

Paper 221-2009

Make Your Tables Pop: Embedding Micrographics in PROC REPORT Output

Pete Lund, Looking Glass Analytics, Olympia, WA

ABSTRACT

It's a very common thing to have a table of information included in a report. It's also very common to have graphics, such as bar charts, in the same report. This paper will look at techniques, available in SAS®, to place the graphics right in the tables.

The SAS/Graph Annotate facility will be used to create small, standalone graphics files. The files can then be placed in a "cell" of a table created by PROC REPORT. This technique allows you to create tables that draw the reader's eyes to the important information much more quickly than scanning through all the text.

INTRODUCTION

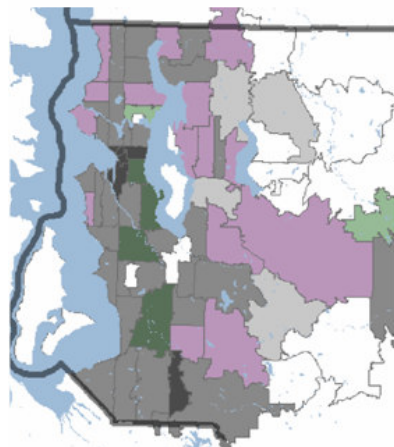
The examples presented in the paper come from a project, funded by the National Institute on Drug Abuse (NIDA), which creates reports detailing measures of alcohol and drug epidemiology. The project uses a web-based user interface, where different measures, time frames and levels of geography can be 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. In many of the reports the tables contain embedded micrographics which are created with the SAS/Graph Annotate facility.

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

AS SIMPLE AS A SQUARE

If we have a map displaying a number of colored regions, we can add something as simple as a colored square to an accompanying table to aid in identifying with which group an table row is associated. This means the reader does not have to go back and forth between the table and the map legend. The map to the left shows the relationship the values of two selected measures for ZIP codes in King County, WA. The darkest polygons are those ZIP codes that ranked high on both measures, the light gray are those that ranked low on both measures. The other colors represent different combinations of high/medium/low on the two measures. (Note: white polygons indicate where results were suppressed due to insufficient data.)



Partial map of King County, WA

On the page(s) following the map is a table which contains the details of the two measures. We could have used the PROC

REPORT code shown below to produce a table with all the information contained in the map. As you can see from table that was created, all the data is in the table, but it's kind of plain and boring.

It doesn't do anything but present the numbers. The code is simple, with no real advanced features other than spanning column headers.

Note: In all of the examples presented in this paper, a number of ODS and formatting options have been removed from the code for the sake of simplicity. As these features become necessary to explain a technique, they will be included in the example code.

```
proc report data=ZIPCodeData nowd;
  columns ZipCode
    ("Selected Measure1" M1Count M1Rate)
    ("Selected Measure2" M2Count M2Rate);

  define ZipCode / 'Zip Code' group;
  define M1Count / 'Count';
  define M1Rate / "Crude Rate";
  define M2Count / 'Count';
  define M2Rate / "Crude Rate";
run;
```

Straightforward PROC REPORT code produces the table to the right – nothing fancy and no easy way to tie the table back to the map, especially if you don't know the location of the ZIP codes.

Zip Code	Selected Measure1		Selected Measure2	
	Count	Crude Rate	Count	Crude Rate
98001	110	392	214	762
98002	286	926	409	1,325
98003	146	336	407	935
98004	31	127	181	743
98005	22	129	99	581
98006	37	102	159	438
98007	58	251	139	603
98008	43	179	136	568
98010	5	n/c	39	719
98011	68	233	198	680

...partial results

The table simply displays the counts and rates for the two measures reported on the map. Wouldn't it be nice if there was something on the table that related back to the colors on the map? That's what we'll do in the next iteration of the report code.

There have been numerous papers presented on different techniques for "traffic lighting" reports. These methods change the color of the text or cell background based on the value of the data. It is most often used to highlight a handful of values that exceed some threshold, making them easy to spot in the table.

