44 Tricks with the 4mat Procedure

Ben Cochran, The Bedford Group, Raleigh, NC

Abstract:
Actually, there probably are not a total of 44 tricks that one can do with the FORMAT procedure. The number was chosen purely for the purposes of alliteration. However, there are some fairly powerful things one can do with this procedure, and this paper serves as a gentle introduction to the FORMAT procedure. In SAS, a format is simply instructions on how to write or display a value. There are dozens, if not hundreds, of formats that ship with the SAS system. While they cover many of the user's needs, they cannot possibly cover all of them. So, the SAS System includes a tool that allows users to create their own formats. That tool is the FORMAT procedure.

Introduction:
PROC FORMAT is a tool that can be used to create more than user defined formats. Used defined INFORMATS as well as 'PICTURES' can be created as well. However, this paper focuses on the formats that one can create. The general syntax is seen in figure 1.

```
proc format options ;
     value name options
          range1 = 'label1'
          range2 = 'label2' ;
run ;
```

Figure 1.

The VALUE statement is the real workhorse of the FORMAT procedure. First, it names the format that is being created. Next, a 'range' is supplied that represents a value and the 'label' represents how the value is to be displayed. Figure 2 shows some examples.

```
value answer 1 = ' Yes '
             2 = ' No ' ;

value gfmt  low - 10 = 'Group 1'
          11 - 20    = 'Group 2'
          21 - high  = 'Group 3' ;
```

Figure 2.

The first example creates a format named 'answer'. If this format is associated with a variable whose value is 1, then the word 'Yes' would be displayed instead of the 1. If the variable has a value of 2, then the word 'No' would be displayed.

The second example creates a format named 'gfmt'. If this format is associated with a variable whose value is less than 10, then the word 'Group 1' is displayed. If the variables' value is between 11 and 20, then the word 'Group 2' is displayed. And finally, if the variables' value is greater than or equal to 21, then the word 'Group 3' is displayed.

Formats for Single Numbers:

One of the most common examples of format usage is with date values, specifically months. Examine the program in figure 3.

```
proc  format ;
     value monfmt
               1 = 'January'
               2 = 'February'
               3 = 'March'
               4 = 'April'
               5 = 'May'
               6 = 'June'
               7 = 'July'
               8 = 'August'
               9 = 'September'
              10 = 'October'
              11 = 'November'
              12 = 'December' ;
run ;
```

Figure 3.

This procedure creates a format named MONFMT. The next thing that needs to be done is to associate it with a variable whose values are from 1 through 12 so that the words for each month are displayed instead of the numbers. This can be done through using a FORMAT statement in a procedure that generates a report, on a PUT statement, or with a PUT function.
This paper only deals with the FORMAT statement used in a PROC Step. Figure 4 illustrates the association in the PRINT Procedure.

```
proc print data = sashelp.mdv;
  where city = 'LISBON' and qtr = 1 ;
  var  city month sales95;
  format month monfmt.;
  sum sales95;
run;
```

Figure 4.

Notice the syntax of the FORMAT statement. The format comes after the variable with which it is being associated. Notice that the MONFMT format ends with a dot. Syntactically speaking, all formats have a dot in them when used on a FORMAT statement. The dot distinguishes the format from a possible variable name. In other words, without the dot, the format name would be a valid variable name. The formatted values can be seen in the MONTH column of the output below (figure 5).

CITY	MONTH	SALES95
LISBON	January	4,523.00
LISBON	February	4,123.00
LISBON	March	4,675.00
		========
		13,321.00

Figure 5.

Formats for a Range of Numbers:

Another common use of formats is to 'translate' a range of numbers into one word or phrase. Figure 6 below illustrates this method.

```
proc format ;
    value incfmt
        LOW – 1000  = 'Small'
        1001 -  5000  = 'Medium'
        5001 -  HIGH = 'Large' ;
run;
```

Figure 6.

When the INCFMT format is associated with a variable whose value is below 1000, the word 'Small' will be displayed. When this format is associated with a variable whose value is between 1001 and 5000, the word 'Medium' will be displayed, and so on. The INCFMT is associated with a variable named INCOME in the code below.

```
proc freq data = sashelp.pm;
    tables region * income /
            norow  nocol nopercent ;
    format income incfmt.;
run;
```

Figure 7.

Examine the PROC FREQ output in figure 8.

Table of REGION by INCOME

REGION	INCOME			
Frequency	Small	Medium	Large	Total
NORTH AMERICA	104	114	18	236
OTHER	184	80	12	276
Total	288	194	30	512

Figure 8.

Instead of having a myriad of columns, this PROC FREQ output now only has three because of the results of the INCFMT format.

Other PROC FORMAT topics that will be covered in the presentation are:

- the FUZZ factor
- the PICTURE statement
- storing FORMATS permanently
- creating formats from SAS data sets
- documenting formats
- maintaining/editing fomats
- version 8 and 9 enhancements

The PICTURE Statement:

You can also create formats with the PICTURE statement. Pictures are valid for numeric variables only. They create a 'template' into which the numbers fit.. The format (picture) names are:

- valid SAS names
- unique – they cannot be the same as other SAS formats or pictures.

