Use FORMATS To Make Your Reports More Informative

David Fickbohm, Fireman's Fund Insurance Company, Novato, CA

ABSTRACT

This talk will introduce beginning SAS users to the idea of using formats. Users will be introduced to several formats that come with BASE SAS. Attendees will learn that formats provide the necessary building blocks for effective, more flexible SAS usage. Users will also be introduced to PROC FORMAT as the key to creating refined SAS reports. Several features of PROC FORMAT will be discussed, including ranges, exclusion ranges, LOW, HIGH, OTHER, and PICTURE.

INTRODUCTION

Any value you wish to print using the SAS system will use a format. A format simply converts data from one representation to another. Many formats come with the SAS System. You can make your own custom formats, if need be. This paper will explain the basics of how the SAS system uses formats to produce useful, readable output. Several standard formats are used as examples. You will see how to create your own specific formats for your own needs.

The SAS systems deals with numeric and character data within its SAS data sets. Numeric data is stored in floating point, a binary form, also known as scientific notation. This method lets the computer operating system do math. Unfortunately, it is not very easy for humans to read.

The number 1 is stored in an IBM mainframe as 4110000000000000. In a PC the number 1 is stored as 3FF0000000000000. These representations are not very user friendly. In fact, they are user hostile.

To make these numbers more user friendly, SAS provides us with FORMAT. The FORMAT converts the floating point data into something recognizable by a human being.

CAPABILITIES AND SYNTAX

The main thing the FORMAT statement does is instruct SAS how to display a variable or a list of variables; the data itself is NOT affected.

The syntax of an output format follows:

```
PROC FORMAT [LIBRARY = ref] [FMTLIB]
        Value fmtname [ ( [DEFAULT=dfltx]
[MAX=maxx] [MIN=minx] ) ]
            Range1 = 'value1'
          [ Range2 = 'value2']
    ;
```

```
PICTURE fmtname [ ( [DEFAULT=dfltx]
[MAX=maxx] [MIN=minx] ) ]
    Range1 = 'picture1' [  ( [FILL = 'fillchar']
              [PREFIX ='prechar'}
              [MULT = scale]
               [NOEDIT] ) ]
          [ Range2 = 'picture2']  [  (rangeoptions) ]
...]
    ;
```

Explanation of Syntax and Comments

- FMTLIB asks for a listing of the contents of the format library, and LIBRARY identifies a LIBNAME statement pointing to a permanent format library.
- The FMTNAME assigns a name to the format. Formats assigned to character variable must begin with a dollar sign ($). You can use a maximum of eight characters. It can consist of numbers and letters. It can not end with a number. You can not use SAS system names, procedure names, or function names.
- DEFAULT, MAX and MIN control the number of columns used when data is displayed. All three are optional.
- DEFAULT tells SAS to use DFLTX columns if no width is specified when a format is used. The range must be from 1 to 40 characters. SAS uses the maximum width of valuex or picturex if DEFAULT is not entered.
- MAX sets the format to a maximum of maxx characters.
- MIN sets the format to a minimum of minx characters.
- The RANGEX sets one or more values that will be displayed as either valuex or picturex. The values can be implied or explicit. Explicit and implied ranges can be combined in a single range specification.
- Rangex must be enclosed in a pair of quotes ('). Values in different ranges can not overlap. A value can only appear once in all the ranges. You can indicate the lowest nonmissing value by using LOW, the highest nonmissing value by using HIGH, and non specified values by using OTHER.

- The PICTUREX sets the template to be laid over the numbers defined in RANGEX. The picture can not be more than 24 characters and must begin with numbers showing the placement of the formatted numbers.

- The PREFIX, FILL, NOEDIT, and MULT options may change from range to range. They provide you with further control to the shape and form of the formatted values.

- PREFIX tells SAS to show PRECHAR to the left of the formatted value, if space permits

- FILL tells SAS to insert FILLCHAR to the left of the formatted value if it is less than the default width. The default fillchar is a blank.

