Paper 086-31

Really Cool Graphics With SAS/GRAPH® - Plotting and Embedding Likert Scales in Your Reports

Rick M. Mitchell, Westat, Rockville, MD

ABSTRACT

The use of Likert scales can be a very effective approach in reporting one's data. While this type of scale is commonly used across a wide-range of professions to show where a given measure fits within a predefined scale, references on how to generate this as a high-quality professional looking graphic in SAS are rare at best. This paper will discuss an approach to generating Likert scale graphics that is unique, yet basic. By utilizing PROC GPLOT within SAS/GRAPH along with a few nifty ODS features, a SAS programmer can easily plot and embed his or her own Likert scales into reports. Users of this approach will undoubtedly get a "Wow, those are really cool graphics!" reaction from their coworkers.

INTRODUCTION

Likert scales are commonly used in behavioral surveys to show where a measurement lies within a given scale that represents ranges such as weak to strong, negative to positive, and low to high. For the beginning SAS programmer, these scales can be difficult to graphically produce with no magical “push-button” options available. Embedding these graphics in a professional looking report can be even tougher. Fortunately, this process is quite simple and the approach discussed in this paper can be easily picked up by a SAS programmer and applied to one’s own work. The approach represents a process that includes ODS HTML designations, basic PROC GPLOT code, and a few simple statements that “trick the eye” and ultimately forces SAS to give the SAS/GRAPH user exactly what he or she wants. This paper will guide the SAS programmer through the necessary steps to achieve the graphics shown in Figure 1 below and beyond!

