Contrasting programming techniques for summarizing voluminous SAS output using the SAS Output Delivery System (ODS) (PROC FREQ as an example)

Stuart Long, Westat
Lawrence Park, Westat

ABSTRACT

SAS® ODS provides programmers with the ability to extract selected information from a procedure and store it in datasets. Such datasets can then be combined to summarize the results from numerous procedures. The SAS Macro facility can be used to execute and extract information from repetitively called SAS procedures.

"Macro Variable Arrays" can simplify the extraction of information from SAS procedures when identical or similar code is executed repeatedly on a multitude of datasets and/or variables. This paper describes Macro Variable Arrays and contrasts these with other programming methods to obtain summary datasets from ODS output.

Although PROC FREQ is used in this paper, these techniques can serve as a blueprint for extracting and summarizing output from other SAS procedures. This is an advanced tutorial intended for SAS programmers knowledgeable with the SAS Macro facility.

INTRODUCTION

SAS can generate an enormous volume of output, such as when procedures are run repeatedly in analyzing a large number of variables. Often, only a selected, and possibly small, portion of the output is of immediate interest. The SAS Output Delivery System (ODS) provides a convenient mechanism for extracting statistics for customized summarization. ODS can be used in conjunction with macro processing to automate the generation and presentation of summary data from SAS procedures. The information presented in this paper serves as a blueprint for summarizing statistics produced from SAS procedures executed iteratively.

We use PROC FREQ as the example for this paper. The dataset used consists of forty medical conditions, identified by the range "medcond1" – "medcond40", and 23 symptoms, identified by the range "dizziness" -- "absentminded". All variables are dichotomous, where 0=No and 1=Yes. This assignment of 0 and 1 as the dichotomous values is necessary for proper execution of the macros discussed in this paper. Our task is to produce a summary dataset of frequencies, percentages, chi square probabilities and odds ratios with confidence limits for 2x2 tables crossing each outcome with each symptom. Once this summary dataset has been produced, the user can create a summary table using whatever reporting method within the SAS System that they prefer, or the data can be exported for other processing.

Consider the following code:

```
PROC FREQ DATA=mydata;
    TABLES (dizziness--absentminded)*
           (medcond1-medcond40)
           / CHISQ RELRISK;
RUN;
```

The above PROC FREQ will produce 920 2x2 crosstabs and associated statistics, and would generate 920 pages of output in the SAS listing. By using SAS ODS iteratively within a macro, we reduce the output to a 20 page summary report. In this paper we present 3 different methods to do this using: 1) arrays of macro variables, 2) call execute, and 3) by variable processing in PROC FREQ. Each of these is described in detail in the sections that follow, and the appendices provide the full code for each of the methods.

METHOD 1: MACRO VARIABLE ARRAYS

Have you ever wished you could run code like this?

```
ARRAY symptom {*} dizziness--absentminded;
ARRAY disease {*} medcond1-medcond40;
DO i = 1 TO DIM(symptom);
    DO j = 1 TO DIM(disease);
        PROC FREQ DATA=mydata;
          TABLES symptom(i)*disease(j)
                 / CHISQ RELRISK;
        RUN;
    END;
END;
```

Since arrays must be defined and executed within a data step, we see that the above code is syntactically incorrect. However, by defining two sets of Macro variables, we can achieve the above logic within the syntax requirements of the SAS system.

First, let's dissect the structure of a data step array.

```
DATA myset;
  SET mydata;
  ARRAY symptom {*} dizziness--absentminded;
  total_symptoms=0;
  DO i = 1 TO DIM(symptom);
     IF symptom(i)=1 THEN total_symptoms+1;
  END;
RUN;
```

- The name of the array is "symptom".
- The index is "i".
- The array element is identified by "symptom(i)".
- For i=1, symptom(i) = dizziness.

In the example below we use a Macro Variable Array, which has a nearly isomorphic relationship to the structure of a data step array.

```
%LET symptom1=dizziness;
%LET symptom2=nervous;
%LET symptom3=absentminded;
%MACRO arrays;
  %DO i = 1 %TO 3;
      PROC FREQ DATA=mydata;
        TABLES &&symptom&i*medcond1;
      RUN;
  %END; /*%DO Loop*/
%MEND arrays;
```

- The root name of the array is "symptom".
- The index is "i".
- The array element is identified by "&&symptom&i".
- For i=1, &&symptom&i = dizziness.

