%SHOWCOMB: a macro to produce a data set with frequency of combinations of responses from multiple-response data

Ronald Fehd, Centers for Disease Control, and Prevention, Atlanta GA

ABSTRACT

Multiple-response data from survey questionnaires where questions have the instruction "check all that apply" present a challenge to the SAS® software programmer because the number of possible response combinations is two to the power of the number of responses. This paper examines the SAS proc FREQ output data set from a cross-tabulation and discusses the issues in constructing a similar data set for multiple-response data with one variable containing the combination of responses. Issues related to labeling, storage and type of multiple-response variables are discussed.

The SHOWCOMB macro takes as parameters an output data set name which is the prefix of a series of variables containing the multiple-response data. The second parameter may be a list of the multiple-response variables, or the output data set provided by %CHECKALL. See Fehd (1996), (1997), %CHECKALL and %ARRAY.

INTRODUCTION

Simple questions may have complex answers when the question contains the phrase "Check all that apply". This paper reviews the output data set of a proc FREQ cross-tabulation of a series of variables. This data set is used as a model to construct a macro which produces a standardized data set with the frequencies of the combinations of responses in multiple-response data.

The Answers: (Check all that apply)

A: Apple   
  B: Banana 
  C: Cherry  

Common values used for the meaning of 'checked' include: (Y,N), (T,F), etc. The example data uses numeric (1,0). A proc FREQ cross-tabulation is the easy first step in examining multiple-response data. Our proposed process requires saving the output data set.

Program 1

data QUERIES; label Q02A = 'Apple' Q02B = 'Banana' Q02C = 'Cherry';
input Q02A Q02B Q02C;
run;

/slist noprint out = FREQ;

proc PRINT data = FREQ label;

SAS output

<table>
<thead>
<tr>
<th>OBS</th>
<th>Apple</th>
<th>Banana</th>
<th>Cherry</th>
<th>Count</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>8.3333</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>16.6667</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>25.0000</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>25.0000</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>16.6667</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8.3333</td>
</tr>
</tbody>
</table>

Our output is raw data: the values representing 'checked' and 'not checked' must be mentally translated while reading. Our next task is to replace the value for 'checked' in each variable with the variable label. This requires a new set of variables which are character with length of 40, the allowed length of labels. 'Not checked' is irrelevant and is changed to blank.

example intermediate output

<table>
<thead>
<tr>
<th>Label</th>
<th>Label</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q02A</td>
<td>Q02B</td>
<td>Q02C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>8.3333</td>
</tr>
<tr>
<td></td>
<td>Banana</td>
<td>2</td>
</tr>
<tr>
<td>Banana</td>
<td></td>
<td>16.6667</td>
</tr>
<tr>
<td></td>
<td>Banana</td>
<td></td>
</tr>
<tr>
<td>Banana</td>
<td></td>
<td>25.0000</td>
</tr>
<tr>
<td></td>
<td>Apple</td>
<td>3</td>
</tr>
<tr>
<td>Apple</td>
<td></td>
<td>25.0000</td>
</tr>
<tr>
<td></td>
<td>Apple</td>
<td>2</td>
</tr>
<tr>
<td>Apple</td>
<td></td>
<td>16.6667</td>
</tr>
<tr>
<td></td>
<td>Apple</td>
<td>1</td>
</tr>
<tr>
<td>Apple</td>
<td></td>
<td>8.3333</td>
</tr>
</tbody>
</table>

The last step is to concatenate the series of variables into one variable, compress, and delimit the labels with a comma. The data is sorted by descending Count.

example desired output

<table>
<thead>
<tr>
<th>Combinations of Q02</th>
<th>Count</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>3</td>
<td>25.0000</td>
</tr>
<tr>
<td>Banana, Cherry</td>
<td>3</td>
<td>25.0000</td>
</tr>
<tr>
<td>Apple, Banana</td>
<td>2</td>
<td>16.6667</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>16.6667</td>
</tr>
<tr>
<td>Apple, Banana, Cherry</td>
<td>1</td>
<td>8.3333</td>
</tr>
<tr>
<td>Cherry</td>
<td>1</td>
<td>8.3333</td>
</tr>
</tbody>
</table>

Constraints of using cross-tabulation

When processing a cross-tabulation, SAS must allocate a
matrix of N columns, where N is the number of variables. The number of matrix rows allocated is O(2**N) if data is binary-valued: (0,1) and O(3**N) if data contains missing (0,1,..). An additional consideration is the width of each column. The minimum space available for numeric variables is 2 or 3 bytes, depending on operating system. For multiple-response data with large numbers of variables, some optimization of both data storage and matrix size is necessary.

