Paper 96

A Fast Format Macro – How to Quickly Create a Format by Specifying the Endpoints

Christine A. Smiley, Kestnbaum, a KnowledgeBase Marketing Company, Chicago, IL

ABSTRACT

Have you ever needed to create a format for a variable that required a large number of ranges? If so, you have probably discovered how tedious the use of PROC FORMAT can become. This paper illustrates a macro that can create a numeric format easily and quickly using SAS® software. The method used by the macro and the logic behind it are explained. The CNTLIN option in PROC FORMAT is described.

INTRODUCTION

In data analysis, there is often a need to break a continuous variable into a small number of distinct ranges or categories. The output may be a frequency table with the variable split into these ranges, or the result may be means of another variable by these ranges. The set of rules describing the values that fall into each range is called a format. Typically, PROC FORMAT is used to create a format. For variables requiring many numeric ranges, tedious coding can be avoided by using a macro instead. This paper presents a macro that creates a numeric format with little coding.

USING PROC FORMAT

A common method for breaking a variable into categories is to create a format as shown below.

```
proc format fmtlib;
     value dollars
  low   - 50   = 'low to 50'
  50  < - 100  = '50< to 100'
  100 < - 200  = '100< to 200'
  200 < - 400  = '200< to 400'
  400 < - high = '400< to high';
```

In this example, the format is called dollars and has five levels. The FMTLIB option produces a format library that details the format created.

```
-----------------------------------------------------
|   FORMAT NAME: DOLLARS  LENGTH:   13              |
|   NUMBER OF VALUES:    5                          |
|---------------------------------------------------|
|START           |END             |LABEL            |
|----------------+----------------+-----------------|
|LOW             |              50|low to 50        |
|              50<             100|50< to 100       |
|             100<             200|100< to 200      |
|             200<             400|200< to 400      |
|             400<            HIGH|400< to high     |
-----------------------------------------------------
```

CALLING THE MAKEFORM MACRO

A macro can be used to simplify this method. The same task can be accomplished by calling the macro as shown below. The **name** parameter in the macro specifies the name of the format to be created. The **endpoint** parameter is a list of the endpoints for each level of the format.

```
%makeform(name=dollars,
   endpoint=50 100 200 400 high);
```

USING A FORMAT

After calling the makeform macro or using PROC FORMAT, the format can be used as needed.

```
proc freq data=money;
     tables price;
     format price dollars.;
```

```
ITEM PRICE
                                Cumul Cumul
PRICE      Freq    Perc  Freq  Percent
---------------------------------------
low to 50   276   55.2   276   55.2
50< to 100  132   26.4   408   81.6
100< to 200  67   13.4   475   95.0
200< to 400  16    3.2   491   98.2
400< to high  9    1.8   500  100.0
```

THE CNTLIN OPTION IN PROC FORMAT

The macro works by using the CNTLIN option in PROC FORMAT. A data set provides the foundation of the format to be created. The input data set for use in the CNTLIN option must contain certain variables. These variables mirror information shown in the format library. The FMTNAME variable gives the name of the format.

```
proc print data=macdata;
```

```
START    END     LABEL          FMTNAME
low      50      low to 50      dollars
50       100     50< to 100     dollars
100      200     100< to 200    dollars
200      400     200< to 400    dollars
400      high    400< to high   dollars
```

With this data set and the CNTLIN option, a format can be created that is equivalent to the format produced using the VALUE statement in PROC FORMAT.

```
proc format cntlin=macdata;
```

THE MAKEFORM MACRO CODE

The macro works by creating the macdata data set based on the name and endpoints listed in the macro call. The full text of the macro is shown below.

```
%macro makeform(name, endpoint);

data macdata;
  length start $ 8 end $ 8 label $ 20 ;
  fmtname = "&name";

retain i;
i = 1;

do while
(scan("&endpoint", i, " ") ne ' ');

      end = scan("&endpoint", i, " ");

      start = lag(end);
      if start = ' ' then start = 'low';

      label = left ( trim(start) ||
              '< to ' || left(end) );
      if start='low' then
      label = left ( trim(start) ||
              ' to ' || left(end) );

      output;
      i = i +1;
end;
drop i;

proc format cntlin=macdata;

%mend;
```

STEP 1 – BEGIN THE MACRO

The %MACRO statement is used to begin the macro definition. The name of the macro will be makeform, and it will have two parameters: **name** and **endpoint**.

```
%macro makeform(name, endpoint);
```

STEP 2 – DEFINE THE VARIABLES

A data set called macdata is created based on the parameters given in the macro call. The LENGTH statement defines the START, END, and LABEL variables as character variables with appropriate lengths. The FMTNAME variable is taken directly from the **name** parameter of the macro. Note that double quotes must be used so that the **name** parameter will resolve to the value specified in the macro call.

```
data macdata;
 length start $8 end $8 label $20;
 fmtname = "&name";
```

STEP 3 – START THE DO WHILE LOOP

The variable `I` is simply a counter. It will be retained and set to 1 at the beginning.

```
retain i;
i = 1;
```