All items in the Macro Variable Array can be mapped to the data step array, except the "&&" syntax controls the final resolution of the Macro Variables in the array. The advantage of the Macro Variable Array is that the programmer can create iterative code, which executes outside of the data step on a list of variables in a manner similar to iterative code that is executed on a list of variables within a data step.

CREATING MACRO VARIABLE ARRAYS

Below are three methods to create Macro Variable Arrays within a program.

Method 1: %LET is used to assign values to macro variables. By incrementing the index, or suffix, of the names of macro variables to which elements of the array are assigned, you can create a usable Macro Variable Array. Below we create an array of 23 symptoms, where the index is incremented by one, for each successive element assigned to the array:

```
%LET row_1 =dizziness   ; %LET row_2 =nervous  ;
%LET row_3 =speechProb  ; %LET row_4 =nausea   ;
%LET row_5 =depression  ; %LET row_6 =lowEnergy;
%LET row_7 =tasteChange ; %LET row_8 =sweating ;
%LET row_9 =appetiteLoss; %LET row_10=headache ;
%LET row_11=fastHeart   ; %LET row_12=numbHands;
%LET row_13=badBalance  ; %LET row_14=irritable;
%LET row_15=unConscious ; %LET row_16=trembling;
%LET row_17=blurredSight; %LET row_18=insomnia ;
%LET row_19=concentrate ; %LET row_20=armsWeak ;
%LET row_21=nightVision ; %LET row_22=twitching;
%LET row_23=absentminded;
```

The Macro Variable Array with elements row_1 to row_23 contains the values dizziness - - absentminded.

Method 2: If your values are already defined with elements that contain incremental suffixes, you can use a macro to make assignments:

```
%MACRO assign(__root,__n);
  %DO i = 1 %TO &__n;
      %GLOBAL col_&i;
      %LET col_&i=&__root&i;;
  %END;
%MEND assign;
%assign(medcond,40)
```

The above code generates 40 %LET statements which assign the 40 elements in medcond1 - medcond40 to 40 macro variables in the range of col_1 to col_40. It is identical to executing the following code;

```
%LET col_1 =medcond1 ;  %LET col_2 =medcond2 ;
%LET col_3 =medcond3 ;  %LET col_4 =medcond4 ;
%LET col_5 =medcond5 ;  %LET col_6 =medcond6 ;
%LET col_7 =medcond7 ;  %LET col_8 =medcond8 ;
%LET col_9 =medcond9 ;  %LET col_10=medcond10;
%LET col_11=medcond11;  %LET col_12=medcond12;
%LET col_13=medcond13;  %LET col_14=medcond14;
%LET col_15=medcond15;  %LET col_16=medcond16;
%LET col_17=medcond17;  %LET col_18=medcond18;
%LET col_19=medcond19;  %LET col_20=medcond20;
%LET col_21=medcond21;  %LET col_22=medcond22;
%LET col_23=medcond23;  %LET col_24=medcond24;
%LET col_25=medcond25;  %LET col_26=medcond26;
%LET col_27=medcond27;  %LET col_28=medcond28;
%LET col_29=medcond29;  %LET col_30=medcond30;
%LET col_31=medcond31;  %LET col_32=medcond32;
%LET col_33=medcond33;  %LET col_34=medcond34;
%LET col_35=medcond35;  %LET col_36=medcond36;
%LET col_37=medcond37;  %LET col_38=medcond38;
%LET col_39=medcond39;  %LET col_40=medcond40;
```

Method 3: Although it requires less execution time to assign values to a macro array outside of a data step by using %LET statements, it may be preferable to make the assignments within a data step with the aid of data step arrays.

```
DATA _NULL_;
  SET mydata;
  ARRAY symptoms {*} dizziness--absentminded;
  ARRAY diseases {*} medcond1-medcond40;
  CALL SYMPUT("dim_row",DIM(symptoms));
  CALL SYMPUT("dim_col",DIM(diseases));
  DO i = 1 TO DIM(symptoms);
    CALL SYMPUT("row_" || LEFT(i),
                VNAME(symptoms(i)));
  END;
  DO i = 1 TO DIM(diseases);
    CALL SYMPUT("col_" || LEFT(i),
                VNAME(diseases(i)));
  END;
  STOP;
RUN;
```

