# Using ANNOTATE MACROS as Shortcuts

Arthur L. Carpenter
California Occidental Consultants

## Abstract

ANNOTATE macros can provide a shortcut when creating an ANNOTATE data set using assignment statements. To be used properly you need to understand how they work and what they will do for you. They will not abrogate your need to understand how the process of creating the data set works. Indeed you need to have a good understanding of how the ANNOTATE data set is constructed before you should attempt to use these macros.

## Keywords

ANNOTATE, macro, annomac, SAS/GRAPH

## Introduction

The process of creating observations in an ANNOTATE data set can be simplified through the use of ANNOTATE macros. Since macros are executed before the data step is compiled and executed, these macros can be used to generate the assignment statements that you would otherwise create yourself.

These macros are predefined to give the user the ability to control all of the basic variables associated with given ANNOTATE function. When controlling a function the macro name usually takes on the name of the function that is to be defined. Using these macros can eliminate some of the tedium associated with the use of assignment statements.

When using ANNOTATE macros it is important to remember that the macro call will be resolved into a series of DATA step assignment statements. These are the assignment statements that you could have written if you had chosen not to use the macro.

There are two types of ANNOTATE macros those that prepare or manage the environment and those that define functions. Primarily you will want to use macros to replace the series of assignment statements associated with a particular function.

Macros that prepare the environment include:

| | |
|---|---|
| %ANNOMAC | Always required - compiles all other ANNOTATE macros and makes them available for use. |
| %DCLANNO | Specifies the correct length for all ANNOTATE variables. |
| %SYSTEM | Used to define the type of coordinate reference system by assigning values to the variables XSYS, YSYS, and HSYS. |

Macros used to replace assignment statements associated with functions include:

| | |
|---|---|
| %BAR | Creates a fillable rectangle. |
| %CIRCLE | Draws an empty circle. |
| %DRAW | Draws a line to a specific point. |
| %LABEL | Write text at the specified location. |
| %MOVE | Moves to a specific point without drawing. |
| %POLY | Begins drawing a polygon. |
| %POLYCONT | Continues drawing a polygon. |

ANNOTATE macros cannot be used unless the %ANNOMAC macro has been called somewhere in the job (prior to the calls of any other ANNOTATE macro). This macro compiles all of the remaining ANNOTATE macros which are then added to WORK.SASMACR.

Unlike the calls to many functions all arguments to the macros must be specified. Even when you want to use the default value for the argument, the default value must be included (missing values can be used in most cases to achieve the default value). The macro uses the argument to build one or more assignment statements. Since very little checking is done by the macro, blank arguments are more likely to cause errors than to result in the default value for a particular option.

The arguments for the macros may be constants (numbers or character strings), variable names, or literal strings. You will need to consult the documentation in the SAS/GRAPH Reference Manual (Vol. 1 pp.570-587) to determine which is expected for a particular argument.

| | |
|---|---|
| constant | use a number or a quoted string. |
| variable name | the variable needs to be on the PDV and is not quoted. |
| literal strings | these strings will be placed inside of quotes by the macro and so are not quoted in the macro call. The reference manual indicates which arguments are to be literals. |

Two typical ANNOTATE macros are highlighted below to give you a general feel for the syntax and usage.

**%SYSTEM**

The syntax for the %SYSTEM macro is:

```
%system(xsys, ysys, hsys)
```

where each argument is a literal and can be 1 through 9 and A, B, or C. These values correspond to the coordinate reference systems *e.g.* XSYS='3' is the absolute percentage of the Graphics Output Area.

The following portion of a SASLOG was generated using the MPRINT system option. It shows the statements generated by the %SYSTEM macro. The macro requests that the 'Graphics Output Area - percentage' be used as the basis for the coordinates. Notice that the third argument is not specified. This results in HSYS being missing.

```
182  %system(3,3);
MPRINT(SYSTEM):   XSYS = "3";
MPRINT(SYSTEM):   YSYS = "3";
MPRINT(SYSTEM):   HSYS = "";
```

HSYS is used by several ANNOTATE functions when establishing the coordinate system or units to use when requesting such things as a height for a character or a length of a line. The documentation should be consulted as to which functions use HSYS. The %LABEL and %SLICE macros (shown below) both use HSYS.

