Paper 042-2013

Can You Create Another PowerPoint for Me? How to Use Base SAS® and DDE to Automate Snappy PowerPoint Presentations

Scott Koval, Pinnacle Solutions Inc.; Mitchell Weiss, Maguire Associates Inc.

ABSTRACT

Your supervisor appreciates your wonderful and informative SAS ® reports. How many times have you heard, “Great! Now can you compile all the SAS reports into a PowerPoint presentation?” At that moment you wish you could press a button to automate the process because SAS programmers spend way too much time updating PowerPoint slides. This paper offers solutions to make your life easier by building upon techniques from Vyverman (2005 that discussed using SAS to take advantage of Dynamic Data Exchange (DDE) to write through MS Excel to MS PowerPoint. The goal is to free data analysts from PowerPoint tyranny by enabling efficient and repeatable PowerPoint presentations.

INTRODUCTION

BASE SAS provides analysts with many opportunities to improve efficiency. SAS Macros and custom web reporting allow for the optimization of processes whose purpose is to produce all manners of analytical output. In some cases, however, flexibility and familiarity with Microsoft Office products is required. While many SAS users may feel comfortable generating reports and graphics within SAS, the vast majority of business practitioners are dependent upon Microsoft Office. Luckily, through the Dynamic Data Exchange feature of Windows, analytical professionals are able to efficiently send output from SAS to Microsoft Office programs (for example, Word and Excel) for use by their supervisors.

There are many reference papers highlighting the use of DDE to write to Excel and Word, but there are very few that explain how to produce slides in PowerPoint. Vyverman (2005) goes into detail about using DDE to efficiently produce PowerPoint slides. Vyverman points out that PowerPoint is not DDE compliant, however an ingenious workaround (which uses the Microsoft suite of products ability to communicate between different programs) allows for PowerPoint slides to be created and updated through commands issued in Excel.

This paper expands upon Vyverman’s to update the necessary keystroke commands that correspond with recent versions of Microsoft PowerPoint. Additionally, we introduce some new elements to the process which include the ability to parameterize the code to allow for repeatability with semi-customization and the utilization of the Find/Replace feature to make project-wide changes.

BUSINESS PROBLEM

In most companies it is not too difficult to imagine a scenario where the supervisor of an analytical project wants not only the direct output from the analysis, but also a fully compiled PowerPoint presentation. For some business processes these presentations may share many similarities in terms of content and structure. However, the production of PowerPoint presentations can occupy tremendous

amounts of an analyst's time that could be reallocated into further analysis or even the addition of new projects. While some companies may have standardized PowerPoint presentation templates, there is still a significant amount time that is required to update the material on each slide.

Given some standard elements to reproduce, using SAS and DDE to export results to Microsoft products such as Word and Excel can greatly improve efficiency. However, the ability to push those results to PowerPoint may still require a portion of the work to be done manually, as DDE does not readily communicate with MS PowerPoint.

Figure 1. Title Slide Template

Figure 2. Graphic Content Slide Template

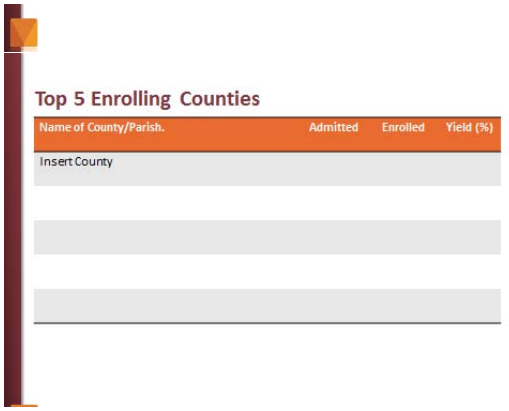


Figure 3. Table Content Slide Template

POWERPOINT PRESENTATION TEMPLATE

The sample presentation is an enrollment report for a fictitious university in California. (The example enrollment data can be found in the appendix). The admissions and enrollment teams at this University want to see where their students are coming from; however, some people prefer visual displays of data, while others want to see the information in a table.

The first step to enable SAS to reproduce PowerPoint presentations is to create the template for the presentation. For this example we will update three slides, which should cover many of the elements that may need to be reproduced.

- The first slide is the title slide. Here, we put placeholders for the university's name and the years that report covers. We will have SAS use Find/Replace within PowerPoint to update the information in the presentation.
- The second slide has a placeholder for the year and an image of a county choropleth that was created using `proc gmap`.
- The third and final slide has a placeholder text as well as a table, which shows the top counties where enrolling students reside.

