Paper DV01-2014

Categorical AND Continuous – The Best of Both Worlds

Kathryn Schurr, M.S., Spectrum Health-Healthier Communities, Grand Rapids, MI

Ruth Kurtycz, Spectrum Health-Healthier Communities, Grand Rapids, MI

ABSTRACT

Visually representing data such as blood pressure has often presented a challenge for health professionals. Do we treat blood pressure as a continuous or categorical variable when interpreting this type of data? When continuous data is depicted graphically, minute changes allow decision makers to identify positive or negative trends over time; categorical data on the other hand permits one to efficiently identify abnormal levels that facilitate prompt assignment of treatment. Although each type of graphic has its distinct uses, there has been no visual representation that simultaneously displays blood pressure's categorical and continuous nature.

We have devised a bi-variate scatter-band plot to show categorical changes in parameters such as blood pressure, while still maintaining the integrity and the information density of the continuous variable. This graphic allows the user to observe the changes in the data through the scatterplot, while colored bands show the categorical nature of each data point. To utilize the method discussed here, users should have a basic understanding of PROC SGPLOT and access to SAS v9.3.

Introduction

Blood pressure is a measure of one's overall health and is one of the most common indicators for a person's risk of developing chronic diseases such as heart disease, stroke, and kidney failure. Table 1 below delineates the various blood pressure categories as set forth by the American Heart Association and provides guidelines for classifying a person's blood pressure reading.

TABLE 1

Systolic Reading	Action Category	Diastolic Reading
Higher than 210	**Emergency** See health care provider or hospital emergency room within 24 hours	Higher than 120
160-209	**Urgent High, Stage 2** See your health care provider within 1 week. If you are being treated for high blood pressure, keep following the treatment as ordered by your doctor.	100-119
140-159	**High, Stage 1** See your health care provider within 1-2 months. If you are being treated for high blood pressure, keep following the treatment as ordered by your doctor.	90-99
120-139	**Pre-hypertension or High Normal** Have your blood pressure re-checked within 6 months. Consider lifestyle changes to improve your blood pressure.	80-89
Lower than 120	**Normal** If you are being treated for high blood pressure, follow up with your doctor on a regular basis. If you are not being treated for high blood pressure, have your blood pressure re-checked within 2-years.	Lower than 80

As a physiological parameter, blood pressure shows natural variation in repeated measurements. In the case of an individual with an initial blood pressure reading of 119/79; this individual's blood pressure would be categorized in the normal range. Say that a month from that measurement, another reading is taken. The second reading is 124/76. In this instance, the new classification would be High-Normal blood pressure or Pre-Hypertensive. Because in this example only one individual is being evaluated, a conclusion could be drawn that the change in the person's blood pressure is most likely due to natural variation and not in fact due to an underlying malady. To evaluate multiple individuals for changes due to natural variation, using the categories of Table 1, decision makers might use the style represented by Table 2. This table shows the categorization of the systolic measurement at an initial screening compared with the categorization of a second screening result some time later.

TABLE 2

Systolic	**Reading At Rescreening**				
Reading At Initial Screening	**Normal**	**Pre-Hypertensive**	**High, Stage 1**	**Urgent High, Stage 2**	**Total**
Normal	83	26	1	0	110
Pre-Hypertensive	27	36	7	0	70
High, Stage 1	0	11	3	1	15
Urgent High, Stage 2	0	0	0	1	1
Total	110	73	11	2	196

By looking at the table above, the highlighted cell shows that 26 individuals increased from their initial screening of Normal to Pre-Hypertensive. By evaluating these individuals further, it is found that out of the original 26, 7 individuals had a blood pressure change of less than 10 mmHg. This paper is not meant to discuss what change would be considered diagnostically significant; however, those 7 patients' results comprise roughly 25% of the increase from Normal to Pre-Hypertensive blood pressure. When looking at the table from a decision maker's perspective, it would be difficult to apply a treatment or action based on a categorical change that could in fact equate to very small differences numerically.

Instead of analyzing the data categorically, what if the actual numeric changes between screenings were considered? Table 3 shows the numeric change results below for the same 196 individuals as Table 2.

TABLE 3

Change in Systolic Blood Pressure									
N	Mean	Std Dev	Minimum	10th Pctl	25th Pctl	Median	75th Pctl	90th Pctl	Maximum
196	-0.35	12.87	-31.00	-16.00	-8.00	-1.50	7.00	17.00	50.00

By looking at Table 3, we see that the mean change in Systolic Blood Pressure was a decrease of 0.35 mmHg, which means that the population's systolic blood pressure decreased on average. The table percentiles also hint at dissimilar distributions of systolic change experienced by subgroups of individuals. However, this method for evaluating change in blood pressure is also flawed because it is highly susceptible to extreme values; there is also no simple way to identify if the individual who increased by 50 systolic points ended up in an Urgent High category or a High Category, both scenarios are possible. Though the exact numeric representation of the systolic blood pressure variable is preserved, there is very little information here that could influence decision makers into a treatment action one way or another.

Because both methods have flaws in their ability to help decision makers, we have devised a visual bi-variate graphic that captures both the categorical nature of blood pressure while preserving the underlying numerical value of the measurement. We call this graphic the Scatter-Band Plot.

Getting Started

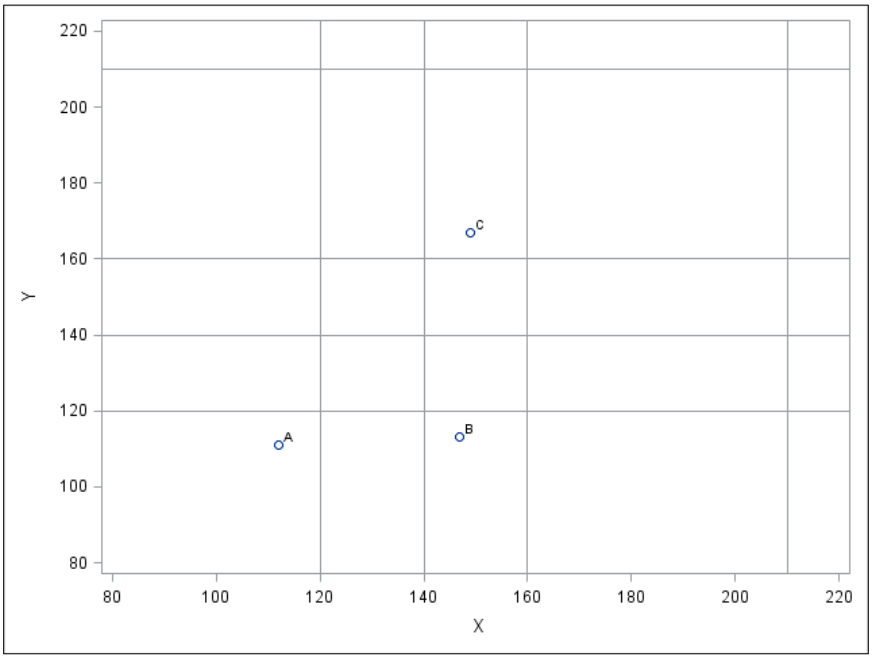
The first graphic is a scatter plot showing the difference in systolic blood pressure at two different times. The first measurement (Variable X) was taken at an initial screening, and the rescreening value (Variable Y) was taken 6 months later. The first and most complex part of the graph is determining where the categorical lines on the graphic should be. To begin, the sample data step below includes hypothetical data from three patients; Patient A, Patient B, and Patient C.

```
DATA Example;
      INPUT Patient $ X Y;
      CARDS;
A 112 111
B 147 113
C 149 167
;
RUN;
```

This data can be graphed using PROC SGPLOT. Before the three patients' results are plotted, categorical reference lines are added to the graphic using Table 1 as a guide. The lines are drawn so that they represent the boundaries of various stages of hypertension. Figure 1 results from the following code:

```
PROC SGPLOT DATA = Example;
      SCATTER X = X Y = Y / DATALABEL = Patient;
      REFLINE 120 140 160 210;
      REFLINE 120 140 160 210 / AXIS = X;
      YAXIS VALUES = (80 100 120 140 160 180 200 220);
      XAXIS VALUES = (80 100 120 140 160 180 200 220);
RUN;
```

FIGURE 1



In the figure above, the lines representing the categorical boundaries have been plotted in addition to the sample data. By looking at the individual patients, where the bands should be placed becomes a bit clearer. Patient A began in the 'Normal' range and also was 'Normal' at their rescreening. Patients falling within the main diagonal made by the categorical boundaries on this graphic will have not changed their hypertensive stage between the initial screening and rescreening. Patient B was classified as 'High, Stage 1' hypertensive at their initial screening and 'Normal' at the time of rescreening. This change represents a two-category shift to the right for the individual (they improved their hypertension by two categories). Any patient that falls below the main diagonal has a better systolic blood pressure at their rescreening than they did at the initial screening. Using that same logic to evaluate Patient C we find that the initial hypertensive category was 'High, Stage 1' and the rescreening hypertensive category was 'High, Stage 2'. Any patient that falls in a box above the main diagonal will have a hypertensive categorical shift in the negative direction.

Banding

With the logic used above, the general location of the bands has been determined. To further delineate the change over time, colors may be added to the graphic according to the boundaries previously discussed. The simplest bands to implement will be the bands representing the 'No Change' individuals, or the individuals whose rescreening category matched their initial screening. These patients fall on the main diagonal. To begin, the diagonal band data will be created in the data step below.

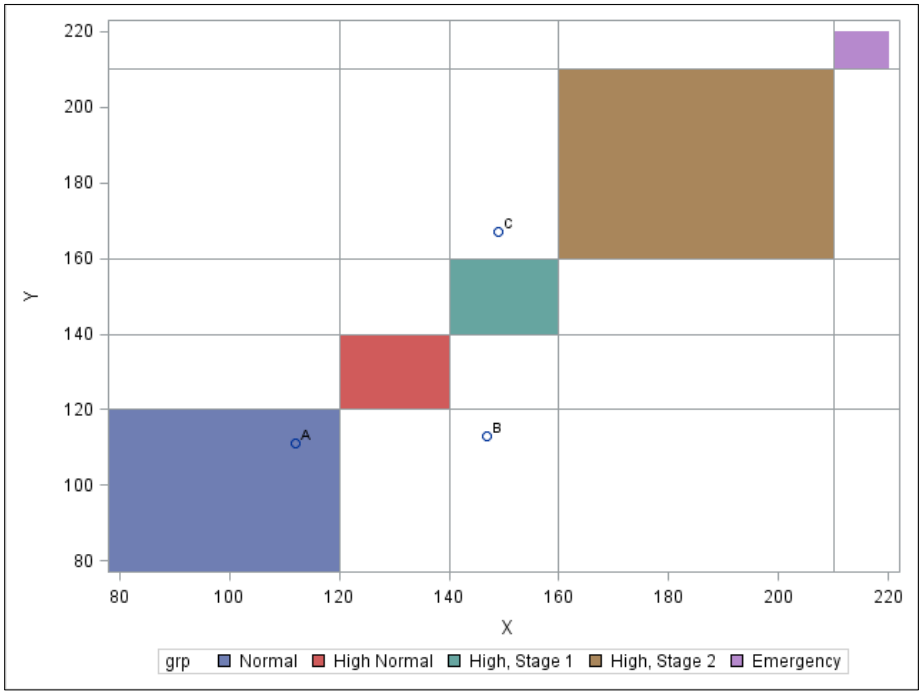
```
DATA systolic;
      INPUT sys losys hisys grp $ 13-25;
      CARDS;
70  70  120 Normal
120 70  120 Normal
120 120 140 High Normal
140 120 140 High Normal
140 140 160 High, Stage 1
160 140 160 High, Stage 1
160 160 210 High, Stage 2
210 160 210 High, Stage 2
210 210 220 Emergency
220 210 220 Emergency
;
RUN;
```

The systolic data step contains the banding data for each classification located on the main diagonal. Each hypertension group is defined by two lines; the first line gives the lower boundary for the x axis (sys), the lower and upper boundaries for the y axis (losys & hisys), and the name of the hypertensive group. The second line follows the same pattern only the value of sys represents the upper boundary for the x axis. If there are not two lines per group, the bands do not display correctly.

Now that the first banding information has been created, the patient data must be merged with the bands so that the scatter plot and band plot can be placed on the same graphic. The data step below and following PROC SGPLOT code result in Figure 2.

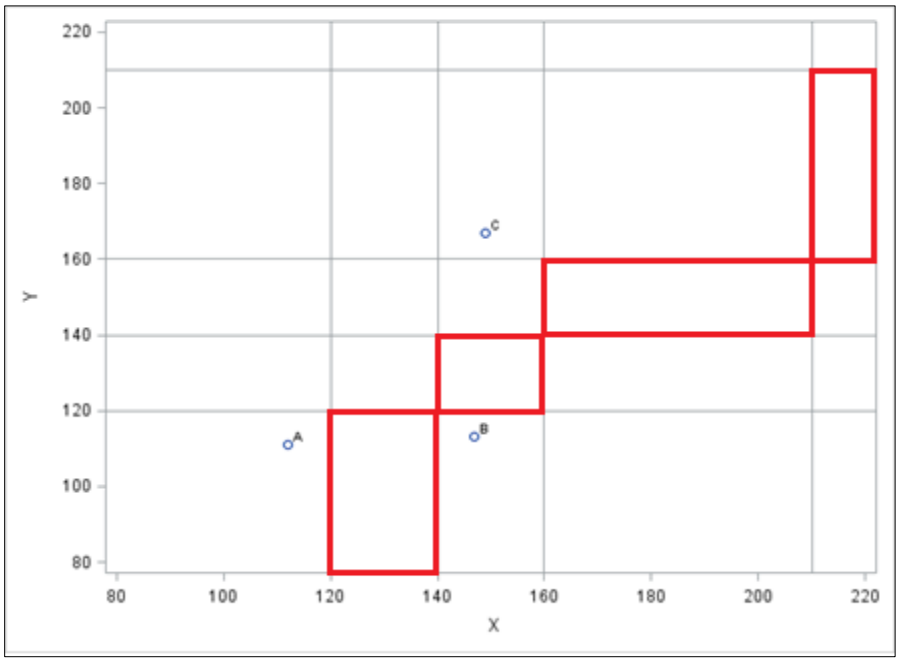
```
DATA Example_2;
      MERGE Example Systolic;
RUN;

PROC SGPLOT DATA = Example_2;
      BAND X = sys LOWER = losys UPPER = hisys / GROUP = grp NAME = "bp";
      SCATTER X = X Y = Y / DATALABEL = Patient;
      REFLINE 120 140 160 210;
      REFLINE 120 140 160 210 / AXIS = X;
      YAXIS VALUES = (80 100 120 140 160 180 200 220);
      XAXIS VALUES = (80 100 120 140 160 180 200 220);
RUN;
```

FIGURE 2



Now that the diagonal has been appropriately demarcated, the next step is to fill in the negative changes. This becomes a bit more complex because a decrease in one hypertensive stage is located in more than one position on the graphic. This is also true for a decrease in two or more hypertensive stages. In Figure 3, the decrease in one hypertensive stage boxes have been highlighted to show where they fall on the graphic.

FIGURE 3



To fill in those classifications, a data step similar to systolic is required. The data step is below.