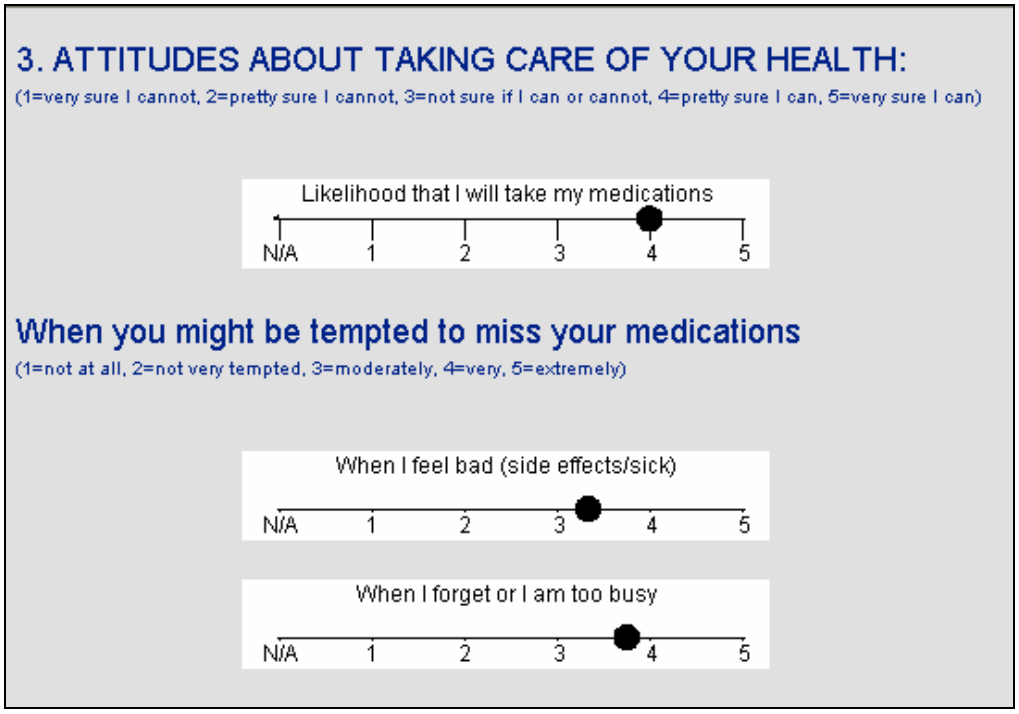

Figure 1 – Really Cool Graphics

BORING DEFAULT GRAPHICS

If one were to issue basic PROC GPLOT code within SAS/GRAPH, the user would generally be presented with unimpressive if not unprofessional looking graphics. Despite the vast variety of statements and options that users can apply, they can often be labor intensive and complicated and one may sometimes just be willing to settle for what SAS provides – boring default graphics. Let us take an example of basic code that can plot one variable (A which is a constant equal to zero) and a macro variable &XNAME into which the user can feed the desired analysis variable as shown in the code below:

```
proc gplot data=&dsn;
   plot a*&xname /  noframe nolegend;
   title1 "&tname";
run;
```

If one were to do nothing else beyond the code that is presented, the result would be a basic image (see Figure 2 below) – nothing fancy, only SAS defaults, and no special options nor statements.

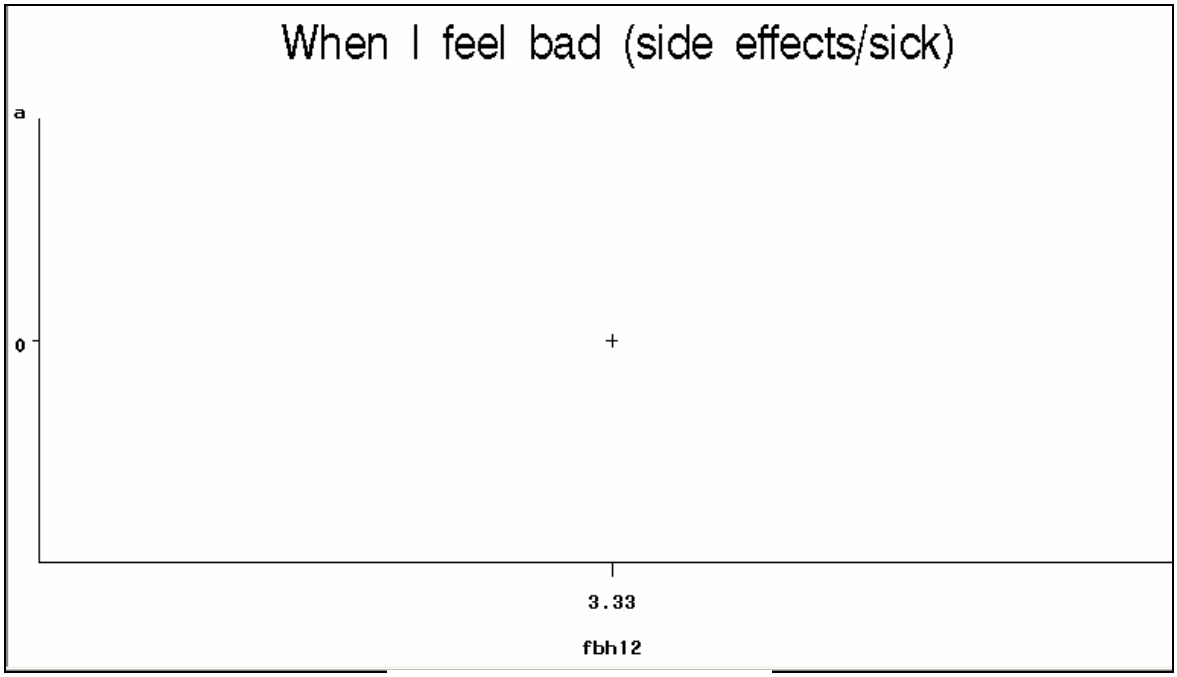

Figure 2 – Default Graphic Output

GENERATING REALLY COOL GRAPHICS

Let us now walk through a series of steps within the author's COOLGRAPH macro that will transform the boring default graphic shown in Figure 2 to the really cool graphic that was shown previously in Figure 1. These steps will help generate a Likert scale that may then be embedded within a given HTML report.