**%LABEL**

The syntax for the %LABEL macro is:

```
%label(x,y,text,color,angle,
rotate, size, style, position)
```

where

| | |
|---|---|
| x & y | specify coordinates for the text string |
| text | text string or character variable containing string to be placed |
| color | literal (quotes are not used) - color of the text |
| angle | number or numeric variable - writes text at an angle |
| rotate | number or numeric variable - rotates individual characters of the text |
| size | number or numeric variable - specifies the text size |
| style | literal - font to be used for the text |
| position | literal - position of text relative to the X,Y coordinate |

The following portion of the SASLOG shows the statements generated by the %LABEL macro.

```
106       %label(50,75,
      'Home Wanted',blue,.,.,4,script);
MPRINT(LABEL):   X = 50;
MPRINT(LABEL):   Y = 75;
MPRINT(LABEL):   ANGLE = .;
MPRINT(LABEL):   ROTATE = .;
MPRINT(LABEL):   SIZE = 4;
MPRINT(LABEL):   STYLE = "script";
MPRINT(LABEL):   TEXT = 'Home Wanted';
MPRINT(LABEL):   IF "" =: '*' THEN ;
MPRINT(LABEL):   ELSE POSITION = "" ;
MPRINT(LABEL):   IF "blue" =:'*' THEN ;
MPRINT(LABEL):   ELSE COLOR = "blue";
MPRINT(LABEL):   FUNCTION = "LABEL   ";
MPRINT(LABEL):   OUTPUT;
```

In the above example the last argument of the %LABEL macro call was left blank. An examination of the code (bolded above) shows that this did not result in an error for this argument. As a general rule it is not wise to leave arguments blank.

## Building a GSLIDE

The example below creates three labels using ANNOTATE macros. The equivalent assignment statements have also been included (but commented out) to show what code the ANNOTATE macros are producing. Of course if the commented code is removed the program becomes much shorter.

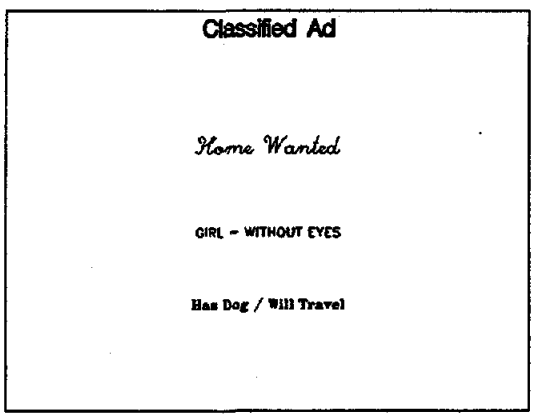
```
%annomac

* USE PROC GSLIDE AND ANNOTATE TO
* CREATE A CLASSIFIED AD FOR ANNIE.;
DATA ANNIE;
LENGTH FUNCTION COLOR STYLE $8;

*RETAIN XSYS YSYS '5';
%system(5,5)

*COLOR='BLUE';
*STYLE='SCRIPT';
*SIZE=4;
*TEXT='Home Wanted              ';
*Y=75;
*OUTPUT;
%label(50,75,'Home Wanted              ',
       blue,0,0,4,script);

*SIZE=2;
*Y=50;
*STYLE='DUPLEX';
*TEXT='GIRL - WITHOUT EYES';
*OUTPUT;
%label(50,50,'GIRL - WITHOUT EYES',
       *,0,0,2,duplex);

*Y=30;
*STYLE='TRIPLEX';
*COLOR='GREEN';
*TEXT='Has Dog / Will Travel';
*OUTPUT;
%label(50,30,'Has Dog / Will Travel',
       green,0,0,2,triplex);
run;

PROC GSLIDE ANNO=ANNIE;
TITLE1 F=SWISS H=3 'Classified Ad';
run;
quit;
```

After removing the commented code, the DATA step that creates the ANNOTATE data set ANNIE becomes:

```
* USE PROC GSLIDE AND ANNOTATE TO
CREATE A CLASSIFIED AD FOR ANNIE.;
DATA ANNIE;
LENGTH FUNCTION COLOR STYLE $8;
%system(5,5)
%label(50,75,'Home Wanted              '
       ,blue,0,0,4,script);
%label(50,50,'GIRL - WITHOUT EYES',
       *,0,0,2,duplex);
%label(50,30,'Has Dog / Will Travel',
       green,0,0,2,triplex);
run;
```

The length of the TEXT variable in the previous DATA step is set in the first %LABEL by padding the string with blanks. It is generally smarter to use a LENGTH statement.

Although the new program is much shorter (fewer statements) than one that uses assignment statements, it is not more efficient from the computer's point of view. We are now using the %SYSTEM macro to write assignment statements to assign the values to XSYS and YSYS when the RETAIN would be quicker. Also the macros create and assign values to a number of variables that are not needed and really should be dropped. These include HSYS, POSITION, ANGLE, and ROTATE.

