Proc Report, the Graph Template Language, and ODS Layouts: Used in Unison to Create Dynamic, Customer-Ready PowerPoints
Amber Carlson, Amelia Stein, and Xiaobin Zhou, Curriculum Associates LLC.

ABSTRACT
Twice a year, we create PowerPoint decks and supplemental tables for over 100 customers to present data on their system performance to help inform their decision-making. We use one SAS® program to create PowerPoint slides that incorporate the corporate template and include dynamic-editable tables, charts, titles, footnotes, and embedded hyperlinks that open additional drill-down data tables in either PDF or Excel format. These additional data tables are saved in automatically created categorized folders.
In this SAS program we first employ SAS styles, ODS Layout, and ODS PowerPoint to format the slides and automate creation. Macros and X Command are also utilized to create categorized folders for organization. Finally, the Graph Template Language, Proc Report, and ODS PDF are utilized to create the customer-specific charts and tables for the main deck and the supplemental tables that are linked by hyperlinks on the corresponding slides of the PowerPoint. This program starts from the raw data source and the output is a complete customer deck that is ready for presentation.
In this paper we share examples of how to create a completely customized PowerPoint deck using SAS styles and ODS Layout. We also share tips and tricks that we have learned regarding what works and what does not in the ODS PowerPoint destination. In addition, we demonstrate the program flow to highlight each type of functionality required to create a multi-level custom report.

INTRODUCTION
PowerPoint is an effective tool for broad communication with data visualizations. However, creating 100 customized data-driven PowerPoints with a standardized look-and-feel is no easy feat. In the past, we used SAS® to aggregate the data and export the aggregations into an excel template, which was then populated into PowerPoint using a third-party Visual Basic Application. This process was error prone and required manual review of every presentation.
By adding an ODS PowerPoint procedure to our existing SAS program, we are now able to aggregate the data, create charts and tables, and output formatted slides in one program, without sacrificing the professional design. This paper describes the methods we use to fully automate the creation of PowerPoints with high quality data visualizations without leaving the SAS environment. First, we walk through the basic building blocks of creating a PowerPoint in SAS. This includes data organization and the development of stylized charts and tables using the Graph Template Language and PROC REPORT. Next, we discuss how to use ODS LAYOUT, as well as ODS Language Statements to create custom slide layouts with titles, footnotes, and visualizations. Lastly, we touch upon how to add customizable features using Macros, to create dynamic presentations.

CREATING A POWERPOINT PRESENTATION
Figure 1 shows the two slides that were created using ODS PowerPoint and the Sashelp.Bweight dataset.¹

Figure 1. ODS PowerPoint Output

¹ We are using SAS 9.4 M3. An earlier version may not have all of the functions described.
This paper will walk through the code used to create these slides. Not only will we discuss the steps we took, but we will also highlight options available for customization. For the complete code, please see the Appendix.

**BASIC BUILDING BLOCKS**

Before we begin formatting and organizing the layout of our PowerPoint, the first step we take is to create a dataset that contains the aggregated data and labels necessary to create the desired charts. Although SAS is capable of aggregating data within PROC REPORT and PROC TEMPLATE, we find it more convenient for quality assurance and formatting purposes to start with a summary dataset. Figure 2 provides a snapshot of the summary data set we used to create the slide in Figure 1.A and Figure 3 shows the summary data set we used to create the slide in Figure 1.B. Please see the code in the appendix on lines 8-59 for organizing the data.

![Figure 2. Summary Data Set for Figure 1.A](image)

![Figure 3. Summary Data Set for Figure 1.B](image)

After creating the summary dataset, the next step is to create the charts using the Graph Template Language (GTL). By using a PROC TEMPLATE statement in conjunction with a PROC SGRENDER statement, GTL allows you to create fully customizable charts and graphs.

