Paper 60-2010

More to it than Meets the Eye:
Creating Custom Legends that Really Tell a Story

Pete Lund, Looking Glass Analytics, Olympia, WA

ABSTRACT

Merriam-Webster defines a legend as “an explanatory list of the symbols on a map or chart.” We’re used to seeing just that – a table of symbols or a series of colored boxes with a word or two explaining what the symbols or colors mean. In almost every case, this is sufficient. However, what if a legend did more than just explain the colors on the map – what if it had some descriptive or analytic of its own?

This paper discusses techniques to build custom legends using the SAS/Graph Annotate facility. Two real-world examples will be presented. One creates a typical legend and then adds quantitative, descriptive information to it. The other creates a graphic that not only defines the colors on the map, but also stands alone as an analytic display.

INTRODUCTION

The examples presented in the paper come from a project, funded by the National Institute on Drug Abuse (NIDA), which creates reports detailing measures of alcohol and drug epidemiology. The project uses a web-based user interface, where different measures, time frames and levels of geography can be selected. For example, a user might select a report detailing the change in per-capita admissions to alcohol treatment between 2004 and 2006 by zip code. These selections are made on a web page and passed back,, via SAS/Intrnet, to a SAS-based back end which creates a PDF report that is passed back to the browser.

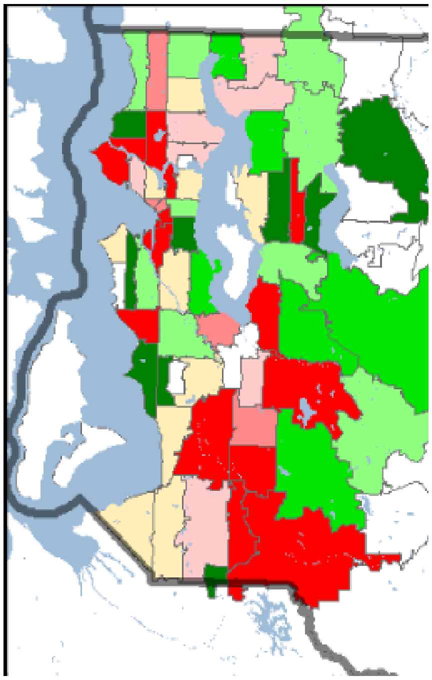
The resulting report contains a map of rate change values, with an accompanying legend and explanatory notes, and a supporting table with details for each zip code. All of the different reports for this project are similar in that all have a map and related detail tables. One important note – we did not use SAS/Graph to create the maps. SAS makes a call to a map server, forwarding some of the parameters passed back from the web site, which creates an image file containing the map. This image file is then included in the PDF with the SAS-generated output.

Because SAS is not creating the map, we cannot use the LEGEND statement to create the legend. The map server could have created the legend and made it part of the map graphic, however there was little control over the appearance and placement of that legend. That led us to use the SAS/Graph Annotate facility to create a separate, stand-alone graphic file containing the legend. That also allowed us to think “outside the box” and do a bit more than just a traditional legend.

Note: in this paper, for reasons of confidentiality and privacy, the real measures being displayed in the maps and tables are masked and reported in ways such as “selected measure” or “measure 1.”

STARTING OUT SIMPLE

Our first example compares the rates for a selected measure in two different years. The map displays the difference between the rate in the first year and the rate in the second year selected. Areas in which rates went up are shown in various shades of red and areas where the rates went down are shown in shades of green. You can see this on the map snippet of King County, WA zip code shown below. Notice that there are also tan areas, where there was very little change and a few white areas as well. The white zip codes are those that too few cases to reliably compute rates, so the information is



suppressed. As we'll see later, this affords us a nice opportunity to customize the legend a bit, in a way not easy to do with a LEGEND statement.

There are always seven ranges of rate differences displayed on the map; three in red, three in green and one tan. There is an algorithm that examines the distribution of rate change values and assigns each data value a mapping value of 1 through 7 corresponding to the . A dynamic format is build which contains the text label with the range of values that correspond to each of the mapping values. For our example, this generated format (BarLabel) would be equivalent to the one shown here. There is also a static format (BarColor) that defines the colors to be used for each value.

```
value BarLabel
   1 = '21 and above'
   2 = '13 to 20'
   3 = '5 to 12'
   4 = '-4 to 4'
   5 = '-5 to -12'
   6 = '-13 to -20'
   7 = '-21 and below';
```

Note: this format is dynamically generated, not "hard-coded"

These labels and colors defined in these formats will be used to build a legend for the map. Most often we would use SAS/Graph LEGEND statements but, as noted above, SAS/Graph is not used to create the map. We will use the Annotate facility to create all the components of the legend and much more.

A Quick Introduction to The SAS/Graph Annotate Facility

SAS/Graph procedures can create many different types of charts, plots and maps. There are a number of mechanisms for adding information to that output including axis, symbol, pattern and legend statements and also procedure-specific options. In addition, the SAS/Graph Annotate facility allows you to define commands to create graphics or to "annotate" other SAS/Graph-generated output with additional graphical elements.

Annotate Data Sets

The Annotate commands are stored in SAS data sets. The data set variables have specific names – not all of the variables will be used for every Annotate commands. Some of the variables include:

- Function – the type of Annotate command
- X and Y – specify the x and y coordinates of the output
- Color – the color of the output
- Text – text to display
- Style – font, bar pattern or image type (depending on the command)

If a variable is not used by a particular command it is ignored, as are any non-Annotate variables in the dataset.

Each observation contains information for a single command, specified in the Function variable. It's often helpful to keep a little history in mind when creating an Annotate data set. Think of the output being generated on a plotter. We might need to move the pen to a specific location on the paper, then draw a line, move the pen again, then add some text, and so on. Commands include,

- Move – moves the "pen" to the specified x,y location
- Label – places text on the page
- Draw – draws a line from the current location to the specified x,y
- Bar – treats the current location as one corner of a bar and the specified x,y as the other corner
- Image – places an image (gif, jpg, png, etc.) on the page

Note: there are a number of Annotate data set variables which define the "environment" of the annotation: the coordinate system to use, whether an element should be placed before or after other graphics, etc. See the Resource section at the end of the paper for details on these variables.

Annotate Macros

There are many different Annotate commands and it can be challenging to remember which commands need which variables. Also, a great number of graphic elements require more than one Annotate command. For example, to draw a bar on the page requires a “move” command, which places the “pen” at one corner of the bar, and a “bar” command, which contains the coordinates for the opposite corner of the bar. SAS has supplied a number of macros that simplify the process of creating the Annotate dataset.

By default, the Annotate macros are not available to be used. Issue a call to the %Annomac macro to make the library of Annotate macros accessible.

Each macro has the parameters necessary to create the needed command(s). For example, the %BAR macro has the following seven parameters:

- X1,Y1 – the first corner of the bar
- X2,Y2 – the second corner of the bar (diagonally opposite from the first
- Color – the color of the bar
- Line – which lines should be drawn
- Style – the fill style of the bar

A call to this macro generates two observations in the data set – first, a “move” command using the X1,Y1 parameters and then a “bar” command using the X2,Y2 and the rest of the parameters.

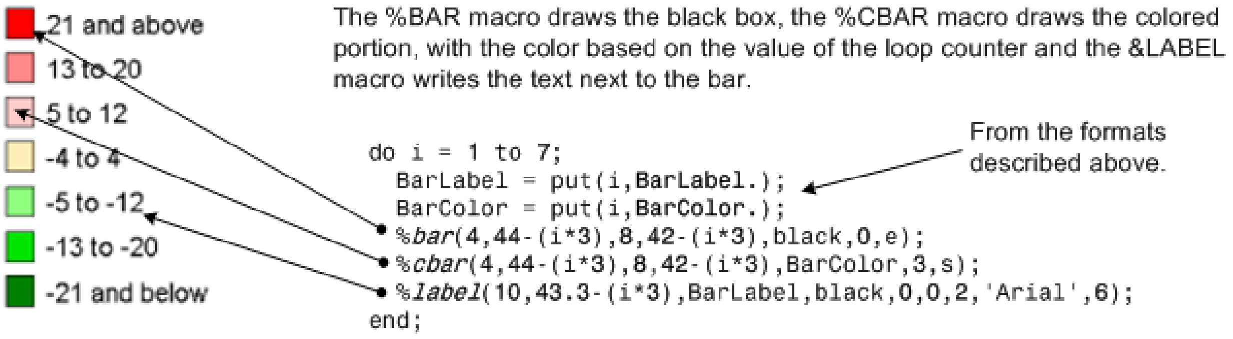
Using Annotate Data Sets with SAS/Graph Procedures

As noted above, Annotate data sets are often used with SAS/Graph procedures that generate output. The ANNO= option on the procedure statement references the dataset to be used. Annotate data sets can be displayed on their own with the GSLIDE procedure. GSLIDE creates no output of its own, but will display annotations, as well as titles and footnotes.