In the above data step, CALL SYMPUT is used to make assignments of values to the macro variables "row_1 – row_23" and "col_1 – col_40". In the statement

```
CALL SYMPUT("row_" || LEFT(i),
            VNAME(symptoms(i)));
```

the first argument in the CALL SYMPUT is the name of the macro variable, and the second argument contains the variable that will be assigned to the macro variable. Where i= 1:

```
"row_" || LEFT(i)  becomes "row_1"
VNAME(symptoms(i)) becomes "dizziness"
```

This statement resolves to:

```
CALL SYMPUT("row_1","dizziness");
```

This is similar to making the following assignment outside of a data step using %LET:

```
%LET row_1=dizziness;
```

Once the code for the data step in example 3 is executed, Macro Variable Arrays are generated which are equivalent to the arrays created in examples 1 and 2. In addition, we have created macro variables containing the dimensions of these Macro Variable Arrays.

USING MACRO VARIABLE ARRAYS TO EXECUTE PROCEDURES ITERATIVELY

The two Macro Variable Arrays created in the previous data step can be used outside of a data step as follows:

```
%MACRO runfreq(dimrow,dimcol,dset);
  %DO i = 1 %TO &dimrow;
      %DO j = 1 %TO &dimcol;
          PROC FREQ DATA=&dset;
            TABLE &&row_&i * &&col_&j
                  / CHISQ RELRISK;
          RUN;
      %END; /*%DO Loop j*/
  %END; /*%DO Loop i*/
%MEND runfreq;

%runfreq(&dim_row,&dim_col,mydata)
```

We know that &dim_row = 23 and &dim_col = 40. Thus the above macro, "runfreq" will generate 23 x 40, or 920 PROC FREQ 2x2 tables. The first three times through the %DO Loop will generate the following SAS code:

```
PROC FREQ DATA=mydata;
  TABLE dizziness*medcond1  / CHISQ RELRISK;
RUN;
PROC FREQ DATA=mydata;
  TABLE dizziness*medcond02 / CHISQ RELRISK;
RUN;
PROC FREQ DATA=mydata;
  TABLE dizziness*medcond03 / CHISQ RELRISK;
RUN;
```

SAS PROCEDURE DATA OBJECTS AND ODS TRACE

The statistics and other information generated by SAS procedures are held internally in several data objects. The components of these objects and their names are specific to each procedure. The ODS TRACE statement gives the programmer a method to identify the names of the results objects that are generated by a given procedure. The following code shows how to use the ODS TRACE statement to identify the names of the data objects generated by PROC FREQ.

```
ODS TRACE ON;
PROC FREQ DATA=mydata;
  TABLES dizziness*medcond1 / CHISQ RELRISK;
RUN;
ODS TRACE OFF;
```

FIGURE 1: SAS Log Listing for TRACE Statement

```
9   ODS TRACE ON;
10  PROC FREQ DATA=mydata;
11   TABLES dizziness*medcond1 /CHISQ ELRISK;
12  RUN;

Output Added:
-------------
Name:      CrossTabFreqs
Label:     Cross-Tabular Freq Table
Data Name:
Path:Freq.Dizziness_by_medcond1.CrossTabFreqs
-------------
Output Added:
-------------
Name:      ChiSq
Label:     Chi-Square Tests
Template:  Base.Freq.ChiSq
Path:      Freq.Dizziness_by_medcond1.ChiSq
-------------
Output Added:
-------------
Name:      FishersExact
Label:     Fisher's Exact Test
Template:  Base.Freq.ChisqExactFactoid
Path: Freq.Dizziness_by_medcond1.FishersExact
-------------
Output Added:
-------------
Name:      RelativeRisks
Label:     Relative Risk Estimates
Template:  Base.Freq.RelativeRisks
Path: Freq.Dizziness_by_medcond1.RelativeRisks

NOTE: There were 2529 observations read from
      the data set mydata.
NOTE: PROCEDURE FREQ used:
      real time          0.44 seconds
      cpu time           0.01 seconds

13   ODS TRACE OFF;
```

Four output objects are generated by this PROC FREQ, based on the options that the programmer has requested. Figure 1 shows how the SAS Log identifies each of these as they are added to the SAS listing.