In addition to the count and rate variables, there is a variable (ColorGroup) which identifies which of the ten colors should be displayed on the map. The format to the right identifies the colors that go with each of these values.

ZIPCode	ColorGroup	M1Count	M1Rate	M2Count	M2Rate
98001	22	110	391.57	213.98	761.705
98002	33	286	926.4401	409.19	1325.48
98003	22	146	336.6309	406.61	934.724
98004	12	31	127.2806	180.75	743.000
98005	12	22	129.0211	99.064	580.999
98006	11	37	101.8464	159.05	437.798
98007	22	58	250.9961	139.42	603.348
98008	12	43	179.2332	136.000	568.000
98010	99	5	n/c	39.000	719.000
98011	22	68	233.3943	198.22	680.338

```
proc format;
  value $PlotColors
    '11' = "cxc8c8c8"
    '12' = "cxbb96bb"
    '13' = "cxaf64af"
    '21' = "cx96bb96"
    '22' = "cx898989"
    '23' = "cx705770"
    '31' = "cx64af64"
    '32' = "cx577057"
    '33' = "cx4b4b4b"
    '99' = 'white';
run;
```

We'll use a similar technique here, but we're going to add another column on the far left of the table which is a square of color matching the color for the ZIP code on the map.

It is now easy see which map group the ZIP code falls into without having to refer back to the map legend. The PROC REPORT in Code Sample 2 has two additional columns of data. It might seem that we'd only need to add the column which will be used to display the color block, but we need one additional variable as well. Let's look at what we did above:

```
proc report data=ZIPCodeData nowd;
  columns ZipCode=RowOrder ColorGroup ZipCode 1
    ("Selected Measure1" M1Count M1Rate)
    ("Selected Measure2" M2Count M2Rate);

  define RowOrder / '' group noprint; 2
  define ColorGroup / '' group
    style=[background=$PlotColors. foreground=$PlotColors.]; 3
  define ZipCode / 'Zip Code' group;
  define M1Count / 'Count';
  define M1Rate / "Crude Rate";
  define M2Count / 'Count';
  define M2Rate / "Crude Rate";
run;
```

We've added two new variables to the column statement and a little trick in the DEFINE statement to get the colored squares in the table.

Zip Code	Selected Measure1		Selected Measure2	
	Count	Crude Rate	Count	Crude Rate
98001	110	392	214	762
98002	296	926	409	1,325
98003	146	336	407	935
98004	31	127	181	743
98005	22	129	99	581
98006	37	102	159	438
98007	58	251	139	603
98008	43	179	136	568
98010	5	n/c	39	719
98011	68	233	198	680

...partial results

1 – Two new variables are added to the COLUMNS statement: RowOrder and ColorGroup. RowOrder is actually an alias (and copy) of ZipCode. ColorGroup contains the color value for the zip.

2 – The RowOrder variable is not printed on the report, but is necessary to retain the zip code order of the table. PROC REPORT orders rows based on the left-most column(s). The ColorGroup is the first that is displayed in the table, but we don't want the rows ordered by that variable. Placing a "copy" of zip code first allows for the rows to remain in the desired order.

3 – The column displays the value of the ColorGroup. But, we don't want to see the actual numeric value, we just want a square of the corresponding color. The STYLE= option allows for defining a number of formatting attributes for the PROC REPORT "cell." In this case, we want to set the background color to the color that goes with the ColorGroup value, using the \$PlotColor format. Also, we'll set the foreground to the same color – this is the color of the text. So, the text is still printed but it has the same color as the background and we get a solid colored square.