This paper is not intended to be a tutorial on Annotate data sets or the Annotate variables, functions and macros, but rather a discussion of a particular use of the Annotate facility. See the resource section at the end of the paper to get more general annotate information.

A Simple Legend…

Back to our example – we could always build a very simple legend using the Annotate macros. We want to have a stacked legend, one colored box and label on top of the next. Remember, the map will always have seven possible regions, so the layout of the legend will always look the same - the labels will change depending on the measure, years and level of geography chosen for the report. We can create the seven legend entries inside a little loop. What needs to change on each pass of the loop is the Y position of the legend components, the color of the box and the text for the label. The color and text will be obtained from the formats described above. Each legend entry is 3 units below the previous one, so the Y parameters are just computed with an offset (i*3) from a fixed starting position.



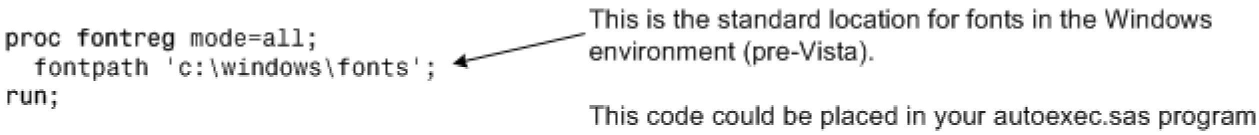
The %BAR macro draws the black box, the %CBAR macro draws the colored portion, with the color based on the value of the loop counter and the &LABEL macro writes the text next to the bar.

From the formats described above.

```
do i = 1 to 7;
  BarLabel = put(i,BarLabel.);
  BarColor = put(i,BarColor.);
  %bar(4,44-(i*3),8,42-(i*3),black,0,e);
  %cbar(4,44-(i*3),8,42-(i*3),BarColor,3,s);
  %label(10,43.3-(i*3),BarLabel,black,0,0,2,'Arial',6);
end;
```

Before continuing, it should be noted that the %CBAR macro is not an Annotate macro. A deficiency of the %BAR macro is that the color parameter must be a color name or RGB value – that is, the color cannot be referenced with a variable. This means that you need to have a separate %BAR call for each color that you want. The %CBAR macro functions like the %BAR macro except that the color parameter can be either a constant or a dataset variable

Another important note: In the %LABEL call above, notice the ‘Arial’ font specification. If you are using a non-SAS/Graph font you must enclose the font name in single quotes. You may also need to run the FONTREG procedure to make fonts available to your SAS session. The following code, submitted once per session, will generally work in the Windows environment:



```
proc fontreg mode=all;
  fontpath 'c:\windows\fonts';
run;
```

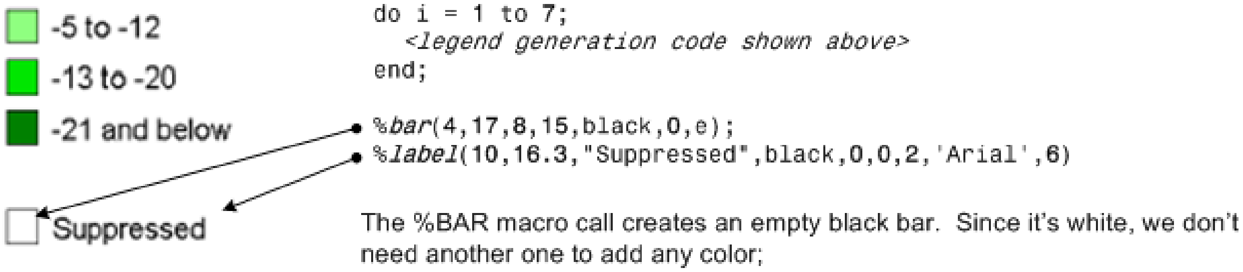
This is the standard location for fonts in the Windows environment (pre-Vista).

This code could be placed in your autoexec.sas program

What we’ve done so far is very much like the legends we can get with SAS/Graph and the LEGEND statement – let’s see if we can kick it up a notch and create a legend that provides a little more information than just the colors and labels for the map.

In this project, for reliability reasons, if there are not enough reported cases in a particular geographic area, the result of the rate calculation is suppressed and the area remains white. The first thing we’ll add to our traditional legend, which would be a challenge for the LEGEND statement, is a white box labeled “Suppressed” that is offset vertically from the rest of the legend entries.

In the code that creates the legend, shown above, the bottom (i=7) legend entry would have had a Y-value ot 23, 44-(7*3). If we would have created the suppressed entry in the same loop, its Y-value would be 20, 44-(8*3). However, we want to offset the suppressed entry a little bit, as it’s not really part of the sequence of values in the rest of the legend. So, following the loop we can add the code to create the white box and the label. The new “Suppressed” entry, plus part of the original legend, is shown here:

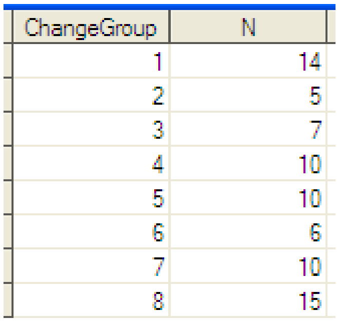


```
do i = 1 to 7;
  <legend generation code shown above>
end;

%bar(4,17,8,15,black,0,e);
%label(10,16.3,"Suppressed",black,0,0,2,'Arial',6)
```

The %BAR macro call creates an empty black bar. Since it’s white, we don’t need another one to add any color;

…AND THEN SO MUCH MORE

One thing that often hard to notice on a shaded, or choropleth, map is how many of the areas fall into each color value. This is especially true when there are a large number of areas (for example, the 77 zip codes that are in King County). The next thing we’ll add to out legend is a histogram denoting the number of areas in each value range.



ChangeGroup	N
1	14
2	5
3	7
4	10
5	10
6	6
7	10
8	15

A small dataset is created which contains an observation for each of the legend entry values (stored in the variable ChangeGroup) - values 1 through 7 represent the seven entries in the legend and value 8 represents the suppressed areas. The count of zip codes for each value is contained in the variable N.

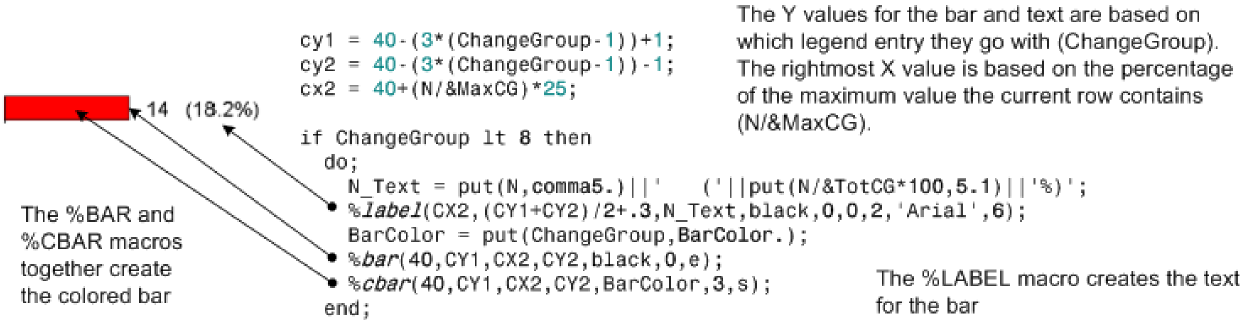
Also, we will analyze this little dataset and create two macro variables:

- &MaxCG: contains the maximum number of areas associated with one of the ChangeGroups (the max of the N variable). In this example, the suppressed areas (ChangeValue 8) has the most with 15.
- &TotCG: contains the total number of areas on the map (the sum of the N variable). There are 77 zip codes on the map.

