

MACROS FOR CONVERTING PROC FREQ AND PROC UNIVARIATE OUTPUT FILES INTO ARRAY FORM FOR DATA ANALYSIS ON A LARGE SCALE NDA CLINICAL TRIAL DATABASE

Hugh Geary, Trilogy Consulting Corporation

INTRODUCTION

Three prospectively randomized clinical trials were conducted to compare the relative benefits of a new chemotherapy for the treatment of adult acute myelogenous leukemia (aml). The data collected from 500 patients served as pivotal evidence in support of a New Drug Application (NDA) submitted to the Food and Drug Administration (FDA). Of critical importance to both the sponsor and the FDA was the timely submission of an accurate analysis integrating all patient safety data.

Statistical analysis for this project required numerous tables of frequency counts. Such tables display frequencies arranged in rows and columns. To save development time macros, FREQ_ARR and UNIV_ARR, were both written to extract statistics from the SAS output data sets of PROC FREQ[®] and PROC UNIVARIATE[®], respectively, and present them in an array form.

For example, suppose data is collected on the response to a drug by several patient groups with each group receiving a different dose level and each patient response being graded on a scale of 1 - 5. The macro, Freq_arr, puts the frequency counts into arrays, dose level by columns and grade by rows. Suppose also that instead of each patient receiving a single grade that there are several attributes of the patient's health each being graded. Further suppose similar data has been collected for each of several drugs and that an array display is to be made for each. By a single invocation the macros described below can generate all of the arrays associated with this crosstabulation. The UNIV_ARR arrays are similar except that there would be a row for each of the statistics: means, medians, standard deviations, ranges, etc.

PROBLEMS TO SOLVE

PROC FREQ does use array format when it prints crosstabulation tables for two or more variables and if no data falls into one of the crosstabs cells then it automatically supplies the zero for that cell's

frequency count. But when using the SAS output data set from PROC FREQ the array form must be constructed and the zero entries for any null cells must be created.

If PROC FREQ has processed some records with missing data, the counts of missing data may need to be presented as an additional row in the crosstabulation array form created by FREQ_ARR.

GENERAL DESCRIPTION

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 the user supplies the labels and format for printing.

This macro collects frequencies found in the OUT=SASdataset 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 tables 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 separate arrays for each value of W. Z is a SAS data set containing a record with the count of occurrences for each combination of W, X and Y. Each such count is to be placed in an array according to the value of W, X and Y. The entry will be zero if the combination does not occur in the data being processed. It may happen that an entire row is zero-filled. Processing the Z data sets to create the array is the main purpose of this macro, but supplying the zeros is one of the major problems solved by the macro.

A more general situation occurs when there is not just one variable X, but several as, X_m, X_{m+1}, X_{m+2}, ... X_n, where m ≤ n, which occur together in the data, all having the same range of values and being processed in exactly the same way. (This is the usual situation in the

author's applications.) In this case all of the crosstabulations, $W \times X_i \times Y$, $m \leq i \leq n$, can be computed by one macro call with tables statements being generated for each value of i .

INPUT/OUTPUT

The input data set needs to supply variables corresponding to the macro variables: W , the X_i and Y although the input data set names may be different.

The resulting output consists of several sets of arrays - one complete set of the $X_i \times Y$, $m \leq i \leq n$, arrays - for each value of W . It is sorted by W and a category variable, $catg$, whose values are the strings, ' X_m ', ' X_{m+1} ', ... ' X_n ', and next by the value of the X_i variable. The variables in the output data set are W , $CATG$, X , Y_1 , Y_2 , ... Y_{max_col} where max_col is the number of columns in the array.

As an input variable Y was not subscripted, but in the output, Y appears with subscripts - one for each value of the variable, Y . The output variable X does not have subscripts because the values taken on by the X_i variables are exactly the same numbers used by X and they are the row numbers or row designators.

PARAMETERS USED IN THE MACRO

In the calling statement,

```
Xfreq_arr (inn,outt,X,m,n,W,Y,min_x=,
           max_x=,max_col=,mis=),
```

Inn and $outt$ represent the input and output data sets. X represents the name of the subscripted variable for which frequency counts are being made. m and n specify the lower and upper extremes of the range of the subscript i for the variables, X_i .

The remaining parameters are keyword parameters and if the user does not supply them in the calling statement, they will be assigned by the macro. As mentioned above a row may have to be created zero-filled if the data set being processed doesn't contribute any counts to a given row. Thus if the first row(s) or last row(s) need to be zero-filled then min_x and/or max_x need to be given to supply

the desired beginning row number and/or the last row number because otherwise they will be unknown. If either of these is not supplied the macro will search through the data and supply the smallest and/or largest X_i value as needed.