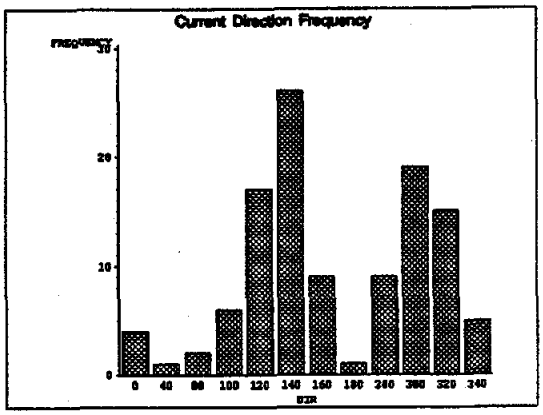



## Creating a windrose plot using %SLICE and %DRAW

The windrose plot takes its name from plots of wind speed and direction. Windrose plots are a type of histogram and are useful when the extreme values of the histogram's midpoint variable are related. Typical applications include any histograms involving direction, clock time, or other cyclical values.

In the example below the frequency of ocean current direction and current speed information was collected over a four month period in 1986 near a power station on the Pacific coast of California. A frequency histogram of the compass bearings fails to highlight the relationship between between the extreme

directions (0-20 degrees and 340-360 degrees). This relationship can be highlighted by the use of the windrose plot (or in this case a current rose).




Each observation in the data set CURRENT represents the average current direction (DIR) and speed (RES) for that day. The statistics, frequency (NOBS) and mean speed (SPEED), are then calculated with a PROC MEANS.

```
proc sort data=current;
by dir;
run;

proc means data=current noprint;
by dir;
var res;
output out=stats n=nobs mean=speed;
run;
```

The current diagram shown below was drawn entirely using ANNOTATE. The %MOVE and %DRAW macros were used to draw the coastline for a visual reference. In this case the coordinates are provided in the code, but more typically they will be provided in a separate data set.

```
%annomac

data anno;
set stats;
retain xsys ysys hsys '5';

* draw the coast line once;
if _n_=1 then do;
   %move(47,80)
   %draw(51,67,black,1,1)
   %draw(65,52,black,1,1)
   %draw(69,43,black,1,1)
   %draw(77,38,black,1,1)
   %draw(78,30,black,1,1)
   %draw(86,18,black,1,1)
   %draw(88,15,black,1,1)
   %label(47,80,'Pacific Coast',
          black,.,.,5,duplex,3)
end;
```

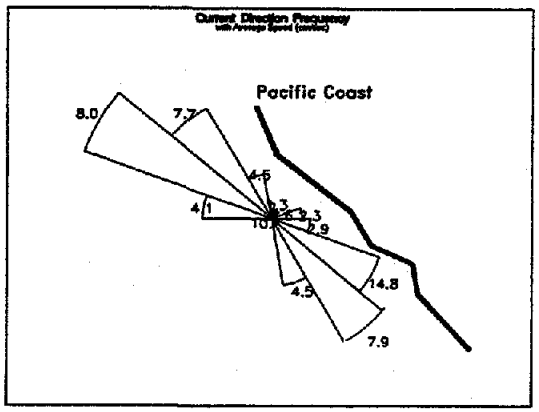
The %SLICE macro is used to create a pie slice for each direction. The direction (DIR) variable is used to determine the orientation of the pie slice and its length is a function of the frequency (NOBS).

The length of the plot lobes (RADIUS) indicates the frequency of that direction (each lobe has an angle of 20 degrees). The lobe is then centered in this case at the point 50,50 percent, and is oriented by using the direction of the current (DIR) to specify the angle.

```
* draw the pie slice;
* adjust the radius of the slice;
radius = nobs*2.0;
%slice(50,50,dir,20,radius,
       blue,empty,both)
```

The average speed associated with each direction is added to each slice as a label using the %PIEXY and %LABEL macros. %PIEXY is used to find the center of a pie slice and determine a point outside of the end of the arc that can be used to attach a label. The second argument of %PIEXY is a multiplier of the radius. In the code below a multiplier of 1.1 is used to move the label just outside of the arc. A multiplier of less than one can be used to put labels inside of the slice. ANNOTATE 'remembers' locations of points from observation to observation in the ANNOTATE data set with the internal coordinate variables XLAST, YLAST, XLSTT, and YLSTT. Functions that write text tend to use the latter two while functions that draw or move use the former (XLAST, YLAST). Values may need to be exchanged between the two sets of coordinates when placing a label at a point established by a function that updates XLAST and YLAST. The %CNT2TXT macro performs this operation.

```
* prepare to add the slice label;
text = put(speed,5.1);
lblang = dir+10;
%piexy(lblang,1.1)
%cntl2txt
%label(.,.,text,black,.,.,4,simplex)
run;

proc gslide anno=anno;
title1 'Current Direction Frequency';
title2 h=1.5 f=simplex
       'with Average Speed (cm/sec)';
run;
quit;
```




This graph does not yet seem finished. There are several fixes that will improve the appearance of this presentation.