We'll again use %BAR, %CBAR and %LABEL macro calls. However, this time we need to actually look at the values in the dataset, not just a static DO…END loop like we used above. Just like we did above, we're going to create colored bars with labels using %BAR and %LABEL macro. The process will be different from the above in a few ways:

- These new bars will be aligned with the ones we already have in the legend, so the offset of the Y parameters will be done exactly the same as we did above. However, this time we'll use the value of the ChangeGroup variable rather than the loop counter (i) to calculate the Y values. See the calculation of CY1 and CY2 below.
- The original bars were all the same size; these bars will have a width based on the value of N relative to the largest value of N, (N/&MaxCG). The widest bar will be 25 units across – see the calculation of CX2 below.
- The text next to the bar is composed of two pieces and is stored in the variable N_TEXT:
    - `put(N,comma5.)` writes out the value of N, with a comma thousands separator
    - `('||put(N/&TotCG*100,5.1)||'%)` writes out the percentage, wrapped in parentheses

In the example below you can see that the right X value of the bar (contained in variable CX2) is based on the current value of N, divided by the maximum value of N (&MaxCG). This makes each bar proportional in width to the maximum, with the largest value having a bar that is 25 units wide.



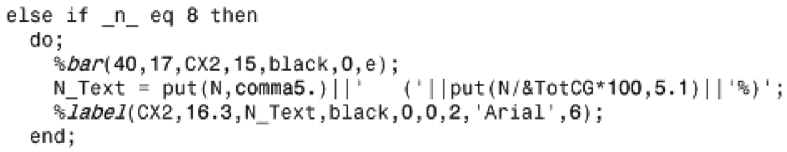
```
cy1 = 40-(3*(ChangeGroup-1))+1;
cy2 = 40-(3*(ChangeGroup-1))-1;
cx2 = 40+(N/&MaxCG)*25;

if ChangeGroup lt 8 then
  do;
    N_Text = put(N,comma5.)||'   ('||put(N/&TotCG*100,5.1)||'%)';
    %label(CX2,(CY1+CY2)/2+.3,N_Text,black,0,0,2,'Arial',6);
    BarColor = put(ChangeGroup,BarColor.);
    %bar(40,CY1,CX2,CY2,black,0,e);
    %cbar(40,CY1,CX2,CY2,BarColor,3,s);
  end;
```

The Y values for the bar and text are based on which legend entry they go with (ChangeGroup). The rightmost X value is based on the percentage of the maximum value the current row contains (N/&MaxCG).

The %BAR and %CBAR macros together create the colored bar

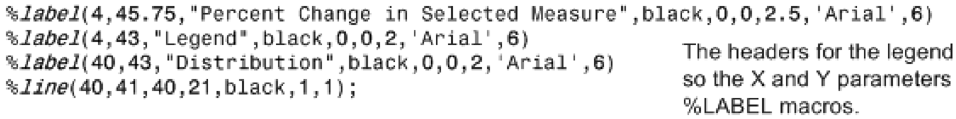
The %LABEL macro creates the text for the bar

Notice that the code above is only executed when the ChangeGroup value is less than 8. When the value is 8, it is the count of suppressed areas. To match the main legend above, we need to offset the suppressed row from the rest of the rows – it was simpler to code a separate section for this. The code is very similar, except that most of the X,Y values can be hard-coded. The exception is the right X value which, like those above, is based on the proportion of the maximum. In this example, there happen to be more suppressed values than any other value, so the suppressed bar will be the maximum 25 units wide.



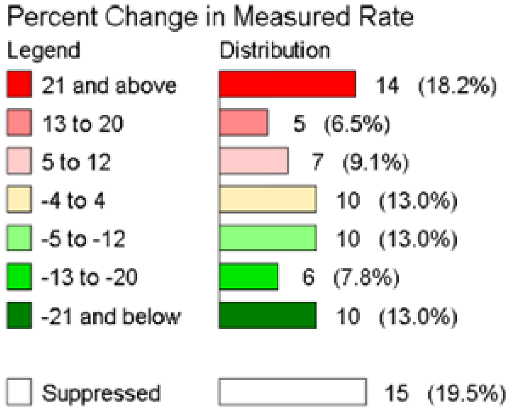
```
else if _n_ eq 8 then
  do;
    %bar(40,17,CX2,15,black,0,e);
    N_Text = put(N,comma5.)||'   ('||put(N/&TotCG*100,5.1)||'%)';
    %label(CX2,16.3,N_Text,black,0,0,2,'Arial',6);
  end;
```

The final steps needed to make our new legend are some headers and a single line on the left edge of the value bars in order to make it more like a histogram. Here's the final result:



```
%label(4,45.75,"Percent Change in Selected Measure",black,0,0,2.5,'Arial',6)
%label(4,43,"Legend",black,0,0,2,'Arial',6)
%label(40,43,"Distribution",black,0,0,2,'Arial',6)
%line(40,41,40,21,black,1,1);
```

The headers for the legend are in a fixed positions, so the X and Y parameters can be hard-coded in the %LABEL macros.



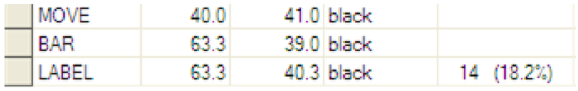
The %LINE macro call above draws a vertical line to the left of all the distribution bars. It helps to give a more histogram-like appearance to that part of the legend.

So, we have a relatively small amount of code that produces this legend based on the data for the particular report. There is nothing hard-coded and this same code is used for zip code, school district and census tract maps.

What we have now is a legend that does much more than let you know what the colors mean. You don't have to try and count the number of areas for each value

executed on the first iteration of the data step.

The histogram portion of the code is called for each observation in the ChangeCounts dataset.



	MOVE	40.0	41.0	black	
	BAR	63.3	39.0	black	
	LABEL	63.3	40.3	black	14 (18.2%)

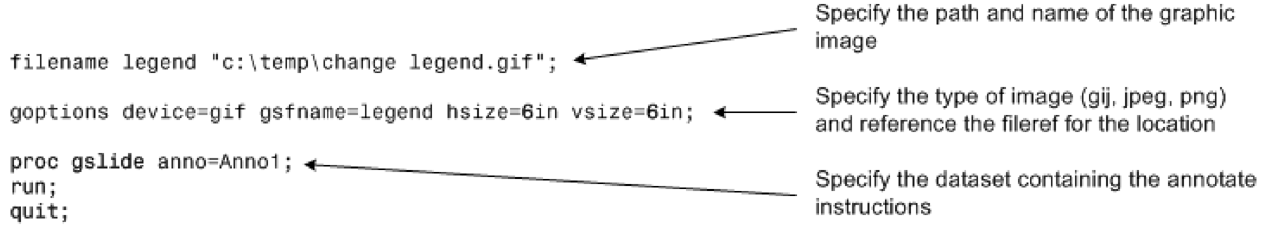
partial output (histogram portion of dataset – top histogram entry)

:We've talked about all these Annotate macro calls as if they are creating the graphics themselves. In reality, all they do is write observations to a dataset. When all the code we've talked about above is done we're left with a dataset containing 98 observations, xx for the "legend" and xx for the "histogram", with the instructions to create a graphic.

CREATING THE GRAPHIC FILE…

How do we actually make the graphic? Well, this is the simplest part so far. The most common use of Annotate datasets is to reference them in a SAS/Graph procedure that is already producing some output. However, as mentioned earlier, we can use the GSLIDE procedure to generate a graphic from the annotate instructions without being attached to any other output.