- NOEDIT tells SAS to print PICTUREX without applying the formatting.

- MULT multiplies the value in RANGE by SCALE. The result is formatted. The default is no multiplying. Use this if you do not want to change large numbers in a dataset but do want to print them as, maybe, thousands of units.

THOUGHTS ON USING FORMATS

- Other than system space limitations, there are no limitations to the number of formats you can use in a SAS program.

- The OTHER range should be placed at the end of a series of ranges. It can be placed anywhere in the series, but putting it as the last range is regarded as good programming style.

- User-created formats, once defined, are used just like system formats.

- User-created formats, may be used in DATA step FORMAT statements. The format must be available when the data step is executed, since the format name is stored in the permanent SAS dataset.

- More than one format can be assigned to a PROC. If more than one FORMAT is assigned to a variable, the one closest to the end of the PROC's statements is used.

THE W.D. AND BEST FORMATS

The simplest format, in my opinion, is the W.D format. Where W is the number of columns and D is the number of columns to the right of the decimal point. The following shows how various values would look using some w.d. formats:

	formats				
VALUE	5.	3.	1	10.1	10.2
-100.00	-100	***	*	-100.0	-100.00
-6.00	-6	-6	*	-6.0	-6.00
0.00	0	0	0	0.0	0.00
1.40	1	1	1	1.4	1.40
1.50	2	2	2	1.5	1 50
1.99	2	2	2	1.9	1.99
100.00	100	100	*	100.0	100.00
1000.00	1000	1E3	*	1000.0	1000.00

Please note several points about the above examples. Note that all values in the various columns line up on the decimal points. (The 1000 is an exception, and is explained below). This is done since most reports want decimals lined up, and it is easier to read and natural to the eye. You will see that rounding occurs when the value cannot be fully shown in the space given by the format. Since format 1.0 does not allow for any decimal places it rounds 1.99 up to 2. In the case of 1000, there are not enough spaces for the number 1000; it is six positions long and only five positions are allocated. SAS deals with this by using a format called BEST. BEST determines the best way to represent the number using the number of positions allocated. In this example BEST chose exponential notation to format 1000.

REPORT-ORIENTED FORMATS

Let us now look at a more specialized set of formats. These are numeric formats used in reporting. Specifically we will explore the use of COMMA, DOLLAR, and NEGPAREN. The small following datastep demonstrates these formats:

```
Data null; file print header = HDR;
  Input  Y @@;
   Put  Y 7.2  + 1  Y comma8.2  +1  Y dollar9.2
              + 1  Y negparen9.2;
      return;
HDR:  put +7  +1 '   comma' +1 '  dollar'
              +1 ' negparen';
cards;
1   10  100.45  4567.89  .25 -.25 –25
;
```

Value	comma	dollar	negparen
1.00	1.00	$1.00	1.00
10.00	10.00	$10.00	10.00
4567.89	4,567.89	$4,567.89	4,567.89
0.25	0.25	$0.25	0.25
-0.25	-0.25	$-0.25	(0.25)

-25.00 -25.00 $-25.00 (25.00)

The formats above, like the w.d are most appropriate for columnar reports. The COMMA and DOLLAR put commas between each three digits. The DOLLAR also includes a leading dollar sign. The NEGPAREN uses parentheses to indicate negative values, a standard method in accounting.

DATE FORMATS

Another group of formats supplied with Base SAS are for date processing.