Numeric binary-valued data where the data is either zero, one or missing may be more effectively stored as character in one byte. The cost of this storage efficiency is that the data must be converted to numeric for usage in many SAS procedures.

Further efficiency may be realized by converting a series of binary-valued variables into an integer where each bit represents one variable. Again, the cost is conversion from compressed data to individual variables for analysis.

A major benefit of using an integer to store multiple-response data is that the width of the cross-tabulation matrix is reduced to one column. This reduces the chance that a production routine would fail.

**Data compression**

Each bit in an integer can be changed from zero to one on the condition that a contributing value is true. This routine has a parameter TRUE which can be changed to accept a value of numeric or character one. Each element in the array is tested for &TRUE and the bit changed accordingly. The exponent of 2 is the array dimension minus the index; this changes bits from left to right, reflecting the left to right pass through the array. This allows the programmer to compare the variable values with the integer produced.

**Macro code excerpt 1**

```plaintext
%LET TRUE = 1;

%3: data: prepare subset of DATA and create Number for FREQ which is the binary-value of all the variables with value = &TRUE;
DATA ZBINNMBR;
  set DATA;
  array CheckAll {* } &VAR_LIST;
  N = 0;
  do I = 1 to dim( CheckAll );
    if CheckAll[ I ] = &TRUE. then
      N = sum( N, 2**( &DIM_VAR. * I ) );
  /*do I */ end;

Saving the labels

In this step the labels of the series are saved to an array of macro variables.
```

**Macro code excerpt 2**

```plaintext
%3.2 create array of labels of series;
length Label $ 40 ; drop Label ;
%DO I = 1 %TO &DIM_VAR ;
  call label(&VARAI . , Label ) ;
  call symput('LBLAI . ,trim(left(Label ))) ;
%END ;

The data is now prepared for proc FREQ of a single variable: the integer containing the data from the series of variables. An output data is saved for the decompression step: ZFRQCOMB.

**Decompression**

Changing the integer to a character variable of combinations of labels is a two-step process. First the integer is changed to a character variable -- BinStrng -- containing zeros and ones. Then this binary string is used to concatenate the labels into one variable -- Label -- which contains the combinations.

In order to save data for large series, where more than 200 characters are needed for the combinations, a second process is carried out. An array of Labels is prepared. Lbls[] always contains the same variable label or is blank. A shorter array of Columns is prepared -- Cols[] -- whose dimension is equal to the maximum number of items checked in the series. Each non-blank Lbls[] is moved to next empty Cols[]. Cols[1] will always contain one label, though it may be different in each combination.

**Macro code excerpt 3**