Since PROC FREQ counts the occurrences of missing data in the data set being processed, a row-number, mis , needs to be assigned for the missing count row. If not supplied the macro will use the value $n+1$.

Max_col is to be used to supply the number of columns in the output array. If not supplied the number will be computed.

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, X_i , to be analyzed. One SAS output data set is created for each variable analyzed.

Computing Keyword Parameters if not specified.

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

Building the Arrays.

Again a %do loop is used to process the SAS output data sets - one pass through the do loop for each Tables statement. Here a single dimensioned 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.

The code for FREQ_ARR follows.

```

***      FREQ_ARR.MAC      ***;

%macro freq_arr
  (inn,outt,X,m,n,W,Y,x_min=,
    x_max=,max_col=,mis=);
proc freq data=&inn;
  %do i=&m %to &n;
    tables &W*&X&i*&Y /out=f_X&i
      noprint;
  %end;

  %* Next Compute the upper and lower
  bounds of the number of rows and
  columns to be used in the output
  array if these bounds have not
  already been given;

  %* The step has no output other
  than the macro variables: min_x,
  max_x, max_col & mis;
data _null_;
  set %do i=&m %to &n;
    f_X&i (rename=(&X&i=xx))
  %end;
  end=last;
  retain m_grd 10000 x_grd m_col;

  if xx>. & "&x_min"="." then do;
    if &x_min>xx then put
      "An X value was found less
      than x-max=&x_max as
      specified in the macro call";
  end;

  if "&x_max"="." then do;
    if &x_max<xx then do;
      put "An X value was found
      greater than x-max=&x_max as
      specified in the macro call";
      put _all_;
    end;
  end;

  if "&max_col"="." then do;
    if &max_col<&Y then do;
      put "A Y value was found
      greater than max_col=
      &max_col. as specified
      in the macro call";
      put _all_;
    end;
  end;

  if xx>. then m_grd=m_grd<>xx;
  x_grd=x_grd<>xx;
  m_col=m_col<>&Y;

  if last then do;
    if "&x_min"="." | m_grd<&x_min
      then call
      symput('x_min',put(m_grd,2.));

```

```

    if "&x_max"="." | x_grd>&x_max
      then call
      symput('x_max',put(x_grd,2.));

    if "&mis"="." do;
      if "&x_max"="." | x_grd>&x_max
        then call
        symput('mis',put(x_grd+1,2.));
      else call
        symput('mis',put(&x_max+1,2.));
    end;

    if "&max_col"="." | m_col>&max_col
      then call
      symput('max_col',
        left(put(m_col,3.)));
  end;

data _null_;
  %if &x_min <= &mis &
    &mis <= &x_max %then
    %put ERROR - the missing
    value grade=&mis is inside
    the x_min x_max range =
    (&x_min.,&x_max.);
  %* put x_min= &x_min. x_max= &x_max.
    max_col= &max_col. mis= &mis.;

  %*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.1-&Y&max_col.);
      set f_X&i;
      by &W &X&i;
      array arr{&max_col.}
        &Y.1-&Y&max_col.;
      length catg $ 3;
      retain catg grade;

  %* initialize the variables;
  %* Note: "." is a valid value;
  if first.&W then do;
    catg="&X&i";
    grade=.;
    do j=1 to &max_col;
      arr{j}=0;
    end;
  end;

  %* Write a zero-filled row if no;
  %* one in the group, W, receives;
  %* the grade, X, for that row;
  do while (grade < &X&i);
    link out;
  end;

  arr{&Y}+count;

```

```

%* When all records with the same
  grade (or &X&i) have been read
  then write the output and then
  reinitialize for the next row;
  if last.&X&i then do;
    link out;
    do j=1 to &max_col;
      arr{j}=0;
    end;
  end;

%* when a tail end grade is not;
%* achieved then write a zero-;
%* filled row for it;
  if last.&W then do while
    (grade <= &x_max);
    link out;
  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";
%end;

proc sort data=&outt;
  by &W catg grade;

%* Assign the desired row # for the;
%* missing data counts row;
data &outt;
  set %do i=&m %to &n; &X&i %end;;
  if grade=.
    then grade=input("&mis",3.);
%mend freq_arr;

```

UNIV_ARR CODE EXPLANATION

The second macro, UNIV_ARR, calls PROC UNIVARIATE, suppresses its printout, extracts statistics from its SAS data set and puts them in an array. This macro will process the same input file as FREQ_ARR.

Statistics such as, median, would come out more naturally in columns with rows representing groups in the population, but UNIV_ARR transposes the output so that there is a row for medians, a row for St. Dev., etc.