ELEMENTS OF SAS PROGRAM

In order to complete the report, there are three main tasks that must be accomplished. First, the map must be created. This is achieved through merging the data with a standard map that is available through SAS and then running the **gmap** procedure. Second, macro variables for the top five enrollment counties are created using a SQL procedure. Third, we utilize DDE to send key commands to PowerPoint via Microsoft Excel.

```
/*     Part 1: Creating Map of California  */

proc template;     *Defines a color ramp from white to orange;
   define style styles.colorrampo;
   parent=styles.default;
   style twocolorramp / startcolor=cxffffff endcolor=orange;
   style graphdata1 from graphdata1 / color=cxffffff;
   style graphdata2 from graphdata2 / color=orange;
   end;
run;

proc sort data=maps.uscounty(where=(state=6)) out=ca_map;
      by county;
run;

data map_merge;
      set sim_enr ca_map;
      by county;
run;

filename mapout 'C:\temp';     *Stores Map in temporary location;
goptions device=png noborder cback=white;
ods html style=colorrampo path=mapout body='Map.htm';
proc gmap map=map_merge;
      id county;
      choro sum_enroll / levels=8 statistic=sum name='Map';
run;
ods html close;
```

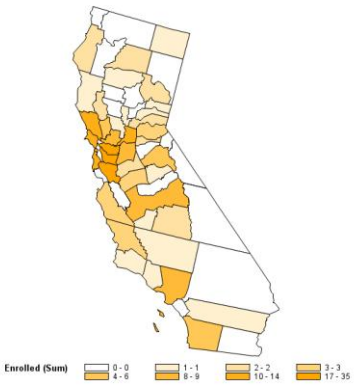


Figure 4. Output Enrollment Map

The initial **proc template** in the above syntax is included in order to create a two color ramp going from white to orange for the choropleth map (Figure 1). The sort procedure takes the data used to draw the map and prepares it to be merged with the enrollment data set. A filename statement is used to point to a temporary location to store the output map. The goptions statement is also used to create a PNG. Finally, a choropleth map displaying enrollment information for each county in California was created using **proc gmap.**

Now that the map is finished, the table can be prepared.

```
/*    Part 2: Prep data for table   */

proc sql outobs=5;
      select countynm, sum_admits, sum_enroll,
            sum_enroll/sum_admits*100 as yield format=8.2
      into :cnty1 - :cnty5, :admit1 - :admit5, :enroll1 - :enroll5,
                  :yield1 - :yield5
      from sim_enr
      order by sum_enroll desc;
quit;
```

For this report we are interested in displaying information on the top five California counties from which students enroll. This piece of code is used to create a series of macro variables to be written into the table on the third slide. In this instance, four column name prefixes (cnty, admit, enroll, and yield) are used along with an increasing numeric suffix. By using this method, the macro variables can be passed iteratively through DDE.

```
/*    Part 3: The DDE   */
options noxsync noxwait xmin;

filename sas2xl dde 'excel|system';

data _null_;
      length fid rc start stop time 8;
      fid = fopen('sas2xl','s');
      if (fid <= 0) then do;
            rc = system('start excel');
            start=datetime();
            stop=start+10;
            do while (fid <= 0);
                  fid=fopen('sas2xl','s');
                  time=datetime();
                  if (time >= stop) then fid = 1;
                  end;
            end;
      rc=fclose(fid);
run;

data _null_;
      rc=system('start powerpnt');
      rc=sleep(3);
run;

data _null_;
      file sas2xl;
      put '[error(false)]';
      put '[workbook.insert(3)]';
run;
```

```
filename xlmacro dde "excel|macro1!r1c1:r10c1" notab lrecl=200;
data _null_;
      file xlmacro;
      put '=wait(now()+"00:00:03")';
      put '=halt(true)';
run;
filename xlmacro clear;
```

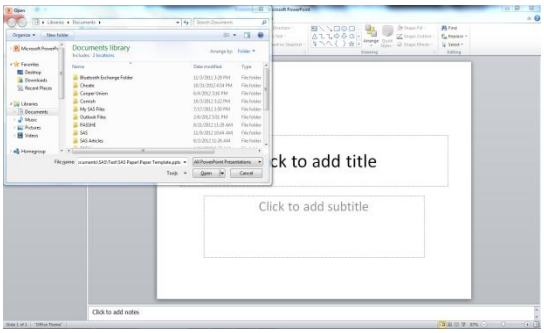
The above syntax appeared in Vyverman (2005). It starts out by first setting the system options that allow SAS to continue processing while the other applications are running. In order for the subsequent commands to flow into PowerPoint we utilize SAS and DDE to open Excel and in turn open PowerPoint. Following this are system function calls that will start Excel and PowerPoint. Finally, DDE is used to create a macro within Excel that can be invoked to create pauses between key commands. This pause is necessary as it allows for the commands issued to Excel to have their intended effect on the PowerPoint template before the next series of commands are issued through the SAS macro.