ADJUSTING THE SIZE OF THE IMAGE

To adjust the various dimensions of the image, one can simply take advantage of the SAS/GRAPH options that are also known as GOPTIONS. These options allow the user to specify the horizontal (HSIZE=n) and vertical (VSIZE=n) sizes of the image as well as a designation of the device (DEVICE=name) that will be used to store the image, in this case a GIF file. If the user does not specify a size unit preference (IN=inches, CM=centimeters, and PT=points), then SAS will use inches as the default. The code to perform this step is shown below:

```
GOPTIONS DEVICE=gif vsize=.5 hsize=3;
```

Note that after the GOPTIONS are applied, all dimensions of the image are reduced in size proportionately as shown in Figure 3 below.

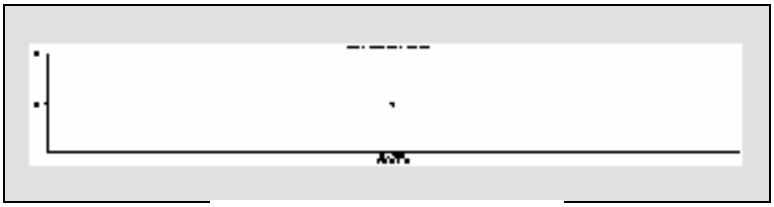

Figure 3 – Adjusted Image Size

So what now? The user has successfully generated an image that can fit nicely within a report, but what good is the image if one cannot read it? We will now proceed with enhancing selected portions of the image.

ADJUSTING THE SIZE OF THE TITLE

By merely telling SAS the height (h=n) of the text, one can bring out the TITLE statement to a more presentable format without modifying the dimensions of the existing image. The code for this step is shown below along with user select preferences of font (font=arial), justification (j=c), and macro text (&tname):

```
title1 h=15 font=arial j=c "&tname";
```

Note that even though our height setting is so large (h=15), all other aspects of the image are unchanged with the exception of a very slight decrease in the size of the vertical axis (Figure 4 below).

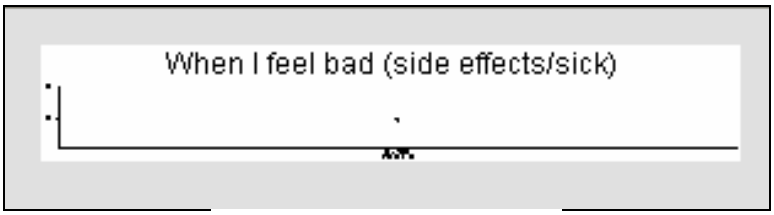

Figure 4 – Adjusted TITLE Size

ADJUSTING THE SIZE OF THE SYMBOL

The user may specify a wide-range of symbols, and in this example we choose to select a filled in circle (v=dot) that is increased significantly in size (h=25) as shown in the code below.

```
symbol1 v=dot font= h=25;
```

Note that the height of the symbol has no effect on the TITLE statement that was just run, nor does it affect the other dimensions of the image (Figure 5 below). One is free to select any type of symbol and any type of height without having to take anything else into consideration besides the constraints of the image itself.

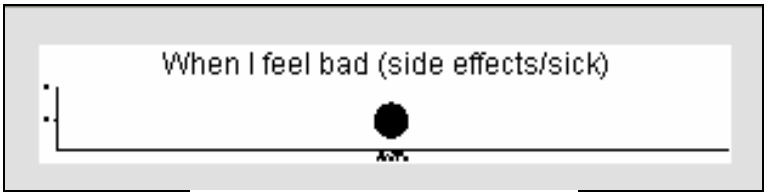

Figure 5 – Adjusted SYMBOL Size

ADJUSTING THE VERTICAL AXIS

We want to plot this point ON the horizontal axis (x-axis), and we want to eliminate the vertical axis (y-axis). Current SAS options do not allow us to achieve our goal by performing just one of these steps by itself. If

one eliminated the vertical axis, then the symbol will have no reference to plot on. Similarly, if one moved the symbol down, then the y-axis would compensate and the symbol still would not land on the x-axis. Therefore, it will be necessary to perform 2 steps:

Step 1 – SHRINKING THE SIZE (MOVING THE SYMBOL DOWN)

To move the symbol down within the image, one can utilize the LENGTH statement as shown below:

```
axis1 length=1;
```

Note that one can barely see the vertical axis now as it has "shrunk" and the symbol moves down with it (Figure 6 below), although its height specifications remain unaffected. The x-axis labels are overwritten by the large size of the symbol, however, the perfectionist SAS programmer may wish to clean this up.

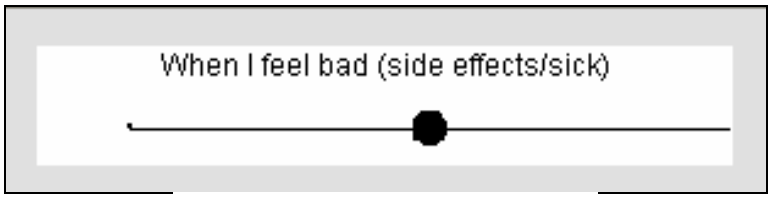

Figure 6 – Shrinking the Vertical Axis

STEP2 – TURNING THE Y-AXIS "OFF"

As noted previously, if the Y-AXIS was actually turned off literally, then the symbol would not plot. But, we can "trick" the recipient of the image to think that it is turned off by simply changing the color of the axis as shown in the code below:

```
axis1 color=white;
```

Note that white has been chosen assuming that a white background is being used (Figure 7 below). If different background colors were used, then the SAS programmer may change the option accordingly.

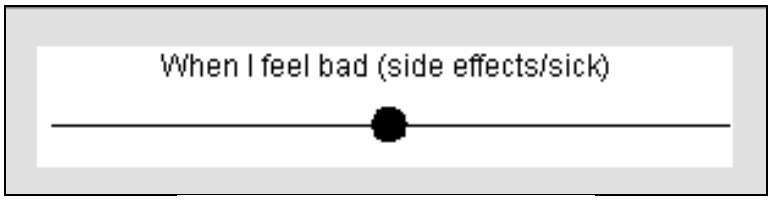

Figure 7 – Vertical Axis Turned "Off"

HORIZONTAL AXIS STATEMENT

Now that the vertical axis has been completed, one needs to modify the horizontal axis to finish things off. The image needs Likert scale values added, and as will be discussed, the TITLE statement is subsequently affected and must be adjusted as well. This results in a 2-step process: (1) add the Likert scale values, and (2) move the TITLE up.

STEP 1 – ADD THE LIKERT SCALE VALUES

Likert scale values can be added by utilizing the AXIS2 statement. To the typical SAS/GRAPH users, many of these options are not extraordinary, so non-familiar users are encouraged to consult the SAS/GRAPH User's Guide. The focus of this code is actually the height statement (h=15) as shown below:

```
axis2 label=none
      order=(0 to 5 by 1)
      value=(font=arial h=15 'N/A' '1' '2' '3' '4' '5')
      minor=none;
```

The height option is inserted within the VALUE option to give compatible values that are in sync with the image and its settings to produce Figure 8 below.

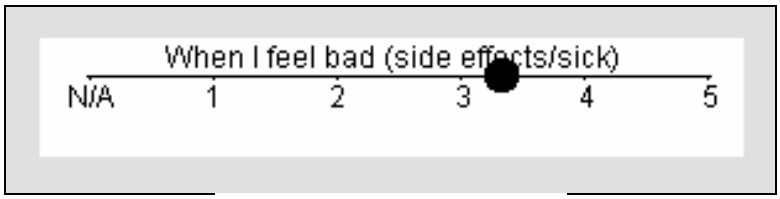

Figure 8 – Likert Values Added

Note that any changes made to the AXIS2 statement do not affect any other portion of the graph. However, while the TITLE did not move, the x-axis did move up in order for the image dimensions to accommodate the increased value sizes. So, our TITLE is now practically sitting on top of the scale requiring us to make one final adjustment.