The syntax of the GSLIDE procedure is the simplest of all SAS procedures – there are no statements and only a single argument: the name of the annotate dataset. The only other things we need are a FILENAME statement giving a reference to the desired location of the graphic image file and a GOPTIONS statement giving the type of image file to create.



```
filename legend "c:\temp\change legend.gif";

goptions device=gif gsfname=legend hsize=6in vsize=6in;

proc gslide anno=Anno1;
run;
quit;
```

Specify the path and name of the graphic image

Specify the type of image (gij, jpeg, png) and reference the fileref for the location

Specify the dataset containing the annotate instructions

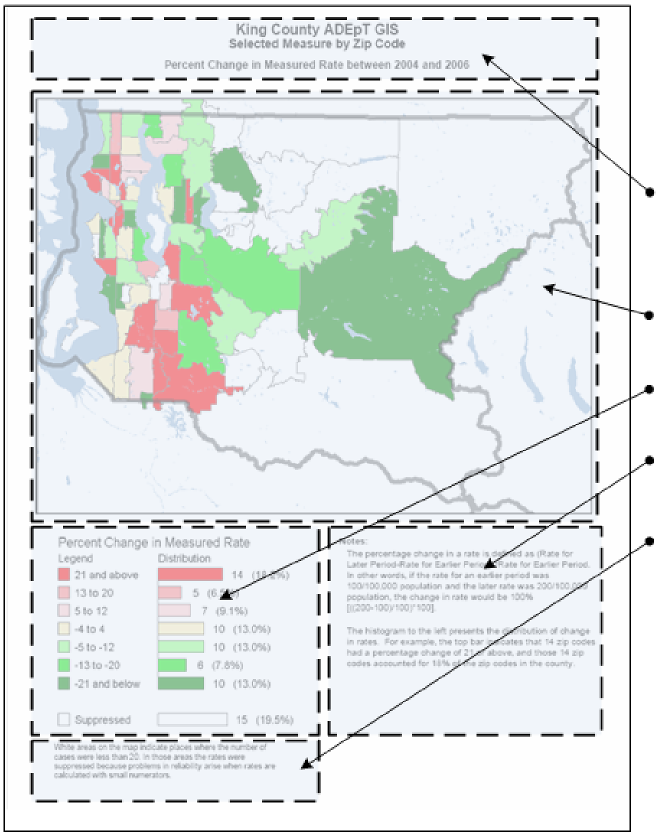
The above code would create a gif image (change legend.gif) of the legend shown above.

…AND PLACING IT ON THE PAGE

Now that we have the image, how do we put it on the page? You can see an example of the report below and in Appendix 1. Notice that there are titles, the map, the legend, and some explanatory text. To get all these pieces where we want them on the page, we will use the following ODS tools:

- ODS PDF – sends output to a PDF file

- ODS LAYOUT and ODS REGION – allows specification of predefined regions on the page where output is placed
- Inline styles – allows use of CSS-like style attributes to control text (font, color, size, etc) and other “cell” attributes
- ODS TEXT – writes text to the page. This text can be formatted (with style references) and has no restriction on length.

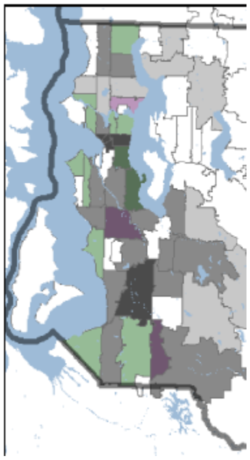

King County ADEpT GIS
Selected Measure by Zip Code
Percent Change in Measured Rate between 2004 and 2006
Percent Change in Measured Rate
Legend
Distribution
21 and above
14 (18.2%)
13 to 20
5 (6.5%)
5 to 12
7 (9.1%)
-4 to 4
10 (13.0%)
-5 to -12
10 (13.0%)
-13 to -20
6 (7.8%)
-21 and below
10 (13.0%)
Suppressed
15 (19.5%)
Notes:
The percentage change in a rate is defined as (Rate for Later Period-Rate for Earlier Period)/Rate for Earlier Period. In other words, if the rate for an earlier period was 100/100,000 population and the later rate was 200/100,000 population, the change in rate would be 100% [((200-100)/100)*100].
The histogram to the left presents the distribution of change in rates. For example, the top bar indicates that 14 zip codes had a percentage change of 21 or above, and those 14 zip codes accounted for 18% of the zip codes in the county.
White areas on the map indicate places where the number of cases were less than 20. In those areas the rates were suppressed because problems in reliability arise when rates are calculated with small numerators.

```
filename urlimage "c:\temp\change map.gif";
filename legend "c:\temp\change legend.gif";

ods pdf file=<filename> style=journal;

title1 f='Arial/bold' h=14pt 'King County ADEpT GIS';
<other titles>

ods layout start;

ods region x=.35in y=.2in height=7in width=7.5in;
ods pdf text= "~S={preimage=urlimage}";

ods region x=.5in y=6.2in height=4in width=7.5in;
ods pdf text= "~S={preimage=legend}";

ods region x=4.5in y=6.2in height=3.5in width=3.5in;
ods pdf text="<text to right of legend>";

ods region x=.5in y=9in height=1in width=3.2in;
ods pdf text="<text below legend>";

ods layout end;

ods _all_ close;
ods listing;
```

You can see in the code above that we do not have any SAS procedure output. As we did with the GSLIDE to create the legend image, a FILENAME statement provides a reference to both the map image (change map.gif) and the legend (change legend.gif). The map and legend are then placed on the page with a PREIMAGE style attribute to place the graphic files in the “cell” defined by the ODS REGION. For much more information on ODS, see the reference to *PDF Can be Pretty Darn Fancy…* in the Resources section.

The Evolution of a Legend

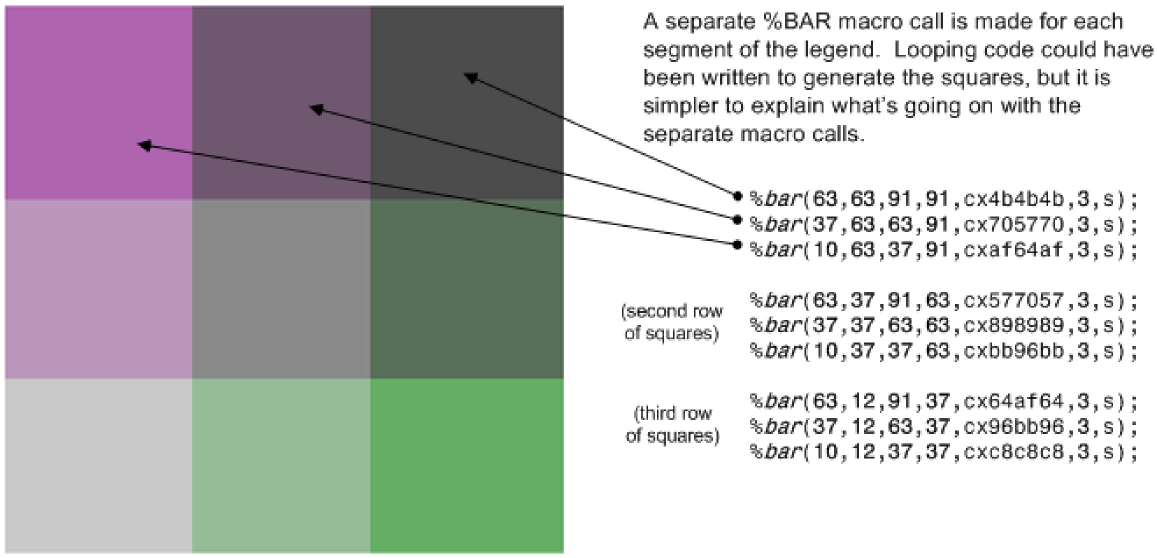
In another report, two measures are selected and every area (i.e., zip code) is ranked as high, medium or low for each of them. The medium range is defined as values within ½ standard deviation of the mean. The high range is more than ½ standard deviation above the mean and the low is less than ½ standard deviation below the mean. A map displays the nine possible combinations (high/high, high/medium, high/low, medium/high, medium/medium, medium/low, low/high, low/medium, and low/low).



The large number of values makes a legend a challenge, but the natural 3x3 pattern led to a choice of display that evolved to become a standalone analytic tool. The first goal of a legend is to explain what’s on the map. In this case, we wanted it to be clear that the low values were light and high values were dark, that the values for one measure tended to be green and for the other tended to be purple.

The portion of the map shown here demonstrates the nine colors plus, as we had in the first example, the white suppressed areas.

Again, the map is not created by SAS/Graph so we turned to Annotate to create a legend for the report. The first attempt at a legend was just to create a set of nine colored squares with %BAR macro calls, similar to what we've already seen.