**Basic clustered bar chart**

Basic syntax for the clustered bar chart in Figure 1.A is below, it creates the clustered bar chart in Figure 4.A.1

```sas
proc template;
  define statgraph double_bar;
  begin graph;
    layout overlay;
      barchart x=Education y=Avg_Weight / group=Marital_Status groupdisplay=cluster;
      endlayout;
    endgraph;
  end;
run;
proc sgrender data=Education template=double_bar; run;
```

2 Depending on the environment that you are working in, you may encounter the following error, “Template ‘double_bar’ was unable to write to template store!” To resolve this error, insert the following line of code before the PROC TEMPLATE statement:

```sas
ods path(prepend) work.template(update);
```
Proc Report, the Graph Template Language, and ODS Layouts: Used in Unison to Create Dynamic, Customer-Ready PowerPoints, continued

Figure 4. Clustered Bar Charts

Customized clustered bar chart

To improve the visual appeal of the chart, we add style options, such as custom colors and data labels, to create the clustered bar chart in Figure 4.A.2.

```
ods graphics / border=off;
proc template;
define statgraph double_bar;
begingraph / pad=0
designwidth=9in
designheight=4in
datacolors=(CXB5C8EB CX004281)
datacontrastcolors=(CXB5C8EB CX004281)
opaque=false;
layout overlay /
walldisplay=none
xaxisopts=(display=(line tickvalues) tickvalueattrs=(color=(CX595959)))
yaxisopts=(display=none linearopts=(viewmin=3100) offsetmin=0 offsetmax=.2);
barchart x=Education y=Avg_Weight / barlabel=true barlabelformat=comma9.
barlabelattrs=(size=10pt) name='Education'
barwidth=.8 clusterwidth=.6
group=marital_status groupdisplay=cluster;
discretelegend 'Education' /
border=false
location=outside
valign=bottom
displayclipped=true
valueattrs=(size=10pt)
opaque=false;
endlayout; endgraph; end; run;
```

This statement removes the border around the chart.

These style options set the width and height of the chart, as well as the colors of the bars. By adding the option “opaque=false”, we create a translucent background.

 Adds style elements to chart axes.

```
proc sgrender data=Education template=double_bar; run;
```

The reference line for average height and the textbox label in Figure 1.A were added using SERIESPLOT, DRAWRECTANGLE, and DRAWTEXT statements. Detailed code can be found in Appendix lines 75-96. The code for adding the table at the bottom of Figure 1.A can also be found in the Appendix on lines 111-122.

Adding features to charts

As seen in Figure 1.B, simple bar charts, with some added features, can direct attention towards a comparison quickly and effectively. To begin creating the chart in Figure 1.B, we start with the basic syntax below, which creates the bar chart in Figure 5.B.1:

```
proc template;
define statgraph bar_w_arrow;
begingraph;
layout overlay;
```
Proc Report, the Graph Template Language, and ODS Layouts: Used in Unison to Create Dynamic, Customer-Ready PowerPoints, continued

```sas
   barchart x=Smoker_Status y=Avg_Weight;
endgraph; end; run;
```

```sas
   proc sgrender data=Smoking template= bar_w_arrow; run;
```

To add the n-counts below the chart, we first create Count_Label and Label_Y in our summary dataset. Count_Label houses the text that will be displayed for the n-counts and Label_Y contains the y-coordinate for the location of the labels. See the appendix lines 37 and 38 for the addition of these variables. We then used a SERIESPLOT to add the label by using the Count_Label variable as the markercharacter and the Label_Y variable as the y-coordinate.

In order to place the labels at the desired location on the chart, LAYOUT LATTICE and LAYOUT OVERLAY statements are used. The LAYOUT LATTICE statement allows for the combination of two plots along one x-axis. Below is the code that adds n-counts, bar labels, and axis style options to the existing comparative bar chart.

```sas
   ods graphics / border=off;
   proc template;
   define statgraph bar_w_arrow;
   begingraph / pad=0 designwidth=6.5in designheight=4in opaque=false;
   layout lattice / rowdatarange=data columns=1
columndatarange=union rowweights=(.9 .1);
   layout overlay /
   walldisplay=none
   xaxisopts=(display=(line tickvalues) tickvalueattrs=(color=CX595959 size=10pt) gridattrs=(thickness=10) offsetmin=.3 offsetmax=.3) yaxisopts=(display=none linearopts=(viewmin=3100 offsetmin=0 offsetmax=.1);
   barchart x=Smoker_Status y=Avg_Weight /
   barlabel=true barlabelformat=comma9.
   barlabelattrs=(size=10pt) fillattrs=(color=CX004281) outlineattrs=(color= CX004281) barwidth=.5;
   endlayout;
   layout overlay /
   walldisplay=none
   xaxisopts=(display=none)
   yaxisopts=(display=none linearopts=(viewmin=0) offsetmin=0.1 offsetmax=0);
   scatterplot x=smoker_status y=label_y /
   markercharacter=count_label labelstrip=true
   markercharacterattrs=(size=10 color=black family="Calibri");
   endlayout;
   endlayout; endgraph; end; run;
```