The code for FREQ_ARR follows.

```

***          UNIV_ARR.MAC          ***;

* This macro calls PROC UNIVARIATE and
  transposes the SAS output so that the
  statistics, such as, say, means, will
  appear in a row in the output instead
  of a column.;
* inn - input dataset with W, Y and the
  variables Xm, ... Xn where m and n are
  the first and last of the set of con-
  secutative integers used to subscript
  the variables under study.;
* outt - output dataset with variable W,
  catg (category values are: 'Xm',...
  'Xn') and Y1,Y2, ... Yhi where hi is
  the number of values of W which is,
  likewise, the number of columns in the
  output table.;
* W (group) is the major sort key of inn;
* Y is the minor sort key of inn and is
  the number of an output column into
  which the statistics for the variables
  Xm, ... Xn will go;

%macro univ_arr (inn,outt,X,m,n,W,Y,hi);

proc sort data=&inn out=bbb; by &W &Y;
  %* inn input is W &Y & the
    categories: X1, X2, ... Xn;
  %* uni the univariate output is &W &Y
    & stats on each of X1, X2, ... Xn;
proc univariate data=bbb noprint;
  var &x&m-&x&n;
  by &W &Y;
  output out=uni
    mean=mean&m-mean&n
    median= med&m-med&n
    min= min&m-min&n
    max= max&m-max&n
    std= std&m-std&n
    n= n&m-n&n;

data %do i=&m %to &n;
  U&i (keep=&W &Y n&i mean&i med&i std&i
    min&i max&i)
  %end;;
  set uni;
* i.e. here each very long record of uni is
  broken down into n-m+1 records which will
  eventually be transposed into columns;
* each Ui is derived from Xi and
  has &W &Y and stats for Xi;

* The following is an explicit example of
  the U&i above;
*data u1
  (keep=&W &Y n1 mean1 med1 std1 min1 max1)
  u2
  (keep=&W &Y n2 mean2 med2 std2 min2 max2)
  u3
  (keep=&W &Y n3 mean3 med3 std3 min3 max3);

```

- * TRANSPOSE THE U_i VECTORS IN COLUMNS;
- * input is values of &Y down and stats across whereas output is stats down and values of &Y across;

```
%do i=&m %to &n;
  data u&i(keep=&W catg &Y.1-&Y&hi stat);
  set u&i;
  by &W;
  length stat $7;

  retain mn1-mn&hi md1-md&hi sd1-sd&hi
         mil-mi&hi mx1-mx&hi nn1-nn&hi;

  array uni{6,&hi} mn1-mn&hi md1-md&hi
         sd1-sd&hi mil-mi&hi mx1-mx&hi nn1-nn&hi;

  if first.&W then
    do k=1 to 6;
      do l=1 to &hi;
        uni{k,l}=0;
      end;
    end;
  catg="&x&i";

  uni{1,&Y}=mean&i;
  uni{2,&Y}=med&i;
  uni{3,&Y}=std&i;
  uni{4,&Y}=min&i;
  uni{5,&Y}=max&i;
  uni{6,&Y}=n&i;

  if last.&W then do;
    %do j=1 %to &hi; &Y&j=mn&j; %end;
    stat='Mean';
    output;
    %do j=1 %to &hi; &Y&j=md&j; %end;
    stat='Median';
    output;
    %do j=1 %to &hi; &Y&j=sd&j; %end;
    stat='St.Dev.';
    output;
    %do j=1 %to &hi; &Y&j=mi&j; %end;
    stat='Min';
    output;
    %do j=1 %to &hi; &Y&j=mx&j; %end;
    stat='Max';
    output;
    %do j=1 %to &hi; &Y&j=nn&j; %end;
    stat='n';
    output;
  end;
%end;

data &outt;
  set %do i=&m %to &n; u&i %end;;
  by &W catg;
* interleaving the Ui by &W & catg to
  produce outt;
%mend univ_arr;
```

To illustrate these two macros a program has been written which calls both macros to process the following data set producing the table on the following page.

In the input data set class=1 is the 7th grade, 2 is the 8th grade, etc. The X1 column contains math scores, X2 contains Science scores and X45 contains the student ages. The variable, W, is named City here. Not all of the variables are used in this example.

Suppose the variables X1, X2, X3, X4 and X5 had been created originally, then naming two new variables X45 and X46 would cause their output in the table to automatically come between the outputs of X4 and X5.

