ABSTRACT

Multiple-response data from survey questionnaires where questions have the instruction "check all that apply" present a challenge to the SAS software programmer. For a simple question, the answer may be either A or B; the sum of percent response is 100%. For the series of variables in a multiple-response question, the answer may be both A and B; because response rate for each variable in the series is dependent upon the other variables, the sum of the percent response may be greater than 100%.

This paper examines the PROC Freq output data set and discusses the construction of a standardized data set containing frequency of response information for multiple-response data. The algorithm illustrated here encompasses both those presented in [2] and [3] as macros. Combining the processes and optimizing the algorithm has reduced the number of statements by two thirds.

INTRODUCTION

Simple questions may have simple answers. Simple questions may have complex answers when the question contains the phrase "Check all that apply". This paper reviews the output data set of PROC Freq on a single variable with a format. This data set is used as a model to construct a macro which produces a standardized data set with the frequencies of each variable in the series of multiple-response data.

THE ANSWER: (CHOOSE ONLY ONE)

Programmers may or may not be consulted when survey questionnaires are designed. When data are delivered we hope that the answers are in this form:

A:Apple  B:Banana  C:Cherry  ...

When data are simple, the programming exercise is straightforward. SAS software provides PROC Format to translate variable values into information. PROC Freq can create an output data set, which can be standardized by adding the format as a variable when saved for later use.

Program 1: Choose Only One

```
data Q01; length Label $ 40;
set Freq; Label = Put(Q01,$Q01.);
DATA ChzOnly1; input Q01 $ 1;
cards;
proc PRINT data = Q01; sum _numeric_
PROC Format; value $Q01 'A' = 'Apple'
'B' = Banana' 'C' = 'Cherry';
PROC Freq data = ChzOnly1;
tables Q01 / noprint
out = Freq;
```

This is the data set that we will use as a model for the information in a multiple-response series. It contains the information Value (Q01), Label, Count, and Percent of one variable. The internal relationships of this data are that the sum of Count is equal to the number of responses and the sum of Percent is 100%. These internal relations may not hold for multiple-response data.
THE ANSWERS: (CHECK ALL THAT APPLY)

☐ Apple  ☐ Banana  ☐ Cherry

What values mean "Checked"? The most common sets include: (Y,N), (T,F), (X,), etc. Here we use (1,0).

The next example data set illustrates multiple responses. The variables have labels, which will be used in place of the format. We use the same PROC Freq output option and examine the result.

```
Program 2: Check All
DATA Q02; set Q02A Q02B Q02C;

DATA Test; label Q02A = 'Apple'
   Q02B = 'Banana'
   Q02C = 'Cherry';
input Q02A Q02B Q02C; cards;
  0 . . 1 20.0
  1 . . 4 80.0
PROC Freq data = Test;
  tables Q02A / noprint out = Q02A;
  tables Q02B / noprint out = Q02B;
  tables Q02C / noprint out = Q02C;
```

This data set has some of the desired information. We will eliminate the observations with values of zero, which represent 'not checked', combine the series of variables into one variable and save the label. This process will be repeated for each of the individual data sets of the series.

```
Program 3: Standardize FREQ data set
DATA Q02;
drop Q02A;
length Value $ 8 Value Label Count Percent
retain Value 'Q02A';
set Q02A;
where Q02A = 1;
call label( Q02A, Label);
```

This is the desired product: a data set, or object, with a known set of variable names where each observation contains the name of the multiple-response variable, its label, the count and percent response. The sum of Count is greater than the number of observations; the sum of Percent is greater than 100%.

```
Program 4
proc FREQ data = QUERIES;
tables Q02A * Q02B * Q02C / list noprint out = FREQ;
proc PRINT data = FREQ label noobs;
```

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.

```
Program listing 2
Label Label Label
Q02A Q02B Q02C Count Percent
----- ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
Cherry 1 8.3333
Banana 2 16.6667
Banana 3 25.0000
Apple 3 25.0000
Apple 2 16.6667
Apple 1 8.3333
```
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.

<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>

PARAMETERS FOR THE MODULE

This module is used as a parameterized %include file. It requires a set of parameters for input and output.

The set of variables is defined by a common prefix, type in (char, num), variable length, and the value to be tested as True.

The Label or Suffix parameter provides a means of switching from reporting the suffixes of the variables in the list, a narrower and more meaningful report for the developer, to the use of the variable labels.

NOTES

Fehd's earlier algorithm, [2], used a macro array, [1], to produce a PROC Freq output data set of each variable. Each data set was processed and concatenated as in the examples above. The current algorithm combines both processes of [2] (400 lines) and [3] (500 lines); it replaces multiple procedure usage with an array of accumulator variables to create the Check-All and -Combination data set.