ANOTHER EXAMPLE OF COLOR – WITH TEXT

Another report in the project ranks areas on a selected measure from the area with the highest rate to the one with the lowest. The areas are grouped into five categories, each containing about 1/5 of the values. The following example shows reports that rank a measure across school districts.

```
proc report data=Ranks nowd;
  columns RateRank SchDist
    ("Selected Measure" VarCount VarRate);

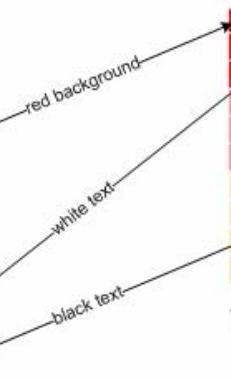
  define RateRank / "Rank" group 1
    style=[background=colorlist. foreground=textcolor.];
  define SchDist / "School District";
  define VarCount / "Count";
  define VarRate / "Rate*";
run;
```

```
proc format;
  value ColorList
    1-3 = "cxff0000"
    4-6 = "cxff9999"
    7-10 = "cxffeeb8"
    11-13 = "cx9999ff"
    14-16 = "cx0000ff"
    17-19 = "white";

  value TextColor
    1,2,3 = 'white'
    other = 'black';
run;
```

Rank	School District	Selected Measure	
		Count	Rate*
1	Auburn	523	720
2	Seattle	3,245	560
3	Highline	662	543
4	South Central	90	539
5	Kent	785	524
6	Renton	429	407
7	Shoreline	225	344
8	Federal Way	424	338
9	Tahoma	69	198
10	Snoqualmie Valley	61	183

*Rate per 100,000 Population



The same technique we've just described will be used to place a colored square to the left of the district, with each rank group having a different color. In this case, however, we want the rank order value to be displayed inside the square.

The formats in the example above are actually dynamically generated based on the number of geographic areas to be displayed. In the example above, there are 19 school districts shown in the table, and on the map. Three of these ended up with values being suppressed and will have a white color. The other 16 are split into five groups; four of these groups have three values each and the other has four.

1 – We again use the STYLE= option on the DEFINE statement to get a background color for the cell, using the ColorList format. As noted, we want to see the values of the RateRank variable, so we don't want to use the same format for the foreground color, as we did above. If we left the FOREGROUND attribute out of the style designation, the text color would default to black. Since some of the background colors are dark we can't just let the text color remain black or it wouldn't show clearly against the darker background colors. So, we'll use a different format for the text color which sets the text to white when the background is dark and to black for the lighter backgrounds.

EMBEDDING ACTUAL GRAPHICS IN PROC REPORT

The first page of the report mentioned above contains a map, a legend and the "top ten" rank table just shown. The following pages of the report show a table of all of the geographies on the map, in rank order. As in our first example, we could just present the data that goes into the map. Everything is there

```
proc report data=RankMainTable nowd;
  columns RankVal SchDist
           ("Selected Measure" VarCount VarRate);

  define RankVal / 'Rank' group;
  define SchDist / '' group "School District";
  define VarCount / 'Count';
  define VarRate / "Crude Rate*";
run;
```

Note: One thing is omitted from the above code - to get the colored background in the spanning column header an "inline style" designation is made. An escape character is defined with the following statement:

```
ODS ESCAPECHAR='~';
```

The escape sequence S={...} defines style attributes for the text that follows. The column header text above was created with the following code:

```
"~S={background=cxe0e0e0}Selected Measure"
```

The escaped text does not show up on the page, but rather the gray background color (cxe0e0e0) is applied.

See the note to the left of the table to see how the colored background of the "Selected Measure" header was done.

Rank	School District	Selected Measure	
		Count	Crude Rate*
1	Auburn	523	720
2	Seattle	3,245	560
3	Highline	662	543
4	South Central	90	539
5	Kent	785	524
6	Renton	429	407
7	Shoreline	225	344
8	Federal Way	424	338
9	Tahoma	69	198
10	Snoqualmie Valley	61	183
11	Enumclaw	49	182
12	Riverview	33	180
13	Lake Washington	266	159
14	Bellevue	183	155
15	Issaquah	93	108
16	Northshore	117	101
n/a	Mercer Island	5	n/c
n/a	Skykomish	8	n/c
n/a	Vashon Island	11	n/c

for the reader to see, but it's rather dull.

However, what if we could embed a histogram, showing the relative values of the ranked measure, right in the PROC REPORT output?

There are two things we need to do here: first, we need to make the graphics and then, we need to get them into the table. We will use the Annotate facility of SAS/Graph to create a graphic for each district in the report, dividing the rate by the maximum rate among all districts to get the relative rate that will be displayed in the bars..