The DO WHILE statement starts a loop that will go through the **endpoint** parameter of the macro one word at a time. The SCAN function is used to separate the list of endpoints into distinct words. In this case, the SCAN function searches through the **endpoint** parameter to find the i*th* word, where words are separated by spaces. The DO WHILE loop will continue as long as the current word is not missing. That is, the loop will continue until there are no more words.

```
 do while
(scan("&endpoint", i, " ") ne ' ');
```

STEP 4 – CREATE THE END VARIABLE

The SCAN function is used as described above to directly pull out each endpoint.

```
end = scan("&endpoint", i, " ");
```

STEP 5 – CREATE THE START VARIABLE

The start value for each level will be the same as the endpoint of the previous level. The LAG function is used to get the endpoint of the previous level. Each time the LAG function is executed, the current value of the variable END is saved to a queue and the previously saved value is returned. On the first execution of the LAG function, a missing value is returned. To fix this, the macro checks for a missing value for the variable START and then sets it to 'low'.

```
start = lag(end);
if start = ' ' then start = 'low';
```

STEP 6 – CREATE THE LABEL VARIABLE

To create the LABEL variable, the macro uses the TRIM function to remove trailing blanks from the character variable START. The character string '< to' will be inserted next. The LEFT function is used to left align the character

variable END. The concatenation operator is used to string together the character expressions. A final LEFT function is used to left align the combined result.

```
label = left ( trim(start) ||
        '< to ' || left(end) );
```

For the first level of the format, the left endpoint will be included in the range, so the label is adjusted slightly.

```
if start='low' then
label = left ( trim(start) ||
        ' to ' || left(end) );
```

STEP 7 – COMPLETE THE DO WHILE LOOP

An OUTPUT statement is used to create an observation for each level of the format. Next the counter variable `I` is incremented by one and the next level is started. Then the END statement finishes the DO WHILE loop. The counter variable will not be needed once the data step completes, so it is dropped.

```
   output;
   i = i +1;
end;
drop i;
```

STEP 8 – CREATE THE FORMAT AND FINISH THE MACRO

PROC FORMAT is used with the CNTLIN option to create the format. The macro is completed with the %MEND statement.

```
proc format cntlin=macdata;

%mend;
```

USING THE MAKEFORM MACRO

Another example of using the makeform macro is shown below. The macro call specifies that the format should be named ages and should have five levels.

```
%makeform (ages, 18 35 45 65 high);

proc freq data=insured;
   tables insage;
   format insage ages.;
```

Note that the data set name is specified in the PROC FREQ following the macro call. Since the macro creates a data set during its execution, the most recently created data set will be the macdata data set. You should be careful to specify the appropriate data set in any procedures following the macro call. Also note that the final endpoint is listed as 'high'. This ensures that all values will fall into a range of the format. If you know the largest value in the data, then you can use that value instead. Example output from the PROC FREQ is shown here.

INSAGE	Freq	Percent	Cumul Freq	Cumul Percent
low to 18	990	1.3	990	1.3
18< to 35	13681	18.6	14671	19.9
35< to 45	19968	27.1	34639	47.0
45< to 65	25507	34.6	60146	81.6
65< to high	13544	18.4	73690	100.0

ADVANCED VERSION OF THE MAKEFORM MACRO

An expanded version of the makeform macro is shown below. This version adds code that prints text to the log file for use in PROC FORMAT. If you need a slightly different or more complex format than the macro can handle, a first run of the macro will produce text that can then be edited easily to run through PROC FORMAT.

```
%macro makeform(name, endpoint);
options nonotes;

data _null_;
      put ' ';
      put 'proc format;';
      put "value &name";

data macdata;
      length start $ 8 end $ 8 label $ 20 ;
      fmtname = "&name";
      retain i;
      i = 1;
do while (scan("&endpoint", i, " ") ne ' ');
       end = scan("&endpoint", i, " ");
       start = lag(end);
       if start = ' ' then start = 'low';
       label = '>'||left(trim(start))||' to '||left(end));
       if start='low' then
                label = left(trim(start))||' to '||left(end));
       output;
       if start='low' then put start '- ' end '= ' "label";
       else put start '< - ' end '= ' "label";
       i = i +1;
end;
drop i;

data _null_;
      put ';';
 put 'Specify data set name in all procedures below.';

proc format cntlin=macdata;
options notes;
 %mend;
```

CONCLUSION

The makeform macro allows you to create a format by simply specifying the format name and the endpoints of the levels. Each endpoint will be included in its level. The next level will start with values greater than the previous endpoint. With the expanded version of the macro, the

statements in the log help you update the format in case you need to go beyond the defaults of the macro. Using this macro is quite easy and avoids many of the syntax errors that arise when coding a new format.

REFERENCES

SAS Institute Inc. (1990), *SAS® Language: Reference, Version 6, First Edition,* Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1997), *SAS® Macro Language: Reference, First Edition,* Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990), *SAS® Procedures Guide, Version 6, Third Edition,* Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Thank you to the following people for their suggestions and contributions to this paper: Heather Cole, Stephanie Papaioannou, Chris Rellinger, and Ken Smiley.

CONTACT INFORMATION

Please contact the author with any questions or comments:

Christine A. Smiley
Kestnbaum, a KnowledgeBase Marketing Company
55 W. Wacker Drive, Suite 1210
Chicago, IL 60601-1612
Phone: 312-782-1360 x232
Fax: 312-782-1362
E-mail: cas@kestnbaum.com