%CHECKALL: a macro to produce a frequency of response data set 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. 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 SAS software proc FREQ output data set and discusses the construction of a standardized data set containing frequency of response information for multiple-response data for graphics presentation.

The CHECKALL macro takes as parameters an output data set name, and a list of the multiple-response variables. The output data set contains the variable names and labels of the multiple-response variables. This data set is used by the SHOWCOMB macro to report the frequency of combinations of response. See Fehd (1996), (1997), %SHOWCOMB and %ARRAY.

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 the 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 CHZONLY;   input Q01 $ 1;   cards;
proc FORMAT;   value $Q01 'A' = 'Apple'
               'B' = 'Banana'
               'C' = 'Cherry';
proc FREQ data = CHZONLY;
tables Q01 / nprint out = FREQ;
data Q01;   length Format $ 40;
set FREQ;   Format = Put(Q01,$Q01.);
proc PRINT data = Q01;   sum _numeric_;

SAS output:

<table>
<thead>
<tr>
<th>Q01 Format</th>
<th>Count</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>A  Apple</td>
<td>1</td>
<td>16.6</td>
</tr>
<tr>
<td>B  Banana</td>
<td>2</td>
<td>33.3</td>
</tr>
<tr>
<td>C  Cherry</td>
<td>3</td>
<td>50.0</td>
</tr>
<tr>
<td>6</td>
<td>100.0</td>
<td></td>
</tr>
</tbody>
</table>

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 (Format), Count, and Percent of one variable. The internal relationships of this data set 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)

A:Apple    B:Banana    C:Cherry

What values mean “Checked”? The most common sets include: (Y,N), (T,F), (X,<dot>), 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 QUERIES;   label Q02A = 'Apple'
                Q02B = 'Banana'
                Q02C = 'Cherry';
input Q02A Q02B Q02C;   cards;   ;
proc FREQ data = QUERIES;
Our task now is to replicate this process for any series. The minimum parameter would be the name of the series of variables. In the test data presented here that name is the prefix of each of the series’ variables. This parameter is used as the output data set name. For one-time usage, a list of variables would be the second parameter. In production usage, data sets with the series of variable names are prepared from subsets of a proc CONTENTS output data set. The data set names are in the form: V&SERIES. See the test data provided with the macro below. In order to process each variable in the series, the list is converted to a series of macro variables with the macro %ARRAY routine, which accepts either a list or a data set as input. See Fehd (1996) %ARRAY. The default list for CHECKALL will be a data set previously prepared, with the option of accepting a user-prepared list.

Macro step 1:
%IF %‘ALIST’=%‘DATA’ %THEN
%ARRAY(VAR=DATA.,&SERIES.,VAR=Name);
%ELSE %ARRAY(VAR=&LIST, .);

The macro call %ARRAY(VAR=&LIST) performs the following macro variable assignments:

%LET VAR1 = Q02A;
%LET VAR2 = Q02B;
%LET VAR3 = Q02C;
%LET DIM_VAR = 3;

If the LIST is provided in a data set, then the macro variables must be concatenated in order to be used in the SAS language keep and array statements. The parameter LIST has a dummy value; it is used to contain the concatenated list.

Macro step 2:
%MACRO VAR_LIST;
%DO I = 1 %TO DIM_VAR; &VAR&I. %END; %MEND;
%IF %‘ALIST’=%‘DATA’ %THEN %LET LIST = %VAR_LIST;

Data set information and subsetting, TRUE

Since the internal consistency of these data sets is known to be unstable, then both the number of observations and response rate must be saved to be stored with the data set. An observation is output when one or more of the variables is ‘checked’. In order to accommodate different sets of data values, a parameter TRUE is provided. The TRUE parameter allows values to be picked from multi-valued data such as (‘A’, ‘B’, ‘C’,..). For the next example program the value of TRUE is numeric one. The test data provided with the macro contains a series of variables where values of ‘checked’ are equal to character one, numeric one, and another series with ‘X’ meaning ‘checked’.

Macro step 3:
* macro parameter: TRUE=1
DATA CHECKALL;
retain NmbRChkd NmbRResp 0;
set LIBRARY.&DATASET
mobs = NmbRobs ;
NmbRChkd = 0;
array CheckAll[*] &LIST;
do I = 1 to dim(CheckAll) until(NmbRChkd);
  NmbRChkd + (CheckAll[I] = &TRUE.); end;
if NmbRChkd then do; output;
  NmbRResp + 1; end;
call symput('NMBROBS',trim(left(put(NmbRobs,8.))));
call symput('NMBRRESP',trim(left(put(NmbRresp,8.))));
run;
%LET PCTRESP = %eval(100* &NMBRRESP /&NMBROBS);

Loop through the series of variables

The macro %ARRAY returned the dimension of the series of variables to be processed in the variable DIM_VAR. This allows the following loop definition; each variable is accessed by the convention &VAR&I., where VAR is the name of the macro array and I is the index.

%DO I = 1 %TO &DIM_VAR;
  *... (keep = &VAR&I.); %END;

As in Program 2 above, each variable is run through proc FREQ and output to a data set. This data set is standardized and concatenated to a work data set for the next step.

Macro step 4:

%DO I = 1 %TO &DIM_VAR;/*---------------------------*/
proc FREQ data = CHECKALL (keep = &VAR&I.); tables &VAR&I. / noprint out = FREQ;

DATA FREQ; length Value $ 8 Label $ 40;
retain Value ' &VAR&I.';
set FREQ;
where &VAR&I. = &TRUE;
call label(&VAR&I,Label);

%IF &I = 1 THEN %DO;
  DATA SERIES; set FREQ;
%END;
%ELSE %DO;
  proc APPEND base = SERIES data = FREQ; %END;
/*--------------------------- %DO I=1:&DIM_VAR*/ %END;

Optimizing for storage

While the data sets created by this routine have the same set of variables, it is necessary for later processing to optimize their storage length. This data step checks the maximum width of the Label. In the last data step these values will be used to change the width of Label.

Macro step 6:

DATA _NULL_;
retain WLabel 0;
set SERIES;
WLabel = max(WLabel, length(Label));
call symput('WLABEL',trim(left(put(WLabel,2.))));

Save the data for later usage

As the data is saved to a permanent data set the width and format of Label are added.

Macro step 7:

DATA LIBRARY.&SERIES.;
attrib Label length = $ WLABEL format = $Char&WLABEL..;
set SERIES;

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 the order to be able to produce a data set which describes the series in the same fashion as a single variable. For similar processing of combinations of responses to multiple-response data see Fehd, (1996), %SHOWCOMB.

REFERENCES


Fehd, Ronald (1997) %ARRAY, %CHECKALL, %SHOWCOMB: Proceedings of the Twenty-Second Annual SAS Users Group International Conference.

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
4770 Buford Hwy NE  MS-G25
Atlanta GA 30341-3724  voice: 770/488-4316
e-mail: RJF2@phpdlS1.em.cdc.gov (D eL S one)
SAS-L archives: send e-mail
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.