A separate %BAR macro call is made for each segment of the legend. Looping code could have been written to generate the squares, but it is simpler to explain what's going on with the separate macro calls.

```
%bar(63,63,91,91,cx4b4b4b,3,s);
%bar(37,63,63,91,cx705770,3,s);
%bar(10,63,37,91,cxaf64af,3,s);
```

(second row of squares)

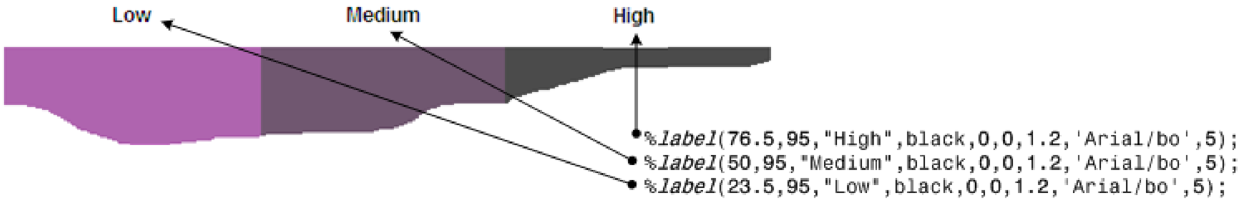
```
%bar(63,37,91,63,cx577057,3,s);
%bar(37,37,63,63,cx898989,3,s);
%bar(10,37,37,63,cxbb96bb,3,s);
```

(third row of squares)

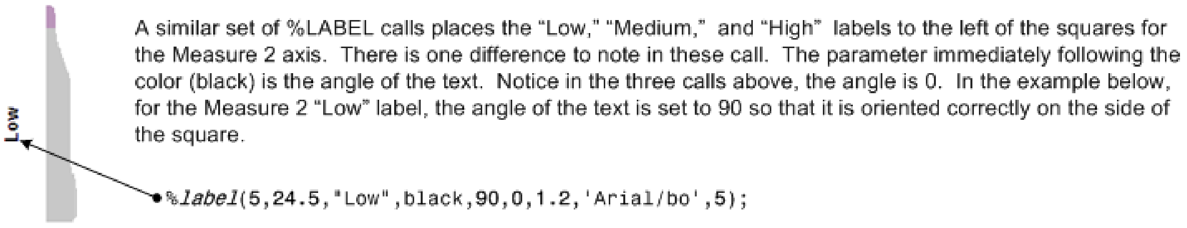
```
%bar(63,12,91,37,cx64af64,3,s);
%bar(37,12,63,37,cx96bb96,3,s);
%bar(10,12,37,37,cxc8c8c8,3,s);
```

The question then was how to label the squares. We could put the "Low," "Medium," and "High" labels inside the bars themselves, but since there are two labels for each that could be confusing. We decided to place the labels above and to the left of the squares.

The %LABEL macro call generate the Low, Medium and High text that placed above the squares for Measure 1.



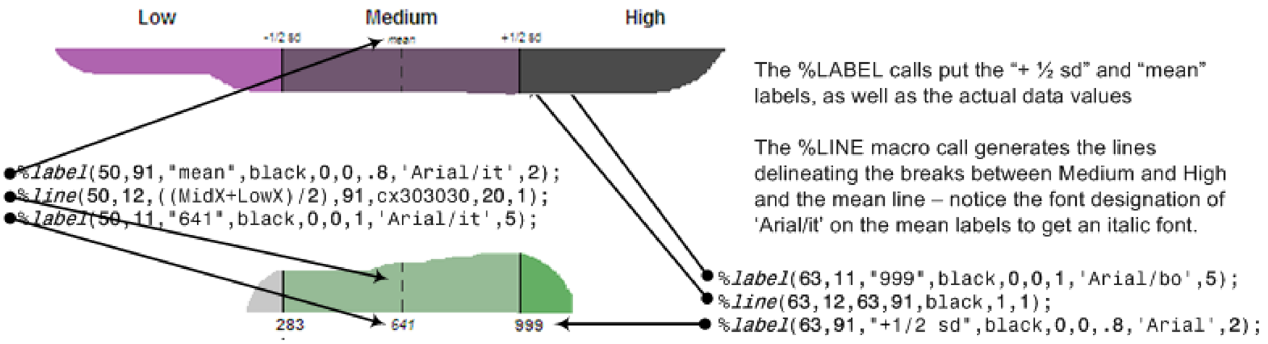
```
%label(76.5,95,"High",black,0,0,1.2,'Arial/bo',5);
%label(50,95,"Medium",black,0,0,1.2,'Arial/bo',5);
%label(23.5,95,"Low",black,0,0,1.2,'Arial/bo',5);
```



A similar set of %LABEL calls places the "Low," "Medium," and "High" labels to the left of the squares for the Measure 2 axis. There is one difference to note in these call. The parameter immediately following the color (black) is the angle of the text. Notice in the three calls above, the angle is 0. In the example below, for the Measure 2 "Low" label, the angle of the text is set to 90 so that it is oriented correctly on the side of the square.

```
%label(5,24.5,"Low",black,90,0,1.2,'Arial/bo',5);
```

Once these labels were on the legend it seemed prudent to add some more text to explain what labels like "High" meant. So, for example, we use another %LABEL macro calls to add "+ ½ sd, " "- ½ sd" and "mean" labels to the top of the squares. Other %LABEL calls add the actuals value to the same boundary on the bottom of the squares. We also use another annotate macro, %LINE, to add a black line between the color boundaries to help set them apart and another %LINE macro to put a dashed line along the mean. Notice that the parameter following the color (black) on the %LINE macros is the line type: 1 for solid, 20 for dashed.



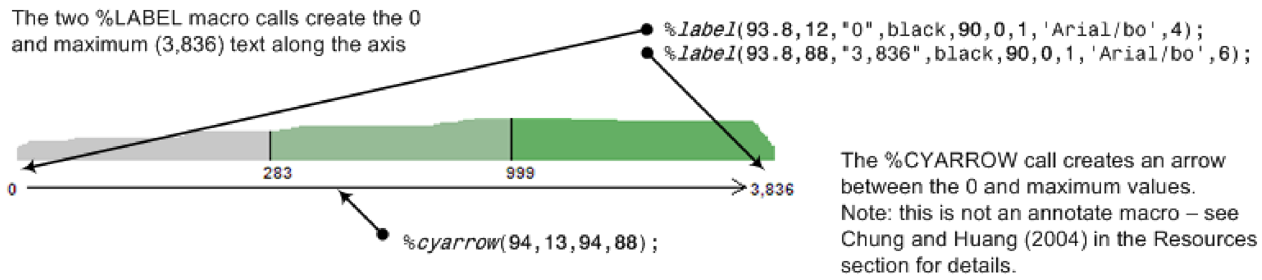
The %LABEL calls put the "+ ½ sd" and "mean" labels, as well as the actual data values

The %LINE macro call generates the lines delineating the breaks between Medium and High and the mean line – notice the font designation of 'Arial/it' on the mean labels to get an italic font.

```
%label(50,91,"mean",black,0,0,.8,'Arial/it',2);
%line(50,12,((MidX+LowX)/2),91,cx303030,20,1);
%label(50,11,"641",black,0,0,1,'Arial/it',5);
```

```
%label(63,11,"999",black,0,0,1,'Arial/bo',5);
%line(63,12,63,91,black,1,1);
%label(63,91,"+1/2 sd",black,0,0,.8,'Arial',2);
```

We'll use four sets of the standard deviation boundary macro calls; between low and medium and between medium and high on each axis. We'll also have two sets of the mean line/label macro calls; one for each axis. We've now added almost all the information about the actual values, except for one. The "high" range of values is open ended and the maximum values for the measures are not shown. Again, the %LABEL macro is used to put a "0" on the left edge of the legend square the maximum value on the right edge. The %CYARROW macro is called to place an arrow between the two values.



The two %LABEL macro calls create the 0 and maximum (3,836) text along the axis

```
%label(93.8,12,"0",black,90,0,1,'Arial/bo',4);
%label(93.8,88,"3,836",black,90,0,1,'Arial/bo',6);
```

The %CYARROW call creates an arrow between the 0 and maximum values. Note: this is not an annotate macro – see Chung and Huang (2004) in the Resources section for details.

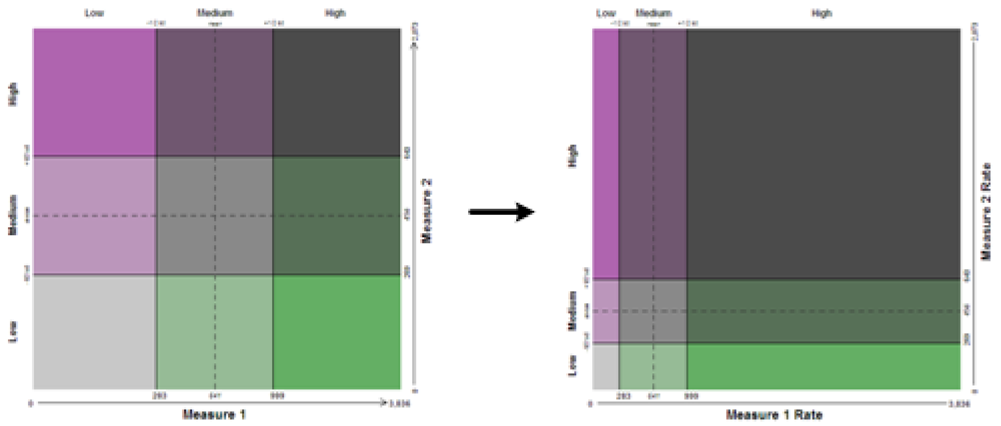
```
%cyarrow(94,13,94,88);
```

The final task is to label the axes – two additional %LABEL statements are used to place the text "Measure 1" below the arrow on the bottom of the legend and "Measure 2" next to the arrow on the right side of the legend.

