILE ppt;
INPUT;
PUT _infile_;
run;

FILENAME ppt clear;
%sysexec(erase /q "&path.*.png");
```

The last two statements are to delete the temporary file and PNG images in ‘C:\temp’ folder.



	Resources		Products		
DATE	COAL	PETROL	CHEMICAL	VEHICLES	TOBACCO
MAY89	82486	12712	23244	21025	2,351
JUN89	78544	13077	24264	19346	2,581
JUL89	66269	12384	21819	11786	1,399
AUG89	90824	12350	22661	19082	2,028
SEP89	84618	12625	23602	20127	2,554
OCT89	89574	12777	22187	20217	2,230
NOV89	86965	12357	21805	20385	2,543
DEC89	72554	12738	22353	16653	2,827
JAN90	90276	13268	22131	13065	1,417
FEB90	81756	12571	23190	20275	1,829
MAR90	91292	12442	24548	21776	2,617
APR90	82880	12687	24252	20260	2,222
MAY90	86200	12995	23506	22523	2,663
JUN90	84276	13039	24736	23033	2,557
JUL90	79535	13035	22049	14133	1,726
AUG90	91515	16683	24493	20110	2,468
SEP90	82813	18752	25487	19682	2,744
OCT90	93078	19604	24260	22197	2,393
NOV90	86461	18201	23929	17212	3,179
DEC90	75487	16080	23031	11784	2,346
JAN91	85810	14935	23701	15467	1,548
FEB91	82592	13261	24205	17002	1,826
MAR91	85012	12838	24200	15952	2,944
APR91	79324	13509	24971	18767	1,817
MAY91	79917	14352	24560	20605	2,906
JUN91	76896	14136	24992	19809	3,130
JUL91	79720	13672	22566	14233	1,792
AUG91	88818	14394	24037	19311	2,456
SEP91	81504	14406	25047	20827	2,954
OCT91	90230	14587	24115	23388	2,390
NOV91	81644	14271	23034	20181	3,454
DEC91	79244	12981	22590	14344	2,825

Figure 5. The Professional-Looking PowerPoint Report

There are eight arguments in AddPictures() statement as follows:

AddPicture (FileName, LinkToFile, SaveWithDocument, Left, Top, Width, Height, Anchor)

The 1st argument is the file name of the figure. The 2nd and 3rd arguments command PowerPoint t to make a physical copy of the source figure or a link. The pair value for physical copy with link is (-1,-1), without link is (0, -1). If you use the former, the new images always overwrite the old ones whenever you update the images with same file names.

The 4th and 5th arguments are the X and Y coordinates of the starting position of the figure from left border and top border, calculated in points (1 inch =72 points or 1cm=28.35 points).

The 6th and 7th arguments are to set the size of figure (width and height). In this example, we control the papersize option, the padding value, and DPI resolution value to best the Powerpoint slide resolution without increasing the file size.

The 8th argument is 'anchor', the range to which the picture is bound. If Anchor is specified, the anchor is positioned at the beginning of the first paragraph in the anchoring range. If this argument is omitted, however, the anchor is placed automatically and the picture is positioned relative to the top and left edges of the page. If interested, readers can refer to Microsoft Developer Network (MSDN) for further information.

CONCLUSION

By combining SAS with DDE, Proc Template, GTL, ODS Printer, Proc Report, Proc SGPlot, and VBScript, the whole automation process become time-saving, less error-prone, and more efficient.

This paper presents sample code for each step to explain the whole automation process from password protected Excel raw data to profession-looking PowerPoint report.

REFERENCES

- Huang, Ya. *Automate PowerPoint Slide Generation with ODS and VBScript.* SAS Global Forum 2010. Paper 228-2010
- Li, Don. *Tired of Defining Axis Scale for SAS Graphs? A Solution with an Automatic Optimizing Approach*. SAS PharmaSUG , 2003
- Microsoft Developer Network (MSDN). *VBScript Language Reference.* http://msdn.microsoft.com/en-us/library/d1wf56tt(v=vs.84).aspx Retrieved on August 15th, 2013.
- Smith, Curtis A. *Importing Excel Files Into SAS Using DDE.* WUSS Proceedings 2010
- SAS Document, *Sample 31328: Creating password-protected Excel files with SAS® software*. 2008. http://support.sas.com/kb/31/328.html Retrieved on August 10th, 2013.
- SAS Institute. *SAS® 9.2 Language Reference Concepts (2nd Edition).* 2010. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The author would like to thank Albert Tang, Sam Zheng, Shirley Xu, Chong E. Teoh, Charles Stone, and Yibo Zhang for their input of ideas and contributions to this paper.

I appreciate all the technical support and advice from the posts on SAS-L and SAS Community.

RECOMMENDED READING

- Carpenter, Art. *Carpenter's Complete Guide to the SAS REPORT Procedure.* SAS Institute, May 2007
- Cohen, John and Shields, James. *SAS DDE: How We Make Our Customers Happy and Still Use SAS®.* NESUG Proceedings 2004
- Johnson, Misty. *Macro Variables Make Life Easier: Applications with SAS® to Excel*. MWSUG Proceedings 2010. Paper 40-2010
- Microsoft Support. *Macrofun.exe File Available on Online Services*. http://support.microsoft.com/kb/128185
- Vyverman, Koen*. Using Dynamic Data Exchange to Export your SAS® Data to MS Excel-Against All ODS, Part I* Proceedings of SUGI **26,** 20**01.**
- Vyverman, Koen. *Fancy MS Word Reports Made Easy: Harnessing the Power of Dynamic Data Exchange-Against All ODS, Part II*. Proceedings of SUGI 28**,** 2003**.**
- Vyverman, Koen. *A Matter of Presentation: Generating PowerPoint Slides from Base® SAS using Dynamic Data Exchange.* Proceedings of SUGI 30**,** 2005**.**

CONTACT INFORMATION

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

Name:	Jeff Hao
Enterprise:	JPMorgan Chase & Co.
Address:	1111 Polaris Parkway
City, State ZIP:	Columbus, OH 43240
Work Phone:	614-213-0378
Fax:	614-217-5781
E-mail:	Jianjun.hao@jpmchase.com

COPYRIGHT STATEMENT

DISCLAIMER

The views and opinions expressed herein are solely those of the author and do not necessarily represent those of JPMorgan Chase & Co. JPMorgan Chase & Co. does not endorse, recommend, or promote any of the computing architectures, platforms, software, programming techniques or styles referenced in this paper.