A demonstration of the PICTURE statement can be seen below in figure 9. The first step in figure 9 is a DATA Step with a CARDS statement which reads in-stream data. Next, a PROC FORMAT step creates two formats using a PICTURE statement. And finally, the formats
are associated with the proper variables in the PROC PRINT step using a FORMAT statement.

```
data  numbers ;
  input phone_1 money;
        cards;
9198184004   +52513653
8888311191    -123.45
;

proc format ;
  picture  pict_a
      low – high = ' 000)000-0000'
      ( prefix = ' (' ) ;
  picture  pict_b
      low - - 0  =  ' 0,000,009.99 – '
      0 -  high  =  ' 0,000,009.99 + '
      prefix = '$'  fill = '*' ;
 run;

proc print data = numbers;
   format phone_1  pict_a.
          money    pict_b. ;
run;
```

Figure 9.

In the PICT_A format, the LOW – HIGH syntax indicates that any number is to be used in this format. Once the numbers are placed in this 'template', the prefix is applied.

In the PICT_B format, the 'low – 0' represents any negative number up to but not including 0. This is translated to the number followed by a negative sign. Then, any positive number is displayed with a following positive sign. Notice the PREFIX and FILL options. The PREFIX option places one dollar sign in front of the value, while the FILL option 'fills' the available space in front of the value with dashes.

In the PROC PRINT step , examine the syntax of the FORMAT statement. Notice that PICTURES are associated with variables in the FORMAT statement. The output from PROC PRINT can be seen in figure 10.

Obs	phone_1	money
1	(919)818-4004	**$2,513,653.00+
2	(888)831-1191	--------$123.45-
3	(803)319-1055	----$543,755.93-

Figure 10.

Permanent Formats

You can store user defined formats permanently by using the LIBRARY option on the PROC step. The example in figure 11 illustrates creating two different formats that are stored permanently in the SAS_3 library. The name of the catalog defaults to FORMATS.

```
proc format library=sas_3;
   value $reg_fmt '  50' - ' 200' = 'Southeast'
                  ' 250' - ' 500' = 'Northeast'
                  ' 550' - ' 700' = 'Midwest'
                  ' 750' - ' 850' = 'Western'
                  other           = 'International';
   value amt_fmt  low - 75     = 'Small'
                  76  - 150    = 'Medium'
                  151 - high   = 'Large';
run;
```

Figure 11.

Suppose you want to create a format, but at the same time, you want to 'borrow' values from an existing format (SAS supplied or user defined). The next example shows you how to accomplish this task by 'nesting' formats.

In this case, we want to 'translate' all the AGE values up to 12 to be 'Real Young'. Next, we want to 'translate' AGE values of 13 and 14 to 'Small'. Then, all other AGE values we want to display as ROMAN numerals.

```
proc format;
     value demofmt
     low - 12 = 'Real Young'
     13  - 14 = 'Small'
     other = [roman12.];
run;
proc print data=sashelp.class;
    format age demofmt.;
run;
```

Figure 12.

Notice the values for ‘Other’ in the PROC FORMAT code.

Applying the DEMOFMT format to a familiar data set gives us the following output.

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	Small	69.0	112.5
2	Alice	F	Small	56.5	84.0
3	Barbara	F	Small	65.3	98.0
4	Carol	F	Small	62.8	102.5
5	Henry	M	Small	63.5	102.5
6	James	M	Real Young	57.3	83.0
7	Jane	F	Real Young	59.8	84.5
8	Janet	F	XV	62.5	112.5
9	Jeffrey	M	Small	62.5	84.0
10	John	M	Real Young	59.0	99.5
11	Joyce	F	Real Young	51.3	50.5
12	Judy	F	Small	64.3	90.0
13	Louise	F	Real Young	56.3	77.0
14	Mary	F	XV	66.5	112.0
15	Philip	M	XVI	72.0	150.0
16	Robert	M	Real Young	64.8	128.0
17	Ronald	M	XV	67.0	133.0
18	Thomas	M	Real Young	57.5	85.0
19	William	M	XV	66.5	112.0

Figure 13.

Notice the AGE column.

Other features will be illustrated in the presentation, including:

- creating informats
- creating formats from datasets
- documenting formats
- editing/maintaining formats
- table lookup techniques
- version 8 and 9 enhancements

Conclusion

If the output from your SAS program was a cake, then proc format would be the icing. Not only are there a number of ways to use the FORMAT procedure to effect the appearance of the output from your SAS programs, you can also do much,
much more.

The author can be reached at:
Ben Cochran
The Bedford Group
3216 Bedford Avenue
Raleigh, NC 27607
(919) 831-1191

bedfordgroup@nc.rr.com
www.bedford-group.com