After adding the LAYOUT LATTICE statement and additional LAYOUT OVERLAY statements, the image in Figure 5.B.2 is outputted. Adding style elements such as colors, bar labels, and axis options, to the existing code, creates the chart in Figure 5.B.3. To create the final output, we add a DRAWARROW statement to produce a downward arrow in the non-smoker bar to signify the decrease in average weight.

```sas
   Drawarrow x1="Non-Smoker" y1=3402 x2="Non-Smoker" y2=3161 / drawspace=datavalue
   arrowheadshape=barbed lineattrs=(color=CXB5C8EB thickness=6);
```

To see the complete code, please see the Appendix lines 137-161.
Figure 5. Creating an Enhanced Bar Chart

SLIDE LAYOUT AND OUTPUT

This next section will walk through the steps to design the layout and add text to PowerPoint slides, utilizing the charts and graphs that we have already created.

Using ODS LAYOUT GRIDDED, we can combine the code above for charts and table with ODS Language Statements to create an organized PowerPoint slide. The ODS LAYOUT GRIDDED statement allows you to structure each slide with multiple output and text. The skeletal layout for the slide in Figure 1.A is illustrated below in Figure 6.A and is created by the following code:

```sas
/*Title*/
proc odstext;
p "Title";
p "Text";
r
run;
ods layout gridded rows=2;
  ods region row=1 width=9in;
    /* Insert Code for Graph */
    ods region row=2 width=9.5in;
      /* Insert Code for Table*/
      /* Insert Code for Footnote*/
ods layout end;
```

To add titles and footnotes to our slides, we use PROC ODSTEXT, which allows for greater flexibility in formatting than TITLE or FOOTNOTE statements.

The layout for Figure 1.B is illustrated in Figure 6.B. It is created using multiple (and nested) ODS LAYOUT statements. The code used to produce the slide layout in Figure 6.B is below.

```sas
proc odstext;
```
Proc Report, the Graph Template Language, and ODS Layouts: Used in Unison to Create Dynamic, Customer-Ready PowerPoints, continued

Before finalizing the layout of the PowerPoint slide and the text that you want to add, it is important to keep in mind that PowerPoint slides are 10in by 7in with a .25in margin that cannot be changed. The layout design, as well as the size of the charts and tables must fit within the dimension of the slide. If all charts and tables within an ODS LAYOUT statement do not fit on one PowerPoint slide, then SAS will continue printing your layout on the next slide. SAS will not overlap objects (text, tables, pictures, graphs) on the slide. Furthermore, if a layout does not fill the entire PowerPoint slide, SAS will end the layout and continue the next layout on the same slide. To avoid this, add the following code after the ODS LAYOUT END statement to force a page break:

```sas
data _null_;
dcl odsout obj(); run;
```

Once the layouts of the PowerPoint slides are complete, the slides can be output to PowerPoint using the following basic syntax:

```sas
ods powerpoint options(backgroundimage="C:\Documents\SESUG_Background.png" File="C:\Documents\Customer Presentation.pptx"); /*Insert code for the first PowerPoint Slide*/ /*Insert code for the second PowerPoint Slide*/ ods powerpoint close;
```

As demonstrated above, one of the strengths of using ODS PowerPoint is having the ability to change the background image of any slide using options in the ODS PowerPoint statement. Here we have used the SESUG template background image, but a corporate background could just as easily be used.

