ABSTRACT

The FREQ procedure without instruction from the user will provide easily read formatted crosstabulation tables. However if another organization of the output, such as a two dimensional display of the frequencies is needed, one can build into a macro nearly as much automatic control so that, say, a compact tabular arrangement can be created for printing the PROC FREQ results without giving the macro any more information than what already has to be provided to PROC FREQ.

To have greater flexibility one must use the PROC FREQ output data set and design an original display from scratch. The purpose of this paper is to write such a macro with emphasis on computing the number of rows and columns, handling of missing values and missing rows and detecting some of the data keying errors that may distort these computations.

INTRODUCTION

Frequencies produced by the FREQ procedure often need to be displayed in table form. But the PROC FREQ log output (Figure 1.) and the PROC FREQ output file option (Figure 2.) often are not suitable. In order to increase efficiency a macro, FREQ ARR, was created to call PROC FREQ, extract the counts from the PROC FREQ output data sets and organize the output in an array form (Figure 3.).

For example, suppose the students were all given proficiency exams in mathematics and the scores need to be reported by letter grade and class. The first row of the table should have the number of students getting an A in the Freshman, Sophomore, Junior and Senior
classes; in the second row should be those getting a B, etc. The output data set in this case will have 6 records; one for each of the grades A, B, C, D and F and the count of the missing grades. On each record will be 6 variables - one containing the name, MATH, one containing the letter grade, and four giving the count for each of the four classes with this letter grade.

In the following we address the more general case in which we treat:

1) a three-way table,
2) treating several courses instead of just math in one macro call and,
3) how, by using keyword parameters, errors in computing dimensions of the array can be avoided and certain outliers in the data can be identified by error messages.

CONSTRUCTING THE 2-DIMENSIONAL ARRAYS

The macro, FREQ ARR, suppresses the printout of PROC FREQ, but provides a simple means for putting the PROC FREQ frequency counts into tables for which you supply the labels and format for printing.

This macro collects frequencies found in the Output dataset option of a PROC FREQ 3-way crosstabulation and arranges them in an array such that the numbers are positioned in rows and columns ready to be printed in a tabular form.

The statement for this crosstabulation is of the form:
```
tables W*X*Y /out=Z noprint;
```

There is a row in the array for each value of X and a column for each value of Y. There are distinct arrays for each value of W. Z is Output data set containing a record with the count of occurrences for each combination of W, X and Y. Thus each such count is to be placed into an array cell indexed by the values of X, Y and W. The entry will be zero if the combination does not occur in the data being processed.

PROC FREQ uses array format when it prints crosstabulation tables for two or more variables as in Figure 1, and if no data falls into one of the cross-tab cell frequency count. But when using the PROC FREQ Output data set the array form must be constructed and the zero entries for any null cells must be created. It may happen that an entire row is zero-filled. Processing the Z data set to create the array is the main function of this macro. Supplying the zeros is one of the major problems solved by it.

PROCESSING SEVERAL X VARIABLES IN ONE MACRO CALL

A more general situation occurs when there is not just one variable X, but several as, Xn, Xn+1, ... Xn, m<=n, which occur together in the data, all having the same range of values and using the same values of Y. (This is the usual situation in the author's applications.) In this case all of the crosstabulations,
```
W*Xn*Y, m<=i<=n,
```
can be computed by one macro call with tables statements being generated for each value of i.

In order to identify the table resulting from each Xi, a variable, named catg,
```
catg='Xi', where m<=i<=n,
```
is created by the macro. Catg is useful as a sort variable for determining the order in which the arrays (tables) will be printed.

SHALL FREQ ARR COMPUTE HOW MANY ROWS AND COLUMNS IT NEEDS?

The macro can be further automated to compute the number of rows and columns that are needed to display the frequency tables. In the example in Figure 3, first and last rows are determined by minimum and maximum grades of all of the classes portrayed in the table, that is, these rows are determined by the min and max of all the values of the variable, X. If several Xi are
passed in one call to FREQ.ARR then we use the min and max of all of the numbers in these Xi, \( m \leq i \leq n \).

It is being assumed that the values of \( X_{\text{min}} \) and \( X_{\text{max}} \) as well as those of other keyword parameters are integers.