OBS	CITY	CLASS	X1	X2	X3	X4	X45	X46	X5
1	1	1	.	3	2	1	12	1	.
2	1	1	3	3	2	2	11	1	.
3	1	1	.	3	2	2	12	1	3
4	1	1	2	3	1	2	12	1	2
5	1	1	3	2	4	4	12	2	2
6	1	1	2	4	2	4	13	2	4
7	1	1	2	4	2	3	12	2	3
8	1	1	2	1	4	1	12	1	2
9	1	1	2	2	0	1	11	2	2
10	1	1	2	1	3	2	13	2	2
11	1	2	.	0	4	3	14	1	2
12	1	2	2	1	2	3	13	1	.
13	1	2	.	3	2	2	13	1	4
14	1	2	2	2	2	3	14	2	3
15	1	2	2	2	1	3	12	1	4
16	1	2	3	3	2	3	13	1	3
17	1	2	0	4	1	2	14	2	4
18	1	2	4	3	4	1	13	1	2
19	1	2	2	4	3	0	13	2	2
20	1	2	.	3	1	3	14	2	3
21	1	3	4	2	2	3	14	1	2
22	1	3	3	3	3	4	14	2	3
23	1	3	2	2	2	1	15	1	2
24	1	3	4	3	1	0	14	1	2
25	1	3	4	3	2	3	13	2	1
26	1	3	2	2	3	3	15	2	1
27	1	3	3	4	1	3	15	2	3
28	1	3	3	3	4	1	13	1	.
29	1	3	4	1	4	4	15	2	3
30	1	3	3	3	1	1	16	1	2
31	1	4	3	3	4	2	15	2	3
32	1	4	3	3	2	2	16	2	3
33	1	4	3	1	2	4	17	2	2
34	1	4	2	2	3	1	16	2	2
35	1	4	.	1	3	3	15	2	3
36	1	4	4	3	2	2	15	2	.
37	1	4	3	2	2	2	16	1	1
38	1	4	2	2	2	1	16	1	.
39	1	4	.	.	1	2	18	2	2
40	1	4	2	2	2	1	16	1	.
41	1	5	3	3	3	2	17	1	3
42	1	5	2	2	1	2	15	2	2
43	1	5	2	2	4	3	16	1	2
44	1	5	3	3	2	1	16	1	3
45	1	5	3	2	1	2	16	2	2
46	1	5	3	2	2	3	16	1	.
47	1	5	2	3	1	4	17	1	2
48	1	5	2	3	4	4	17	1	2
49	1	5	.	4	.	2	17	1	4
50	1	5	3	1	1	3	17	1	1
51	1	6	3	3	4	2	16	2	2
52	1	6	3	2	4	2	17	1	2
53	1	6	3	2	3	1	17	2	1
54	1	6	2	3	2	2	17	2	2
55	1	6	2	.	4	2	17	1	3
56	1	6	.	2	0	3	18	1	.
57	1	6	3	1	1	4	17	2	3
58	1	6	4	3	3	2	17	1	2
59	1	6	4	3	4	1	17	1	2
60	1	6	3	2	4	3	18	2	3

TABLE XXX

PROC FREQ and PROC UNIVARIATE Output Arranged by FREQ_ARR and UNIV_ARR

		7th Grade	8th Grade	Freshman	Sophomore	Junior	Senior
Math	A	0	1	4	1	0	2
	B	2	1	4	4	5	5
	C	6	4	2	3	4	2
	D	0	0	0	0	0	0
	F	0	1	0	0	0	0
	Missing	2	3	0	2	1	1
Math	Mean	2.3	2.1	3.2	2.8	2.6	3.0
	Median	2.0	2.0	3.0	3.0	3.0	3.0
	St.Dev.	0.5	1.2	0.8	0.7	0.5	0.7
	Min	2	0	2	2	2	2
	Max	3	4	4	4	3	4
	n	8	7	10	8	9	9
Science	A	2	2	1	0	1	0
	B	4	4	5	3	4	4
	C	2	2	3	4	4	4
	D	2	1	1	2	1	1
	F	0	1	0	0	0	0
	Missing	0	0	0	1	0	1
Science	Mean	2.6	2.5	2.6	2.1	2.5	2.3
	Median	3.0	3.0	3.0	2.0	2.5	2.0
	St.Dev.	1.1	1.3	0.8	0.8	0.8	0.7
	Min	1	0	1	1	1	1
	Max	4	4	4	3	4	3
	n	10	10	10	9	10	9
Age	Mean	12.0	13.3	14.4	15.8	16.4	17.1
	Median	12.0	13.0	14.5	16.0	16.5	17.0
	St.Dev.	0.7	0.7	1.0	0.6	0.7	0.6
	Min	11	12	13	15	15	16
	Max	13	14	16	17	17	18
	n	10	10	10	10	10	10
Age	18	0	0	0	0	0	2
	17	0	0	0	1	5	7
	16	0	0	1	6	4	1
	15	0	0	4	3	1	0
	14	0	4	3	0	0	0
	13	2	5	2	0	0	0
	12	6	1	0	0	0	0
	11	2	0	0	0	0	0

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. indicates USA registration.

Hugh Geary
630 Morning Street
Worthington, Ohio 43085
(614) 885-3164