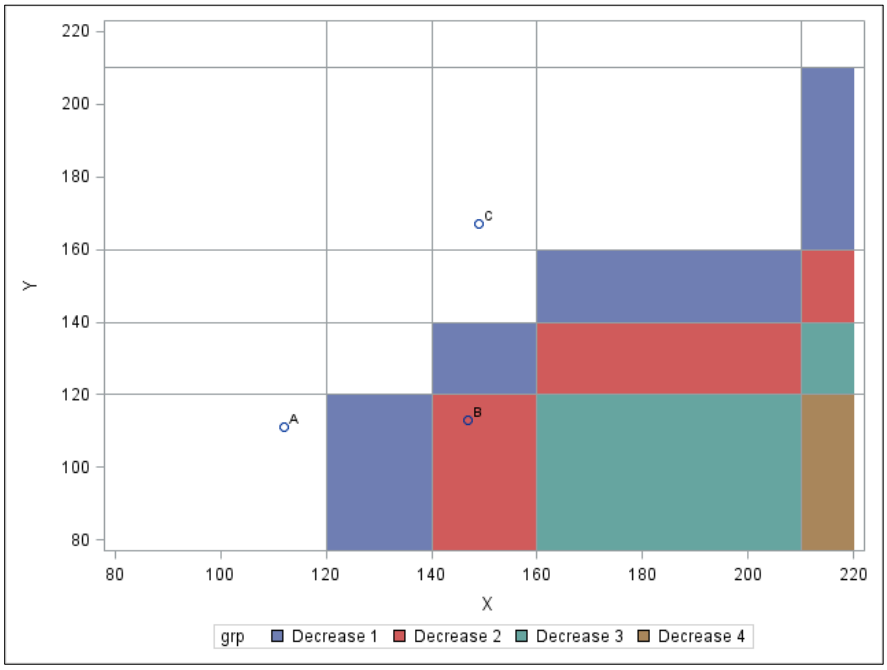
```
DATA Systolic_2;
      INPUT sys losys hisys grp $ 13-25;
      CARDS;
120 70  120 Decrease 1
140 70  120 Decrease 1
140 120 140 Decrease 1
160 120 140 Decrease 1
160 140 160 Decrease 1
210 140 160 Decrease 1
210 160 210 Decrease 1
220 160 210 Decrease 1
140 70  120 Decrease 2
160 70  120 Decrease 2
160 120 140 Decrease 2
210 120 140 Decrease 2
210 140 160 Decrease 2
220 140 160 Decrease 2
160 70  120 Decrease 3
210 70  120 Decrease 3
210 120 140 Decrease 3
220 120 140 Decrease 3
210 70  120 Decrease 4
220 70  120 Decrease 4
;
RUN;
```

Because there is more than one 'box' outlined in Figure 3, each 'box' representing a decrease of the same magnitude needs two lines. Figure 3 shows four 'boxes' that need to be banded as the same category. Following the formula previously mentioned, the dataset Systolic_2 contains 8 lines for 'Decrease 1'. Similarly, there are 6 lines for 'Decrease 2', 4 lines for 'Decrease 3' and 2 lines for 'Decrease 4'.

Now that the banding code has been created for the negative changes, we must again merge this data with the patient data to create the appropriate graphic. The data step and PROC SGPLOT code result in Figure 4.

```
DATA Example_3;
      MERGE Example Systolic_2;
RUN;

PROC SGPLOT DATA = Example_3;
      BAND X = sys LOWER = losys UPPER = hisys / GROUP = grp NAME = "bp";
      SCATTER X = X Y = Y / DATALABEL = Patient;
      REFLINE 120 140 160 210;
      REFLINE 120 140 160 210 / AXIS = X;
      YAXIS VALUES = (80 100 120 140 160 180 200 220);
      XAXIS VALUES = (80 100 120 140 160 180 200 220);

RUN;
```

FIGURE 4



As previously mentioned, Patient B is shown in the appropriate box portraying that they decreased two hypertensive stages between their initial screening and their rescreening.

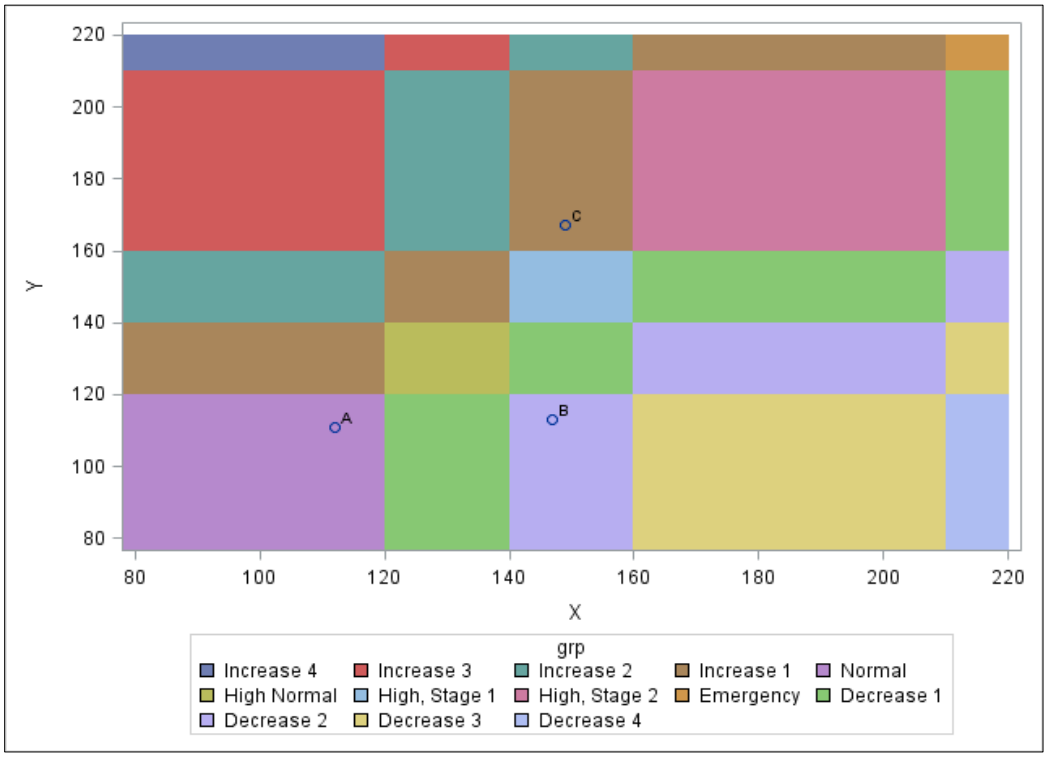
Using the same methodology as above, now the final portion of the banding dataset, representing the 'Increase' stages, can be created. We combine the other two banding datasets with the latest one to create a final succinct banding dataset containing all the appropriate bounds and labels. This final dataset is given below.

```
DATA systolic;
      INPUT sys losys hisys grp $ 13-25;
CARDS;
70  210 220 Increase 4
120 210 220 Increase 4
70  160 210 Increase 3
120 160 210 Increase 3
120 210 220 Increase 3
140 210 220 Increase 3
70  140 160 Increase 2
120 140 160 Increase 2
120 160 210 Increase 2
140 160 210 Increase 2
140 210 220 Increase 2
160 210 220 Increase 2
70  120 140 Increase 1
120 120 140 Increase 1
120 140 160 Increase 1
140 140 160 Increase 1
140 160 210 Increase 1
160 160 210 Increase 1
160 210 220 Increase 1
210 210 220 Increase 1
70  70  120 Normal
120 70  120 Normal
120 120 140 High Normal
140 120 140 High Normal
140 140 160 High, Stage 1
160 140 160 High, Stage 1
160 160 210 High, Stage 2
210 160 210 High, Stage 2
210 210 220 Emergency
220 210 220 Emergency
120 70  120 Decrease 1
140 70  120 Decrease 1
140 120 140 Decrease 1
160 120 140 Decrease 1
160 140 160 Decrease 1
210 140 160 Decrease 1
210 160 210 Decrease 1
220 160 210 Decrease 1
140 70  120 Decrease 2
160 70  120 Decrease 2
160 120 140 Decrease 2
210 120 140 Decrease 2
210 140 160 Decrease 2
220 140 160 Decrease 2
160 70  120 Decrease 3
210 70  120 Decrease 3
210 120 140 Decrease 3
220 120 140 Decrease 3
210 70  120 Decrease 4
220 70  120 Decrease 4
;
RUN;
```

Once the banding data has been successfully created, combining the final banding data with the example patient data creates the data set needed to produce the full banded scatter plot. The data step and PROC SGPLOT code result in Figure 5.

```
DATA Example_4;
      MERGE Example Systolic;
RUN;

PROC SGPLOT DATA = Example_4;
      BAND X = sys LOWER = losys UPPER = hisys / GROUP = grp NAME = "bp";
      SCATTER X = X Y = Y / DATALABEL = Patient;
      YAXIS VALUES = (80 100 120 140 160 180 200 220);
      XAXIS VALUES = (80 100 120 140 160 180 200 220);
RUN;
```

FIGURE 5



Coloring

Although Figure 5 displays all relevant bands, the default colors make this graph difficult to read. Instead of the default, let's apply a color gradient also addresses the relative severity of the change in systolic blood pressure. One choice for the color-coded gradient would show all increases in Hypertension as a shade of red, all decreases in Hypertension stages a shade of green, and all non-changes a shade of yellow. More intense colors represent a more severe change in the increase and decrease groups, and represent a more serious categorization in the no change groups.

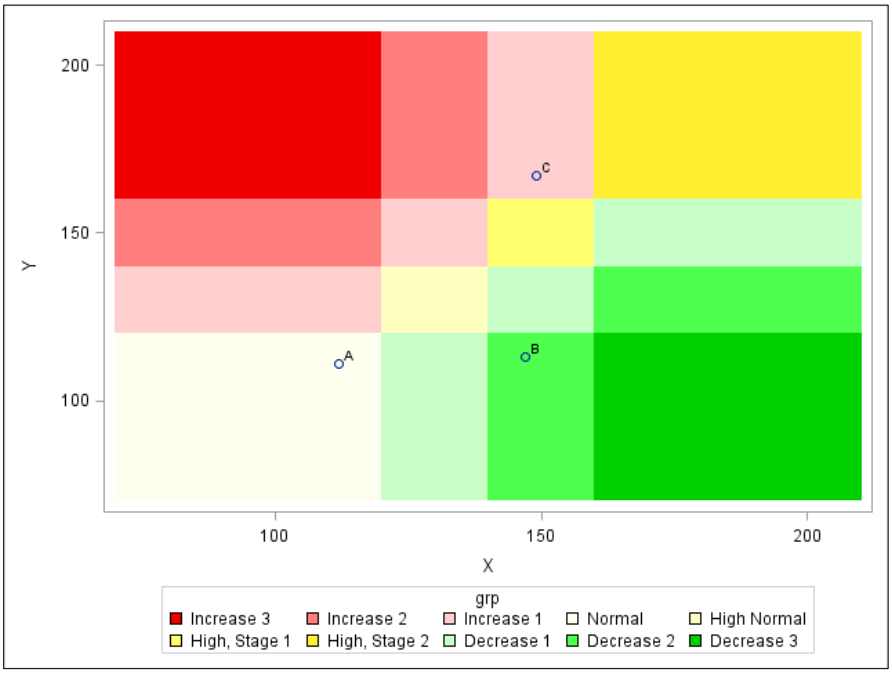
After some research and experimentation, it was found that using PROC TEMPLATE to color the graphic was not advisable due to the generally large number of colors needed to display all categories and changes (13 are needed for this graphic). Instead an attribute dataset will be implemented; an attribute datastep has three necessary variables, Value, Fillcolor, and ID. The 'Value' variable is filled in with the category names from the band dataset. The 'Fillcolor' variable specifies what color code is to be used for each value or category. The final variable, 'ID,' is used to ensure that SAS knows which value to include for a particular graphic. The datastep below creates the attribute dataset. For this graph, the categories of Emergency, Decrease 4, and Increase 4 were removed due to the lack of data in those categories.

```
DATA myattrmap;
      INPUT value $1-13 fillcolor $15-22 ID $24-27;
      CARDS;
Normal         CXFFFFF0 myid
High Normal    CXFFFFC0 myid
High, Stage 1 CXFFFF70 myid
High, Stage 2 CXFFEF30 myid
Increase 1     CXFFCFCF myid
Increase 2     CXFF7F7F myid
Increase 3     CXEF0000 myid
Decrease 1     CXC8FFC8 myid
Decrease 2     CX4FFF4F myid
Decrease 3     CX00CF00 myid
;
RUN;
```

We can now use the attribute dataset in the PROC SGPLOT code to add colors to our graph. The results are shown in Figure 6.

```
PROC SGPLOT DATA = Example_5 DATTRMAP = MyAttrMap;
      BAND X = sys LOWER = losys UPPER = hisys / GROUP = grp NAME = "bp"
      ATTRID = myid;
      SCATTER X = X Y = Y / DATALABEL = Patient;
RUN;
```

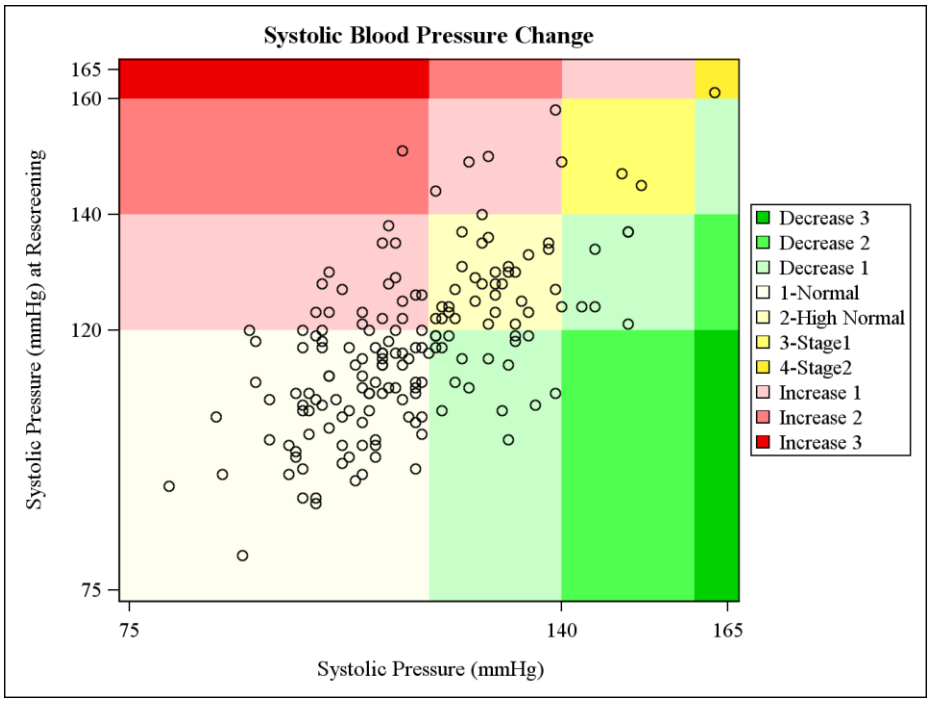
FIGURE 6



Adding in Real Patient Data

Now that the banding is completed and the coloring is suggestive of the information to be displayed, the sample data is exchanged for the real patient data. The final graphic, Figure 7, shows the true change in Systolic Blood Pressure in patients within the "Programa Puente" Program.

FIGURE 7



The graphic above helps decision makers quickly understand what part of the patient population is decreasing their blood pressure and by how much, as well as how many patients are staying roughly the same or increasing. Because the process to create this graphic is time consuming and may be different for different types of categorical data, we have developed a macro for users so that they may easily make their own versions of our Scatter – Band Plot.

Macro Creation – Pre vs. Post

The macro described below is a simple and easy way for users to create a Scatter – Band plot that will graph the values of a Pre vs. Post measurement, the code in its entirety is in Appendix I. Any type of categorical data can be used in this macro, here we will use categorical and numeric measures of diastolic blood pressure. Later on, the paper will discuss other versions of the Scatter – Band Plot.

```
DATA parameters;
      INPUT low high class $ 32.;
CARDS;
0 80 Normal
80 90 Prehypertension
90 100 Stage1
100 120 Stage2
120 . Emergency
;
RUN;
```

The parameters data step contains the class names, and the lower and upper limit of each class. The user must input their desired classes and limits. Changing the names of the variables low, high, or class will cause errors in the macro. The user can specify as few as two or as many as nine classes. Within each class, the value of low must be less than the value of high. Both the first value of low and the last value of high can be left as a '.' to include any data points from negative to positive infinity. The macro will choose appropriate end points for these groups based on the data minimum and maximum values. High and low are numeric variables, and class is qualitative. The macro will automatically calculate the minimum low and the maximum high based on the data.

```
%LET title=Diastolic Blood Pressure Change;
%LET labely=Diastolic Pressure (mmHg) at Rescreening;
%LET labelx=Diastolic Pressure (mmHg);
DATA bp;
      SET health.dbp_sbp;
      %LET scatter1=dbp; %LET scatter2=dbp_rscrn;
RUN;
```

If the user wishes to define a more descriptive title or axis labels they may do so using the %LET statements above. If left blank, the graphic will have no title or axis labels.