\( X_{\text{min}} \) and \( X_{\text{max}} \) may be passed as keyword parameters. If they are passed the macro will compare all \( X \) values with them, delete any counts that are not within these bounds, write a message to the log file and put the rest in the table. If either \( X_{\text{min}} \) or \( X_{\text{max}} \) is not passed, then the macro will compute what it needs from the data.

Similarly, FREQ.ARR will compute \( \text{Max col} \) which is the maximum of all of the values of \( Y \) or, that is, the number of columns of the array if \( \text{Max col} \) is not passed. Otherwise it will check if any \( Y \) values exceed \( \text{Max col} \), delete any counts having \( Y \) values that do, write a message to the log file and continue processing the rest. The array created by the macro will have \( \text{Max col} \) columns and if there is no data associated some value of \( Y \), the column will be zero filled.

**MISSING DATA**

Occasionally data contains missing values and since PROC FREQ keeps count of these, FREQ.ARR devotes one row of its cross-tabulation array to displaying the counts of missing observations.

The keyword, Mis, designates the number of the row into which counts of missing observations are to be placed. This value should not be in the range from \( m \) to \( n \), but may be before or after. If no value for Mis is specified, the value \( n+1 \) is used placing this count in the last row of the array.

**THE CALLING STATEMENT**

\[
\%\text{freq\_arr} (\text{inn, outt, } X, m, n, W, Y, \text{min}_x =, \text{max}_x =, \text{max}\_\text{col} =, \text{mis} =)
\]

\( \text{Inn} \) and \( \text{outt} \) represent the input and output data sets. The input data set needs to supply variables corresponding to the macro variables, \( W \), the \( Xi \) and \( Y \). (The actual variable names in the data set may be different.)

The resulting output consists of several sets of arrays - one complete set of the \( X_i \times Y \) arrays, \( m \leq i \leq n \), for each value of \( W \). The output is sorted by \( W \) and a category variable, \( \text{catg} \), whose values are the strings, \( 'Xm' \), \( 'Xm+1' \), \ldots \( 'Xn' \), and next by the value of the \( Xi \) variable. The variables in the output data set are \( W \), \( \text{catg} \), \( X \), \( Y_1 \), \( Y_2 \), \ldots \( \text{Ymax\_col} \) where \( \text{max\_col} \) is the number of columns in the array.

As an input variable \( Y \) was not subscripted; in the introductory example variable \( Y \) took on a value for each year of high school - 1, 2, 3 and 4. But in the output \( Y \) appears with subscripts - one for each value of the variable, \( Y \); that would be \( Y_1 \) for year 1 of high school, \( Y_2 \) for year 2, etc. with the appropriate labels over the columns.

The output variable \( X \) does not have subscripts because each of the \( Xi \) take exactly the same range of values and so can be denoted without a subscript. In the example exams in any other courses would all have the same grades: \( A, B, C, D \) and \( F \). Thus \( X \) is subscripted in the input but not in the output.

**FREQ.ARR CODE EXPLANATION**

The Tables Statement.

FREQ.ARR calls PROC FREQ immediately and uses a %do loop to create the appropriate number of Tables statements. There is one Tables statement for each variable, \( Xi \), to be analyzed. One output data set is created for each variable analyzed.

Next all of the PROC FREQ output data sets are concatenated together so that
the whole output can be searched to find the minimum and maximum Xi value and the largest column number.

BUILDING THE ARRAYS

A %do loop is used to process the output data sets — one pass through the do loop for each table statement. Here a single dimension array, arr, is used to accumulate all of the values for one line, or row, of output. Arr is always zero-filled before accumulation begins and is output by code that is reached by a link statement from three places in the macro.

CONCLUSION

The FREQ_ARR macro is a post processor for PROC FREQ organizing the frequency counts into tabular arrays ready for printing in a report. Several variables in the input dataset provided they each have the same range of values can be processed by one call to the macro. The keyword parameters if supplied will control the dimensions of the output array; any count not associated with "in range" X & Y values will be deleted with error message written to the log. Any keyword parameter not supplied will be computed from data values found in the input data set.

The intent of the flexibility built into the keyword parameters is to make a robust macro capable of giving the user as much information as can be extracted without failing to run to completion.