BUT IT'S NOT QUITE RIGHT

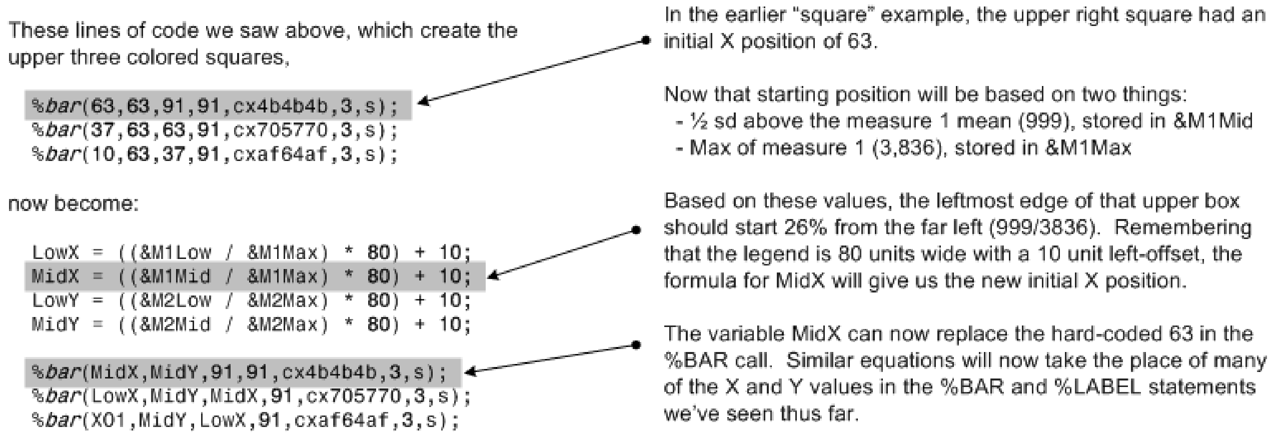
We've used a total of 9 %BAR, 24 %LABEL, 6 %LINE and 2 %CYARROW macro calls – a total of 41. All the information is there, but something seems a little off. Now that we've added the numbers, it's quite obvious that the "squares" shouldn't be – the ranges are not of equal size. The next step in the process was to take all the X,Y parameters that we've hard-coded so far and make them based on the actual values of the data so that the colored areas of the legend are proportional to the data.

On the left is the legend as it's been described so far and on the right is what we get when the size of the colored areas and the placement of the lines are based on the data values. Remember, there's no



dataset used here – all this is done in a single pass through a datastep. Here's how that was done.

The mean, +½ standard deviation, -½ standard deviation and maximum values for each of the two measures are stored in macro variables. For example, &M1Mean, &M1Low, &M1Mid and &M1Max are these values for the Measure 1 variable.



These lines of code we saw above, which create the upper three colored squares,

```
%bar(63,63,91,91,cx4b4b4b,3,s);
%bar(37,63,63,91,cx705770,3,s);
%bar(10,63,37,91,cxaf64af,3,s);
```

now become:

```
LowX = ((&M1Low / &M1Max) * 80) + 10;
MidX = ((&M1Mid / &M1Max) * 80) + 10;
LowY = ((&M2Low / &M2Max) * 80) + 10;
MidY = ((&M2Mid / &M2Max) * 80) + 10;

%bar(MidX,MidY,91,91,cx4b4b4b,3,s);
%bar(LowX,MidY,MidX,91,cx705770,3,s);
%bar(X01,MidY,LowX,91,cxaf64af,3,s);
```

In the earlier "square" example, the upper right square had an initial X position of 63.

Now that starting position will be based on two things:
- ½ sd above the measure 1 mean (999), stored in &M1Mid
- Max of measure 1 (3,836), stored in &M1Max

Based on these values, the leftmost edge of that upper box should start 26% from the far left (999/3836). Remembering that the legend is 80 units wide with a 10 unit left-offset, the formula for MidX will give us the new initial X position.

The variable MidX can now replace the hard-coded 63 in the %BAR call. Similar equations will now take the place of many of the X and Y values in the %BAR and %LABEL statements we've seen thus far.

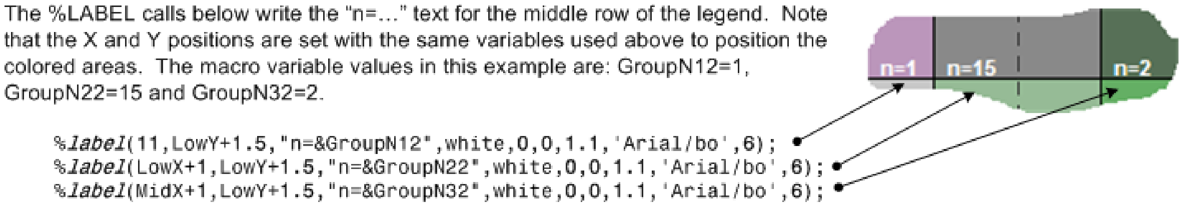
The bottom left corner of the legend square is not at 0.0, in order to have room for the text labels (for example, "Low," "Medium," "High). Also, we need room on top and to the right as well. So, the offset is 10 (out of 100) units in all directions, leaving the colored box to be 80x80.

All that is said to make sense of the equations for calculating the X and Y parameters of the %BAR, %LABEL and %LINE macros. The above example should help make this a little more clear.

ADDING SOME DATA TO THE LEGEND – PART 1

There are two ways we are going to add some data-specific information to this legend. First, similar to adding a histogram to our first legend, we will add a text entry to each colored area of the legend with the count of data points for that legend. Once again, we will use %LABEL statements to do that.

In addition to the macro variables containing the + and – ½ standard deviation and maximum values, there are also a set of nine macro variables that contain the count for each "cell" of the legend. These variables are called GroupN11 – GroupN33, with the first digit representing the horizontal location (1=left, 2=middle, 3=right) and the second digit representing the vertical location (1=bottom, 2=center, 3=top). So, GroupN12 would be the left center cell of the legend.



The %LABEL calls below write the "n=…" text for the middle row of the legend. Note that the X and Y positions are set with the same variables used above to position the colored areas. The macro variable values in this example are: GroupN12=1, GroupN22=15 and GroupN32=2.

```
%label(11,LowY+1.5,"n=&GroupN12",white,0,0,1.1,'Arial/bo',6);
%label(LowX+1,LowY+1.5,"n=&GroupN22",white,0,0,1.1,'Arial/bo',6);
%label(MidX+1,LowY+1.5,"n=&GroupN32",white,0,0,1.1,'Arial/bo',6);
```

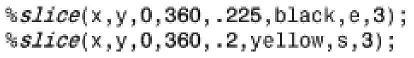
Notice that the text values are offset slightly from the positions used to draw the colored areas and the lines (+1.5) so that the text does not hit the lines.

This report, like the first one, has suppression rules and the areas that are suppressed remain white on the map. But, unlike the first legend, there really isn't a place on this legend to reference this. So, we'll just use one more %LABEL call to create a line of text that notes the number of suppressed areas that are on the map.

Adding Some Data to the Legend – Part 2

Up to now we've written 51 macro calls that create a dataset with 82 observations. All of them are executed in a single iteration of a datastep – there is no input dataset needed. Now's the time for some fun – we'll add a lot more information to the legend, using only two additional macro calls, with the dataset that's used to create the map.