A QUICK GUIDE 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 statement and 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 all 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 x,y 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 command 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 and a “bar” command. 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

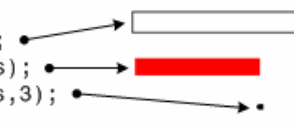
As noted above, Annotate data sets are often used with SAS/Graph procedures that generate output. The ANNO= option on the procedure statement references the dataset to be used. Annotate data sets can be displayed on their own with the GSLIDE procedure. GSLIDE creates no output of its own, but will display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

CREATING THE RANK BARS

Let's say we're building the bar for people living in the Highline School District in the report above. Its rate on the measure selected for the report is 543 per 100,000 total population. We see that the highest rate among the districts is 720. This means that the Kent rate is 75% of the maximum rate, so we want a bar that extends across 75% of the black box. The other piece of information that we need is the color of the bar – in this case, the bar will be red.

```
data Anno;
  ❶ %bar(0,0,100,100,black,0,e);
  ❷ %bar(0,10,75,90,cxff0000,0,s);
  ❸ %slice(75,50,0,360,6,black,s,3);
run;
```



There are actually three things we need to do to create the rank bars that we will embed into the PROC REPORT output. Each of these tasks can be done with a call to one of the Annotate macros and can be seen in the simple data step below:

1 – The first %BAR macro writes the instructions for creating an empty black rectangle that serves as a border. The x,y coordinates for the corners of the bar are 0,0 and 100,00.

2 – The second %BAR macro will create a colored bar, the width of which is proportional to the current observation relative to the maximum rate for the report. In this case, the Highline rate of 543 is 75% of the maximum rate of 720. The x,y coordinates of the colored portion are 0,0 and 75,100 creating a bar that is 75/100 the width of the outline box.

The color of the bar (cxff0000, red) is based on the value as well, depending on which fifth of the rankings it falls. In the real code, both the width and the color are passed as macro variables – more on this in the next section.

3 – The %SLICE macro creates a “pie” slice. Here, we're creating a 360° slice, a complete circle, which is placed at 75,50. This is halfway up the end of the colored bar. The “dot” is placed there to make the end of the bars easy to identify if a black and white printer does not render the different colors well.

What this little data step does is create a dataset with five observations containing the instructions to create a graphic file. To actually create the graphic requires one additional step.

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 Bar "...\\Highline.gif";
goptions device=gif gsfname=Bar hsize=1.5in vsize=.2in;
proc gslide anno=anno;
run;
quit;
```

Specify the path and name of the graphic image

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

Specify the dataset containing the annotate instructions

The above code would create a gif image (Highline.gif), shown here.



Please note that in the actual application that produces these reports the code above is included in a macro that is called for each geography value (i.e., school district). A number of the annotate macro parameters are either dataset variables or macro variables, as is the name of the gif file to be created. Hard-coded values were used in the example for the sake of clarity. In reality, there would be a separate gif file created for each school district, with the bar size, color and gif file name all passed as macro parameters.

USING THE RANK BARS

How do we get those to display in a table? It's actually quite simple, as you can see in the PROC REPORT code below, which is only slightly modified from what we saw above.

```
proc report data=RankMainTable nowd;
  columns RankVal SchDist
           ("Selected Measure" VarCount VarRate ShowBar);

  define RankVal / 'Rank' group;
  define SchDist / '' group "School District";
  define VarCount / 'Count';
  define VarRate / "Crude Rate*";
  define ShowBar / "Relative Rate" computed;