The bp data step contains the data to be classified. The data set must contain two numeric variables that will be plotted against each other. The user must assign the two variables of interest to the macro variables scatter1 and scatter2. In this case we chose to use diastolic blood pressure (pre) and diastolic blood pressure as a rescreening (post).

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
```

The macro %GRAPH contains five parameters: the first is the parameter data set, the second is the numeric scatter plot data, and the final three are optional color choices. The user can choose up to three colors for the graph by entering a number 1 through 9 (1=Red 2=Yellow and Orange 3=Green 4=Blue 5=Purple 6=Grey 7=Rainbow 8=Springtime 9=Storm); grey is the default color base. All gradients are displayed from light to dark but can be used in reverse if the negative color base number is specified. Entering a number less than negative nine or greater than nine for a colorbase will cause errors. If a zero is entered the macro will treat it as a missing colorbase. If less than three colors are needed, the user should enter a '.' for the additional color bases. For example, %GRAPH(perm,data,1,4,.) will create a graph using Red and Blue as the colors. Failure to enter a '.' for a missing color base will cause errors in the macro.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
%MACRO SORT(data,var,nodupkey,decending);
      PROC SORT DATA=&data &nodupkey;
                  BY &decending &var;
      RUN;
%MEND;
…
%MEND;
```

The sort macro allows us to shorten the code needed to sort a dataset. The data option specifies the dataset to be sorted and var option specifies the variable on which the sort ordering will be made. In addition, the user may choose to remove duplicate observations in the sorting variable or sort the dataset in descending order.

It should be noted that nesting macros has been known to lengthen processing time because the inner macro is redefined every time the outer macro is called. However, because nested macros allow for greater programming flexibility and because the nested macros used here are fairly short, we have decided not to define them outside the %GRAPH macros.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA order;
      SET &perm;
      class=CAT(_N_,'-',class);
      IF low^=LAG(high) THEN LOW=LAG(high);
RUN;
DATA _NULL_;
      SET order(KEEP=Class OBS=1);
      CALL SYMPUT ('class', INPUT (class, $32. ) );
RUN;
…
%MEND;
```

The order data step cleans the parameter data set by numbering the classes for ease of sorting and by ensuring that the lower bound of one class is equal to the upper bound of the previous class. If the lower and upper bounds are not equal the macro corrects the lower bound. The macro variable class will be used later to assign a group name to the scatter plot observations.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA nomiss;
      SET &data;
      IF &scatter2=. OR &scatter1=. THEN DELETE;
RUN;
PROC MEANS DATA=nomiss MIN MAX NOPRINT;
      VAR &scatter1 &scatter2;
      OUTPUT OUT=data_means MIN(&scatter1 &scatter2)=mina minb
      MAX(&scatter1 &scatter2)=maxa maxb;
RUN;
DATA data_limits;
      SET data_means(DROP=_TYPE_ _FREQ_);
            datamin=MIN(mina,minb);
            datamax=MAX(maxa,maxb);
            datamin=ROUND((datamin-10),10);
            datamax=ROUND((datamax+10),10);
      KEEP datamin datamax;
RUN;

DATA _NULL_;
      SET data_limits;
      CALL SYMPUT ('datamin', INPUT (datamin, BEST. ) );
      CALL SYMPUT ('datamax', INPUT (datamax, BEST. ) );
RUN;
…
%MEND;
```

The nomiss data step deletes data points that have missing data for one or both scatter variables. After running PROC MEANS to obtain the data minimum and maximum values for the scatter variables, the highest maximum and the lowest minimum is kept and rounded to the nearest ten. We then add and subtract ten from the minimum and maximum respectively to ensure no data points are omitted from the graph. The data minimum and maximum are later assigned to macro variables that will define the graph limits.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA one;
….
      IF low<min THEN low=min;
      IF high=. OR high>max THEN high=max;
      diff=high-low;
            IF diff<0 THEN DO;
                  %PUT One or more classes may have been deleted due to lack
of data;
                  DELETE;
            END;
…
RUN;
…
%MEND;
```

The code, found in the data step one, replaces the user entered class limits with the data minimum and maximum values where appropriate. If the first or last class defined is found to have no data in it, that class is deleted and a warning is outputted to the log. Middle classes will be displayed regardless of a lack of data.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
%MACRO BLOWUP(numclass);
      DATA MultiSet;
            SET %DO I = 1 %TO &numclass;
            one(KEEP=x) %END;;
      RUN;
%MEND;
%BLOWUP(&numclass);

DATA three;
      SET one nodupkey;
      DO I= 1 TO (2*&numclass);
            OUTPUT;
      END;
RUN;
…
%MEND;
```

After the macro variable numclass has been defined as the number of classes, the macro %BLOWUP is created to set the x variable to MultiSet as many times as there are classes. Data step three then expands the low and high variables to double the number of classes. This is the first step in setting up the data set which will define the colored bands in the final graph.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA four;
      RETAIN x low high class;
      MERGE three MultiSet;
      value=I;
      DROP I;
            IF MOD(value,2)=1 THEN value=value+1;
            value=value/2;
RUN;
DATA final;
      SET four;
      BY class;
      RETAIN count;
      IF FIRST.class THEN count+1;
            value=value-count;
            groupnum=0-value;
            group_num=(SQRT(groupnum*groupnum));
            LENGTH group $32.;
                  IF groupnum<0 THEN group=CAT('Decrease ', group_num);
                  ELSE IF groupnum>0 THEN group=cat('Increase ', group_num);
                  ELSE IF groupnum=0 THEN group=class;
      KEEP x low high group groupnum;
RUN;
…
%MEND;
```

In data step four, the value variable is defined using the do loop variable I, and adjusted to reflect the order of the bands along the horizontal axis within each of the classes. In data step final, the count variable reflects the order of the classes. Using the value and count variables, we are able to manipulate the data so that the groupnum variable is a numerical representation of the degree to which values of the scatter variables differ from each other. The group names are assigned by concatenating the words

'Increase" or 'Decrease' with a positive version of the groupnum variable. If the scatter variables do not differ, the group name is equal to the user inputted class name.

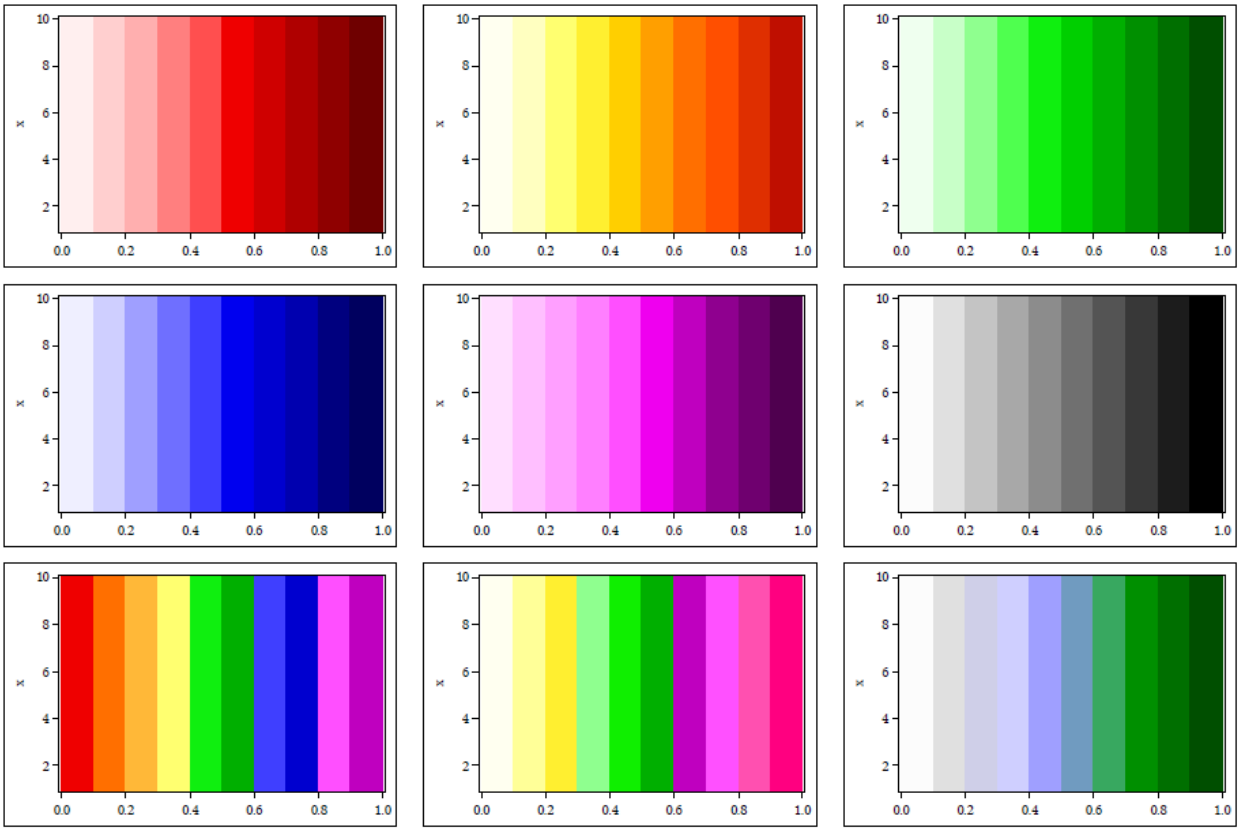
```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA scatter;
      MERGE final(DROP=groupnum) nomiss;
      IF group='' THEN group="&class";
RUN;
…
%MEND;
```

The scatter data set contains all variables needed to construct the final graph. The class group assigned to missing group values does not affect which group a data point is assigned to. It is simply there to create consistency in the graph's colors. The final set of variables in the scatter dataset are x, low, high, group, and the two scatter variables. The difference between the x variables of two observations specifies the width of the bands, while the low and high variables define the height. The group variable contains the group name associated with each observation, and the scatter variables contain the data points needed to construct the scatterplot.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
%IF &colorbase1=0 %THEN %LET colorbase1=.;
%IF &colorbase2=0 %THEN %LET colorbase2=.;
%IF &colorbase3=0 %THEN %LET colorbase3=.;
%IF &colorbase1=. %THEN %DO;
      %LET colorbase1=6;
      %LET colorbase2=6;
      %LET colorbase3=6;
%END;
%IF &colorbase2=. %THEN %LET colorbase2=&colorbase1;
%IF &colorbase3=. %THEN %LET colorbase3=&colorbase2;

%IF &colorbase1<0 %THEN %LET colorbase1_2=(&colorbase1*-1);
%ELSE %LET colorbase1_2=&colorbase1;
%IF &colorbase2<0 %THEN %LET colorbase2_2=(&colorbase2*-1);
%ELSE %LET colorbase2 2=&colorbase2;
%IF &colorbase3<0 %THEN %LET colorbase3_2=(&colorbase3*-1);
%ELSE %LET colorbase3_2=&colorbase3;
…
%MEND;
```

Here the user's color choices are validated. If no colors are specified, the default for all three colors is gray. If one color is specified, that color will be used instead of gray. If two colors are specified, the third color will be the same as the second.Colorbase1_2, colorbase2_2 and colorbase3_2 are positive versions of the user specified color bases and will be used when creating the graph attribute map because negative numbers are not supported. For our graph, we decided on a stop light theme for indicating diastolic blood pressure levels, where green indicates a decrease in blood pressure at the rescreening, red is an increase, and yellow is no change. A health care professional would be able to see at a glance that data points in the green are good and points in the red indicate a problem.

FIGURE 8



There are six different, single colored gradients, each containing ten shades of a base color. There are also three multicolor gradients meant to give a graph more color when only one gradient is required. The six base colors are: red, yellow and orange, green, blue, purple, and gray. The multicolor gradients are rainbow, springtime, and storm. The color schemes are displayed above. All gradients were created from lightest to darkest shades; the user can reverse the gradient by entering the color base number as a negative.

The gradients were created by using the SAS RBG color code, where each color is defined by its red, green, and blue components; the components are added together to create the final color. For each gradient, the code for the base color was found using SAS documentation and we could then decrease or increase the color amounts to obtain lighter or darker shades of the base color. Once the ten colors were specified, they could be adjusted to create a more visually appealing final gradient. The codes for the base colors are; RED=CXFF0000, YELLOW=CXFFFF00, ORANGE=CXFF6F00, GREEN=CX00FF00, BLUE=CX0000FF, PURPLE=CXFF00FF, GRAY=GRAY70.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA colors;
      color1='CXFFEFEF';  color2='CXFFCFCF';  color3='CXFFAFAF';
color4='CXFF7F7F';
      …
      color85='CX9F9FFF'; color86='CX709BC0'; color87='CX38A860';
color88='CX008F00';
      color89='CX006F00'; color90='CX004F00';
RUN;
PROC TRANSPOSE DATA=colors OUT=colors NAME=colors;
      VAR color1-color90;
RUN;
…
%MEND;
```

The color data step contains the code for all the colors displayed in the gradients mentioned above. Within macros, statements like DATALINES; and CARDS: are not supported so we use PROC TRANSPOSE to make the dataset 90x1 instead of 1x90.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA color_1;
      SET assign_colors(WHERE=(&colorbase1_2=color));
      color_base=1;
RUN;
%IF &colorbase1<0 %THEN %DO;
      %SORT(color_1,colors, ,decending);
%END;

…
DATA choose_colors;
      SET color_1 color_2 color_3;
RUN;
…
%MEND;
```

The three colors chosen by the user are assigned to separate data sets and given a color base variable equal to their matching color base macro variable. This helps preserve the correct number of colors when less than three colors are specified. If the user specified color base was negative the data is sorted in descending order and the colors will be used from darkest to lightest. The three color datasets are then recombined in the choose_colors data step.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
%IF &colorsum<&numboxes %THEN %PUT Graph requires more colors than specified.
Please input additional color bases or reduce the number of classes;
…
%MEND;
```

If the total number of available colors in the choose_colors data is less than the total number of colors needed to accurately display the graph, a warning is outputted to the log for the user to input more color choices or reduce the number of classes. The graph will still be displayed with SAS calling on its default coloring scheme to fill in the bands without a defined color.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
%IF &colorbase1=&colorbase2 %THEN %DO;
      DATA final_colors;
            SET choose_colors;
            BY color_base;
            IF FIRST.color_base THEN count=0;
            count+1;
                  IF color_base=1 THEN DO;
                        IF count>&numclass THEN DELETE;
                  END;
                  ELSE IF color_base=2 THEN DO;
                        IF count<=&numclass THEN DELETE;
                        IF count>=(2*&numclass) THEN DELETE;
                  END;
                  ELSE IF color_base=3 THEN DO;
                        IF count<=&numclass THEN DELETE;
                        IF count>=(2*&numclass) THEN DELETE;
                  END;
      RUN;
%END;
…
%MEND;
```

When one or no colors are specified, the first data step is executed. The colors dataset is trimmed to the correct number of colors for the GRAPH using the number of classes and the count variable. The no change groups are colored with the first part of the gradient and the remaining colors are shared by the increase and decrease groups. Each gradient only contains ten unique colors so data with more than five groups will not be colored correctly.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
%IF &colorbase1^=&colorbase2 %THEN %DO;
      %IF &numclass>5 %THEN %DO;
            …
      %END;
      %IF &numclass<=5 %THEN %DO;
            DATA less_colors;
                  SET choose_colors;
                  BY color_base;
                  IF FIRST.color_base THEN count1=0;
                  count1+1;
                  onlyeven=MOD(count1,2);
                        IF color_base^=1 THEN DO;
                              IF onlyeven ^= 0 THEN DELETE;
                        END;
            RUN;
            DATA final_colors;
                  SET less_colors;
                  BY color_base;
            …
            RUN;
      %END;
%END;
…
%MEND;
```

If two or more colors are specified, then the second data step is executed. When more than five classes

are specified the colors are used in user specified order for each colorbase. If five or less classes are specified the number of colors is reduced by deleting the odd numbered colors in the less_colors data step before the final_colors step is executed; this helps make the colored bands in the final graph easier to differentiate.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA myattrmap;
      RETAIN ID value fillcolor;
      ID='myid';
      MERGE color_values(RENAME=(group=value)) final_colors(KEEP=COL1
RENAME=(COL1=fillcolor));
      IF value='' THEN DELETE;
      KEEP ID value fillcolor;
RUN;
…
%MEND;
```

An attribute map is then created to use in the final GRAPH. The color_values data set contains the names of each group and the final_colors data contains the codes for the colors chosen by the user. Using an attribute map allows users to utilize more colors than assigning colors in PROC SGPLOT.