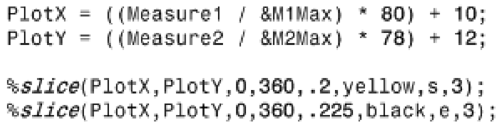
The map dataset contains one observation for each area on the map and has the values of the two selected measures (Measure1 and Measure2). We're going to use two calls an Annotate macro that we haven't used yet, the %SLICE macro.



```
%slice(x,y,0,360,.225,black,e,3);
%slice(x,y,0,360,.2,yellow,s,3);
```

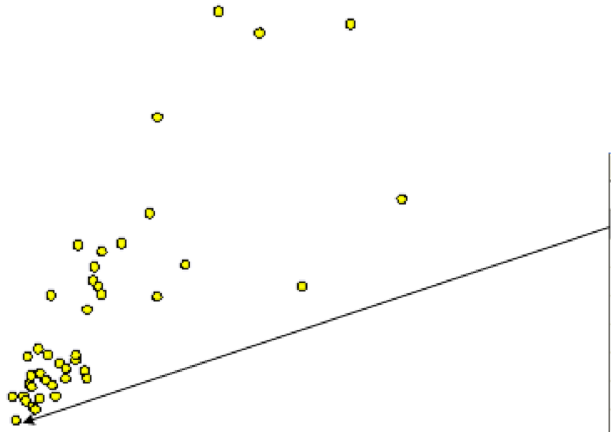
The %SLICE macro creates a "pie" slice – here we want slices that start at the 0 position and continue for 360°, or a complete circle.

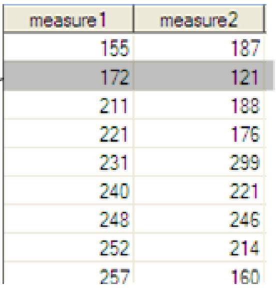
The two calls gives us a yellow "dot" that is set off a bit by the black outline. If we call those two macros for each observation in the map dataset, we get annotate instructions that will draw a dot for each area on the map.



```
PlotX = ((Measure1 / &M1Max) * 80) + 10;
PlotY = ((Measure2 / &M2Max) * 78) + 12;

%slice(PlotX,PlotY,0,360,.2,yellow,s,3);
%slice(PlotX,PlotY,0,360,.225,black,e,3);
```

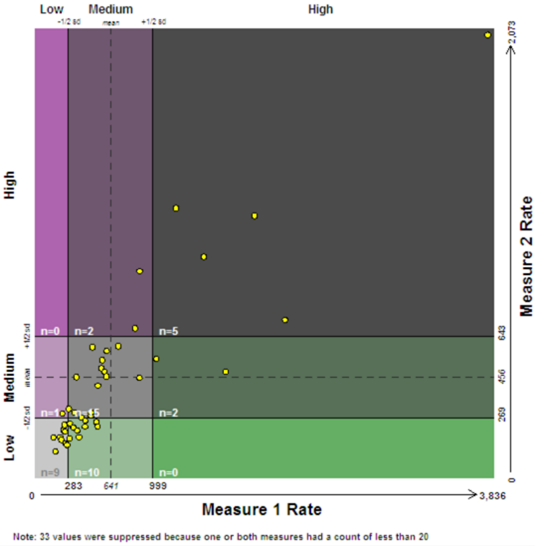
The dataset below contains one observation for each area on the map, with the values for the two selected measures. As we loop through the data, the code to the left is executed. The ratios of the Measure1 and Measure2 values to their maximums are computed and used to determine where to place the dot for that area. By using the same size and offset that we did for the legend, the pattern of dots fits the same size area that the legend does.





measure1	measure2
155	187
172	121
211	188
221	176
231	299
240	221
248	246
252	214
257	160

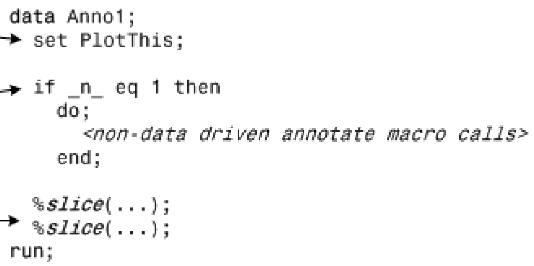
For example, the maximum value for Measure1 is 3,836 and for Measure2 is 2,073. The dot for the highlighted area should be 4.5% (172/3836) from the left and 5.8% (121/2073) from the bottom of the legend area.



Note: 33 values were suppressed because one or both measures had a count of less than 20

ata

ry



```
data Anno1;
  set PlotThis;

  if _n_ eq 1 then
    do;
      <non-data driven annotate macro calls>
    end;

  %slice(...);
  %slice(...);
run;
```

Because all the observations are in the same dataset and scaling of the legend colored areas and the placement of the dots are based on the same algorithms, the dots and the legend can overlay each other. We now have a legend that not only explains the colors on the map, but provides a scatter plot of the distribution of data points as well.

We ended up writing a data step with total of 53 macro calls which generated 170 observations containing annotate instructions – not really a lot of work to get this legend.

The same techniques as noted in the first example were used to create the files and generate the report. The GSLIDE procedure was used to turn the annotate instructions into a GIF file. That file, along with the map and explanatory text were placed into a PDF inside ODS LAYOUT regions with ODS TEXT statements. As above, the PREIMAGE style attribute was used to reference and place the image files on the page. A copy of the final report can be found in *Appendix B – Comparison Map Example* at the end of the paper.

CONCLUSION

A legend can do so much more than just reference the colors or symbols on a map or chart. We've seen a couple examples of legends that have some descriptive and analytic capabilities of their own. The SAS/Graph Annotate facility put an incredible amount of power and flexibility at the programmer's disposal,

The advent of ODS LAYOUT to place data anywhere on a page, in the printer destinations, allows for creation of reports that mix text and graphics – and maybe even your own legend!

RESOURCES

There are a couple of resources that will help you in your quest to learn about the SAS/Graph Annotate facility.

Annotate: Simply the Basics, by Art Carpenter, SAS Publishing, 1999

The SAS Online Docs as http://support.sas.com/onlinedoc/913/docMainpage.jsp

- Click on SAS/Graph Reference and then, The Annotate Facility

For some more details on using ODS and PDF in general and ODS LAYOUT in particular:

PDF can be Pretty Darn Fancy: Advanced ODS Options for PDF Output, by Pete Lund, Proceedings of the 31st Annual SAS Users Group International, San Francisco, CA, 2006 (http://www2.sas.com/proceedings/sugi31/092-31.pdf)

ACKNOWLEDGEMENTS

Thanks to David Kelley and the ODS guys at SAS: Dan O'Connor, Eric Gebhart, Kevin Smith, Scott Huntley, and Tim Hunter and to Chevell Parker in SAS Tech Support. Your encouragement to the reporting that we do and responses to questions and issues is always appreciated.

AUTHOR CONTACT INFORMATION

Please let me know other things you've thought of to do with the Annotate Facility!

Pete Lund
Looking Glass Analytics
215 Legion Way SW
Olympia, WA 98501
360-528-8970 voice