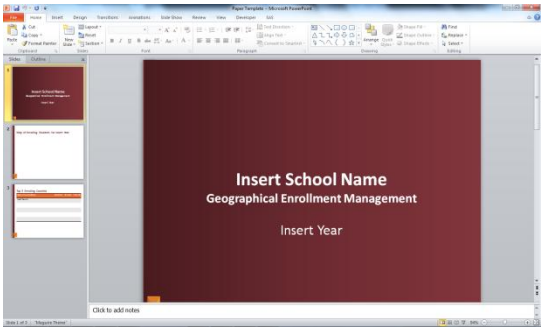
The following code is a simple macro that was built to automate the PowerPoint report.

```
%macro pptx_dde(sch_name=, yr=);
data _null_;
      file sas2xl;
      /*    Part 1: Open Report Template File   */
      put '[app.activate("microsoft excel - Book1")]';
      put '[app.activate("presentation1 - microsoft powerpoint")]';
      put '[run("macro1!r1c1")]';
      put '[send.keys("%",true)]';
      put '[send.keys("f",true)]';
      put '[send.keys("o",true)]';
      put '[send.keys("C:\Users\skoval\Documents\SAS\Test\SAS Paper\Paper
Template.pptx",true)]';
      put '[run("macro1!r1c1")]';
      put   '[send.keys("{return}",true)]';
      /*    Part 2a: Replacing the School Name  */
      put '[run("macro1!r1c1")]';
      put '[send.keys("^h",true)]';
      put '[send.keys("Insert School Name",true)]';
      put '[send.keys("{tab}",true)]';
      put '[send.keys("'"&sch_name"'",true)]';
      put '[run("macro1!r1c1")]';
      put '[send.keys("+{tab}",true)]';
      put '[send.keys("+{tab}",true)]';
      put '[send.keys("a",true)]';
      put '[run("macro1!r1c1")]';
      put '[send.keys("{enter}",true)]';
      put '[send.keys("{esc}",true)]';
      /*    Part 2b: Replacing the Year   */
      put '[run("macro1!r1c1")]';
      put '[send.keys("^h",true)]';
      put '[send.keys("Insert Year",true)]';
      put '[send.keys("{tab}",true)]';
      put '[send.keys("'"&yr"'",true)]';
      put '[run("macro1!r1c1")]';
      put '[send.keys("+{tab}",true)]';
      put '[send.keys("+{tab}",true)]';
```

```
        put '[send.keys("a",true)]';
        put '[run("macro1!r1c1")]';
        put '[send.keys("{enter}",true)]';
        put '[send.keys("{esc}",true)]';
        /*    Part 3a: Inserting Map  */
        put '[send.keys("{pgdn}",true)]';
        put '[run("macro1!r1c1")]';
        put '[send.keys("%",true)]';
        put '[send.keys("n",true)]';
        put '[send.keys("p",true)]';
        put '[run("macro1!r1c1")]';
        put '[send.keys("C:\temp\Map.png",true)]';
        put '[send.keys("{return}",true)]';
        put '[run("macro1!r1c1")]';
        /*    Part 3b: Resizing the Image   */
        put '[send.keys("%",true)]';
        put '[send.keys("jp",true)]';
        put '[send.keys("sz",true)]';
        put '[run("macro1!r1c1")]';
        put '[send.keys("{tab}",true)]';
        put '[send.keys("{tab}",true)]';
        put '[send.keys("5.83",true)]';
        put '[send.keys("+{tab}",true)]';
        put '[run("macro1!r1c1")]';
        /*    Part 3c: Repositioning the Image    */
        put '[send.keys("{down}",true)]';
        put '[send.keys("{tab}",true)]';
        put '[send.keys(".83",true)]';
        put '[send.keys("{tab}",true)]';
        put '[send.keys("{tab}",true)]';
        put '[send.keys("1.17",true)]';
        put '[send.keys("{return}",true)]';
        put '[send.keys("{esc}",true)]';
        put '[run("macro1!r1c1")]';
        /*    Part 4: Inserting Data into Table   */
        put '[send.keys("^f",true)]';
        put '[send.keys("Insert County",true)]';
        put '[run("macro1!r1c1")]';
        put '[send.keys("{return}",true)]';
        put '[send.keys("{esc}",true)]';
        %do i = 1 %to 5 %by 1;
              put '[send.keys("'"&&cnty&i"'",true)]';
              put '[send.keys("{tab}",true)]';
              put '[send.keys("'"&&admit&i"'",true)]';
              put '[send.keys("{tab}",true)]';
              put '[send.keys("'"&&enroll&i"'",true)]';
              put '[send.keys("{tab}",true)]';
              put '[send.keys("'"&&yield&i"'",true)]';
              %if &i < 5 %then %do;
                    put '[send.keys("{tab}",true)]';
              %end;
        %end;
        put '[run("macro1!r1c1")]';
        put '[send.keys("{esc}",true)]';
        put '[send.keys("{esc}",true)]';
run;
%mend;
```