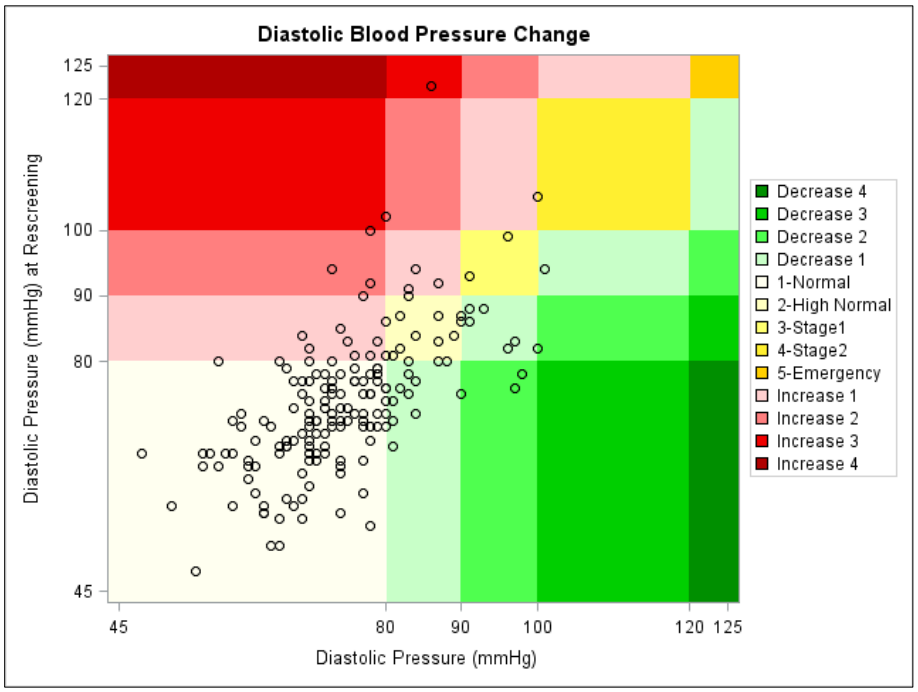
```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
      DATA axis_values;
            MERGE order data_limits;
            datamin2=datamin+5; datamax2=datamax-5;
            …
            KEEP x;
      RUN;
      PROC SQL NOPRINT;
        SELECT x
        INTO :list SEPARATED BY ' '
        FROM axis_values;
      QUIT;
      %MACRO AXIS(list);
            XAXIS VALUES=(&list) LABEL="&labelx";
            YAXIS VALUES=(&list) LABEL="&labely";
      %MEND;
…
%MEND;
```

The axis_values data step is very similar to data step one, but instead of changing the datamin and datamax by ten, we change them by five; this makes the final graph display more visually appealing. Using the axis_values data step and PROC SQL, we create a list of numbers that define the cutoff points of each class. This list variable is then used in the %AXIS macro to create the labels for both the x and y axis in the final graph.

```
%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);
…
PROC SGPLOT DATA=scatter DATTRMAP=myattrmap;
   BAND X=x LOWER=low UPPER=high /
        GROUP=group NAME="groups" ATTRID=myid;
   SCATTER Y = &scatter2 X = &scatter1/MARKERATTRS=(COLOR=black);
   %AXIS(&list);
   KEYLEGEND "groups" / POSITION=right ACROSS=1;
   TITLE "&title ";
RUN;
…
%MEND;
```

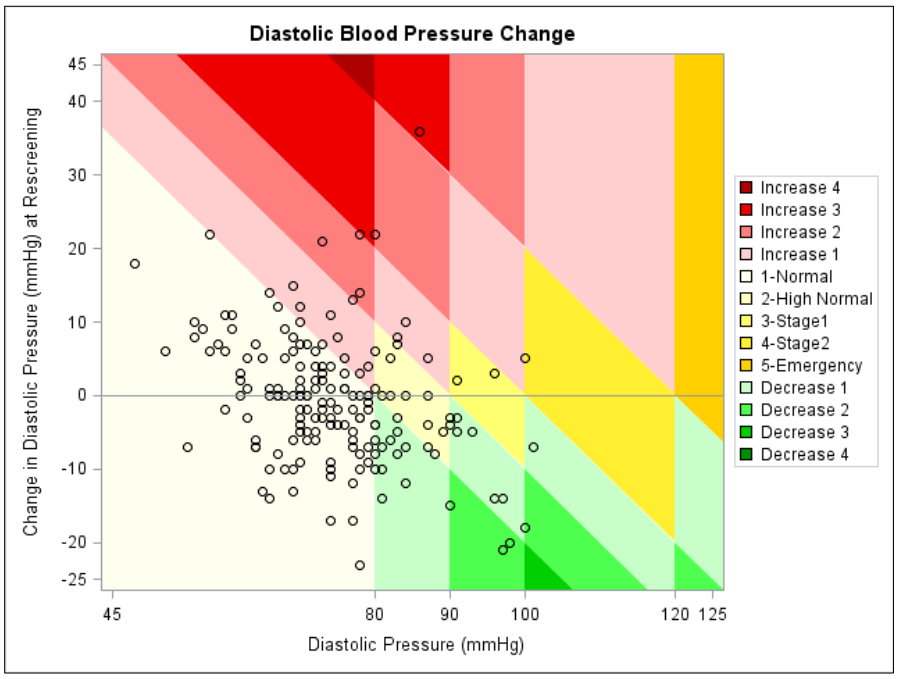
Finally we code for the actual graph as seen below, specifying the scatter data set and the attribute map for the colors. The legend is placed in a vertical box to the right of the graph; by default it would appear in a horizontal box under the graph. The data points appear as black, open circles. The graph below is similar to the graph in Figure 7 except it uses measurements of diastolic blood pressure at an initial screening and a rescreening instead of systolic.

FIGURE 9



Initial Screening vs. Change

FIGURE 10



The graphic above displays how much a patient's diastolic blood pressure reading changes from initial to rescreening measurement. This graphic allows decision makers to quickly identify the numeric magnitude and direction of the changes, and what category the reading is at the initial screening and rescreening. Our second macro described below creates a Scatter – Band plot that will graph the values of a Pre vs. Change measurement. As before, the x variable will be the diastolic blood pressure reading at the initial screening. The y variable will be the change in diastolic blood pressure between the initial and rescreening measurements. The change in diastolic blood pressure can be positive or negative so the graph includes a reference line at y=0. Like the previous graph, vertical lines mark the boundaries between the diastolic blood pressure categories. The lines for change variable are diagonal instead of horizontal because the magnitude of change required to move from one category to another depends on the initial categorical and numeric measurement. The lines are found using a system of regression equations to solve for values of y at each categorical boundary for diastolic blood pressure. The black circles represent the patient's initial diastolic blood pressure and change at rescreening while the colored bands indicate the categorical change in the patient's diastolic reading.

To simplify the process for users, a macro has been developed that will allow users to create this graphic. This macro's code in its entirety is in Appendix II.

Macro Creation – Initial Screening vs. Change:

```
%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA one;
      RETAIN y;
      …
      y+diff;
RUN;
…
%MEND;
```

This data step one is almost exactly like the data one in the previous macro with the exception of the y variable, which is the sum of the differences between the low and high variables. This y will be essential in constructing the regression lines in the final graph.

```
%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);
…
PROC MEANS DATA=nomiss MIN MAX NOPRINT;
      VAR difference;
      OUTPUT OUT=deff_means MAX=maxdiff MIN=negdiff;
RUN;
DATA data_limits2;
      SET deff_means(DROP=_TYPE_ _FREQ_);
      maxdiff=ROUND((maxdiff+10),10);
      negdiff=ROUND((negdiff-10),10);
      KEEP maxdiff negdiff;
RUN;
…
%MEND;
```

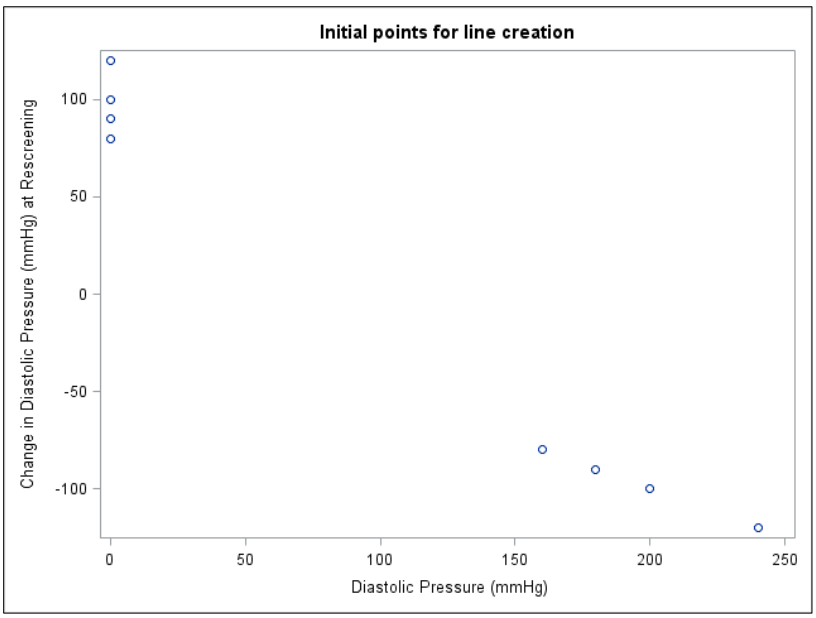
Like the previous macro, this macro uses PROC MEANS to create macro variables for the maximum and minimum data values. Unlike the previous macro, here we also create macro variables for the minimum and maximum values of the difference variable, which is calculated from subtraction of the scatter variables. These two macro variables will be useful for defining the graph axes later.

```
%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA y1;
      SET one(KEEP=y);
      y=y+&datamin;
      x=0;
      IF _N_>=&numclass THEN DELETE;
RUN;
…
DATA y2;
      SET one(KEEP=high y RENAME=(high=x));
      y=(y+&datamin)*-1;
      x=x*2;
      IF _N_>=&numclass THEN DELETE;
RUN;
…
%MEND;
```

Now we will create two data points for each of diagonal reference lines. We will need one less line than the number of classes specified; the line `IF _N_>=&numclass THEN DELETE;` ensures that the

correct number of regression lines will be created. Each data point will be found from the user entered class boundaries. The y1 data step uses the y variable added to the data minimum to create the y intercept of the regression lines. The y2 data step uses the y and high variables (defined as y and x respectively) to create the second set of data points for each line. The data points from y1 and y2 are displayed in the graph below.

FIGURE 11



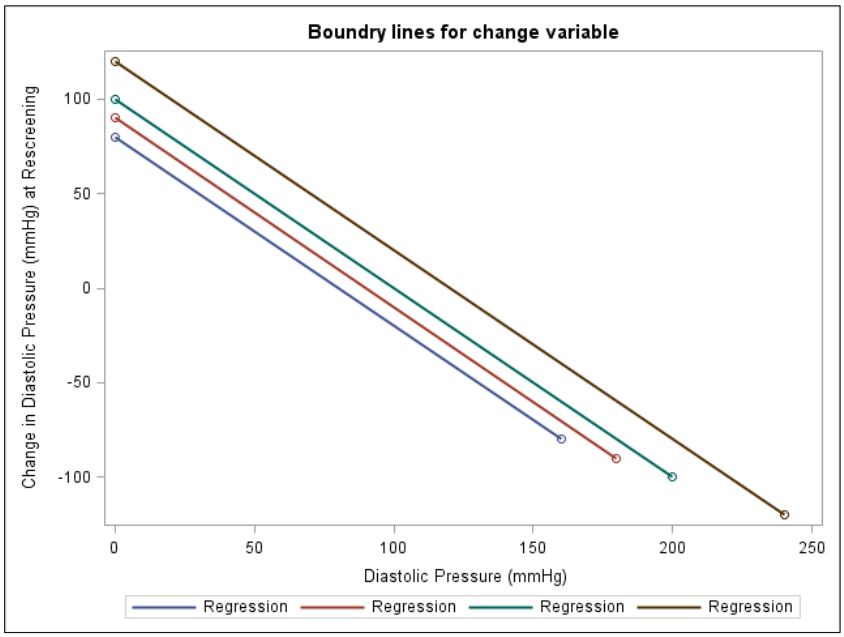

```
%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA ys;
      SET y_1 y_2;
RUN;
DATA xs;
      SET x_1 x_2;
RUN;
…
DATA TWO;
      SET xs mid(KEEP=x);
      ARRAY xcoord _ALL_;
      DO OVER xcoord;
            IF _N_>2 then xcoord=x;
      END;
      DROP x;
RUN;
…
%MEND;
```

After the data points are transposed, the ys dataset contains only the y values of the data points and xs contains only the x values. In data step two, the x values for the regression lines are combined with the x values for the final graph. When the equations for the lines are calculated, a predicted value of y will be found for each of the graphing x variables.

```
%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);
…
%MACRO LINES;
      %DO I=1 %TO &oneless;
            MODEL y_&i=x_&i;
            OUTPUT OUT=pred&i P=pred;
      %END;
%MEND;
PROC REG DATA=three NOPRINT;
      %LINES;
RUN;
QUIT;
…
%MEND;
```

The %LINES macro runs the regression for each set of xy data points and outputs a data set containing the predicted values of the x graphing variables. The graph below shows the final lines that will define the categorical boundaries of the change variable. Now we have the predicted values of y for each of the class boundaries for diastolic blood pressure, we will be able to use two observations to define and categorize each colored band in the final graphic.

FIGURE 12




```
%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);
…
%MACRO MULTIPERD;
 DATA Multiperd;
 set
 %DO I = 1 %TO &oneless;
 pred&i(KEEP=pred firstobs=3) %END;;
 RUN;
 %MEND; %MULTIPERD;
```

```
%MACRO MULTIX;
 DATA Multix;
 SET
 %DO I = 1 %TO &oneless;
 mid(KEEP=x class) %END;;
 RUN;
 %MEND; %MULTIX;
…
%MEND;
```

The %MULTIPERD and %MULTIX macros are similar to the %BLOWUP macro seen in the previous graph macro. In %MULTIPERD each data set containing the predicted values of x is assigned to the multiperd data set. The first observation is set to three because the first two values are the x coordinates for the regression line, not the predicted values. In %MULTIX the graphing x variable and the class names are set to the Multix data set exactly one less time than the number of classes.