STEP 2 – MOVE THE TITLE UP

Once again, we will need to perform an extra "trick" to finish off the image. This can be accomplished by simply adding another title statement (or multiple statements) with empty space that effectively moves up the actual TITLE as shown in the code below:

```
title1 h=15 font=arial j=c "&tname";
title2 h=30;
```

This example uses a very large height setting (h=30), although the user may either increase or decrease this number to get the desired look for the image (Figure 9 below).

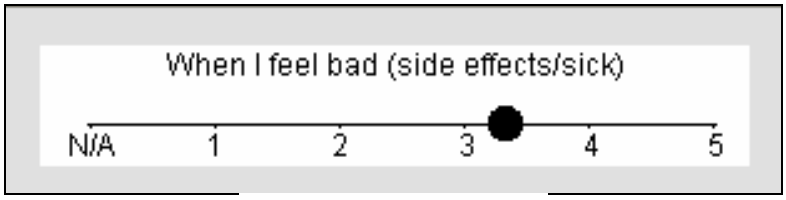

Figure 9 – TITLE Moved Up

The COOLGRAPH Macro

This paper has discussed several simple steps to achieve a professional looking Likert scale graphical image. The combination of these steps forms the macro, COOLGRAPH. As noted already, the code itself is not "earth-shattering," and it is more the sizing of the various image components that drives the process to create just the right look. Below is the COOLGRAPH macro code in its entirely:

```
%macro COOLGRAPH(dsn=sdata, tname=, xname=, majortick=none);
   symbol1 v=dot font= h=25;
   GOPTIONS DEVICE=gif vsize=.5 hsize=3;
   proc gplot data=&dsn;
      plot a*&xname / noframe nolegend vaxis=axis1 haxis=axis2;
      axis1 length=1 color=white;
      axis2 label=none
            order=(0 to 5 by 1)
            value=(font=arial h=15 'N/A' '1' '2' '3' '4' '5')
            minor=none
            major=&majortick;
      title1 h=15 font=arial j=c "&tname";
      title2 h=30;
   run;
%mend COOLGRAPH;
```

BEYOND GRAPHICS: IMBEDDING IMAGES AND TEXT IN A REPORT

Now that you've learned to wow your colleagues with really cool graphics, this paper will now explain how to easily embed these graphics and other text within an HTML report. To accomplish this, one must integrate 3 tasks: (1) designating an output source, (2) controlling page breaks, (3) and inserting text and images.

DESIGNATING AN OUTPUT SOURCE

For the purposes of this paper, only the HTML destination is discussed. The code to designate an output source would be as follows:

```
ODS HTML FILE='C:\REALLYCOOLGRAPHICS.HTML';

* INSERT YOUR OWN REPORT HERE;

ODS HTML CLOSE;
```

Users who are interested in finding out how this approach may work for another given output source (e.g. RTF and PDF) are encouraged to use the code presented in this paper as a starting point. The particular task that the author performed this approach for required an HTML report, and some steps may or may not be appropriate for getting the same look in another format.

CONTROLLING PAGE BREAKS

After generating any type of SAS report, table, and/or graph, there is a default page break that separates each of these items. If the SAS programmer chooses to incorporate multiple items on a single page, then a basic understanding of how to control these page breaks is required. For the example discussed in this paper, it was desired to use both a page break and no page break within select portions of the report.