Figure 5. Opening the Template





PART 1: OPENING THE TEMPLATE

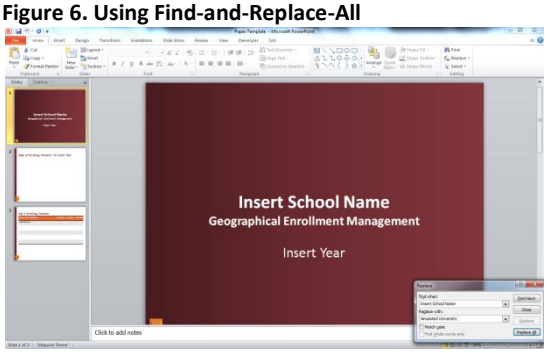
There are four main parts of this syntax. The code listed underneath Part 1 sends a series of keystrokes to open the PowerPoint file template file. The keystrokes used in this program correspond to Microsoft PowerPoint 2010 and are used to interact with the ribbon structure. By inputting the percent sign (`%`) followed by an `f` and an `o`, it is navigating to the file tab attempting to open a file. Following this sequence and the brief pause, the template's location is input into the **File Open** window. Again, a brief pause was used to insure that the file has time to load before the next part of the code can run.

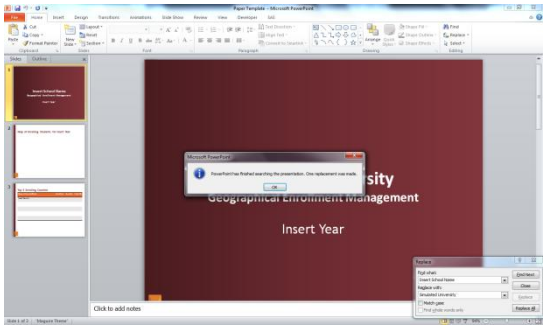
PART 2: FIND-AND-REPLACE-ALL

Beginning with the second part of the program, the find-and-replace-all feature is used to change the name of the school and year. These two values are parameters at the beginning of the macro. By using this method, it is relatively simple to change placeholders with values that are needed for the report. In this program the placeholders for School Name and Year are replaced by Simulation University and 2013.

While this example contains only a few values that need to be replaced, this can have a much greater impact on larger, more comprehensive reports. This is also a useful technique for dealing with titles and footnotes.

Figure 6. Using Find-and-Replace-All





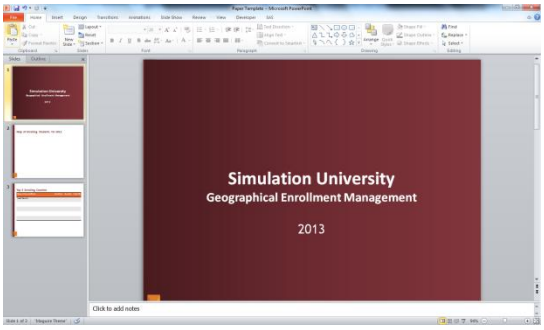
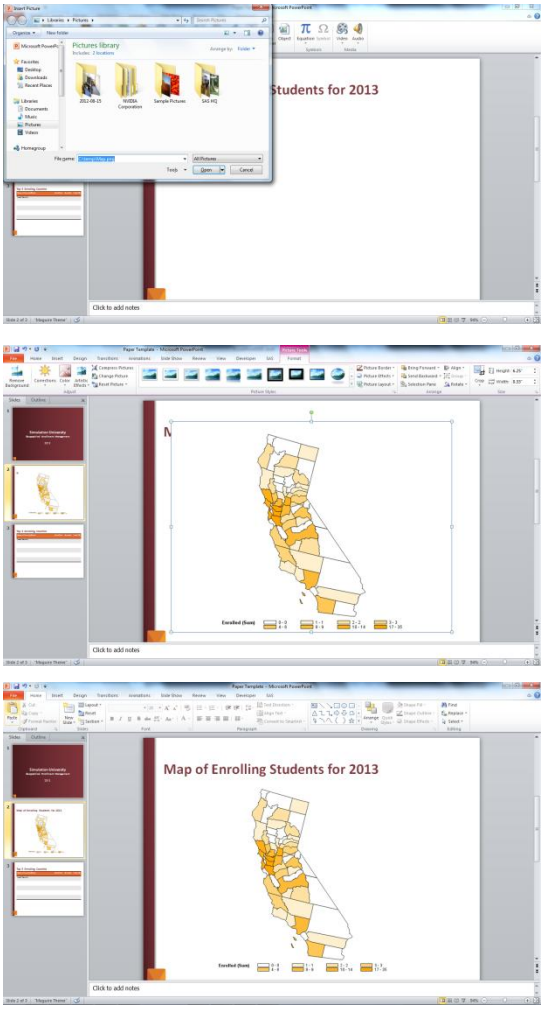