Figure 2 shows the default SAS listing from the PROC FREQ call. Referring back to Figure 1, it is easy to match the components of the listing to their names generated by the ODS TRACE request. As noted in the introduction, our task is to summarize column frequencies and percents, the Mantel-Haenszel chi square probability, and the odds ratios with confidence limits. These statistics are the 8 highlighted items in Figure 2. The problem is to extract only these components from the much larger set of information printed

FIGURE 2: PROC FREQ Output

```
The Frequency Procedure

Table of Dizziness by medcond1

Dizziness(Dizziness episodes)
       medcond1(Diagnosed with Arthritis)
Frequency|
Percent  |
Row Pct  |
Col Pct  | 0) No  | 1) Yes  |   Total

0) No    |  1698  |    79   |   1777
         | 67.14  |  3.12   |  70.26
         | 95.55  |  4.45   |
         | 71.11  | 56.03   |

1) Yes   |   690  |    62   |    752
         | 27.28  |  2.45   |  29.74
         | 91.76  |  8.24   |
         | 28.89  | 43.97   |

Total       2388      141      2529
            94.42     5.58    100.00


Statistics for Table of Dizziness by medcond1

Statistic                    DF    Value    Prob
─────────────────────────────────────────────────
Chi-Square                    1   14.4856  0.0001
Likelihood Ratio Chi-Square   1   13.5495  0.0002
Continuity Adj. Chi-Square    1   13.7729  0.0002
Mantel-Haenszel Chi-Square    1   14.4799  0.0001
Phi Coefficient                    0.0757
Contingency Coefficient            0.0755
Cramer's V                         0.0757

Fisher's Exact Test
─────────────────────────────────
Cell (1,1) Frequency (F)        1698
Left-sided Pr <= F            0.9999
Right-sided Pr >= F        1.587E-04

Table Probability (P)      8.009E-05
Two-sided Pr <= P          2.813E-04

Estimates of the Relative Risk (Row1/Row2)
Type of Study              Value 95% Confidence
                                        Limits
─────────────────────────────────────────────────
Case-Control (Odds Ratio)   1.93   1.36    2.72
Cohort (Col1 Risk)          1.04   1.01    1.06
Cohort (Col2 Risk)          0.53   0.39    0.74
Sample Size = 2529
```

GENERATING ODS OUTPUT DATASETS

The SAS ODS OUTPUT statement allows the programmer to save the results from procedure objects to datasets. We use this to extract the desired results from PROC FREQ into three datasets. In the code below, PROC FREQ is executed within two nested %DO loops inside a macro, and the macro "onerec" is called to further process the output datasets.

```
%MACRO runfreq( dimrow = ,
                dimcol = ,
                dset =  ,
                out =   ,
              );
  %DO i = 1 %TO &dimrow;
    %DO j = 1 %TO &dimcol;
      ODS OUTPUT CrossTabFreqs = __ctab
                 ChiSq         = __chi
                 RelativeRisks = __rrisk;
      PROC FREQ DATA=&dset;
        TABLES &&row_&i * &&col_&j
               / CHISQ RELRISK;
      RUN;
      ODS OUTPUT CLOSE;
      %onerec(row=&&row_&i, col=&&col_&j)
    %END; /* %DO Loop j */
  %END; /* %DO Loop i */
%MEND runfreq;
```

FIGURE 3: PROC PRINT of ODS Datasets

```
Dataset=__ctab
                                      R    C
    d      m              F           o    o
    i      e              r           w    l
    z      d        _     e     P     P    P    M
    z      c     _  T     q     e     e    e    i
    i      o     T  A     u     r     r    r    s
    n      n     Y  B     e     c     c    c    i
O   e      d     P  L     n     e     e    e    s
b   s      0     E  E     c     n     n    n    i
s   s      1     _  _     y     t     t    t    n
                                                g
──────────────────────────────────────────────────
1 0) No  0) No  11 1 1698  67.14 95.55 71.10 .
2 0) No  1) Yes 11 1   79   3.12  4.44 56.02 .
3 0) No       . 10 1 1777  70.26  .     .    .
4 1) Yes 0) No  11 1  690  27.28 91.75 28.89 .
5 1) Yes 1) Yes 11 1   62   2.45  8.24 43.97 .
6 1) Yes      . 10 1  752  29.73  .     .    .
7      . 0) No  01 1 2388  94.42  .     .    .
8      . 1) Yes 01 1  141   5.57  .     .    .
9      .      . 00 1 2529 100.00  .     .    0

Dataset=__chi
Obs Statistic                       DF Value Prob
──────────────────────────────────────────────────
1   Chi-Square                       1 14.48 0.0001
2   Likelihood Ratio Chi-Square      1 13.54 0.0002
3   Continuity Adj. Chi-Square       1 13.77 0.0002
4   Mantel-Haenszel Chi-Square       1 14.47 0.0001
5   Phi Coefficient                  _  0.07  _
6   Contingency Coefficient          _  0.07  _
7   Cramer's V                       _  0.07  _

Dataset=__rrisk
Obs      StudyType           Value Lower Upper
                                   CL    CL
──────────────────────────────────────────────────
1  Case-Control (Odds Ratio) 1.93  1.36  2.72
2  Cohort (Col1 Risk)        1.04  1.01  1.06
3  Cohort (Col2 Risk)        0.53  0.39  0.74
```