  compute ShowBar / char;
    call define(_col_, 'style', "style=[preimage=' " || SchDist || ".gif']");
  endcomp;
run;
```

1 – We've added one new variable to the COLUMNS statement. It does not exist in the incoming dataset. It's included inside the parentheses with the count and rate variables so the "Selected Measure" header will cover it as well.




















2 – The DEFINE statement identifies this new variable as a COMPUTED column, though it's not necessary to be explicit about this, and we give the column some header text.

3 – Because it's a computed variable, we need to have a COMPUTE block of code to define it. Normally we would have some logic and an assignment statement to the a value to the new variable. Here, we're

just going to leave it with the default missing value. The CHAR designation makes ShowBar a character variable, so its missing value is a blank space rather than a period.

4 – Since we're not giving ShowBar a value, what do we do? We use CALL DEFINE to apply a style attribute to the cell. We've used STYLE to give background and foreground colors, now we'll use it to place an image in the cell. The PREIMAGE attribute is used to place an image before any other content in the cell. The contents of this cell are blank, so the image is all there is.

Remember, the image files have the same name as the school district. Since the SchDist column appears to the left of this column we have access to its value. We can just append a ".gif" to the SchDist value and the image for the current row is placed in the cell. The results can be seen in the table below.

Rank	School District	Selected Measure		
		Count	Crude Rate*	Relative Rate
1	Auburn	523	720	
2	Seattle	3,245	560	
3	Highline	662	543	
4	South Central	90	539	
5	Kent	785	524	
6	Renton	429	407	
7	Shoreline	225	344	
8	Federal Way	424	338	
9	Tahoma	69	198	
10	Snoqualmie Valley	61	183	
11	Enumclaw	49	182	
12	Riverview	33	180	
13	Lake Washington	266	159	
14	Bellevue	183	155	
15	Issaquah	93	108	
16	Northshore	117	101	
n/a	Mercer Island	5	n/c	
n/a	Skykomish	8	n/c	
n/a	Vashon Island	11	n/c	

This new column really adds some pop to the table. Since the table rows are in rank order you can see at a glance the pattern of the data. Also, you can see quite quickly that ranks 2-5 are virtually identical and that it drops off from there. Your eye is drawn to the numbers that are most important or interesting.

Also, notice the last three rows in the table have no bar. These are the suppressed districts where, due to small n's, notes were not calculated. Calling the annotate code above is problematic,

even if no colored bar or dot was produced, because we always get the black outline bar. There are a couple simple ways we could handle this. In the macro that contains the annotate code is a test for suppressed values. If this test is true, none of the annotate macro calls in the data step are made. In essence, you end up with this:

```
data Anno;
run;
```

This creates a dataset with no observations. When the GSLIDE reads this empty file it does just what you told it to do, creates an image file with nothing in it. So, even the suppressed rows have an image file – it's just empty.

ADDITIONAL INFORMATION IN THE GRAPHICS

The geography selection in this next report is different from the previous two examples. Rather than selecting a type of geography (like zip code or school district) the user selects a single area, such as zip code 98101, and receives a summary report for that area. Information on rates and ranks for over 20 measures is included in the report.

There is a lot of data crunching needed – not only are counts and rates for each measure provided, but also the rank for the selected area among all the areas for each measure. This means we have to process all the data, not just the data for the selected area.


```
proc report data=ReportRows nowd;
  columns Measure
    ("ZIP Code 98101 (Population: 10,448)" GeogPersons GeogRate ReportedRank) ❶
    ("King County (Population: 1,835,525)" CountyPersons CountyRate); ❷

  define Measure / '';
  define GeogPersons / 'N';
  define GeogRate / 'Crude Rate';
  define ReportedRank / "Rank";