In the PowerPoint created, elements such as text and tables are fully editable. Charts, on the other hand, are static images. To assure that PowerPoint slides are formatted properly we recommend the following five tips:

1. When creating charts, use DESIGNWIDTH and DESIGNHEIGHT statements in graph template to control the size of your chart.
2. When creating tables, use DEFINE statements in PROC REPORT to control the column widths so that the desired number of columns will fit on one slide. This is important when a table has too many columns or the

Figure 6. PowerPoint Slide Layouts

By adding blank text and adjusting the size of the font, we can add extra space between charts and text.

- Nested Layout (orange box Figure 6.B)
- Overall Slide Layout (blue box in Figure 6.B)

Add a background image or corporate template.
columns are too wide for the PowerPoint slide. By default SAS will break up the table onto multiple slides. Similarly, if your table has too many rows that won’t fit on one PowerPoint slide, by default, the table will continue onto the next slide.

In these situations, we recommend two solutions: 1) Export the table to Excel or PDF and add a hyperlink to the file on the PowerPoint slide where the data are referenced. Hyperlinks can be added using ODS Language Statements, as well as TITLE and FOOTNOTE statements. 2) Create multiple table segments with a certain number of rows or columns in each segment. Then output the segments seamlessly on separate PowerPoint slides. We prefer the former solution, whereby we add hyperlinked text to our footnote. Please see Appendix lines 185-195.

3. Use ODS language statements to format slides and add space between visualizations and text. By adding blank text and altering the font size, you are able to control the spacing of your visualizations and control where objects are output on the slide.

4. Code the width of your regions within the ODS LAYOUT statement. Coding the width of your regions gives you more control when outputting many objects onto one slide. It is important, however, to make sure that your objects are no larger than the size of the region.

USING MACRO VARIABLES TO ADD DYNAMIC FEATURES

In many instances, you may need different versions of PowerPoint decks for different customers. For example, one customer might need the full PowerPoint deck, while another may only require certain analyses. Rather than creating multiple versions of code, we found that the most flexible way is to code each slide inside a macro then only call the macros for the required slides when the deck is created.

In the full code in the appendix on lines 97-136 and 162-200, we have encased the layout of each PowerPoint slide in a macro using the following syntax:

```sas
%macro ed_level_marital_status(); /* Enter code for slide 1 */ %mend;
```

We are then able to evoke these macros only when needed. Macro %IF-%THEN-%ELSE statements can also be used to dynamically select the slides that are created. For example, consider the following code:

```sas
%let analysis=slide1;
%macro slide_output();
%if "&analysis"="slide1" %then %do;
%ed_level_marital_status();
%end;
%else %if "&analysis"="slide2" %then %do;
%non_smoker_vs_smoker();
%end;
%else %do;
%ed_level_marital_status();
%non_smoker_vs_smoker();
%end;
%mend;
```

This code allows the programmer to choose which slides to output. Changing the text in the %LET statement to “slide1”, “slide2”, or “both”, will affect which slide(s) are printed. By using dynamic selection, many nuances can be accounted for.

Strategically placed macro variables also enable you to make quick design changes. For example, on lines 61-68 in the appendix, we have created macro variables for all of the colors that we use throughout the presentation. This way, if a color needs to be changed, these lines can be modified and the color scheme throughout the PowerPoint will update as desired.

The one drawback to creating a single program for use with multiple customers is that it can be difficult to predict all of the nuances in the data. Therefore, creating a program that outputs dynamic PowerPoints takes time and in-depth knowledge of the data. That being said, having the ability to output customized/dynamic PowerPoint slides as needed can be a valuable, time-saving tool.

X COMMAND

One final enhancement to the overall process that we have found helpful is the use of X command. X command can be used to create directories and folders to store PowerPoint decks and additional data files. This is useful when you are outputting multiple PowerPoint decks for different audiences or when you are outputting additional data that is hyperlinked to your PowerPoint. The X command can be used to create a separate folder for each PowerPoint deck so that you can stay organized. An example of how you can create directories using X command is below.
%let customer=ABC Corporation;
%let main_dir=C:\Documents;
Options noxwait;
X "%str(md "&main_Dir\Reports\&Customer.%")";
%let report=&main_dir\Reports\&Customer;

Using X command to create a folder and setting a macro to use the newly created folder to save files in.

This code will create a folder for each specified customer within the predefined main directory. You could also create subfolders for different types of supplementary materials such as Excel, PDF, etc.