TURNING THE PAGE BREAK OFF

Using PROC TEMPLATE to generate a customized STYLE will provide a mechanism for a user to turn off page breaks as needed. The following code updates a SASUSER template and then declares a new STYLE template called BREAKOFF in which the page break default is changed to undefined (pagebreakhtml = _undef_) as shown in the code below:

```
* BREAKOFF - removes the page break;

libname sasuser2 'c:\';
ods path sasuser2.tmplmst(update) sashelp.tmplmst(read);

proc template;
   define style BREAKOFF;
      parent=styles.default;
      style body from body / pagebreakhtml=_undef_;
   end;
run;
```

Once the template is defined, then the programmer need only issue STYLE=BREAKOFF in an ODS HTML statement whenever appropriate as shown below:

```
ODS HTML STYLE=BREAKOFF;
```

TURNING THE PAGE BREAK ON

In contrast to the BREAKOFF STYLE that was defined, one need only create another STYLE that merely contains the defaults (which initiate the page break after each report, table, and/or graph that is produced). The code to achieve this is shown on the next page:

```
* BREAKON - turns the page break back on;

proc template;
   define style BREAKON;
      parent=styles.default;
      style body from body;
   end;
run;
```

Similar to STYLE=BREAKOFF, the programmer merely needs to issue an ODS HTML statement with the specified STYLE of BREAKON when wishing to go back to the default page break as shown below:

```
ODS HTML STYLE=BREAKON;
```

INSERTING TEXT AND IMAGES

With all of the pieces now available, one may insert a variety of images and text within the same page of a report by taking advantage of a few basic tricks. This will allow the SAS programmer to spice things up and build on the graphical images that have been discussed in this paper. By surrounding the graphics with various lines of high quality text as well as performing variations on the existing graphics or other graphics, the SAS programmer may begin to "wow" coworkers. We will now touch on the necessary steps to insert text and images by (1) using the COOLTEXT macro to include text within the report, and (2) using a &majortick macro variable as an example to show how one may add variations to the Likert scale.

Text Within the Report (the COOLTEXT macro)

Adding text to a report can give it that high quality professional look that users may be tempted to pursue with other software packages. Even though SAS may not have the perfect procedure to easily allow a programmer to display a combination of customized graphics and text easily, one can take advantage of an approach of running PROC REPORT with all variables "turned off" and with only the TITLE statements (with ODS options) being utilized. Basically, the macro COOLTEXT that is shown below defines a single variable (BLANKLINE) and instructs PROC REPORT not to print this variable (option NOPRINT). So, the variable is not used, but PROC REPORT still generates the TITLE statements. In this case, the programmer may use just 1 TITLE statement or multiple TITLE statements along with any ODS options that will give these TITLE statements the desired look. Although there is a limit of 10 TITLE statements within a SAS procedure, one can easily get around this limitation by appropriately controlling page breaks as discussed earlier in this paper. One wishing to learn more about this approach in greater detail is encouraged to read, "Stranded on a Deserted Island With Nothing But TITLE Statements," which was presented by the author at both NESUG 2004 and SUGI 30. The code, macro call, and output (Figure 10) for such an approach are shown below:

The Code

```
%macro COOLTEXT(titlestr1=, titlestr2=, titlestr3=, titlestr4=,
                titlestr5=, titlestr6=, titlestr7=, titlestr8=,
                titlestr9=, titlestr10=);
   proc report data=sdata missing nowindows split='!'
      style(report)=[background=white foreground=black fs=.1]
      style(header)=[background=white foreground=darkblue fs=.1]
      style(column)=[fs=.1 just=center];
      columns blankline;
      define blankline   / noprint center color=red width=96
         style=[cellwidth=400 background=white];
      title1 &titlestr1.;
      title2 &titlestr2.;
      title3 &titlestr3.;
      title4 &titlestr4.;
      title5 &titlestr5.;
      title6 &titlestr6.;
      title7 &titlestr7.;
```

```
      title8 &titlestr8.;
      title9 &titlestr9.;
      title10 &titlestr10.;
   quit;
%mend COOLTEXT;
```