```plaintext
%6. data: recode FREQ output:
convert Number to Combinations;
DATA ZFRQCOMB ;
array Lbls [* ] $ Lbl1 - Lbl &DIM_LBL ;
array Cols [* ] $ Col1 - Col &MAXCHKD ;
set ZFRQCOMB ;
  Delimitr = ' ' ; /*change to ' ' after 1st */
%6.1 change Number to binary string
  loop: change binary string to Label and Lbl*;
  BinStrng = put( N , binary &DIM_LBL . ) ;
  %DO I = 1 %TO &DIM_LBL ; /*-----------------------*/
    if substr( BinStrng , 1 , I ) = '1'
      then do ; Label = left( trim( Label )
        !! Delimitr
          !! ' &LBLAI . ' ) ;
        Delimitr = ' , ' ;
      LblI = &LBLAI ;
      end ;
/*-----------------------*/
  if length( Label ) = 200 then
    Label = ' ' ; /*substr( Label , 1 , 199 ) ;
    EmptyCol = 1 ; /*fill Cols from Lbl*/
    do I = 1 to dim( Lbls ) ; /*------------------------*/
      if Lbls [ I ] ne ' ' then do ;
        Cols [ EmptyCol ] = Lbls [ I ] ;
      EmptyCol = EmptyCol + 1 ;
      end ;
/*------------------------*/
    do I = 1 : dim( Lbls ) ; /* end;*/
```
Caution: a constraint on number of variables

Bytes contain eight bits. Eight bytes should allow this routine to handle 64 variables. SAS software (Win3.1 V6.11) stores very large integers \((2^{*52^+})\) in scientific notation (E-notation). This is not an accurate bit-string of the number representing the combinations. Therefore SHOWCOMB fails to return accurate results for more than 51 variables. A check of number of variables is provided.

CONCLUSION

When analyzing multiple-response data, a cross-tabulation from proc FREQ serves as a model to present the data. Users prefer to be able to read the labels of variables in each combination. Compressing the data saves system resources and allows large number of variables to be processed in a similar manner. When the data is decompressed, variable labels can be substituted for the variable values.

REFERENCES


Fehd, Ronald (1997) %ARRAY, %CHECKALL,

ACKNOWLEDGMENTS

These routines were developed over a period of five years while I crunched the numbers of survey data collected by the Model Performance Evaluation Program (MPEP) of the Division of Laboratory Systems, Public Health Practice Program Office of the Centers for Disease Control, and Prevention, Atlanta, Georgia. John Hancock, chief, Information Services Activity, encouraged me to write up these routines. I wish to thank Sharon Blumer, David Cross, Thomas Hearn, and William Schalla of the MPEP group for repeating the questions about the data often enough and in enough variation that I was finally able to comprehend the underlying pattern.

SAS is a registered trademark of SAS Institute, Inc. In the USA and other countries, © indicates USA registration.

Author: Ronald Fehd

Centers for Disease Control and Prevention

4770 Buford Hwy NE, MS-G25

Atlanta GA 30341-3724

e-mail: RJF2@phplds1.em.cdc.gov (D E L S one)

SAS-L archives: send e-mail to: SAScontrib@saserv.uga.edu

for %ARRAY subject: cntb0031: download

for %CHECKALL subject: cntb0032: download

for %SHOWCOMB subject: cntb0033: download

REFERENCES


Fehd, Ronald (1997) %ARRAY, %CHECKALL,

Acknowledgments

These routines were developed over a period of five years while I crunched the numbers of survey data collected by the Model Performance Evaluation Program (MPEP) of the Division of Laboratory Systems, Public Health Practice Program Office of the Centers for Disease Control, and Prevention, Atlanta, Georgia. John Hancock, chief, Information Services Activity, encouraged me to write up these routines. I wish to thank Sharon Blumer, David Cross, Thomas Hearn, and William Schalla of the MPEP group for repeating the questions about the data often enough and in enough variation that I was finally able to comprehend the underlying pattern.

SAS is a registered trademark of SAS Institute, Inc. In the USA and other countries, © indicates USA registration.

Author: Ronald Fehd

Centers for Disease Control and Prevention

4770 Buford Hwy NE, MS-G25

Atlanta GA 30341-3724

e-mail: RJF2@phplds1.em.cdc.gov (D E L S one)

SAS-L archives: send e-mail to: SAScontrib@saserv.uga.edu

for %ARRAY subject: cntb0031: download

for %CHECKALL subject: cntb0032: download

for %SHOWCOMB subject: cntb0033: download

Acknowledgments

These routines were developed over a period of five years while I crunched the numbers of survey data collected by the Model Performance Evaluation Program (MPEP) of the Division of Laboratory Systems, Public Health Practice Program Office of the Centers for Disease Control, and Prevention, Atlanta, Georgia. John Hancock, chief, Information Services Activity, encouraged me to write up these routines. I wish to thank Sharon Blumer, David Cross, Thomas Hearn, and William Schalla of the MPEP group for repeating the questions about the data often enough and in enough variation that I was finally able to comprehend the underlying pattern.

SAS is a registered trademark of SAS Institute, Inc. In the USA and other countries, © indicates USA registration.

Author: Ronald Fehd

Centers for Disease Control and Prevention

4770 Buford Hwy NE, MS-G25

Atlanta GA 30341-3724

e-mail: RJF2@phplds1.em.cdc.gov (D E L S one)

SAS-L archives: send e-mail to: SAScontrib@saserv.uga.edu

for %ARRAY subject: cntb0031: download

for %CHECKALL subject: cntb0032: download

for %SHOWCOMB subject: cntb0033: download
null