  define CountyPersons / 'N';
  define CountyRate / 'Crude^Rate';
run;
```

	Zip Code 98101 (Population: 10,448)			King County (Population: 1,835,525)	
	N	Crude Rate ¹	Rank ²	N	Crude Rate
Primary Measures³	100	957	6 of 54	4,918	268
Primary A					
Sub-Measure A1	49	469	3 of 43	2,709	148
Sub-Measure A2	32	306	3 of 14	977	53
Sub-Measure A3	19	n/c	n/a	1,232	67
Primary B					
Sub-Measure B1	83	1,430	6 of 53	4,377	479
Sub-Measure B2	48	1,033	4 of 44	2,903	315
Primary C					
Sub-Measure C1	29	278	4 of 26	1,653	90
Sub-Measure C2	73	699	6 of 57	4,841	264
Sub-Measure C3	29	278	3 of 12	786	43
Secondary Measure A	80	766	2 of 26	1,600	87

Two sets of spanning headers covering:

- ❶ – the selected zip code values
- ❷ – the total county values

Notice the “~n” in the header text. As noted earlier, the tilde has been defined as an escape character. The escape sequence ~n inserts a new line character – this is how the two lines are created in the spanning header text.

You can see in the example to above that we have the count, rate and rank for each measure along with the same numbers for the county as a whole. Note also that suppression rules are in effect and that the rates and ranks are not displayed for any measure with a count of less than 20. This is also the reason that the number of ranked areas is different from measure to measure (i.e., “...of 54”, “...of 43”). These counts show the number of non-suppressed areas for each measure on the report.

Once again we will add a little graphic to add some pop to the table. This time, however, we’ll do a little more than we did above. In the previous example all the data necessary to create the graphic was already in the table. Here, the images that are created will add some additional information that is not presented in the numbers on the table.

We can see in the table above the rate of each measure. What we can’t see is how the rate, on which the ranks are based, measures up to the highest ranked zip code for a particular measure. So, here’s what we’ll do: for each measure the following macro variables are set:

- &GeogRate – the rate of the measure for the selected zip code
- &CountyRate – the rate of the measure for the county as a whole
- &MaxRate – the maximum rate of the measure among all zip codes

These values are used in creating the following annotate dataset.

```
data anno;
  ❶ %bar(1,1,100,31,cxffda6b,0,s);
  ❷ %bar(1,1,100,31,black,0,e);
  ❸ GP = (&GeogRate / &MaxRate) * 100;
  ❹ CP = (&CountyRate / &MaxRate) * 100;
  ❺ %bar(1,1,GP,31,cx6b90ff,0,s);
  ❻ %slice(CP,16,0,360,30/6,black,s,3);
run;
```

A very similar technique to that already described, but with a few things to note:

- ❶ – Create a solid bar to serve as the background color.

2 – As before, create a solid black outline bar.

Note: these two %BAR calls are in this order so that the black outline bar shows up on top of the yellow colored bar. The instructions are performed in the order of the dataset observations. Since the two bars have the same x,y coordinate pairs, which ever comes last will overlay on the first.

3 – Calculate the percentage of the maximum rate for the measure that this zip code has. In this example, it's 957 / 3087 or 31%.

4 – Do the same calculation for the county rate. Here it's 268 / 3087 or 9%.

5 – Use the GP value (see step 3 above) as the second X parameter in the %BAR call to create a blue colored bar that covers 31% of the yellow bar denoting the rate for this zip code relative to the maximum rate.

6 – Use a %SLICE macro call to place a black dot at the relative position of the county rate. The x position is based on the CP variable (see step 4 above).

Finally, in the same way as described above, use the GSLIDE procedure to create an image file, using the measure name as the file name. The data step above, once processed, would create the following image file:



PrimaryMeasures.gif

As before, the data step above is called for each measure that will be displayed in the table and a separate graphic is created for each.

INTO THE TABLE – ONE MORE TIME

We'll use the same technique described above to get the graphics into the PROC REPORT table. Below is the code shown above with the additions necessary to add the graphics to the table.

```
proc report data=ReportRows nowd;
  columns Measure
    ("ZIP Code 98101 (Population: 10,448)" GeogPersons GeogRate ShowGraph ReportedRank)
    ("King County (Population: 1,835,525)" CountyPersons CountyRate);

  define Measure / '';
  define GeogPersons / 'N';
  define GeogRate / 'Crude Rate';
  define ShowGraph / '0 -----> max';
  define ReportedRank / "Rank";

  define CountyPersons / 'N';
  define CountyRate / 'Crude^Rate';