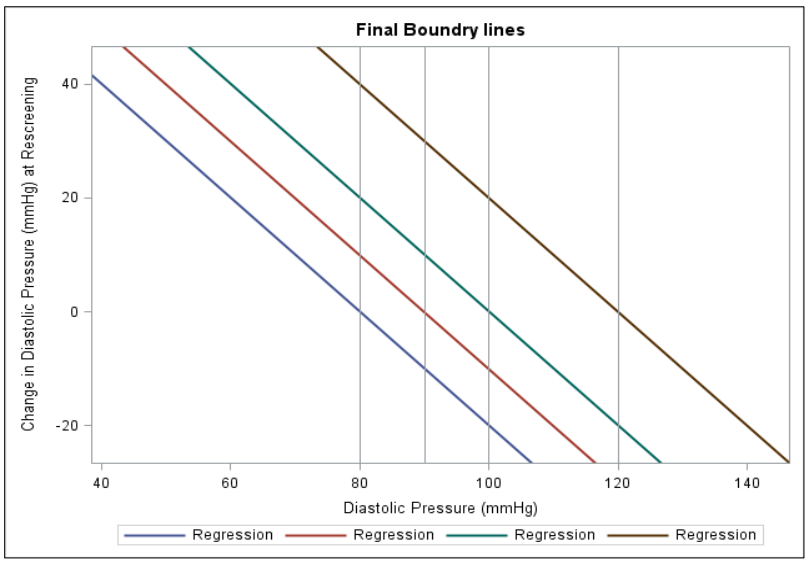
```
%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA mindiff;
      RETAIN pred;
      SET mid(KEEP=x class);
      IF _N_=1 THEN pred=&negdiff;
RUN;
DATA maxdiff;
      RETAIN pred;
      SET mid(KEEP=x class);
      IF _N_=1 THEN pred=&maxdiff;
RUN;
DATA lows;
      RETAIN x;
      SET mindiff multiboth;
RUN;
DATA highs;
      RETAIN x;
      SET multiboth maxdiff;
RUN;
…
%MEND;
```

The min and maxdiff data sets contain the values for the most extreme groups in the final graph. For both groups, the pred variable is set to equal the most extreme predicted value. The min and max data sets are then added to the beginning and end of the predicted data set respectively. Within these data sets, the perd variable now contains all the necessary values for the low and high variables.

```
%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA four;
RETAIN x low high class value;
      MERGE lows(RENAME=(pred=low)) highs(RENAME=(pred=high));
      LABEL low='low';
      LABEL high='high';
      IF low>high THEN low=high;
            IF x=&datamin THEN value=0;
            value+1;
RUN;
…
%MEND;
```

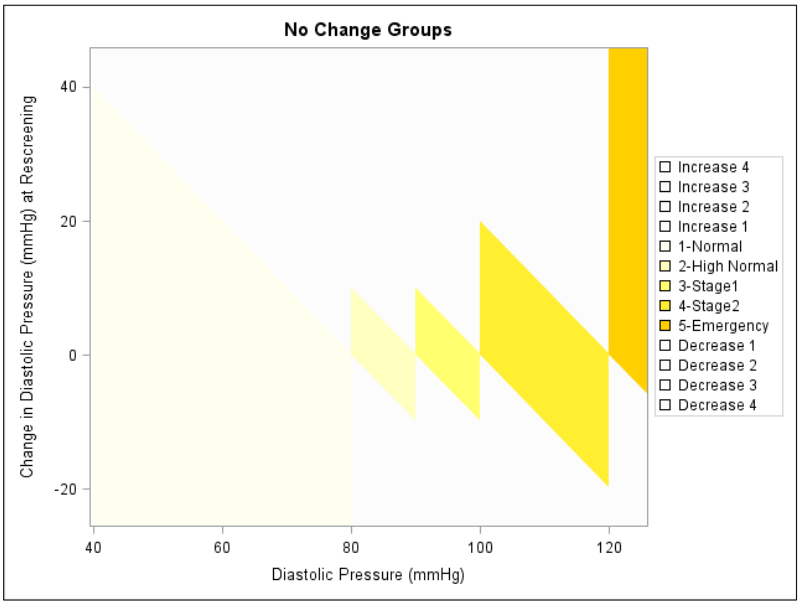
In data step four, the data sets containing all the low and high values are renamed and merged. The data set now contains all the numeric variables needed to create the bands in the final graph. The final, uncolored bands are displayed in the graph below. The intersection of the vertical and diagonal lines represent the different categories of diastolic blood pressure and the possible changes between the initial and rescreening measurements.

FIGURE 13



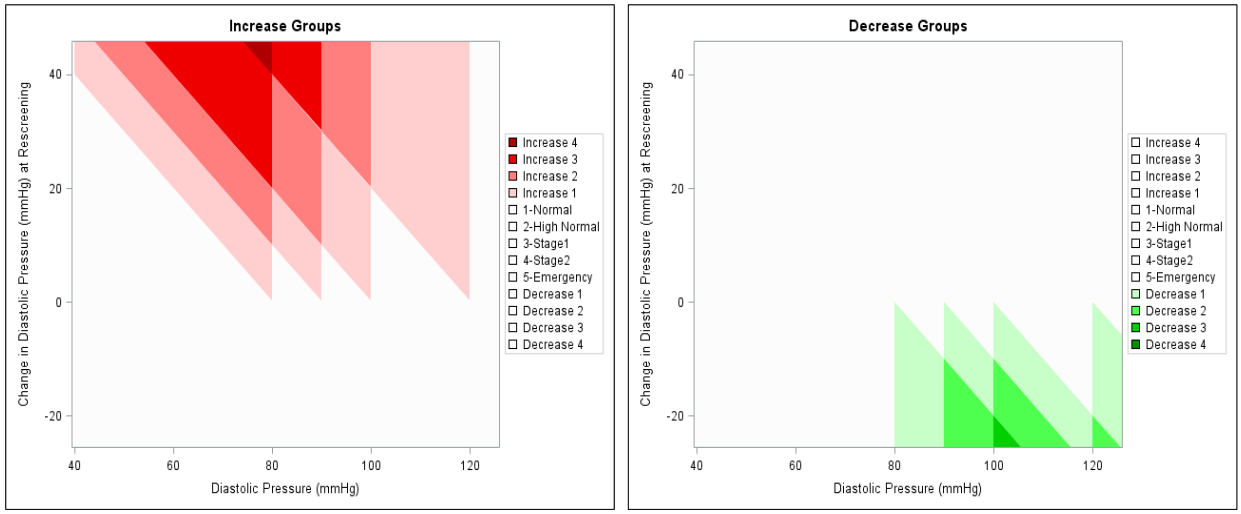
We chose to use a coloring scheme similar to the one used in the original macro graph; shades of yellow would represent patient's whose category did not change between their screening and rescreening while shades of red and shades of green would represent patients who increased and decreased respectively.

FIGURE 14



In Figure 14 the no change groups for diastolic blood pressure have been highlighted in shades of yellow. Like before, the groups on the diagonal are the patients who did not shift categories between their screening and rescreening measurements; however because we are using a change variable instead of the actual rescreening value the no change diagonal is tilted until it is parallel and centered around the y=0 axis. Figures 15 and 16 below show the increase groups in red and the decrease groups in green. The code for selecting these colors exactly mirrors the coloring code discussed in macro 1.

FIGURE 15 AND 16




```
%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);
…
DATA xaxis_values;
      MERGE order data_limits;
      …
      KEEP x;
RUN;
%SORT(xaxis_values,x,nodupkey, );
PROC SQL NOPRINT;
  SELECT x
  INTO :listx SEPARATED BY ' '
  FROM xaxis_values;
QUIT;
%MACRO XAXIS(listx);
      XAXIS VALUES=(&listx) LABEL="&labelx";
%MEND;

DATA yaxis_values;
      MERGE order data_limits;
      …
      KEEP pred;
RUN;
%SORT(yaxis_values,pred,nodupkey, );
PROC SQL NOPRINT;
  SELECT pred
  INTO :listy SEPARATED BY ' '
  FROM yaxis_values;
QUIT;
```

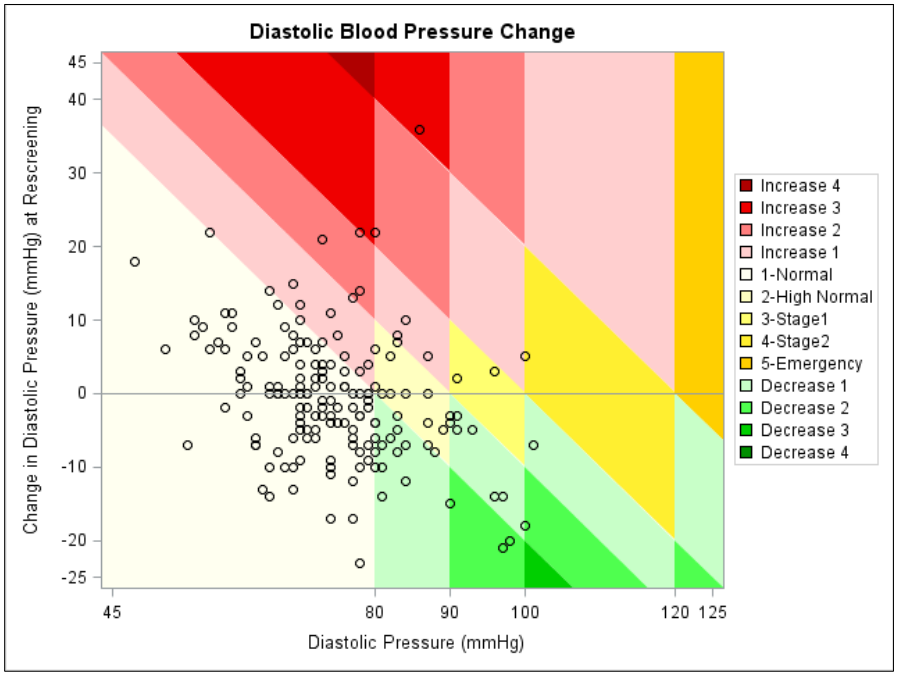
```
%MACRO YAXIS(listy);
      YAXIS VALUES=(&listy) LABEL="&labely";
%MEND;
…
%MEND;
```

Finding the axis values is similar to the previous macro but because this GRAPH is more complicated, we have to find and output the values for the x and y axis separately. The final %XAXIS and %YAXIS macros define the axis labels in the final graph very similarly to the %AXIS macro in the previous graph.

```
%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);
…
PROC SGPLOT DATA=scatter DATTRMAP=myattrmap;
   BAND X=x LOWER=low UPPER=high /
        GROUP=group NAME="groups" ATTRID=myid;
   SCATTER Y = Difference X = &scatter1/MARKERATTRS=(COLOR=black);refline 0;
   %XAXIS(&listx); %YAXIS(&listy);
   KEYLEGEND "groups" / POSITION=right ACROSS=1;
   TITLE "&title ";
RUN;
…
%MEND;
```

The code for the final graph is very similar to the PROC SGPLOT code in the previous macro. One difference is refline 0; which repositions the GRAPH so the x=0 axis appears in the middle of the graph instead of along the bottom. The other is the addition of the %XAXIS and %YAXIS macros to precisely define the limits of the graph. The final result is seen in figure 10 as shown below.

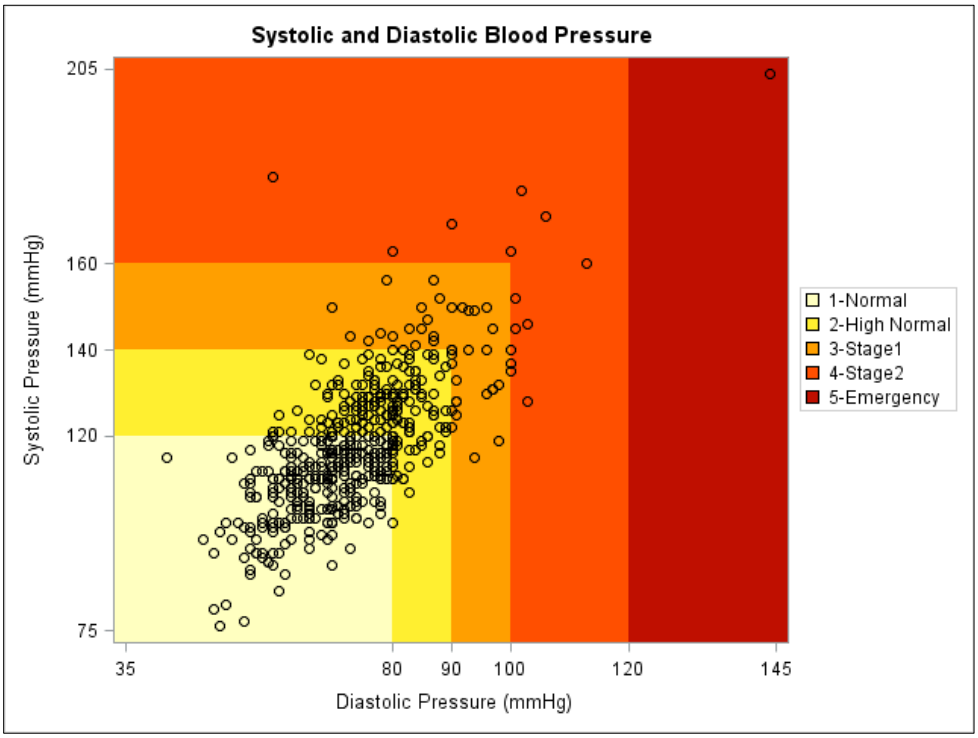
FIGURE 10



Macro – Two Different Categories:

The graphic below has been designed to help a decision maker to quickly evaluate a patient's health status based on both their systolic and diastolic blood pressure readings. Our third macro described below creates a Scatter – Band plot that will graph the values of Systolic vs. Diastolic blood pressure, this macro's code in its entirety is in Appendix III. The x variable will be the diastolic blood pressure and the y variable will be the systolic blood pressure. Each colored band represents a blood pressure category and the black circles represent the patient's numeric systolic and diastolic measurement. The construction and interpretation of this graph are very similar to the initial macro graph.

FIGURE 17




```
DATA peramaters;
      INPUT lowx highx lowy highy class $ 32.;
CARDS;
0 80 0 120 Normal
80 90 120 140 Prehypertension
90 100 140 160 Stage1
100 120 160 210 Stage2
120 . 210 . Emergency
;
RUN;
```

The parameters data step is slightly different than the previous macros in that it contains the class names, and the lower and upper class limits for both the diastolic and systolic blood pressures. The variables names ending with x correspond to the first graphic variable (in this case diastolic) and the y variables correspond the second (systolic). Both variables must have an equal number of classes and the classes must share a name.

```
%MACRO GRAPH3(perm,data,colorbase);
```

The macro %GRAPH3 only calls for one color base instead of three. The color base numbers are defined the same way.

```
%MACRO GRAPH3(perm,data,colorbase);
…

DATA vertical;
      SET one(RENAME=(lowy=low highy=high class=group) FIRSTOBS=3);
            low=&dataminy;
RUN;
PROC SORT DATA=one NODUPKEY OUT=one_nodup;
      BY x;
RUN;
DATA horizontal;
      SET one_nodup(RENAME=(lowy=low highy=high class=group) DROP=x
FIRSTOBS=2)
      one_nodup(RENAME=(lowy=low highy=high class=group) FIRSTOBS=2);
            IF x=. THEN x=&dataminx;
RUN;
%SORT(horizontal,low, ,)
…
DATA final;
      SET horizontal vertical;
RUN;

…
%MEND;
```

The horizontal and vertical bars are defined from the data step one. They are then combined to create the final dataset. The data now contains all the numeric variables needed to create the bands in the final graph.

```
%MACRO GRAPH3(perm,data,colorbase);
…

%IF &numclass>5 %THEN %DO;
      DATA final_colors;
            SET choose_colors;
            IF _N_>&numclass THEN DELETE;
      RUN;
%END;
%IF &numclass<=5 %THEN %DO;
      DATA less_colors;
            SET choose colors;
            onlyeven=MOD(_N_,2);
                  IF onlyeven ^= 0 THEN delete;
      RUN;
      DATA final_colors;
            SET less_colors;
                  IF _N_>&numclass THEN DELETE;
      RUN;
%END;
```

```
...
%MEND;
```

When five or fewer classes are specified, every other color is selected from the chosen gradient. Unlike the other macros, here we use the built-in _N_ variable instead of a count variable because there is only one color base.