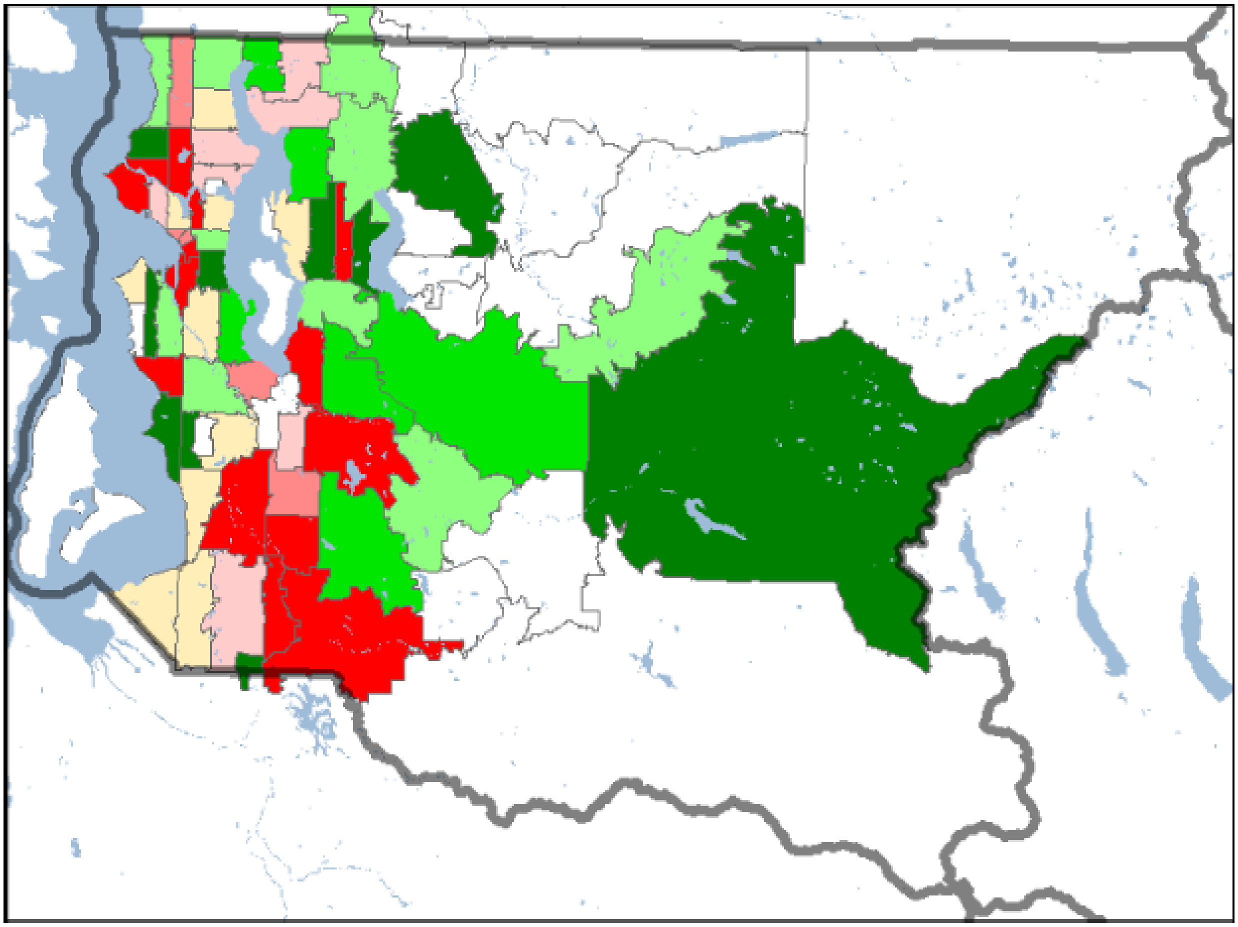
360-570-7533   fax
pete.lund@lgan.com
www.lgan.com

Appendix A – Change Map Report Example

King County ADEpT GIS

Selected Measure by Zip Code

Percent Change in Measured Rate between 2004 and 2006



Percent Change in Measured Rate

Legend		Distribution
🟥	21 and above	14 (18.2%)
🟥	13 to 20	5 (6.5%)
🟥	5 to 12	7 (9.1%)
🟨	-4 to 4	10 (13.0%)
🟩	-5 to -12	10 (13.0%)
🟩	-13 to -20	6 (7.8%)
🟩	-21 and below	10 (13.0%)
□	Suppressed	15 (19.5%)

White areas on the map indicate places where the number of cases were less than 20. In those areas the rates were suppressed because problems in reliability arise when rates are calculated with small numerators.

Notes:

The percentage change in a rate is defined as (Rate for Later Period-Rate for Earlier Period) /Rate for Earlier Period. In other words, if the rate for an earlier period was 100/100,000 population and the later rate was 200/100,000 population, the change in rate would be 100% [((200-100)/100)*100].

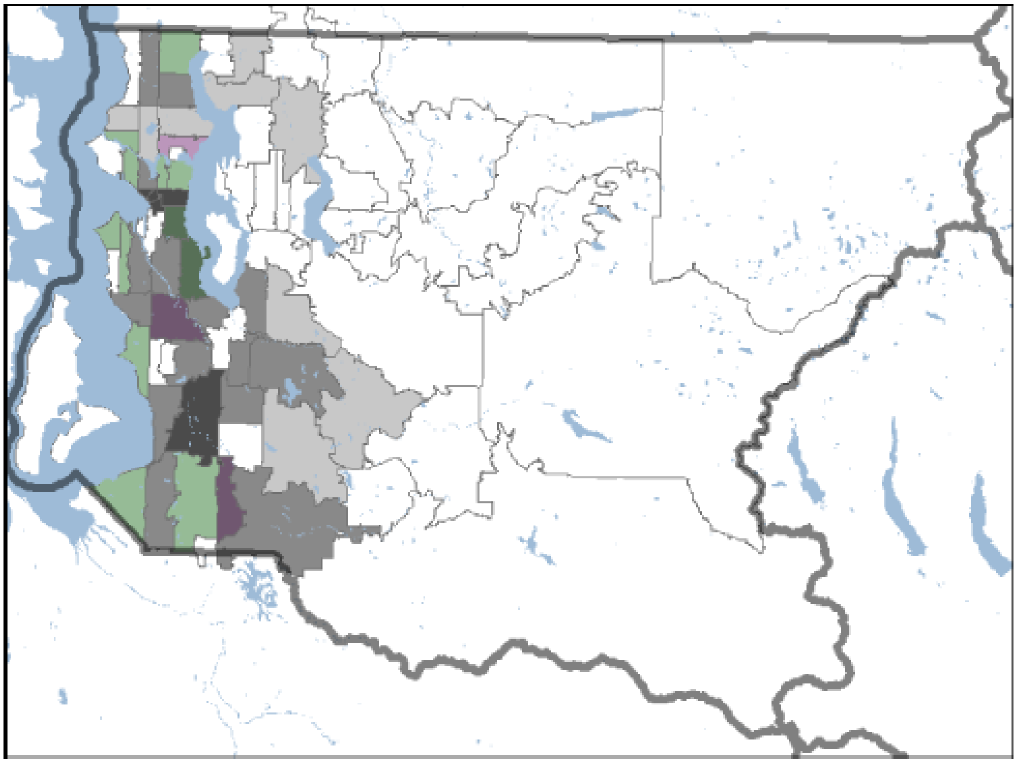
The histogram to the left presents the distribution of change in rates. For example, the top bar indicates that 14 zip codes had a percentage change of 21 or above, and those 14 zip codes accounted for 18% of the zip codes in the county.

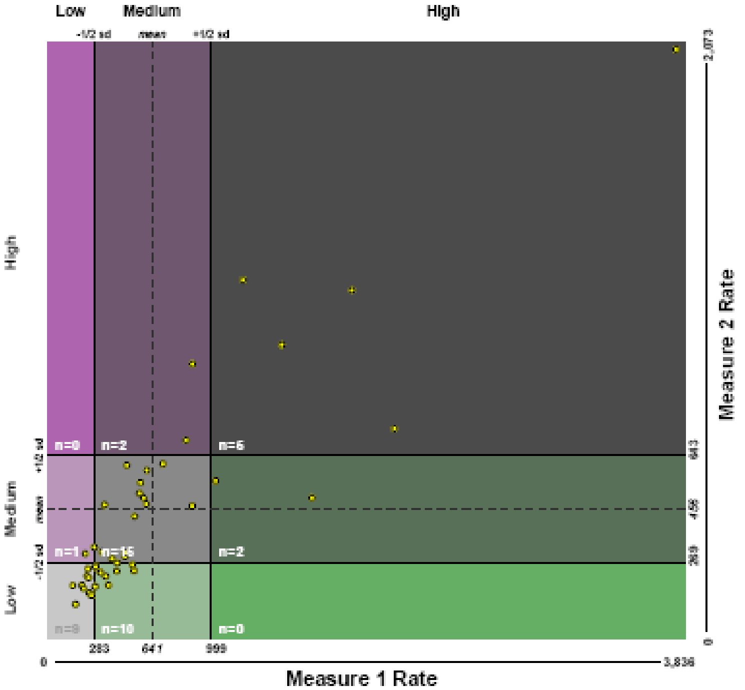
Appendix B – Comparison Map Example

King County ADEpT GIS

Classification of
Measure 1 Rates and Measure 2 Rates

2006 - by Zip Code





Note: 33 values were suppressed because one or both measures had a count of less than 20

Notes:

The map displays zip codes across King County based on their cross-classification on two user-selected measures. Each zip code is classified as falling into one of three categories on each measure, either high, medium or low. Areas fall within the medium category if their value on a particular measure is within one-half of a standard deviation above or below the mean. High is defined as being greater than one-half of a standard deviation above the mean and low as one-half of a standard deviation below the mean. The perpendicular dotted lines show the value of each mean and their intersection, as well as the range of values defining each category. Points in the legend identify the relative location of each zip code within this classification scheme. Nine different colors uniquely symbolize all possible cross-classifications.

Suppression occurs when rates are based on fewer than 20 admissions and zip codes where one or more rates are suppressed are shown in white.

The table on the following page(s) displays counts and rates for each zip code in the county. Micro graphs show the relative location of each user-selected measure.