The datasets generated by the ODS OUTPUT statements are printed in Figure 3. The statistics we want are high-lighted. In the macro "onerec" (below), these statistics are extracted from the three datasets and combined into a one-record summary dataset. This summary dataset is then appended to a cumulative summary dataset, which will contain observations for all 920 2x2 CrossTabs executed by the original macro call. The programmer can then generate a summary report using any facility within SAS or export the data to any external reporting software desired. The following code for the macro "onerec" is continuous from the %MACRO statement to the %MEND statement:

```
%MACRO onerec(row=, col=);
```

First, the statistics to keep in the summary dataset are specified.

```
    DATA __onerec (KEEP = row_var col_var
                          n_col1 n_col2
                          p_col1 p_col2
                          Prob OddsRatio
                          LowerCl UpperCl);
      LENGTH row_var col_var $ 32;
      row_var = VNAME(&row);
      col_var = VNAME(&col);
```

Second, the information from the dataset created from the CrossTabFreqs table __ctab are extracted into the one-record summary dataset (__onerec). The SET statement reads the __ctab dataset and identifies two records, based on the condition specified in the WHERE clause, from which frequencies and percents will be kept for the one record output by the DATA Step,. The Frequency and ColPercent values for the two rows of __ctab are stored in different elements of the n_col and p_col arrays:

```
        ARRAY n_col (0:1);
        ARRAY p_col (0:1);
        DO UNTIL ( eof_ctab );
            SET __ctab (WHERE = (&row = 1 AND
                                 &col IN (0 1) ) )
                END = eof_ctab;
            n_col [ &col ] = Frequency;
            p_col [ &col ] = ColPercent;
        END;
```

Third, the record in __chi containing the probability for the Mantel-Haenszel Chi-Square statistic is output to the __onerec dataset:

```
        DO UNTIL ( eof_chi );
            SET __chi (WHERE =
                (SUBSTR(statistic,1,6) = "Mantel") )
                END = eof_chi;
        END;
```

Finally, the Case-Control Odds Ratio and confidence limits are extracted from the Relative Risks dataset, "__rrisk":

```
        DO UNTIL ( eof_rrisk );
            SET __rrisk end = eof_rrisk;
            IF (substr(studytype, 1 , 4 ) = "Case")
            THEN OddsRatio = Value;
        END; RUN;
```

Once all the statistics from the three output datasets have been read into the __onerec dataset, they are appended to a cumulative summary dataset (specified by the macro variable &out):

```
    PROC APPEND BASE = &out DATA = __onerec
              FORCE; RUN;
%MEND onerec;
```

METHOD 2: CALL EXECUTE

Let's reexamine the code presented at the start of this paper:

```
ARRAY symptom {*} dizziness--absentminded;
ARRAY disease {*} medcond1-medcond40;
DO i = 1 TO DIM(symntom);
   DO j = 1 TO DIM(disease);
      PROC FREQ DATA=mydata;
        TABLES symptom(i)*disease(j)
               / CHISQ RELRISK;
      RUN;
   END;
END;
```

As noted, this code is syntactically incorrect. However, SAS procedure calls can be generated within Data Steps by using the CALL EXECUTE routine. Using this, we can run the PROC FREQ iteratively within nested Data Step DO Loops with the code below. The code generated by the repeated use of CALL EXECUTE is stored in a temporary buffer and is executed after the completion of the data step.

```
DATA _NULL_;
 SET mydata;
 ARRAY symptom {*} dizziness--absentminded;
 ARRAY disease {*} medcond1-medcond02;
 DO i = 1 TO DIM(symptom);
   row=VNAME(symptom(i));
   DO j = 1 TO DIM(disease);
      col=VNAME(disease(j));
      CALL EXECUTE ('PROC FREQ DATA=mydata;' ||
                    'TABLES ' || row || '*' ||
                    col || ' / CHISQ RELRISK;' ||
                    ' RUN;');
      END;
   END;
   STOP;
RUN;
```