Conclusion

Bi-variate graphics with a corresponding set of easy to customize macros gives users another tool to visually portray data that has both qualitative and quantitative aspects. For health care professionals, this graphic has been a very important tool to help decision makers view on a much larger scale, how populations are shifting and by how much. As with most visual displays, tables are also useful in conjunction with graphics; however, our graphics succinctly capture the overall shift in the population, the degree of the shift, and where different subgroups of patients fall on the graphic.

References:

Mofeel SAS community. (2004, November). Setting dataset multiple times . Retrieved April, 2014, from http://www.mofeel.net/1169-comp-soft-sys-sas/15045.aspx

Sample 44184: Add a gradient shaded background to SAS/GRAPH. (2011, September 1). In *SAS*. Retrieved April, 2014, from http://support.sas.com/kb/44/184.html

Saughter, S. J., & Delwiche, L. D. (2004). SAS macro programming for beginners. Retrieved April, 2014, from http://www2.sas.com/proceedings/sugi29/243-29.pdf

Stroupe, J. (2003). Nine Steps to Get Started using SAS® Macros. Retrieved April, 2014, from http://www2.sas.com/proceedings/sugi28/056-28.pdf

SAS support. (2014). %MACRO statement. In *SAS*. Retrieved April, 2014, from http://support.sas.com

SAS support. (2014). The SGPLOT procedure. In *SAS*. Retrieved April, 2014, from http://support.sas.com

SAS support. (2014). Using SG attribute maps to control visual attributes. In *SAS*. Retrieved April, 2014, from http://support.sas.com

CONTACT INFORMATION

Your comments and questions are much appreciated. Contact the authors at:

Kathryn Schurr, M.S.
Statistical Database Analyst
Spectrum Health Corporate
Healthier Communities
665 Seward Avenue NW, Suite 110
Grand Rapids, MI 49504
Work Phone: 616.391.2983
Work E-Mail: kathryn.schurr@spectrumhealth.org

APPENDIX I

```
options mlogic mprint;
DATA parameters;
      INPUT low high class $ 32.;
CARDS;
0 80 Normal
80 90 High Normal
90 100 Stage1
100 120 Stage2
120 . Emergency
;
RUN;

%LET title=Diastolic Blood Pressure Change;
%LET labely=Diastolic Pressure (mmHg) at Rescreening;
%LET labelx=Diastolic Pressure (mmHg);
DATA bp;
      SET health.dbp_sbp;
      %LET scatter1=dbp; %LET scatter2=dbp_rscrn;
      if sbp<50 or sbp_rscrn<50 then delete;
RUN;

%MACRO GRAPH(perm,data,colorbase1,colorbase2,colorbase3);

%MACRO SORT(data,var,nodupkey,decending);
      PROC SORT DATA=&data &nodupkey;
                  BY &decending &var;
      RUN;
%MEND;

DATA order;
      SET &perm;
      class=CAT(_N_,'-',class);
      IF low^=LAG(high) THEN LOW=LAG(high);
RUN;
DATA _NULL_;
      SET order(KEEP=Class OBS=1);
      CALL SYMPUT ('class', INPUT (class, $32. ) );
RUN;


DATA nomiss;
      SET &data;
      IF &scatter2=. OR &scatter1=. THEN DELETE;
RUN;

PROC MEANS DATA=nomiss MIN MAX NOPRINT;
      VAR &scatter1 &scatter2;
      OUTPUT OUT=data_means MIN(&scatter1 &scatter2)=mina minb
      MAX(&scatter1 &scatter2)=maxa maxb;
RUN;
DATA data_limits;
```

```
      SET data_means(DROP=_TYPE_ _FREQ_);
            datamin=MIN(mina,minb);
            datamax=MAX(maxa,maxb);
            datamin=ROUND((datamin-10),10);
            datamax=ROUND((datamax+10),10);
      KEEP datamin datamax;
RUN;
DATA _NULL_;
 SET data_limits;
 CALL SYMPUT ('datamin', INPUT (datamin, BEST. ) );
 CALL SYMPUT ('datamax', INPUT (datamax, BEST. ) );
RUN;

DATA one;
      MERGE order data_limits;
            RETAIN min max;
            IF _N_=1 THEN DO;
                  min=datamin;
                  max=datamax;
            END;
            IF low<min THEN low=min;
            IF high=. OR high>max THEN high=max;
            diff=high-low;
                  IF diff<0 THEN DO;
                        %PUT One or more classes may have been deleted
due to lack of data;
                        DELETE;
                  END;
            DO I=low TO high BY diff;
                  x=I;
                  OUTPUT;
            END;
      KEEP x low high class;
RUN;

PROC SORT DATA=one(DROP=x) OUT=one_nodupkey NODUPKEY;
      BY low;
RUN;
DATA classes;
      SET one_nodupkey;
      n=1;
RUN;
PROC MEANS DATA=classes SUM NOPRINT;
      VAR n;
      OUTPUT OUT=sums(DROP= _TYPE_ _FREQ_) SUM=numclass;
RUN;
DATA _NULL_;
      SET sums;
      CALL SYMPUT ('numclass', INPUT (numclass, BEST. ) );
RUN;

%MACRO BLOWUP(numclass);
      DATA MultiSet;
```

```
            SET %DO I = 1 %TO &numclass;
            one(KEEP=x) %END;;
      RUN;
%MEND;
%BLOWUP(&numclass);

DATA three;
      SET one_nodupkey;
      DO I= 1 TO (2*&numclass);
            OUTPUT;
      END;
RUN;

DATA four;
      RETAIN x low high class;
      MERGE three MultiSet;
      value=I;
      DROP I;
            IF MOD(value,2)=1 THEN value=value+1;
            value=value/2;
RUN;
%SORT(four,class, , );

DATA final;
      SET four;
      BY class;
      RETAIN count;
      IF FIRST.class THEN count+1;
            value=value-count;
            groupnum=0-value;
            group_num=(SQRT(groupnum*groupnum));
            LENGTH group $32.;
                  IF groupnum<0 THEN group=CAT('Decrease ', group_num);
                  ELSE IF groupnum>0 THEN group=cat('Increase ',
group_num);
                  ELSE IF groupnum=0 THEN group=class;
            KEEP x low high group groupnum;
RUN;
%SORT(final,groupnum, , );

DATA scatter;
      MERGE final(DROP=groupnum) nomiss;
      IF group='' THEN group="&class";
RUN;

%IF &colorbase1=0 %THEN %LET colorbase1=.;
%IF &colorbase2=0 %THEN %LET colorbase2=.;
%IF &colorbase3=0 %THEN %LET colorbase3=.;
%IF &colorbase1=. %THEN %DO;
      %LET colorbase1=6;
      %LET colorbase2=6;
      %LET colorbase3=6;
%END;
```

```
%IF &colorbase2=. %THEN %LET colorbase2=&colorbase1;
%IF &colorbase3=. %THEN %LET colorbase3=&colorbase2;

%IF &colorbase1<0 %THEN %LET colorbase1_2=(&colorbase1*-1);
%ELSE %LET colorbase1_2=&colorbase1;
%IF &colorbase2<0 %THEN %LET colorbase2_2=(&colorbase2*-1);
%ELSE %LET colorbase2_2=&colorbase2;
%IF &colorbase3<0 %THEN %LET colorbase3_2=(&colorbase3*-1);
%ELSE %LET colorbase3_2=&colorbase3;

PROC SORT DATA=final(DROP=groupnum) OUT=numboxes;
     BY group;
RUN;
DATA numboxes;
     SET numboxes;
     BY group;
     fgroup=FIRST.group;
RUN;
PROC MEANS DATA=numboxes SUM NOPRINT;
     VAR fgroup;
     OUTPUT OUT=sums2(drop= _type_ _freq_) SUM=numboxes;
RUN;
DATA _NULL_;
     SET sums2;
     CALL SYMPUT ('numboxes', INPUT (numboxes, BEST. ) );
RUN;

DATA colors;
     color1='CXFFEFEF';  color2='CXFFCFCF';  color3='CXFFAFAF';
color4='CXFF7F7F';
     color5='CXFF4F4F';  color6='CXEF0000';  color7='CXCF0000';
color8='CXAF0000';
     color9='CX8F0000';  color10='CX6F0000';
     color11='CXFFFFF0'; color12='CXFFFFC0'; color13='CXFFFF70';
color14='CXFFEF30';
     color15='CXFFCF00'; color16='CXFF9F00';     color17='CXFF6F00';
     color18='CXFF4F00';
     color19='CXDF2F00';   color20='CXBF0F00';
     color21='CXEFFFEF'; color22='CXC8FFC8';     color23='CX8FFF8F';
     color24='CX4FFF4F';
     color25='CX0FEF0F';   color26='CX00CF00';   color27='CX00AF00';
     color28='CX008F00';
     color29='CX006F00';   color30='CX004F00';
     color31='CXEFEFFF';   color32='CXCFCFFF';   color33='CX9F9FFF';
     color34='CX6F6FFF';
     color35='CX3F3FFF';   color36='CX0000EF';   color37='CX0000CF';
     color38='CX0000AF';
     color39='CX00007F';   color40='CX00005F';
     color41='CXFFDFFF';   color42='CXFFBFFF';   color43='CXFF9FFF';
     color44='CXFF7FFF';
     color45='CXFF4FFF';   color46='CXEF00EF';   color47='CXBF00BF';
     color48='CX8F008F';
     color49='CX6F006F';   color50='CX4F004F';
```

```
      color51='GRAYFC';     color52='GRAYE0';     color53='GRAYC4';
      color54='GRAYA8';
      color55='GRAY8C';     color56='GRAY70';     color57='GRAY54';
      color58='GRAY38';
      color59='GRAY1C';     color60='GRAY00';
      color61='CXEF0000'; color62='CXFF6F00'; color63='CXFFB838';
color64='CXFFFF70';
      color65='CX0FEF0F'; color66='CX00AF00'; color67='CX3F3FFF';
color68='CX0000CF';
      color69='CXFF4FFF'; color70='CXBF00BF';
      color71='CXFFFFF0'; color72='CXFFFF98'; color73='CXFFEF30';
color74='CX8FFF8F';
      color75='CX0FEF00'; color76='CX00AF00'; color77='CXBF00BF';
color78='CXFF50FF';
      color79='CXFF50B0'; color80='CXFF0080';
      color81='GRAYFC';     color82='GRAYE0';     color83='CXCFCFE8';
color84='CXCFCFFF';
      color85='CX9F9FFF'; color86='CX709BC0'; color87='CX38A860';
color88='CX008F00';
      color89='CX006F00'; color90='CX004F00';
RUN;
PROC TRANSPOSE DATA=colors OUT=colors NAME=colors;
      VAR color1-color90;
RUN;

DATA assign_colors;
      SET colors;
      total=1;
      color=1;
      IF _N_ >10 THEN color=2;
      IF _N_ >20 THEN color=3;
      IF _N_ >30 THEN color=4;
      IF _N_ >40 THEN color=5;
      IF _N_ >50 THEN color=6;
      IF _N_ >60 THEN color=7;
      IF _N_ >70 THEN color=8;
      IF _N_ >80 THEN color=9;
RUN;

DATA color_1;
      SET assign_colors(WHERE=(&colorbase1_2=color));
      color_base=1;
RUN;
%IF &colorbase1<0 %THEN %DO;
      %SORT(color_1,colors, ,decending);
%END;

DATA color_2;
      SET assign_colors(WHERE=(&colorbase2_2=color));
      color_base=2;
RUN;
%IF &colorbase2<0 %THEN %DO;
      %SORT(color_2,colors, ,decending);
```

```
%END;

DATA color_3;
     SET assign_colors(WHERE=(&colorbase3_2=color));
     color_base=3;
RUN;
%IF &colorbase3<0 %THEN %DO;
     %SORT(color_3,colors, ,decending);
%END;

DATA choose_colors;
     SET color_1 color_2 color_3;
RUN;

PROC MEANS DATA=choose_colors NOPRINT SUM;
     VAR total;
     OUTPUT OUT=color_sum(DROP=_TYPE_ _FREQ_) SUM=colorsum;
RUN;
DATA _NULL_;
     SET color_sum;
     CALL SYMPUT ('colorsum', INPUT (colorsum, BEST. ) );
RUN;

%IF &colorsum<&numboxes %THEN %PUT Graph requires more colors than
specified. Please input additional
color bases or reduce the number of classes;

%SORT(choose_colors,color_base, , );
%IF &colorbase1=&colorbase2 %THEN %DO;
     DATA final_colors;
          SET choose_colors;
          BY color_base;
          IF FIRST.color_base THEN count=0;
          count+1;
               IF color_base=1 THEN DO;
                    IF count>&numclass THEN DELETE;
               END;
               ELSE IF color_base=2 THEN DO;
                    IF count<=&numclass THEN DELETE;
                    IF count>=(2*&numclass) THEN DELETE;
               END;
               ELSE IF color_base=3 THEN DO;
                    IF count<=&numclass THEN DELETE;
                    IF count>=(2*&numclass) THEN DELETE;
               END;
     RUN;
%END;
%IF &colorbase1^=&colorbase2 %THEN %DO;
     %IF &numclass>5 %THEN %DO;
          DATA final_colors;
               SET choose_colors;
               BY color_base;
               IF first.color_base THEN count=0;
```

```
                count+1;
                      IF color_base=1 THEN DO;
                            IF count>&numclass THEN DELETE;
                      END;
                      ELSE IF color_base=2 THEN DO;
                            IF count>=&numclass THEN DELETE;
                      END;
                      ELSE IF color_base=3 THEN DO;
                            IF count>=&numclass THEN DELETE;
                      END;
          RUN;
     %END;
     %IF &numclass<=5 %THEN %DO;
          DATA less_colors;
                SET choose_colors;
                BY color_base;
                IF FIRST.color_base THEN count1=0;
                count1+1;
                onlyeven=MOD(count1,2);
                      IF color_base^=1 THEN DO;
                            IF onlyeven ^= 0 THEN DELETE;
                      END;
          RUN;
          DATA final_colors;
                SET less_colors;
                BY color_base;
                IF FIRST.color_base THEN count2=0;
                count2+1;
                      IF color_base=1 THEN DO;
                            IF count2>&numclass THEN DELETE;
                      END;
                      ELSE IF color_base=2 THEN DO;
                            IF count2>=&numclass THEN DELETE;
                      END;
                      ELSE IF color_base=3 THEN DO;
                            IF count2>=&numclass THEN DELETE;
                      END;
          RUN;
     %END;
%END;

%SORT(final_colors,color_base, , );
PROC SORT DATA=final(KEEP=group) OUT=color_values NODUPKEY;
     BY group;
RUN;
DATA myattrmap;
     RETAIN ID value fillcolor;
     ID='myid';
     MERGE color_values(RENAME=(group=value)) final_colors(KEEP=COL1
RENAME=(COL1=fillcolor));
     IF value='' THEN DELETE;
     KEEP ID value fillcolor;
RUN;
```

```
DATA axis_values;
      MERGE order data_limits;
      datamin2=datamin+5; datamax2=datamax-5;
            RETAIN min max;
            IF _N_=1 THEN DO;
                  min=datamin2;
                  max=datamax2;
            END;
            IF low<min THEN low=min; IF low>max THEN low=max;
            IF high^=. AND high<min THEN high=min; IF high=. OR
high>max THEN high=max;
            diff=high-low; IF diff<=0 THEN DELETE;
            DO I=low TO high BY diff;
                  x=I;
                  OUTPUT;
            END;
      KEEP x;
RUN;
%SORT(axis_values,x,nodupkey, );
PROC SQL NOPRINT;
  SELECT x
  INTO :list SEPARATED BY ' '
  FROM axis_values;
QUIT;
%MACRO AXIS(list);
      XAXIS VALUES=(&list) LABEL="&labelx";
      YAXIS VALUES=(&list) LABEL="&labely";
%MEND;

ODS GRAPHICS / RESET;
PROC SGPLOT DATA=scatter DATTRMAP=myattrmap;
   BAND X=x LOWER=low UPPER=high /
        GROUP=group NAME="groups" ATTRID=myid;
   SCATTER Y = &scatter2 X = &scatter1/MARKERATTRS=(COLOR=black);
   %AXIS(&list);
   KEYLEGEND "groups" / POSITION=right ACROSS=1;
   TITLE "&title";
RUN;
%MEND GRAPH;
```