A SAS date value is actually a day offset from January 1, 1960. The offset can be negative or positive depending on whether the date is prior to or after January 1, 1960. As an example my birthday of April 16, 1946 converts to a SAS date value of –5008. The opening day of WUSS99, October 20, 1999 converts to a SAS date value of 14537. Just as the floating point number SAS uses needs to be converted to something we humans can read, so do SAS date values. The data step below shows several different date formats.

```
data _null_;
   input x @@;
     put 'date:' x date7. +1 x date9.;
     put 'weekdate:' x weekdate.;
     put 'mmddyy:' x mmddyy6. + 1 x mmddyy8.;
     put 'julian:' x julian5. + 1 x julian7.;
  cards;

date:07JAN60 07JAN1960
weekdate: Thursday, January 7, 1960
mmddyy:010760 01/07/60
julian:60007 1960007
date:10DEC59 10DEC1959
weekdate: Thursday, December 10, 1959
mmddyy:121059 12/10/59
julian:59344 1959344
date:19JUL60 19JUL1960
weekdate: Tuesday, July 19, 1960
mmddyy:071960 07/19/60
julian:60201 1960201
date:16APR46 16APR1946
weekdate: Tuesday, April 16, 1946
mmddyy:041646 04/16/46
julian:46106 1946106
date:22OCT99 20OCT1999
weekdate: Friday, October 20, 1999
mmddyy:102299 10/20/99
julian:99293 1999293
```

Date values can be shown many different ways depending on the format. The width of a format also changes the look of the formatted value. Just as when w.d performed rounding, the various date formats produce different results when different widths are used. As an example, DATE7 formats 6 as 06JAN90 while DATE9 shows it as 06JAN1990, the two extra characters allowing for the century.

Using weekdate without a width shows the default width concept. The default width is 29, which is made up of 9 positions for the month, 9 positions for the day, leaving 11 positions for the day and year. If you do not specify a width, the default is used. It is always a good idea to supply a width to control the presentation of your output. Supplying a width lets you set the width of each column in your report.

INTRODUCTION TO PROC FORMAT

We have seen how the formats supplied by SAS work. Now let us consider formats that we can create ourselves.

Consider producing a report of demographic information by demographic group, when the data shows a specific age. The data step below is an example:

```
DATA DEMOGRAF;
  Input Name $ Age Sex $ Income ;
Cards:
        Bill    40      M       2
        Jill    38      F       3
        Larry   26      M       1
        Joe     24      F       5
Run;
```

The data step above puts each person's data specific age into a variable called age, and their income category into salary.

The user of the report wants the age information by age group and the salary information presented in a more user friendly fashion. This can be handled by the following.

```
PROC FORMAT;
  Value AGE
     Low – 17  =  'child       '
     18 – 34   =  'young adult'
     35 - 49   =  'Adult       '
     50 - 64   =  'Older Adult'
     65 – high =  'Senior      ';


  Value SALARY
     1       = '<20K'
     2       = '20K – 30K'
     3       = '31K - 40K'
     4       = '41K - 50K'
```

```
5       = '50K +     ';
Run;
```

The PROC FORMAT above divides age into five (5) categories, and gives a useful label to each category. The keyword LOW refers to all values, negative or positive, below the range end of seventeen (17). The keyword HIGH refers to all values greater than the range start of sixty-five (65). Useful labels are also assigned to each salary category.

Two new formats have been produced in the above example, AGE and SALARY These formats work just as the formats provided by SAS institute. They translate SAS data into something meaningful to a person. While my example associates each new format with one variable, a format can be assigned to multiple variables. Consider the demographic data above with the formats just created:

```
PROC PRINT DATA = demograf;
  Format age AGE. Income SALARY.; Run;
```

The above will associate the variables with the formats just created. The age format has the same name as the variable but this is not required. It is a good idea to name a format in a manner that lets you quickly and easily identify it and its use. The format for income does not have the same name. The output produced by the PROC PRINT above follows:

```
OBS  NAME  AGE          SEX  SALARY

1     Bill   Adult          M    20K – 30K
2     Jill   Adult          F     31K - 40K
3    Larry  Young Adult  M   <20K
4    Mary  Young Adult  F    50K+
```

CHARACTER FORMATS

We can also use formats to make character data more useful and informative. Hopefully, character data will already be easy to read, but this is not always the case. An example follows:

```
PROC FORMAT;
  Value $County
     'MR' = 'Marin          '
     'CC' = 'Contra Costa  '
     'SO' = 'Sonoma        '
     'AL' = 'Alameda       '
     'SF' = 'San Francisco '
     'NP' = 'Napa          '
     'LA','ME' = 'Northern       '
     OTHER = "Unknown      ';
```

Character formats are designated by a $ before the remainder of the format name. In the sample above we have saved key strokes by labeling counties. Note also that more than one value can be assigned to a label. Lake and Mendocino counties have both been assigned to Northern. The use of the OTHER keyword is also shown. If a value does not match any of the specified ranges, the label associated with the OTHER clause will be assigned.