Figure 7. Resizing and Positioning the Map



PART 3: RESIZING AND POSITIONING A MAP

The third part of the macro is designed to insert the enrollment map generated earlier in SAS. This starts out very similar to the first part of the macro where the ribbon is navigated—the only difference is that this time the sequence of keystrokes is instead accessing the insert tab and placing a picture.

The pause macro was once again used to slow down the keystroke in order to allow the program to correctly process. When the enrollment map is placed on the slide, it needed to be both resized and repositioned. Parts 3b and 3c of the macro program show the series of keystrokes needed to accomplish this task.

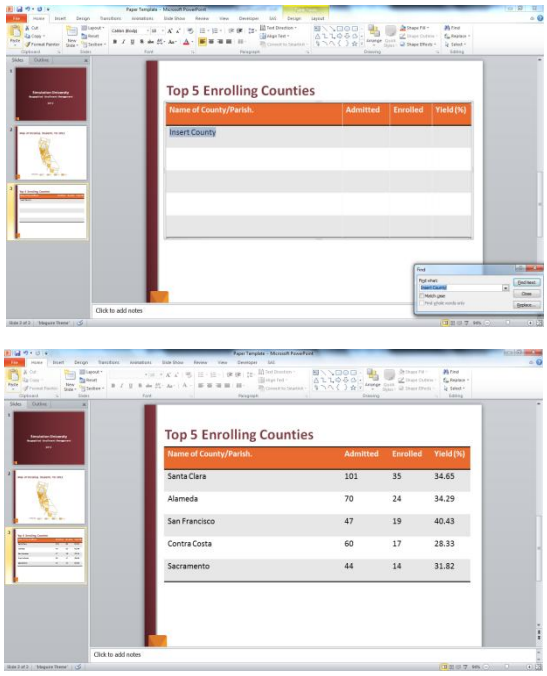
Once again, it is important to note that the pause macro had to be invoked in order to artificially slow down the program and allow time for the keystrokes to be passed.

PART 4: INSERTING DATA INTO A TABLE

In order to reposition the cursor into the table, the `find` feature was used to select another placeholder value. Now that the cursor is in this position, the program uses a loop to cycle through the series of macro variables that were created earlier.

After inserting these values into the table, the report has been completed. While this program does not illustrate it here, the finished product can easily be saved to a specific location by using code to navigate to the file tab.

Figure 8. Inserting Data into the Table



CONCLUSION

Overall, SAS offers a useful method to easily automate the production of PowerPoint files. While this paper covers some of the more common needs for report generation, there are further applications of SAS programming that can be used. One example would be to program conditional logic that will change text within slides. It is also possible to set up a batch job that will automatically pull data and generate PowerPoint reports at given times.

REFERENCES

Roper, Christopher. 2000. "Intelligently Launching Microsoft Excel from SAS, using SCL functions ported to Base SAS." *Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference*. Available at http://www2.sas.com/proceedings/sugi25/25/cc/25p097.pdf.

Vyverman, Koen. 2005. "A Matter of Presentation: Generating PowerPoint Slides from Base® SAS using Dynamic Data Exchange". *Proceedings of the Thirtieth Annual SAS Users Group International Conference*. Available at http://www2.sas.com/proceedings/sugi30/045-30.pdf.

CONTACT INFORMATION

Scott Koval
Pinnacle Solutions, Inc.
426 East New York Street
Indianapolis, IN 46260
E-mail: scott.koval@psiconsultants.com

Mitchell Weiss
Maguire Associates, Inc.
555 Virginia Road, Suite 201
Concord, MA 01742-2727
E-mail: mweiss@maguireassoc.com

PowerPoint slides and data used in this paper are available upon request.