To make use of the ODS OUTPUT datasets described in the previous section, we rewrite the above code to have CALL EXECUTE invoke the macro "runfreq" instead of a direct call to PROC FREQ. The following code will generate 920 macro calls, which are executed after the completion of the Data Step.

```
DATA _NULL_;
  LENGTH rowvar colvar $ 32;
  SET mydata;
  ARRAY col (*) &colvars;
  ARRAY row (*) &rowvars;
  DO i = 1 TO DIM ( col );
    colvar = VNAME ( col[i] );
    DO j = 1 TO DIM ( row );
      rowvar = VNAME ( row[j] );
      CALL EXECUTE ('%runfreq ( DATA = mydata'
                    || " , row = "  || rowvar
                    || " , col = "  || colvar
                    || " , base = summary)"
                    );
    END;
  END;
  STOP;RUN;
```

The first time through the nested DO Loops, the following macro call will be generated at the end of the Data Step:

```
%runfreq ( data = mydata
         , row = dizziness
         , col = medcond1
         , base = summary)
```

The Macro Variable Array version of this program uses two nested %DO Loops to govern the iterative processing of the 920

PROC FREQs. In this version, the iteration is controlled by Data Step DO Loops, which generate the macro call using the CALL EXECUTE statement. The row and column variables in the TABLES statement of the PROC FREQ are standard macro variables that are passed as parameters of the macro call; these parameters replace the Marco Variable Arrays used in the previous method. Below is the revised version of the macro runfreq for use with the CALL EXECUTE statement.

```
%MACRO runfreq (data =  ,
                row = rowvar ,
                col = colvar ,
                base =
                );
    ODS OUTPUT CrossTabFreqs = __ctab
               ChiSq         = __chi
               RelativeRisks = __rrisk;
    PROC FREQ DATA = &data;
       TABLES &row * &col / CHISQ RELRISK;
    RUN;
    ODS OUTPUT CLOSE;
    %onerec(row=&row, col=&col);
%MEND runfreq;
```

The "onerec" macro used here is identical to the one written for the Macro Variable Array version in the previous section.

NOTE: In SAS Version 8.2, using CALL EXECUTE to generate large amounts of code (> 1000 lines) can result in truncation and random failure to execute code. For details and a patch, see http://support.sas.com/techsup/unotes/SN/005/005243.html.

METHOD 3: A SINGLE PROC FREQ

The third method uses an entirely different strategy for solving the task. Here, we use a data step to collect the counts of the cells in each 2x2 table into a single 3-dimensional matrix and process this through a single call to PROC FREQ.

For any given cross-tabulation, each observation on a dataset contributes a count of 1 to a single cell of the table. We use this fact and build a single frequency table for all 23 x 40 cross-tabs in a 3-dimensional array of dimension 23 x 40 x 4. The first 2 dimensions index on the 23 row and 40 column variables, and the third dimension indexes the counts of the 4 cells in the 2x2 cross-tabs for the row and column variables. The values in this array are assigned through two nested DO loops. Inside these loops, one of the 4 cell counters is incremented, depending on the values of the row and column variables. The following code performs this task.

```
RETAIN count 0 ;
ARRAY __col (*) &colvars ;
ARRAY __row (*) &rowvars ;
ARRAY count ( &dimcol , &dimrow , 0:3 );
DO i = 1 TO DIM ( __col ) ;
   DO j = 1 TO DIM ( __row ) ;
      count [i , j , 2*__col[i] + __row[j]] + 1;
   END ;
END ;
```

Therefore, if row variable "NAUSEA" (symptom 4) has the value 0 and column variable "MEDCOND5" has the value 1, then the code within the DO loops resolves to:

```
  count [ 5 , 4 , (2*1 + 0) ] +1;
or
  count [ 5 , 4 , 2 ] + 1;
```

which increments the appropriate counter for the two variables and their combined values.