The Macro Call

```
%COOLTEXT(titlestr1= h=4 font=arialbold j=l "3.  ATTITUDES ABOUT TAKING CARE OF
                                                YOUR HEALTH:",
          titlestr2= h=.75 font=arial j=l "     (1=very sure I cannot, 2=pretty
            sure I cannot, 3=not sure if I can or cannot, 4=pretty sure I can,
            5=very sure I can)")
```

The Output

Figure 10 – Text Within the Report

Variations to the Likert Scale (the &majortick macro variable)

If one wanted to generate similar graphics as produced with COOLGRAPH, but with some slight variation(s), SAS allows the flexibility for such enhancements. As an example of the potential variations that a SAS programmer could perform, we have produced a slightly different graph (Figure 11 below) where major tick marks are shown. This graph, "Likelihood that I will take my medications," is actually the 1st graph produced in the sample report discussed in this paper (Figure 1). It was generated in the same manner with a very slight deviation where the programmer's client had requested that major tick marks be present for that graph only. This was achieved by inserting an additional macro variable, &majortick, to the COOLGRAPH macro (this is the only difference between the 1st graph and the subsequent graphs in Figure 1). In Figure 11, one can see that resolving "`major=&majortick`" in the AXIS2 statement to "`major=(h=10 number=5)`" results in a nice small variation to the graph. Using this concept as an example, one can probably imagine that the "sky is the limit" with endless possibilities being available to the SAS programmer to tailor these graphics to their own special needs.

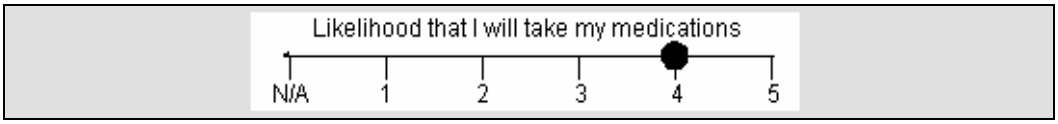

Figure 11 – A Variation, Adding Major Tick Marks

PUTTING ALL OF THE PIECES TOGETHER

Several coding techniques have been discussed individually, which allow the SAS programmer to generate a variety of high-quality graphical and textual images. However, it is the combination of all of these pieces together which produces a unique and professional looking report. Figure 12 on the following page shows how these individual pieces relate to each other in terms of the report as a whole. Both the COOLTEXT and COOLGRAPH macros are called upon within SAS along with multiple page break controls (BREAKON and BREAKOFF) being utilized to produce a report that any SAS programmer should be proud to present.