APPENDIX II

```
DATA parameters;
      INPUT low high class $ 32.;
CARDS;
0 80 Normal
80 90 High Normal
90 100 Stage1
100 120 Stage2
120 . Emergency
;
RUN;

%LET title=Diastolic Blood Pressure Change;
%LET labely=Change in Diastolic Pressure (mmHg) at Rescreening;
%LET labelx=Diastolic Pressure (mmHg);
DATA bp;
      SET health.dbp_sbp;
      %LET scatter1=dbp; %LET scatter2=dbp_rscrn;
      if sbp<50 or sbp_rscrn<50 then delete;
RUN;

%MACRO GRAPH2(perm,data,colorbase1,colorbase2,colorbase3);

%MACRO SORT(data,var,nodupkey,decending);
      PROC SORT DATA=&data &nodupkey;
                  BY &decending &var;
      RUN;
%MEND;

DATA order;
      SET &perm;
      class=CAT(_N_,'-',class);
      IF low^=LAG(high) THEN LOW=LAG(high);
RUN;
DATA _NULL_;
      SET order(KEEP=Class OBS=1);
      CALL SYMPUT ('class', INPUT (class, $32. ) );
RUN;

DATA nomiss;
      SET &data;
      IF &scatter2=. OR &scatter1=. THEN DELETE;
      Difference = &scatter2-&scatter1;
RUN;
PROC MEANS DATA=nomiss MIN MAX NOPRINT;
      VAR &scatter1 &scatter2;
      OUTPUT OUT=data_means MIN(&scatter1 &scatter2)=mina minb
      MAX(&scatter1 &scatter2)=maxa maxb;
RUN;
DATA data_limits;
      SET data_means(DROP=_TYPE_ _FREQ_);
```

```
      datamin=MIN(mina,minb);
      datamax=MAX(maxa,maxb);
      datamin=ROUND((datamin-10),10);
      datamax=ROUND((datamax+10),10);
      KEEP datamin datamax;
RUN;
DATA _NULL_;
      SET data_limits;
      CALL SYMPUT ('datamin', INPUT (datamin, BEST. ) );
      CALL SYMPUT ('datamax', INPUT (datamax, BEST. ) );
RUN;

DATA one;
      RETAIN y;
      MERGE order data_limits;
            RETAIN min max;
            IF _N_=1 THEN DO;
                  min=datamin;
                  max=datamax;
            END;
            IF low<min THEN low=min;
            IF high=. OR high>max THEN high=max;
            diff=high-low;
                  IF diff<0 THEN DO;
                        %PUT One or more classes may have been deleted
due to lack of data;
                        DELETE;
                  END;
            y+diff;
RUN;

DATA mid;
      SET one(KEEP=low high diff class);
      DO I=low TO high BY diff;
            x=I;
            OUTPUT;
      END;
      drop I;
RUN;

DATA classes;
      SET one_nodupkey;
      n=1;
RUN;
PROC MEANS DATA=classes SUM NOPRINT;
      VAR n;
      OUTPUT OUT=sums(DROP= _TYPE_ _FREQ_) SUM=numclass;
RUN;
DATA _NULL_;
      SET sums;
      oneless=numclass-1;
      CALL SYMPUT ('numclass', INPUT (numclass, BEST. ) );
      CALL SYMPUT ('oneless', INPUT (oneless, BEST. ) );
```

```
RUN;

PROC MEANS DATA=nomiss MIN MAX NOPRINT;
      VAR difference;
      OUTPUT OUT=deff_means MAX=maxdiff MIN=negdiff;
RUN;
DATA data_limits2;
      SET deff_means(DROP=_TYPE_ _FREQ_);
      maxdiff=ROUND((maxdiff+10),10);
      negdiff=ROUND((negdiff-10),10);
      KEEP maxdiff negdiff;
RUN;
DATA _NULL_;
      SET data_limits2;
      CALL SYMPUT ('maxdiff', INPUT (maxdiff, BEST. ) );
      CALL SYMPUT ('negdiff', INPUT (negdiff, BEST. ) );
RUN;

DATA y1;
      SET one(KEEP=y);
      y=y+&datamin;
      x=0;
      IF _N_>=&numclass THEN DELETE;
RUN;
PROC TRANSPOSE DATA=y1(KEEP=y) OUT=y_1(DROP=_NAME_) PREFIX=y_;
RUN;
PROC TRANSPOSE DATA=y1(KEEP=x) OUT=x_1(DROP=_NAME_) PREFIX=x_;
RUN;

DATA y2;
      SET one(KEEP=high y RENAME=(high=x));
      y=(y+&datamin)*-1;
      x=x*2;
      IF _N_>=&numclass THEN DELETE;
RUN;
PROC TRANSPOSE DATA=y2(KEEP=y) OUT=y_2(DROP=_NAME_) PREFIX=y_;
RUN;
PROC TRANSPOSE DATA=y2(KEEP=x) OUT=x_2(DROP=_NAME_) PREFIX=x_;
RUN;

DATA ys;
      SET y_1 y_2;
RUN;
DATA xs;
      SET x_1 x_2;
RUN;

DATA TWO;
      SET xs mid(KEEP=x);
      ARRAY xcoord _ALL_;
      DO OVER xcoord;
            IF _N_>2 then xcoord=x;
      END;
```

```
      DROP x;
RUN;

DATA three;
      MERGE ys two;
RUN;

%MACRO LINES;
      %DO I=1 %TO &oneless;
            MODEL y_&i=x_&i;
            OUTPUT OUT=pred&i P=pred;
      %END;
%MEND;
PROC REG DATA=three NOPRINT;
      %LINES;
RUN;
QUIT;

%MACRO MULTIPERD;
 DATA Multiperd;
 set
 %DO I = 1 %TO &oneless;
 pred&i(KEEP=pred firstobs=3) %END;;
 RUN;
 %MEND; %MULTIPERD;

%MACRO MULTIX;
 DATA Multix;
 SET
 %DO I = 1 %TO &oneless;
 mid(KEEP=x class) %END;;
 RUN;
 %MEND; %MULTIX;

DATA Multiboth;
      MERGE multix multiperd;
RUN;
PROC MEANS DATA=Multiboth MAX MIN NOPRINT;
      VAR pred;
      OUTPUT OUT=graph_limit(DROP= _TYPE_ _FREQ_) MAX=maxhigh
MIN=minlow;
RUN;
DATA _NULL_;
      SET graph_limit;
      CALL SYMPUT ('maxhigh', INPUT (maxhigh, BEST. ) );
      CALL SYMPUT ('minlow', INPUT (minlow, BEST. ) );
RUN;

DATA mindiff;
      RETAIN pred;
      SET mid(KEEP=x class);
      IF _N_=1 THEN pred=&negdiff;
RUN;
```

```
DATA maxdiff;
      RETAIN pred;
      SET mid(KEEP=x class);
      IF _N_=1 THEN pred=&maxdiff;
RUN;

DATA lows;
      RETAIN x;
      SET mindiff multiboth;
RUN;
DATA highs;
      RETAIN x;
      SET multiboth maxdiff;
RUN;

DATA four;
RETAIN x low high class value;
      MERGE lows(RENAME=(pred=low)) highs(RENAME=(pred=high));
      LABEL low='low';
      LABEL high='high';
      IF low>high THEN low=high;
            IF x=&datamin THEN value=0;
            value+1;
RUN;
DATA four;
      SET four;
      IF MOD(value,2)=1 THEN value=value+1;
      value=value/2;
RUN;

PROC SORT DATA=four(DROP=value) OUT=foursorted;
      BY class;
RUN;
DATA four;
      MERGE foursorted four(KEEP=value);
RUN;

DATA groups;
      SET four; BY class;
      RETAIN count;
      IF FIRST.class THEN count+1;
            value=value-count;
            groupnum=0-value;
            group_num=(SQRT(groupnum*groupnum));
            LENGTH group $32.;
                  IF groupnum<0 THEN group=cat('Increase ', group_num);
                  ELSE IF groupnum>0 THEN group=cat('Decrease ',
group_num);
                  ELSE IF groupnum=0 THEN group=class;
                  KEEP x low high group groupnum;
RUN;
PROC SORT DATA=groups OUT=final;
      BY groupnum x;
```

```
RUN;

DATA scatter;
      MERGE final nomiss;
      IF group='' THEN group="&class";
RUN;

%IF &colorbase1=0 %THEN %LET colorbase1=.;
%IF &colorbase2=0 %THEN %LET colorbase2=.;
%IF &colorbase3=0 %THEN %LET colorbase3=.;
%IF &colorbase1=. %THEN
%DO;
      %LET colorbase1=6;
      %LET colorbase2=6;
      %LET colorbase3=6;
%END;
%IF &colorbase2=. %THEN %LET colorbase2=&colorbase1;
%IF &colorbase3=. %THEN %LET colorbase3=&colorbase2;

%IF &colorbase1<0 %THEN %LET colorbase1_2=(&colorbase1*-1);
%ELSE %LET colorbase1_2=&colorbase1;
%IF &colorbase2<0 %THEN %LET colorbase2_2=(&colorbase2*-1);
%ELSE %LET colorbase2_2=&colorbase2;
%IF &colorbase3<0 %THEN %LET colorbase3_2=(&colorbase3*-1);
%ELSE %LET colorbase3_2=&colorbase3;

PROC SORT DATA=final(DROP=groupnum) OUT=numboxes;
      BY group;
RUN;
DATA numboxes;
      SET numboxes;
      BY group;
      fgroup=FIRST.group;
RUN;
PROC MEANS DATA=numboxes SUM NOPRINT;
      VAR fgroup;
      OUTPUT OUT=sums2(drop= _type_ _freq_) SUM=numboxes;
RUN;
DATA _NULL_;
      SET sums2;
      CALL SYMPUT ('numboxes', INPUT (numboxes, BEST. ) );
RUN;

DATA colors;
      color1='CXFFEFEF';  color2='CXFFCFCF';  color3='CXFFAFAF';
color4='CXFF7F7F';
      color5='CXFF4F4F';  color6='CXEF0000';  color7='CXCF0000';
color8='CXAF0000';
      color9='CX8F0000';  color10='CX6F0000';
      color11='CXFFFFF0'; color12='CXFFFFC0'; color13='CXFFFF70';
color14='CXFFEF30';
      color15='CXFFCF00'; color16='CXFF9F00';     color17='CXFF6F00';
      color18='CXFF4F00';
```

```
      color19='CXDF2F00';   color20='CXBF0F00';
      color21='CXEFFFEF'; color22='CXC8FFC8';     color23='CX8FFF8F';
      color24='CX4FFF4F';
      color25='CX0FEF0F';   color26='CX00CF00';   color27='CX00AF00';
      color28='CX008F00';
      color29='CX006F00';   color30='CX004F00';
      color31='CXEFEFFF';   color32='CXCFCFFF';   color33='CX9F9FFF';
      color34='CX6F6FFF';
      color35='CX3F3FFF';   color36='CX0000EF';   color37='CX0000CF';
      color38='CX0000AF';
      color39='CX00007F';   color40='CX00005F';
      color41='CXFFDFFF';   color42='CXFFBFFF';   color43='CXFF9FFF';
      color44='CXFF7FFF';
      color45='CXFF4FFF';   color46='CXEF00EF';   color47='CXBF00BF';
      color48='CX8F008F';
      color49='CX6F006F';   color50='CX4F004F';
      color51='GRAYFC';     color52='GRAYE0';     color53='GRAYC4';
      color54='GRAYA8';
      color55='GRAY8C';     color56='GRAY70';     color57='GRAY54';
      color58='GRAY38';
      color59='GRAY1C';     color60='GRAY00';
      color61='CXEF0000'; color62='CXFF6F00'; color63='CXFFB838';
color64='CXFFFF70';
      color65='CX0FEF0F'; color66='CX00AF00'; color67='CX3F3FFF';
color68='CX0000CF';
      color69='CXFF4FFF'; color70='CXBF00BF';
      color71='CXFFFFF0'; color72='CXFFFF98'; color73='CXFFEF30';
color74='CX8FFF8F';
      color75='CX0FEF00'; color76='CX00AF00'; color77='CXBF00BF';
color78='CXFF50FF';
      color79='CXFF50B0'; color80='CXFF0080';
      color81='GRAYFC';     color82='GRAYE0';     color83='CXCFCFE8';
color84='CXCFCFFF';
      color85='CX9F9FFF'; color86='CX709BC0'; color87='CX38A860';
color88='CX008F00';
      color89='CX006F00'; color90='CX004F00';
RUN;
PROC TRANSPOSE DATA=colors OUT=colors NAME=colors;
      VAR color1-color90;
RUN;

DATA assign_colors;
      SET colors;
      total=1;
      color=1;
      IF _N_ >10 THEN color=2;
      IF _N_ >20 THEN color=3;
      IF _N_ >30 THEN color=4;
      IF _N_ >40 THEN color=5;
      IF _N_ >50 THEN color=6;
      IF _N_ >60 THEN color=7;
      IF _N_ >70 THEN color=8;
      IF _N_ >80 THEN color=9;
```

```
RUN;

DATA color_1;
     SET assign_colors(WHERE=(&colorbase1_2=color));
     color_base=1;
RUN;
%IF &colorbase1<0 %THEN %DO;
     %SORT(color_1,colors, ,decending);
%END;

DATA color_2;
     SET assign_colors(WHERE=(&colorbase2_2=color));
     color_base=2;
RUN;
%IF &colorbase2<0 %THEN %DO;
     %SORT(color_2,colors, ,decending);
%END;

DATA color_3;
     SET assign_colors(WHERE=(&colorbase3_2=color));
     color_base=3;
RUN;
%IF &colorbase3<0 %THEN %DO;
     %SORT(color_3,colors, ,decending);
%END;

DATA choose_colors;
     SET color_1 color_2 color_3;
RUN;

PROC MEANS DATA=choose_colors NOPRINT SUM;
     VAR total;
     OUTPUT OUT=color_sum(DROP=_TYPE_ _FREQ_) SUM=colorsum;
RUN;
DATA _NULL_;
     SET color_sum;
     CALL SYMPUT ('colorsum', INPUT (colorsum, BEST. ) );
RUN;

%IF &colorsum<&numboxes %THEN %PUT Graph requires more colors than
specified. Please input additional
color bases or reduce the number of classes;

%SORT(choose_colors,color_base, , );
%IF &colorbase1=&colorbase2 %THEN %DO;
     DATA final_colors;
          SET choose_colors;
          BY color_base;
          IF FIRST.color_base THEN count=0;
          count+1;
               IF color_base=1 THEN DO;
                    IF count>&numclass THEN DELETE;
               END;
```

```
                ELSE IF color_base=2 THEN DO;
                      IF count<=&numclass THEN DELETE;
                      IF count>=(2*&numclass) THEN DELETE;
                ELSE IF color_base=3 THEN DO;
                      IF count<=&numclass THEN DELETE;
                      IF count>=(2*&numclass) THEN DELETE;
                END;
      RUN;
%END;
%IF &colorbase1^=&colorbase2 %THEN %DO;
      %IF &numclass>5 %THEN %DO;
            DATA final_colors;
                  SET choose_colors;
                  BY color_base;
                  IF first.color_base THEN count=0;
                  count+1;
                        IF color_base=1 THEN DO;
                              IF count>&numclass THEN DELETE;
                        END;
                        ELSE IF color_base=2 THEN DO;
                              IF count>=&numclass THEN DELETE;
                        END;
                        ELSE IF color_base=3 THEN DO;
                              IF count>=&numclass THEN DELETE;
                        END;
            RUN;
      %END;
      %IF &numclass<=5 %THEN %DO;
            DATA less_colors;
                  SET choose_colors;
                  BY color_base;
                  IF FIRST.color_base THEN count1=0;
                  count1+1;
                  onlyeven=MOD(count1,2);
                        IF color_base^=1 THEN DO;
                              IF onlyeven ^= 0 THEN DELETE;
                        END;
            RUN;
            DATA final_colors;
                  SET less_colors;
                  BY color_base;
                  IF FIRST.color_base THEN count2=0;
                  count2+1;
                        IF color_base=1 THEN DO;
                              IF count2>&numclass THEN DELETE;
                        END;
                        ELSE IF color_base=2 THEN DO;
                              IF count2>=&numclass THEN DELETE;
                        END;
                        ELSE IF color_base=3 THEN DO;
                              IF count2>=&numclass THEN DELETE;
                        END;
```

```
            RUN;
      %END;
%END;

%SORT(final_colors,color_base, , );
PROC SORT DATA=final(KEEP=group) OUT=color_values NODUPKEY;
      BY group;
RUN;
DATA myattrmap;
      RETAIN ID value fillcolor;
      ID='myid';
      MERGE color_values(RENAME=(group=value)) final_colors(KEEP=COL1
RENAME=(COL1=fillcolor));
      IF value='' THEN DELETE;
      KEEP ID value fillcolor;
RUN;

DATA xaxis_values;
      MERGE order data_limits;
      datamin2=datamin+5; datamax2=datamax-5;
            RETAIN min max;
            IF _N_=1 THEN DO;
                  min=datamin2;
                  max=datamax2;
            END;
            IF low<min THEN low=min; IF low>max THEN low=max;
            IF high^=. AND high<min THEN high=min; IF high=. OR
high>max THEN high=max;
            diff=high-low; IF diff<=0 THEN DELETE;
            DO I=low TO high BY diff;
                  x=I;
                  OUTPUT;
            END;
      KEEP x;
RUN;
%SORT(xaxis_values,x,nodupkey, );
PROC SQL NOPRINT;
  SELECT x
  INTO :listx SEPARATED BY ' '
  FROM xaxis_values;
QUIT;
%MACRO XAXIS(listx);
      XAXIS VALUES=(&listx) LABEL="&labelx";
%MEND;

DATA yaxis_values;
      MERGE Multiperd data_limits2;
      NEGDIFF2=NEGDIFF+5; MAXDIFF2=MAXDIFF-5;
            RETAIN min max;
            IF _N_=1 THEN DO;
                  min=negdiff2;
                  max=maxdiff2;
            END;
```

```
           IF pred<min THEN pred=min; IF pred>max THEN pred=max;
           IF pred<min THEN pred=min; IF pred=. OR pred>max THEN
pred=max;
     KEEP pred;
RUN;
%SORT(yaxis_values,pred,nodupkey, );
PROC SQL NOPRINT;
     SELECT pred
     INTO :listy SEPARATED BY ' '
     FROM yaxis_values;
QUIT;
%MACRO YAXIS(listy);
     YAXIS VALUES=(&listy) LABEL="&labely";
%MEND;

ODS GRAPHICS / RESET;
PROC SGPLOT DATA=scatter DATTRMAP=myattrmap;
   BAND X=x LOWER=low UPPER=high /
        GROUP=group NAME="groups" ATTRID=myid;
   SCATTER Y = Difference X =
&scatter1/MARKERATTRS=(COLOR=black);refline 0;
   %XAXIS(&listx); %YAXIS(&listy);
   KEYLEGEND "groups" / POSITION=right ACROSS=1;
   TITLE "&title";
RUN;

%MEND GRAPH2;
```

