Paper 053-30

Creating a Numeric Format to Suppress Small Numbers

Michael G. Eberhart, Philadelphia Department of Health, Philadelphia, Pennsylvania

ABSTRACT

SAS® provides various numeric formats to present data in reports and tables. SAS does not provide a numeric format that allows for the suppression of small numbers, as is often required when presenting public health data. Creating a new format with PROC FORMAT can be tedious and time consuming when a wide range of numbers is required. This program describes a method to create a new "numeric" format that can be used to replace specific numbers with an asterisk when generating reports.

INTRODUCTION

When presenting public health data to the public, it is often necessary to suppress small numbers to protect patient privacy, especially when presenting data for a small geographic unit such as zip code or census tract. For presentation purposes, it is also desirable to have all other numbers formatted with commas, as in the standard SAS formats such as "comma6.". Typically, small numbers are suppressed with an asterisk in public health reports, with a footnote indicating why the numbers were suppressed. This suppression lets the user know that there was at least one observation, but less than X observations (depending on the desired threshold) without revealing the exact number. However, the asterisk is incompatible with an integer variable such as the number of cases of a specific disease. Simply replacing specific numbers with an asterisk in an IF THEN statement of a data step will cause those numbers to be missing and cause a note to appear in the log (NOTE: Invalid numeric data, '*' , at line X column X.). For this reason, another method must be utilized in order to protect the integrity of the data.

One option is to create a new numeric format that can be applied when generating reports for public consumption, thus preserving the original data. This numeric format will store text values for a range of numbers so that special characters, such as an asterisk, can be used in place of small numbers. In order to create a format that replaces small numbers with an asterisk and presents all other numbers (including zero) in the "comma6." format, the following code was developed.

NUMBERS TO SUPPRESS AND RANGE OF VALUES

For this example, the numbers 1 through 5 will be suppressed to protect patient privacy, and all other numbers will be presented in the "comma6." format. The decision regarding which numbers to suppress is rarely arbitrary, and is usually decided when organizations develop data use policies. When creating a format for this purpose, consideration must be given to the highest number required for the reports so that the format will include all numbers that will be used. If a specific number is not included in the format, it will appear in the table or report as an unformatted numeric integer (e.g. 1000 instead of 1,000). For this example, 99,999 was chosen as the highest number. Since it would be tedious and time consuming to type out a PROC FORMAT statement and include all of the values for numbers between 0 and 99,999, a DO LOOP is used to create a dataset containing all the required values. A separate data step is utilized to create a macro file for execution.

CREATING THE VALUES FOR A RANGE OF NUMBERS

The first step is to create a temporary data set that includes a variable called NUMFMT. Using the COMPRESS and PUT functions, and the concatenation operator, each number from 0 to 99,999 is expressed in the 6. format followed by an equal sign, followed by the same number in the comma6. format. This variable will hold all of the values for each number that would appear in the PROC FORMAT statement (e.g. 0="0" through 99999="99,999").

```
/*Create data set with variable that holds the values for the format*/
data loop;
do i=0 to 99999 by 1;
j=i+1;
Numfmt=COMPRESS(PUT(i,6.)||'='||'"'||PUT(i,comma6.)||'"');
output;
end;
run;
```

The next step is to change the value of NUMFMT only for those numbers that will be suppressed. This is accomplished using IF/THEN statements in a data step. I am sure there's an easier way to do this, but since only five numbers will be suppressed, it takes only 5 lines of code in a data step to make the necessary changes.

```
/*Change the value of numfmt for low numbers*/

data loop; set loop;
if i=1 then Numfmt='1="*"';
if i=2 then Numfmt='2="*"';
if i=3 then Numfmt='3="*"';
if i=4 then Numfmt='4="*"';
if i=5 then Numfmt='5="*"';
run;
```

CREATING FORMAT MACRO

The next data step creates a SAS program file that contains the macro to be executed in order to initialize the format for the session. The output is a file called lownumber.sas that, when executed, creates a format called NUMFMT. At the first observation, the data step inserts text to define (%macro) and name (lownum) the macro and then inserts the value of the variable NUMFMT (0="0"). Subsequent observations of the data step insert only the value of NUMFMT (1="*", 2="*", etc.) into the output file. When the last observation is processed, text is inserted into the output file to run and end (%mend) the macro. Of course, semi-colons are also inserted where required.

```
/*Create macro to produce format*/

Data _null_;
file "c:\programs\lownumber.sas" DROPOVER lrecl=32767;
set loop end=end;
if _n_ = 1 then put @1 '%Macro lownum;'/@1 'Proc Format;' /@2 'Value
NUMFMT';
put @4 NUMFMT;
if end then put @1 'Run;' / @1 '%mend;';
run;
quit;
```

EXECUTING THE FORMAT MACRO

Because the output file from the data step described above produces a macro that is ready to be executed, the macro can then be initiated in any program using the following code:

```
filename programs "c:\programs\lownumber.sas";
      %include programs;

data _null_;
%lownum;
run;
```

Executing the macro in a DATA _NULL_ step allows the format to be initialized without producing 99,999 lines of output in the log file each time the format is utilized.

SUMMARY

Numeric formats that present data as text, allowing for substitution with other text characters, can be used to suppress small numbers to protect patient privacy. Using DO LOOP processing, large numbers of values can be included in a PROC FORMAT statement without requiring the user to type individual values into the procedure.

Using text values in a numeric format allows the user to suppress small numbers while maintaining the integrity of the original data.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Michael Eberhart, MPH
Philadelphia Department of Public Health
500 S. Broad Street
Philadelphia, PA 19146
Work Phone: 215-685-6603
Email: Michael.eberhart@phila.gov