Following is a listing of the macro FREQ_ARR complete with comments. Following that is an output listing.
put xx= "&y=" &y "&w=" &w count=; delete; end;
end;

if "&max_col"=" then do; if &Y>maxcol then do; put "A &Y value was found greater than max col=&max_col.

as specified in the macro call"; put xx= "&y=" &y "&w=" &w count=; delete; end;
end:

if xx>. then m grd=m grd<&x>
x grd=x grd<&x>
m col=m col<&x>
if last then do;

if 

x min"=" then do;
call symput(‘x min’~put(m grd,2.»
if "x max"=" then do;
call symput(‘x max’~put(x grd,2.»
if "mis"=" then
call symput(‘mis’~put(x grd+1,2.»
if "max col"=" then
call symput(‘max col’,left(put(m col,3.»
end:
run;
data null ;
%If %eval (&&x_min<=&&mis) &
%eval(&&mis<=&&x_max)
%then %put ERROR --the missing
value grade=&&mis is inside the x min
x max range = (&&x_min.,&&x_max.).

+%*Put the Proc Freq output into array
form row by row, by W, category & grade;
%do i=&&m %to &&n;
data &&x&i
(keep=&&w catg grade &&y.&i=&&y&max_col.);
set f X&i;
by &&w &&x&i;
array arr{&&max col.}
&&y.1-&&y&max_col.;
length catg $ 3 xmin xmax maxcol 4;
retain catg grade xmin xmax maxcol;

if first.&w then do;
%* initialize the variables;
xmin=input("&x_min",4.);
xmax=input("&x_max",4.);
maxcol=input("&max_col",4.);
catg="&&x&i"
grade=.
%* Note: "&x&i is a valid grade value;
do j=1 to &&max_col;
arr[j]=0;
end;
end;

if (.<&x&i & &&x&i < xmin) | xmax <
&&x&i | maxcol & &Y then delete;
do while (grade < &x&i);
%* Write a zero-filled row if no;
%* one in the group, W, receives the;
%* grade, x, for that row;
link out;
end;

arr[&&y]+count;

if last.&x&i then do;
%* When all records with the same grade;
%* (or &x&i) have been read then output;
%* Reinitializing for the next row;
link out;
do j=1 to &&max_col;
arr[j]=0;
end;
end;

if last.&w then do while
(grade <= &x_max);
%* when a tail end grade is not;
%* achieved then write a zero-;
%* filled row for it;
end;
return;
out: output;
if grade=. then grade=&&x_min.;
else grade=grade+1;
return;
proc print data=&x&i;
title 
&&x&i counts X&i in macro rows";
%mend freq_arr;

proc print data=&&x&i;
title "&x&i counts &x&i in macro rows";
%mend freq_arr;

907
Figure 4.

Math Frequency Tabulations from X1

<table>
<thead>
<tr>
<th>CITY</th>
<th>CLASS1</th>
<th>CLASS2</th>
<th>CLASS3</th>
<th>CLASS4</th>
<th>CATG</th>
<th>GRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x1</td>
<td>.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>x1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>x1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>x1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>x1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>x1</td>
<td>4</td>
</tr>
</tbody>
</table>

Science Frequency Tabulations from X2

<table>
<thead>
<tr>
<th>CITY</th>
<th>CLASS1</th>
<th>CLASS2</th>
<th>CLASS3</th>
<th>CLASS4</th>
<th>CATG</th>
<th>GRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x2</td>
<td>.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>x2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>x2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>x2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>x2</td>
<td>4</td>
</tr>
</tbody>
</table>

English Frequency Tabulations from X3

<table>
<thead>
<tr>
<th>CITY</th>
<th>CLASS1</th>
<th>CLASS2</th>
<th>CLASS3</th>
<th>CLASS4</th>
<th>CATG</th>
<th>GRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x3</td>
<td>.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>x3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>x3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>x3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>x3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>x3</td>
<td>4</td>
</tr>
</tbody>
</table>

History Frequency Tabulations from X4

<table>
<thead>
<tr>
<th>CITY</th>
<th>CLASS1</th>
<th>CLASS2</th>
<th>CLASS3</th>
<th>CLASS4</th>
<th>CATG</th>
<th>GRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x4</td>
<td>.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>x4</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>x4</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>x4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>x4</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>x4</td>
<td>4</td>
</tr>
</tbody>
</table>

Author: Hugh Geary
Trilogy Consulting Corporation
630 Morning Street
Worthington, Ohio 43085

908