CONCLUSION

Creating PowerPoints with sophisticated data analytic results and at the same time meeting style requirements can be accomplished within a powerful SAS program. By adding the procedures outlined in this paper; Graph Template Language, ODS Language Statements, and ODS LAYOUT, as well as an ODS PowerPoint statement, users can conduct the analysis and create PowerPoint slides with professional design directly from their SAS environment. Coupling these procedures with standard SAS elements, such as macros and X command, the SAS program can be enhanced with remarkable flexibility.

Nonetheless, the initial creation of such a program could be very time consuming. In our case, however, using such a program to create 100+ customized presentation significantly reduces the amount of total time. More importantly, it also eliminates the error-prone steps of having to output data and manually create data visualizations. Overall, the benefits of creating a PowerPoint using SAS code are endless. You truly can create a code that “does it all.”

REFERENCES


RECOMMENDED READING

• SAS® 9.4 Graph Template Language User’s Guide. Fourth Edition

CONTACT INFORMATION

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

Amber Carlson
Curriculum Associates Inc.
153 Rangeway Rd
Billerica, MA 01862
ACarlson@cainc.com

Amelia Stein
Curriculum Associates Inc.
AmeliaStein@cainc.com

Xiaobin Zhou
Curriculum Associates Inc.
XZhou@cainc.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.
APPENDIX: SOURCE CODE
1 /* *************** MACROS FOR INDIVIDUAL REPORTS *************** */
2 %let customer=ABC Corporation;
3 %let main_dir=C:\Documents; /*Top level directory for io; *
4 /* *************** CREATE OUTPUT DIRECTORY *************** */
5 options noxwait;
6 x "str(md %"&main_dir\Reports\&customer.")";
7 %let report=&main_dir\Reports\&Customer; /*Directory for exported reports; *
8 /* *************** ORGANIZE DATASETS FOR REPORT SLIDES *************** */
9 /* Education Level and Marital Status */
10 proc sql;
11 create table Education as
12 select distinct (case when MomEdLevel=0 then "High School"
13 when MomEdLevel=1 then "Some College"
14 else "Less than High School" end) as Education,
15 (case when Married=0 then "Single"
16 else "Married" end) as Marital_Status,
17 (case when Married=0 then "Single, N"
18 else "Married, N" end) as N_Description,
19 round(avg(Weight), 1) as Avg_Weight,
20 3402 as Natl_Avg_Weight,
21 count(MomEdLevel) as Count,
22 (case when MomEdLevel=3 then 1
23 when MomEdLevel=0 then 2
24 when MomEdLevel=1 then 3
25 else 4 end) as Sort_Order
26 from sashelp.Bweight
27 group by Married, MomEdLevel; quit;
28 proc sort data=Education; by Sort_Order descending Marital_Status; run;
29 /* Smokers vs Non-Smokers */
30 proc sql;
31 create table Smoking as
32 select distinct (case when MomSmoke=0 then "Non-Smoker"
33 else "Smoker" end) as Smoker_Status,
34 round(avg(Weight), 1) as Avg_Weight,
35 count(MomSmoke) as Count,
36 cats("N = ", put(calculated Count, comma9.)) as Count_Label,
37 0 as Label_Y
38 from sashelp.Bweight
39 group by MomSmoke, MomEdLevel; quit;
40 /* Average Birth Weight by Number of Cigaretts Smoked per Day */
41 proc sql nprint;
42 select Smoking0.Avg_Weight,
43 Smoking1.Avg_Weight,
44 strip(put(Smoking0.Avg_Weight - Smoking1.Avg_Weight, 4.)) into :arrow1, :arrow2, :Smoking_Difference
45 from Smoking0, Smoking1; quit;
46 proc sql; drop table Smoking0, Smoking1; quit;
47 /* Average Birth Weight by Number of Cigaretts Smoked per Day */
48 proc sql;
49 create table Num_Cigs as
select distinct CigsPerDay as Num_Cigs,
    round(avg(Weight),1) as Avg_Weight,
    count(MomSmoke) as Count
from sashelp.Bweight
group by CigsPerDay;
quit