APPENDIX III

```
DATA parameters;
      INPUT lowx highx lowy highy class $ 32.;
CARDS;
0 80 0 120 Normal
80 90 120 140 High Normal
90 100 140 160 Stage1
100 120 160 210 Stage2
120 . 210 . Emergency
;
RUN;

%LET title=Systolic and Diastolic Blood Pressure;
%LET labely=Systolic Pressure (mmHg);
%LET labelx=Diastolic Pressure (mmHg);
DATA bp;
      SET health.Sbpdppatient;
      %LET scatter1=dbp; %LET scatter2=sbp;
RUN;

%MACRO GRAPH3(perm,data,colorbase);

%MACRO SORT(data,var,nodupkey,decending);
      PROC SORT DATA=&data &nodupkey;
                  BY &decending &var;
      RUN;
%MEND;

DATA order;
      SET &perm;
      class=CAT(_N_,'-',class);
      IF lowx^=LAG(highx) THEN lowx=LAG(highx);
      IF lowy^=LAG(highy) THEN lowy=LAG(highy);
RUN;
DATA _NULL_;
      SET order(KEEP=Class OBS=1);
      CALL SYMPUT ('class', INPUT (class, $32. ) );
RUN;

DATA nomiss;
      SET &data;
      IF &scatter2=. OR &scatter1=. THEN DELETE;
RUN;
PROC MEANS DATA=nomiss MIN MAX NOPRINT;
      VAR &scatter1 &scatter2;
      OUTPUT OUT=data_means MIN(&scatter1 &scatter2)=mina minb
      MAX(&scatter1 &scatter2)=maxa maxb;
DATA data_limits;
      SET data_means(DROP=_TYPE_ _FREQ_);
            dataminx=mina;
            datamaxx=maxa;
            dataminy=minb;
```

```
            datamaxy=maxb;
                  dataminx=ROUND((dataminx-10),10);
                  datamaxx=ROUND((datamaxx+10),10);
                  dataminy=ROUND((dataminy-10),10);
                  datamaxy=ROUND((datamaxy+10),10);
      KEEP dataminx datamaxx dataminy datamaxy;
RUN;
DATA _NULL_;
 SET data_limits;
 CALL SYMPUT ('dataminx', INPUT (dataminx, BEST. ) );
 CALL SYMPUT ('datamaxx', INPUT (datamaxx, BEST. ) );
 CALL SYMPUT ('dataminy', INPUT (dataminy, BEST. ) );
 CALL SYMPUT ('datamaxy', INPUT (datamaxy, BEST. ) );
RUN;

DATA one;
      MERGE order data_limits;
            RETAIN minx maxx miny maxy;
            IF _N_=1 THEN DO;
                  minx=dataminx; maxx=datamaxx;
                  miny=dataminy; maxy=datamaxy;
            END;
            IF lowx<minx THEN lowx=minx; IF lowy<miny THEN lowy=miny;
            IF highx=. OR highx>maxx THEN highx=maxx;
            IF highy=. OR highy>maxy THEN highy=maxy;
            diffx=highx-lowx; diffy=highy-lowy;
                  IF diffx<0 AND diffy<0 THEN DO;
                        %PUT One or more classes may have been deleted
due to lack of data;
                        delete;
                  END;
            DO I=lowx TO highx BY diffx;
                  x=I;
                  OUTPUT;
            END;
      KEEP x lowy highy class;
RUN;

DATA vertical;
      SET one(RENAME=(lowy=low highy=high class=group) FIRSTOBS=3);
            low=&dataminy;
RUN;

PROC SORT DATA=one NODUPKEY OUT=one_nodup;
      BY x;
RUN;
DATA horzontal;
      SET one_nodup(RENAME=(lowy=low highy=high class=group) DROP=x
FIRSTOBS=2)
      one_nodup(RENAME=(lowy=low highy=high class=group) FIRSTOBS=2);
            IF x=. THEN x=&dataminx;
RUN;
%SORT(horzontal,low, ,)
```

```
PROC SORT DATA=one(DROP=x) OUT=one_nodupkey NODUPKEY;
      BY lowy;
RUN;
DATA classes;
      SET one_nodupkey;
      n=1;
RUN;
PROC MEANS DATA=classes SUM NOPRINT;
      VAR n;
      OUTPUT OUT=sums(DROP= _TYPE_ _FREQ_) SUM=numclass;
RUN;
DATA _NULL_;
      SET sums;
      CALL SYMPUT ('numclass', INPUT (numclass, BEST. ) );
RUN;

DATA final;
      SET horzontal vertical;
RUN;

DATA scatter;
      MERGE final nomiss;
      IF group='' THEN group="&class";
RUN;


%IF &colorbase=0 %THEN %LET colorbase=6;
%IF &colorbase=. %THEN %LET colorbase=6;
%IF &colorbase<0 %THEN %LET colorbase2=(&colorbase*-1);
%ELSE %LET colorbase2=&colorbase;

DATA colors;
      color1='CXFFEFEF';  color2='CXFFCFCF';  color3='CXFFAFAF';
color4='CXFF7F7F';
      color5='CXFF4F4F';  color6='CXEF0000';  color7='CXCF0000';
color8='CXAF0000';
      color9='CX8F0000';  color10='CX6F0000';
      color11='CXFFFFF0'; color12='CXFFFFC0'; color13='CXFFFF70';
color14='CXFFEF30';
      color15='CXFFCF00'; color16='CXFF9F00';     color17='CXFF6F00';
      color18='CXFF4F00';
      color19='CXDF2F00';   color20='CXBF0F00';
      color21='CXEFFFEF'; color22='CXC8FFC8';     color23='CX8FFF8F';
      color24='CX4FFF4F';
      color25='CX0FEF0F';   color26='CX00CF00';   color27='CX00AF00';
      color28='CX008F00';
      color29='CX006F00';   color30='CX004F00';
      color31='CXEFEFFF';   color32='CXCFCFFF';   color33='CX9F9FFF';
      color34='CX6F6FFF';
      color35='CX3F3FFF';   color36='CX0000EF';   color37='CX0000CF';
      color38='CX0000AF';
      color39='CX00007F';   color40='CX00005F';
```

```
      color41='CXFFDFFF';   color42='CXFFBFFF';   color43='CXFF9FFF';
      color44='CXFF7FFF';
      color45='CXFF4FFF';   color46='CXEF00EF';   color47='CXBF00BF';
      color48='CX8F008F';
      color49='CX6F006F';   color50='CX4F004F';
      color51='GRAYFC';     color52='GRAYE0';     color53='GRAYC4';
      color54='GRAYA8';
      color55='GRAY8C';     color56='GRAY70';     color57='GRAY54';
      color58='GRAY38';
      color59='GRAY1C';     color60='GRAY00';
      color61='CXEF0000'; color62='CXFF6F00'; color63='CXFFB838';
color64='CXFFFF70';
      color65='CX0FEF0F'; color66='CX00AF00'; color67='CX3F3FFF';
color68='CX0000CF';
      color69='CXFF4FFF'; color70='CXBF00BF';
      color71='CXFFFFF0'; color72='CXFFFF98'; color73='CXFFEF30';
color74='CX8FFF8F';
      color75='CX0FEF00'; color76='CX00AF00'; color77='CXBF00BF';
color78='CXFF50FF';
      color79='CXFF50B0'; color80='CXFF0080';
      color81='GRAYFC';     color82='GRAYE0';     color83='CXCFCFE8';
color84='CXCFCFFF';
      color85='CX9F9FFF'; color86='CX709BC0'; color87='CX38A860';
color88='CX008F00';
      color89='CX006F00'; color90='CX004F00';
RUN;
PROC TRANSPOSE DATA=colors OUT=colors NAME=colors;
      VAR color1-color90;
RUN;

DATA assign_colors;
      SET colors;
      total=1;
      color=1;
      IF _N_ >10 THEN color=2;
      IF _N_ >20 THEN color=3;
      IF _N_ >30 THEN color=4;
      IF _N_ >40 THEN color=5;
      IF _N_ >50 THEN color=6;
      IF _N_ >60 THEN color=7;
      IF _N_ >70 THEN color=8;
      IF _N_ >80 THEN color=9;
RUN;

DATA choose_colors;
      SET assign_colors(WHERE=(&colorbase2=color));
RUN;
%IF &colorbase<0 %THEN %DO;
      %SORT(choose_colors,colors, ,decending);
%END;

%IF &numclass>5 %THEN %DO;
      DATA final_colors;
```

```
            SET choose_colors;
            IF _N_>&numclass THEN DELETE;
      RUN;
%END;
%IF &numclass<=5 %THEN %DO;
      DATA less_colors;
            SET choose_colors;
            onlyeven=MOD(_N_,2);
                  IF onlyeven ^= 0 THEN delete;
      RUN;
      DATA final_colors;
            SET less_colors;
                  IF _N_>&numclass THEN DELETE;
      RUN;
%END;

PROC SORT DATA=final(KEEP=group) OUT=color_values NODUPKEY;
      BY group;
RUN;
DATA myattrmap;
      RETAIN ID value fillcolor;
      ID='myid';
      MERGE color_values(RENAME=(group=value)) final_colors(KEEP=COL1
RENAME=(COL1=fillcolor));
      IF value='' THEN DELETE;
      KEEP ID value fillcolor;
RUN;

DATA xaxis_values;
      MERGE order data_limits;
      dataminx2=dataminx+5; datamaxx2=datamaxx-5;
            RETAIN minx maxx;
            IF _N_=1 THEN DO;
                  minx=dataminx2; maxx=datamaxx2;
            END;
            IF lowx<minx THEN lowx=minx; IF lowx>maxx THEN lowx=maxx;
            IF highx<minx AND highx^=. THEN highx=minx;
            IF highx=. OR highx>maxx THEN highx=maxx;
            diffx=highx-lowx; IF diffx<0 THEN DELETE;
RUN;
DATA xaxis;
      SET xaxis_values;
            WHERE diffx>0;
            DO I=lowx TO highx BY diffx;
                  x=I;
                  OUTPUT;
            END;
      KEEP x;
RUN;
%SORT(xaxis,x,nodupkey, );
PROC SQL NOPRINT;
  SELECT x
  INTO :listx SEPARATED BY ' '
```

```
  FROM xaxis;
QUIT;
%MACRO XAXIS(listx);
     XAXIS VALUES=(&listx) LABEL="&labelx";
%MEND;

DATA yaxis_values;
     MERGE order data_limits;
     dataminy2=dataminy+5; datamaxy2=datamaxy-5;
          RETAIN miny maxy;
          IF _N_=1 THEN DO;
               miny=dataminy2; maxy=datamaxy2;
          END;
          IF lowy<miny THEN lowy=miny; IF lowy>maxy THEN lowy=maxy;
          IF highy<miny AND highy^=. THEN highy=miny;
          IF highy=. OR highy>maxy THEN highy=maxy;
          diffy=highy-lowy; IF diffy<0 THEN DELETE;
RUN;
DATA yaxis;
     SET yaxis_values;
          WHERE diffy>0;
          DO I=lowy TO highy BY diffy;
               y=I;
               OUTPUT;
          END;
     KEEP y;
RUN;
%SORT(yaxis,y,nodupkey, );
PROC SQL NOPRINT;
  SELECT y
  INTO :listy SEPARATED BY ' '
  FROM yaxis;
QUIT;
%MACRO YAXIS(listy);
     YAXIS VALUES=(&listy) LABEL="&labely";
%MEND;

ODS GRAPHICS / RESET;
PROC SGPLOT DATA=scatter DATTRMAP=myattrmap;
   BAND X=x LOWER=low UPPER=high /
        GROUP=group NAME="groups" ATTRID=myid;
   SCATTER Y = &scatter2 X = &scatter1/MARKERATTRS=(COLOR=black);
   %XAXIS(&listx); %YAXIS(&listy);
   KEYLEGEND "groups" / POSITION=right ACROSS=1;
   TITLE "&title";
RUN;

%MEND GRAPH3;
```