CONCLUSION

When analyzing a single variable with unique responses procedures FREQ and FORMAT can be combined to produce a data set containing information about that variable. When given a multiple-response data set to analyze, correct labels need to be applied to the variables in order to be able to produce a data set which describes the series in the same fashion as a single variable.

REFERENCES


List of programs on following pages

<table>
<thead>
<tr>
<th>Program</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>test data smryCheckAllTest</td>
<td>1</td>
</tr>
<tr>
<td>output smryCheckAllTest</td>
<td>1</td>
</tr>
</tbody>
</table>

Program: to run test data

options notes source source2 nomprint;
filename Pgm '<directory-reference>';
filename Pgm list;
%Inc Pgm(smryCheckAllTest);

Program: smryCheckAllTest

%Let Source2 = nosource2;

DATA Test;
label ID = 'Id'
   Q02A = 'Apple'   Q02B = 'Banana'   Q02C = 'Cherry'
   N031 = 'One'    N032 = 'Two'      N033 = 'Three'
   X08C = 'Commercial'  X08I = 'In House'  X08M = 'Manufacturer';
input @ 1 Id $char1.
   @ 2 Q02A $char1. @ 3 Q02B $char1. @ 4 Q02C $char1.
   @ 2 N031 1. @ 3 N032 1. @ 4 N033 1.
   @ 5 X08C $char1. @ 6 X08I $char1. @ 7 X08M $char1.; cards;
A100.X.
A110..X
B101X..
B101XXX
A010.
;
%Let Library = Work;     %Let Lib_Out = Work;
%Let Data_Set = Test;    %Let Id_List = Id;
%Let Where_Var = ;       %*names of vars used in where phrase;
%Let Where = 1;          %*default is numeric one : no subset;
%Let Prefix = N03;
%Let Type = num;         %*in (char num) for SQL usage;
%Let Length = 8;         %*of vars in series;
%Let True = 1;           %*char:'1' numeric:1;
%Let Data_Out = zChkAll&Prefix.&Type.;
%Inc Pgm(SmryCheckAll) / &Source2.;

%Let Prefix = Q02;
%Let Type = char;        %*in (char num) for SQL usage;
%Let Length = 1;         %*of vars in series;
%Let True = '1';         %*char:'1' numeric:1;
%Let Data_Out = zChkAll&Prefix.&Type.;
%Inc Pgm(SmryCheckAll) / &Source2.;

%Let Prefix = X08;       %*use Q02 type and length;
%Let True = 'X';         %*marks the spot;
%Let Where_Var = Q02A;
%Let Where = Q02A eq '1';
%Let Data_Out = zChkAll&Prefix.&Type.;
%Inc Pgm(SmryCheckAll) / &Source2.;

%Inc Pgm(SmryReport) / &Source2.;
### Summary

<table>
<thead>
<tr>
<th>MemName</th>
<th>Data Ids</th>
<th>Data Rows</th>
<th>% Rows/ Data</th>
<th>Where</th>
<th>% Rows/ Where</th>
<th>Where Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>zChkAllN03num</td>
<td>TEST</td>
<td>2</td>
<td>5</td>
<td>100.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>zChkAllQ02char</td>
<td>TEST</td>
<td>2</td>
<td>5</td>
<td>100.00</td>
<td>5</td>
<td>100.00</td>
</tr>
<tr>
<td>zChkAllX08char</td>
<td>TEST</td>
<td>2</td>
<td>4</td>
<td>80.00</td>
<td>4</td>
<td>100.00 Q02A eq ’1’</td>
</tr>
</tbody>
</table>

### Details

<table>
<thead>
<tr>
<th>MemName</th>
<th>ValueChr</th>
<th>Label</th>
<th>Count</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>zChkAllN03num</td>
<td>N031</td>
<td>One</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td></td>
<td>N032</td>
<td>Two</td>
<td>2</td>
<td>40.00</td>
</tr>
<tr>
<td></td>
<td>N033</td>
<td>Three</td>
<td>2</td>
<td>40.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>combinations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>One, Three</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>One</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>One, Two</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Two</td>
</tr>
<tr>
<td>zChkAllQ02char</td>
<td>Q02A</td>
<td>Apple</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td></td>
<td>Q02B</td>
<td>Banana</td>
<td>2</td>
<td>40.00</td>
</tr>
<tr>
<td></td>
<td>Q02C</td>
<td>Cherry</td>
<td>2</td>
<td>40.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>combinations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Apple, Cherry</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Apple</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Apple, Banana</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Banana</td>
</tr>
<tr>
<td>zChkAllX08char</td>
<td>X08C</td>
<td>Commercial</td>
<td>2</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td>X08I</td>
<td>In House</td>
<td>2</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td>X08M</td>
<td>Manufacturer</td>
<td>2</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>combinations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Commercial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Commercial, In House, Manufacturer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In House</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Manufacturer</td>
</tr>
</tbody>
</table>

*end of programs, and output of smryCheckAll;*