Paper 212-28

The Use of Formats, Concatenate, and Sum for Reporting on "Check All That Apply" Variables

Lara K. Jungvig, RAND, Santa Monica, CA

ABSTRACT

SAS software always offers more than one way to solve a problem. Using similar data collection instruments from several projects, I approached the problem of how to summarize/display "check all that apply" variables from several different directions. A check all that apply variable occurs when a survey question groups together a set of items with check boxes, instead of asking separate yes/no questions for each item. So, I read in the data as separate variables and produced a frequency of each variable. I concatenated the separate variables into one new variable using a formatted put statement, and then I ran one frequency. Finally, I tried creating a new variable by using a patterned numeric string, formatting that pattern and running a frequency on the new variable. This presentation will utilize basic SAS techniques, in a side-by-side visual.

BACKGROUND

The code I used in this presentation is from sample input statements used to read complex ASCII data in to SAS. The analysis is a medical literature review and meta-analysis. The primary data is abstracted from medical articles using a data collection instrument developed in-house. The data is then entered into an ASCII file and read into SAS. The code included in this presentation reads in the topics of interest in a meta-analysis of antioxidants for prevention and treatment of cancer.

INTRODUCTION

This presentation provides a visual illustration that displays various ways to report "check all that apply" variables, meaning a list of variables where one or more elements in a question may occur. By presenting three different approaches to inputting code and formatting variables side by side, the reader can easily see the pluses and pitfalls associated with each technique.

SAS CODE: SIMPLE, CONCATENATE, FORMAT

Simple input statement and frequencies

Simple input statements and quick frequencies are the obvious "go to" techniques. Easy to write, fast to run, what could be more simple? The problem arises when you need to have these data displayed differently. If the N=196 obs, it's not useful to display separate frequencies that, when totaled, do to add up to 196. Examine the following code and making note that the output is in four tables.

```
/**********************************
Using a simple input statement to read in a flat
ASCII file where there is one record per case.
1 indicates checked box, and missing indicates
unchecked box, then running freqs.
 **********************************/
DATA antiox;
INFILE 'C:\myfiles\rawdata\antiox.dat';

INPUT /
       @1      id                    5.
       @6      VitC                  1.
       @7      VitE                  1.
       @8      CoQ10                 1.;
RUN;

PROC FREQ data = antiox;
       Tables vite vitc coq10;
RUN;
```

```
OUTPUT:
The SAS System
The FREQ Procedure

               Topic: Vitamin E

                                Cumulative Cumulative
vite   Frequency  Percent  Frequency     Percent
-----------------------------------------------
   1       124      100.00     124       100.00

Frequency Missing = 72


                Topic: Vitamin C

                               Cumulative  Cumulative
vitc   Frequency  Percent  Frequency    Percent
-----------------------------------------------
   1       60      100.00       60       100.00

Frequency Missing = 136

               Topic: Coenzyme Q-10

                               Cumulative  Cumulative
Coq10  Frequency   Percent  Frequency    Percent
-----------------------------------------------
  0       148     75.51       148        75.51
  1        48     24.49       196       100.00
```

Using concatenate and formatted put statements

It may not be useful for the observations with different characteristics to be counted several times. For instance, if one observation includes both vitamin C and vitamin E, then it is more accurate to display these data together as one value. By concatenating the four separate variables and adding formatted put statements, I created a new variable named "topic" on which one single frequency may be run. This solves the previous problem of multiple freqs that do not truly represent the total N of studies. It also saves time and paper. The use of formats to recode variables is a useful trick that has broad applications. Examine the following code making note that the output is more efficient and useful.

```
/**************************************
Using concatenate and formatted put statements
to read in a flat ASCII file.
**************************************/
PROC FORMAT;
       format  value vite   1 = "Vitamin E ";
               value vitc   1 = "Vitamin C ";
               value coq    1 = "Coenzyme Q10 ";
RUN;

DATA antiox;
INFILE C:\myfiles\rawdata\antiox.dat';
attrib topic length=$30. label = "Topic";
topic="";
topic = put (vite, vite.)|| put (vitc, vitc.) ||
       put(coq10, coq.);
RUN;

PROC FREQ data=antiox;
       tables topic;
RUN;
```

```
OUTPUT:
The SAS System
The FREQ Procedure

                        Topic
                                                Cumulative
topic                    Frequency     Frequency
-----------------------------------------------------
        .          . Co-Q10    52         52
Vitamin E          .       .   82        134
Vitamin E          . Co-Q10     2        136
        . Vitamin C        .    20        156
Vitamin E Vitamin C        .    39        195
Vitamin E Vitamin C Co-Q10      1        196
```

Using Sum to make a pattern variable

The previous example offered a better tool for data presentation, however the output could be easier to read. In the next example, I create a data display that is more useful than the first example and looks better than the second example. I utilized the sum function with max to create a pattern variable that I formatted so that the output is easily read. I added punctuation to the format, too. The sum function adds up the value for each separate variable that I multiplied by 100, 10, or 1 to create a pattern variable based on binary numbers. This ensures distinct values. The max function takes the maximum value from the list of arguments. Examine the following code noting the use of the pattern to format the variable "topic".

```
/*****************************************
Using Sum, Max and Formats to create and display
a pattern variable.
*****************************************/
PROC FORMAT;
        value topic
     /*ceq*/
          1 = 'Co-Q10'
         10 = 'Vitamin E'
         11 = 'Vitamin E and Co-Q10'
        100 = 'Vitamin C'
        110 = 'Vitamin C and Vitamin E'
        111 = 'Vitamin C, Vitamin E and Co-Q10'
         ;
RUN;
DATA antiox;
INFILE 'C:\myfiles\rawdata\antiox.dat';
attrib topic label = "Topic";
topic = 0;
topic = sum (max (0,vitc*100), max(0,vite*10),
        max(0,coq10));
RUN;

PROC FREQ data=antiox;
        tables topic;
        format topic topic.;
RUN;

OUTPUT:
The SAS System
The FREQ Procedure
                        Topic
                                                Cumulative
topic                        Frequency     Frequency
-----------------------------------------------------
Co-Q10                            52         52
Vitamin E                         82        134
Vitamin E and Co-Q10               2        136
Vitamin C                         20        156
Vitamin C and Vitamin E           39        195
Vitamin C, Vitamin E and Co-Q10    1        196
```

CONCLUSION

In summary, there are different ways to handle the same problem in SAS. While trying to make usable data output, I came across several techniques to handle variables that come from a "check all the apply" question on a survey. Simple freqs of each variable were not helpful in understanding the data. Concatenating those variables and using formatted put statements and running a freq on the new variable produced better data for the research team. One step further was needed, though, to produce output that was easier to read and more professional looking. By using the SUM function with MAX and formatting the pattern variable that was created, the output was ready for analysis. Each technique presented here has advantages and disadvantages. I encourage programmers to try different approaches and to incorporate new techniques in their work.

REFERENCES

1. SAS® Institute Inc. (1999). "SAS OnlineDoc® Version 8," SAS® Institute Inc., Cary, NC.
2. Delwiche, Lora D. and Susan J. Slaughter (1996). "The Little SAS® Book," SAS® Institute, Inc., Cary, NC.

ACKNOWLEDGEMENTS

I would like to thank Beth Roth and Mark Totten for their help on this poster.

CONTACT INFORMATION

Lara K. Jungvig
Programmer/Analyst
RAND Corporation
1700 Main Street
Santa Monica, CA 90407
Phone: (310) 393-0411
Fax: (310) 451-7059
E-mail: Jungvig@rand.org