  compute ShowGraph / char;
    call define(_col_, 'style', "style=[preimage='||Measure||".gif']");
  endcomp;
run;
```

1 – Add the new variable to the COLUMNS statement. Again, this variable does not exist in the incoming dataset and is included under the spanning column header with the count, rate and rank variables.

2 – In the DEFINE statement a column header is added.

3 – Add a COMPUTE block for the new variable that contains a PREIMAGE style attribute to add the graphic.

With the added graphics, the table is a lot more interesting to look at and contains more information.

	Zip Code 98101 (Population: 10,448)				King County (Population: 1,835,525)	
	N	Crude Rate ¹	0 -----> max	Rank ²	N	Crude Rate
Primary Measures ³	100	957		6 of 54	4,918	268
Primary A						
Sub-Measure A1	49	489		3 of 43	2,709	148
Sub-Measure A2	32	308		3 of 14	977	53
Sub-Measure A3	19	n/c		n/a	1,232	67
Primary B						
Sub-Measure B1	83	1,430		6 of 53	4,377	479
Sub-Measure B2	48	1,033		4 of 44	2,903	315
Primary C						
Sub-Measure C1	29	278		4 of 26	1,653	90
Sub-Measure C2	73	699		6 of 57	4,841	264
Sub-Measure C3	29	278		3 of 12	786	43
Secondary Measure A	80	766		2 of 26	1,600	87
...partial results						

THOSE GRAPHICS REALLY HELP!

If we look down a few rows in the table we'll see how adding these graphics tell us a lot more about the ranks. Notice that this zip code was ranked 2nd in each of the measures shown below. In the original table, on the left, that's all you know. When the graphics are added, on the right, you can see that 2nd

Secondary C/E3	22	208	2 of 23		22	208		2 of 23
Secondary C/E4	13	n/c	n/a		13	n/c		n/a
Secondary Measure D	59	565	2 of 55	➔	59	565		2 of 55
Secondary B/D1	43	412	2 of 41		43	412		2 of 41
Secondary B/D2	16	n/c	n/a		16	n/c		n/a
Secondary Measure E	536	5,129	2 of 74		536	5,129		2 of 74
...partial results								

doesn't always mean the same thing. On the "Secondary C/E3" measure this zip code was 2nd, but had a rate almost as high as the maximum rate. On the other measures it was a distant second, with rates just about half that of the highest ranked zip code.

HANDLING NO GRAPHICS – PART II

Notice that in the first look at this table above there are a number of rows with no data (i.e., "Primary A" and "Primary B"). These are header rows for the detail that follows, but they do have an observation in the incoming dataset that contains information on how to format the text in the row. The macro that creates the graphics does not get called for these observations, so we don't create an empty graphic as

we did in the previous example. The PROC REPORT code shown above, which creates the table, would not quite work as is. We'd actually get the following error in the log and a table that's not nearly as visually appealing as we'd like.

```
ERROR: Unable to load image
C:\Temp\Primary A.gif;
default image will be used
instead.
```

	N	Crude Rate ¹	0 -----> max	Rank ²	N	Crude Rate
Primary Measures ³	100	957		6 of 54	4,918	268
Primary A						
Sub-Measure A1	49	469		3 of 43	2,709	148
Sub-Measure A2	32	306		3 of 14	977	63
Sub-Measure A3	19	n/c		n/a	1,232	67
Primary B						

We can use a different technique to get a blank cell where the graphics would otherwise be.

```
compute ShowGraph / char;
  if GeogPersons gt 0 then call define(_col_, 'style', "style=[preimage=' ' || Measure || ".gif' ]");
endcomp; 1
```

1 – Only if there is data in the row (GeogPersons > 0) is the CALL DEFINE executed. If there is no data we just end up displaying the contents of the ShowGraph variable, which is a blank space.

Notice in the example above that right above the second big question mark there is a blank space where the bar should be. This row actually has data, but the rate and rank are suppressed. This observation would have gone through the graphic-making macro and a blank image file would have been created.

CONCLUSION

The SAS/Graph Annotate facility is a powerful tool for creating graphic images. The Annotate macros make building data-based graphics a simple task. The advent of ODS allows us to take those graphics and place them directly in a data table. The embedded graphics really draw the reader's attention to the areas of the report that are interesting.

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>)

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

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

Other brand and product names are trademarks of their respective companies.