EXPLICIT, IMPLIED, AND EXCLUSION RANGES

We have seen both explicit values and implied ranges matching to labels. A explicit value example would be 'MR' = "Marin' and a implied range example would be 18-34 = "Young Adult'.

But what if we have mathematically derived continuos data instead of ordinal data like age ? The user still wants specific cutoffs, but values can only be assigned to one category, since ranges can not overlap. An example follows:

```
 proc format;
 value Deposits
    10-<100 = 'less than 100'
    100-200 = '100 to 200'
   200<-300 = 'above 200 and up to 300';
NOTE: Format DEPOSITS has been output.
 run;

 data _null_;
  input D @@;
   put D = D=Deposits.;
 cards;

D=39 D=less than 100
D=99.99 D=less than 100
D=100 D=100 to 200
D=150 D=100 to 200
D=200 D=100 to 200
D=250 D=above 200 and up to 300
```

The sample above shows that the < symbol indicates that a value is being excluded. The 10 - < 100 says all values between 10 and 100, except precisely 100, will be given a label of less than 100. The < symbol is put after the value starting the range and before the value ending the range. The last two observations also show the exclusion range concept.

Finally you can combine explicit and implied ranges in the same format. The sample below demonstrates this idea:

```
proc format;
  value validgrp 10-15, 20-30, 35, 45 = 'valid'
      other = 'invalid'
    run;
NOTE: Format VALIDGRP has been output.

data _null_;
  input A @@;
    put A = A=validgrp.;
cards;
A=5  A=invalid
A=12 A=valid
A=22 A=valid
A=32 A=invalid
A=45 A=valid
A=60 A=invalid
```

PICTURE FORMATS

The picture format is used to show numeric data in a more user friendly manner. Instead of converting the data, picture formats only reproduce the data with special characters inserted in the data in specific locations, essentially laying a template over the data. Picture formats are not as good at grouping data as value formats. Here is a good example of a picture format: 4158991234 is not really very useable. If I change the data to (415) 899-1234 it is recognizable as a phone number.

Dashes or other characters are often used to separate sections of an account number. An example of each follows:

```
proc format;
   picture phones other = ' 999)999-9999'
(prefix="(");
   picture acct  0-high = '999-99-9999-99';
run;

data _null_;
   input p a @@;
     put p = p = phones. a = a = acct.;
  cards;

P=4158994417 P=(415)899-4417 A=41589912345
A=415-89-9123-45
```

Note the space in the phone picture between the opening quote and the first nine in the picture definition. This space is needed because prefix occupies positions within the picture, It does not add additional positions. If the picture is not wide enough to contain both the value and the prefix, the prefix is truncated or omitted.

CONCLUSION

Formats give you the ability to present your data in a way that makes it informative to the user. This lets you store the data in the most appropriate form, knowing you can present it many different ways with lots of different formats. The formats supplied with Base SAS can deal with most of the common representations, but the FORMAT procedure is available to create formats to meet your needs specifically.

ACKNOWLEDGEMENTS

Thanks to Virginia "Gigi" Gruber for reviewing the topics and content of this paper for clarity, completeness, and usefulness.

SAS is a registered trademark of the SAS Institute Inc., Cary , NC, USA

Any comments, questions, or thoughts on your experiences with FORMATS are most welcome.

David can be contacted at the following address:
Fireman's Fund Insurance Company
777 San Marin Drive
Novato, CA 94998
E-Mail: David_Fickbohm@FFIC.COM