/********************** ORGANIZE POWER POINT SLIDES ******************/
/* Define Colors */
%let transparent=A00000000;
%let dark_blue=CX004281;
%let medium_blue=CX71A5E5;
%let light_blue=CXB5C8EB;
%let lighter_blue=CXD4DFF4;
%let gray=CX595959;
%let red=CXC00000;
/* Set ODS Settings */
ods html close;
opts nodate nonumber center;
opts escapechar='^';
opts graphics /border=off imagefmt=png;
goptions reset=all transparency;

***** Education Level and Marital Status *****
proc template;
define statgraph double_bar;
begingraph /pad=0 designwidth=9in designheight=4in opaque=false
datacolors=(&light_blue &dark_blue) datacontrastcolors=(&light_blue &dark_blue);
layout overlay /walldisplay=none
    xaxisopts=(display=(line tickvalues) tickvalueattrs=(color=&gray))
    yaxisopts=(display=none linearopts=(viewmin=3100 offsetmin=0 offsetmax=0.2));
barchart x=Education y=Avg_Weight /barlabel=true barlabelformat=comma9.
    barlabelattrs=(size=10pt) name='Education' barwidth=.8 clusterwidth=.6
    group=Marital_Status groupdisplay=cluster;
seriesplot x=Education y=Natl_Avg_Weight /lineattrs=(color=&red pattern=1
    thickness=2);
discretelegend 'Education' /border=false location=outside valign=bottom
displayclipped=true valueattrs=(size=10pt) opaque=false;
drawrectangle x=0 y=3402 width=32 height=7 /display=(fill) xspace=datapercen
    yspace=datavalue heightunit=data anchor=center fillattrs=(color=white)
    layer=front;
drawtext textattrs=(size=10pt weight=bold) "Average Birth Weight = 3,402g"
    /x=0 y=3402 anchor=center width=40 xspace=datapercen yspace=datavalue
    border=true borderattrs=(color=&red) justify=center layer=front pad=6pt;
endlayout; endgraph; end;
%macro ed_level_marital_status();
/* Title */
proc odstext;
p "Education Level and Marital Status"
/style=[fontsize=32pt color=&dark_blue fontweight=bold padding=0pt];
p "Average birth weight (g) by mother's education level and marital status."
/style=[fontsize=18pt color=&dark_blue fontweight=bold padding=0pt]; run;
ods layout gridded rows=2;
opts region row=1 width=9in;
/* Graph */
proc sgrender data=Education template=double_bar; run;
opts region row=2 width=9.5in;
proc odstext;
p "" /style=[fontsize=7pt]; run;
/* Table */
proc report data=Education nowd
style(header)={just=center fontsize=10pt color=&gray fontweight=bold}
padding=0pt background=&transparent bordercolor=&gray borderwidth=0.005in
style(column)=[just=center fontsize=10pt color=&gray padding=0pt
background=&transparent bordercolor=&gray borderwidth=0.005in]
style(report)=[just=left fontsize=10pt color=&gray padding=0pt
background=&transparent bordercolor=&gray borderwidth=0.005in];
column ("Mother's Education Level" N_Description) Education,Count;
define N_Description /" " group style=[width=1.7in] order=data;
define Education /" " across style=[width=1.5in] order=data;
define Count /" " analysis format=comma9.; run;

/* Footnote */
proc odstext;
p " " /style=[fontsize=10pt];
p "^{style[fontweight=bold]Note:} The x-axis on the above graph starts at
3,100g. ^n This study uses the Sashelp.Bweight dataset. For additional
information view the ^{style[URL='http://support.sas.com/documentation/cdl/en/statug/66859/HTML/default/viewer.htm#statug_sashelp_sect004.htm']SAS/STAT(R)
13.1 User's Guide.}"
/style=[fontsize=10pt color=&gray padding=0pt]; run;
ods layout end;

/* Page Break */
data _null_; dcl odsout obj(); run;
%mend;