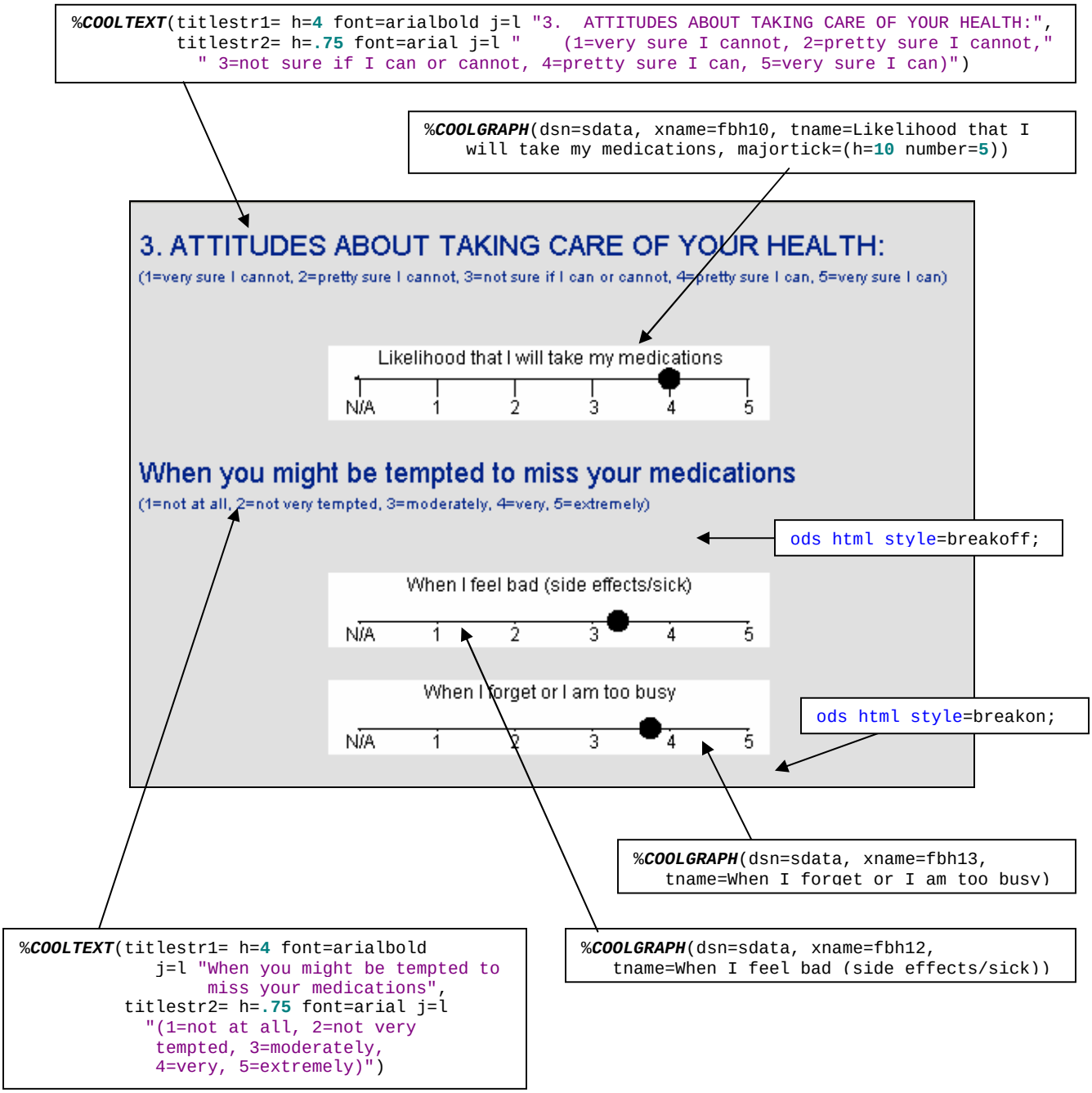

Figure 12 – Putting the Pieces Together

CONCLUSION

This paper has demonstrated how to generate Likert scales and embed these scales within an HTML report. Endless possibilities await the SAS programmer as one may use the approach described in this paper as a solid foundation for branching out in a variety of directions. By fully utilizing the many SAS tools that are available, one may avoid having to jump between multiple software packages to achieve one's project goals. The power and flexibility of SAS allows the SAS programmer to expand on graphical techniques that are basic in terms of programming requirements, yet advanced in terms of looks. Programmers who are

tempted to explore other software packages because of their uncertainty in being able to push SAS beyond its defaults need only nudge SAS a little to fully realize the realm of possibilities that are available to both the beginning and the advanced SAS programmer. So, if you would like to continue to impress your colleagues, take advantage of the skills learned in this paper and apply these to your everyday work. Present these types of professional looking graphical and textual images, and you will undoubtedly get positive reactions of "Wow, those are really cool graphics!"

REFERENCES

- Mitchell, R.M. (2005). Stranded on a Deserted Island With Nothing But TITLE Statements (Paper 069-30). *Proceedings of the 30th Annual SAS Users Group International (SUGI) Conference*.

ACKNOWLEDGEMENTS

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.

DISCLAIMER: The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

CONTACT INFORMATION

Rick M. Mitchell
Westat
1650 Research Boulevard, WB 492
Rockville, MD 20850
(301) 251-4386 (voice)
(301) 738-8379 (fax)
RickMitchell@Westat.com