- The average speed labels are a bit cluttered for the shorter lobes. Conditional processing when assigning these values can eliminate these labels.
- The labels for the 'upcoast' currents tend to fall on or too near the end of their sectors. The ANNOTATE variable POSITION can be made conditional on the orientation of the lobe.

Although the plot was generated the code did not run without errors (warnings). Because the ANNOTATE macros define the length of some of the ANNOTATE variables (*e.g.* STYLE and COLOR) by how they are used the first time, the LENGTH statement is often useful.

```
* Finish the figure;
data anno;
set stats;
length style color $8;
retain xsys ysys hsys '5';
```

This figure adds a label for the position of the power plant and a lobe that acts as a legend for frequency.

```
* draw the coast line and a legend
once;
if _n_=1 then do;
   * Add a sample segment;
%slice(20,15,0,20,20,blue,empty,both)
%label(20,12,'10 Current readings',
       black,.,.,4,simplex,6)
   * draw the coast line;
   %move(47,80)
   %draw(51,67,black,1,1)
   %draw(62,52,black,1,1)
   %draw(65,43,black,1,1)
   %draw(67,38,black,1,1)
   %draw(73,30,black,1,1)
   %draw(84,18,black,1,1)
   %draw(85,15,black,1,1)
   %label(47,80,'Pacific Coast',
          black,.,.,5,duplex,3)
   %label(56,58,'* Power Plant',
          black,.,.,5,duplex,3)
end;
```

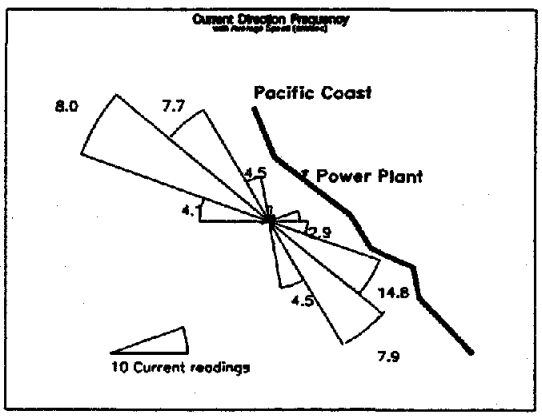
The POSITION of the slice label is changed according to the orientation of the slice and the multiplier has been slightly increased to move the labels further from the end of the slice.

```
* prepare to add the slice label;
text = put(speed,5.1);
* Center the label in the slice (add 10
degrees);
lblang = dir+10;
* Adjust position of the label;
if 0  le dir le 90 then position='3';
else if 90 lt dir le 180 then
position='1';
else if 180 lt dir le 270 then
position='7';
else if 270 lt dir le 360 then
position='9';
%piexy(lblang,1.2)
```

The label for the slice is then conditionally executed for directions with frequencies that are greater than or equal to 5.

```
* Only add a label for slices that are
* more frequent;
if nobs ge 5 then do;
   %cntl2txt
   %label(.,.,text,black,.,.,4,simplex)
end;
run;
```

The resulting diagram contains the same frequency information as does the earlier histogram.



## Summary

ANNOTATE macros are not used to minimize the size of ANNOTATE data sets. They can however be very effective in minimizing the number of statements used to build a ANNOTATE data set. The macros are used to replace a series of assignment statements that would otherwise have to be specified individually.



## Trademark Information

SAS and SAS Quality Partner are registered trademarks of SAS Institute, Inc. in the USA and other countries.
® indicates USA registration.


## About the Author

Art Carpenter's publications list includes two chapters in *Reporting from the Field*, the two books *Quick Results with SAS/GRAPH® Software*, and *Carpenter's Complete Guide to the SAS® Macro Language* and over two dozen papers and posters presented at SUGI, WUSS, and PharmaSUG. Art has been using SAS since 1976 and has served as a steering committee chairperson of both the Southern California SAS User's Group, SoCalSUG, and the San Diego SAS Users Group, SANDS; a conference cochair of the Western Users of SAS Software regional conference, WUSS; and Section Chair at the SAS User's Group International conference, SUGI.

Art is a SAS Quality Partner™ and through California Occidental Consultants he teaches SAS courses and provides contract SAS programming support nationwide.

## Author Contact


Art Carpenter
California Occidental Consultants
PO Box 6199
Oceanside, CA 92058-6199

(760) 945-0613

art@caloxy.com
www.caloxy.com