/***** Smokers vs Non-Smokers *****/ 
proc template;
define statgraph bar_w_arrow;
begingraph /pad=0 designwidth=6.5in designheight=4in opaque=false;
layout lattice /rowdatarange=data columndatarange=union columns=1 rowgutter=0
rowweights=(.9 .1);
layout overlay /walldisplay=none
xaxisopts=(display=(line tickvalues) tickvalueattrs=(color=&gray size=10pt)
gridattrs=(thickness=10 offsetmin=3 offsetmax=.3
offsetmax=0.1);
yaxisopts=(display=none linearopts=(viewmin=3100) offsetmin=0
offsetmax=0.1);
barchart x=Smoker_Status y=Avg_Weight /barlabel=true barlabelformat=comma9.
barlabelattrs=(size=10pt) fillattrs=(color=&dark_blue)
outlineattrs=(color=&dark_blue barwidth=.5;
drawarrow x1="Non-Smoker" y1=&arrow1 x2="Non-Smoker" y2=&arrow2
/drawspace=datavalue arrowheadshape=barbed
lineattrs=(color=&light_blue thickness=6);
endlayout;
layout overlay /walldisplay=none xaxisopts=(display=none)
yaxisopts=(display=none linearopts=(viewmin=0) offsetmin=0.1 offsetmax=0);
scatterplot x=Smoker_Status y=label_y /markercharacter=count_label
labelstrip=true markercharacterattrs=(size=10pt color=black
family="Calibri")
endlayout;
endlayout; endgraph; end; run;

%macro non_smoker_vs_smoker();
/* Title */
proc odstext;
p "Non-Smokers vs Smokers"
/style=[fontsize=32pt fontweight=bold color=&dark_blue padding=0pt];
p "Average birth weight (g), non-smokers compared to smokers."
/style=[fontweight=bold color=&dark_blue fontsize=18pt padding=0pt];
p "" /style=[fontsize=20pt]; run;
ods layout gridded rows=2;
ods region row=1;
ods layout gridded columns=2 column_widths=(6.5in 2.5in) column_gutter=0in;
ods region column=1;
Proc Report, the Graph Template Language, and ODS Layouts: Used in Unison to Create Dynamic, Customer-Ready PowerPoints, continued

/* Graph */
proc sgrender data=Smoking template=bar_w_arrow; run;
ods region column=2;
/* Right side textbox */
proc odstext;
p "The birth weight for babies whose mother did not smoke was on average &Smoking_Difference grams higher than the birth weight for babies born to Mothers who smoked." /style=[fontsize=16pt color=&dark_blue]; run;
ods layout end;
ods region row=2 width=9.5in;
/* Footnote */
proc odstext;
p "^{style [URL='Birth Weight by Number of Cigarettes.xlsx'] Click here for the average birth weight by number of cigarettes smoked per day.} ^n"
ods layout end;
/* Page Break */
data _null_; run;
%mend;
/****************************** CREATE PPT ********************************/
ods powerpoint
options(backgroundimage="&main_dir\Images\SESUG_Background_w_Crabs.png")
file="&report\&customer Report.pptx";
%ed_level_marital_status();
%non_smoker_vs_smoker();
ods powerpoint close;
/************************ CREATE EXCEL BREAK OUT ***************************/
ods excel file="&report\Birth Weight by Number of Cigarettes.xlsx"
options (Sheet_Name="Birth Weight by Cigarettes" frozen_headers='YES' autofilter='ALL' absolute_row_height="45" embedded_titles='YES');
title1 color=black bold height=18pt "Average Birth Weight by ^n Number of Cigarettes Smoked per Day";
/* table*/
proc report data=Num_Cigs nowd
  style(report)=[bordercolor=white fontfamily=Calibri just=center vjust=middle fontsize=10pt]
  style(header)=[background=&dark_blue fontweight=bold fontsize=10pt fontfamily=Calibri foreground=white just=center vjust=middle bordercolor=white]
  style(column)=[background=&lighter_blue bordercolor=white fontfamily=Calibri just=center vjust=middle fontsize=10pt];
column Num_Cigs Avg_Weight Count;
define Num_Cigs / display 'Number of Cigarettes Per Day' flow
  style(column)=[background=&light_blue width=1.75in];
define Avg_Weight / display 'Average Birth Weight (g)' format=comma9. order=data
  style(column)=[width=1.75in];
define Count / display 'Count' format=comma9.
  style(header)=[fontweight=bold width=1in]; run;
ods excel close;