Examine this from the perspective of a single 2x2 table.

```
nausea(Nausea episodes)
           medcond5(Diagnosed with Angina)
Frequency|
Percent  |
Row Pct  |
Col Pct  | 0) No   | 1) Yes    |
         |         |           |
0) No    |         | increment |
         |         | this cell |
         |         | count by 1|
_____|_____|_____|
         |         |           |
1) Yes   |         |           |
         |         |           |
_____|_____|_____|
```

When the final record of the dataset is read, each of the 3680 cell counts is output to a new dataset. This dataset is then processed in the macro "odsmodule", using a single PROC FREQ with a WEIGHT statement to generate the desired statistics.

```
%MACRO buildset(dset);
  DATA q ( KEEP = col row freq colval rowval) ;
    LENGTH col row $ 32 ;
    SET &dset  END = eof ;
    RETAIN count 0 ;

    ARRAY __col (*) &colvars ;
    ARRAY __row (*) &rowvars ;
    ARRAY __count ( &dimcol , &dimrow , 0:3 );
    DO i = 1 TO DIM ( __col ) ;
      DO j = 1 TO DIM ( __row ) ;
        __count [i , j , 2 * __col[i] + __row[j]]
                + 1;
      END ;
    END ;
    IF eof THEN DO  ;
      DO i = 1 TO DIM ( __col ) ;
        col = VNAME ( __col [i] ) ;
        DO j = 1 TO DIM ( __row ) ;
          row = VNAME ( __row[j] ) ;
          DO k = 0 TO 3 ;
            freq = count [ i , j , k ] ;
            colval = INT ( k / 2 ) ;
            rowval = MOD ( k , 2 ) ;
            OUTPUT ;
          END ;
        END ;
      END ;
    END ;
  RUN ;
%odsmodule
%MEND buildset;

%MACRO odsmodule;
  ODS OUTPUT CcrossTtabFreqs = __ctab
             ChiSq           = __chi
             RrelativeRrisks = __rrisk;
  PROC FREQ DATA = q ;
    BY col row NOTSORTED;
    TABLE rowval * colval / CHISQ RELRISK ;
    WEIGHT freq ;
  RUN ;
  ODS OUTPUT CLOSE ;
  RUN;
  %summaryset *see Appendix 3;
%MEND odsmodule;
```

DISCUSSION OF THE THREE METHODS

We have presented 3 solutions to summarize numerous frequency cross-tabs. All three make use of macros and ODS output datasets to extract selected pieces of the output to efficiently process large volumes of information.

The first 2 solutions are very similar; both use looping to iteratively run PROC FREQ, which generates the desired statistics. The Macro Variable Array version uses straightforward macro coding in which nested %DO loops control processing over arrays of macro variables. In contrast, the CALL EXECUTE version uses a data step to build a sequence of SAS statements to run; here, DO loops direct processing of variables in Data Step arrays. The PROC FREQ calls are executed after the completion of the Data Step. An advantage of the Macro Variable Array version is that the variables analyzed may be a mixture of numeric and character. The code in our CALL EXECUTE version would have to be modified somewhat to accommodate mixed variable types.

Both of these methods provide a blueprint for designing other, possibly more complex, summarizations from analyses in SAS. They are easily tailored to process output from procedures other than PROC FREQ and can be adapted to combine and summarize results from multiple procedures[1].

The third method we presented, using a single PROC FREQ, cannot be generalized to other procedures, but provides an interesting contrast to the other 2 solutions. It has the potential advantage in run time; instead of executing 920 PROC FREQs, only one PROC FREQ is run. Thus, if execution time is a significant concern, this method would be preferred for processing frequency tables.

Code for all three methods is presented in appendices one to three. These will be of interest to a programmer. An end user does not necessarily care about the inner workings of the macros, but may be concerned only with obtaining the end product, the summary dataset.   Appendix four shows how the programmer can provide the user with a simple interface to run these macros with a minimum of information. To use the macro, the user needs to provide 1) the name of the dataset to be analyzed, 2) the name of the summary dataset to be produced, and 3) the names of the row and column variables.   Appendix five provides code to produce the summary dataset shown in Figure 5. If the user only wants the summary dataset, then s/he can use the output dataset (Figure 4) as needed.

CONCLUSION

ODS provides the SAS programmer with an efficient method for creating summary datasets from voluminous output. Different SAS programming techniques can be used to take advantage of the ODS utility, depending on the coding preferences of the programmer. The task presented in this paper requires a fair amount of knowledge about the SAS Macro facility on the part of the programmer. However, the end user does not have to be concerned with these programming details and can be provided a simple way to use these macros. Once the final summary dataset has been created, the end user can generate a report with the method provided by the programmer, or with any other SAS reporting facility, or external method if additional reporting needs are required.

DISCLAIMER: The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

ACKNOWLEDGEMENTS

REFERENCES

1. Long, S., Darden, R. (2003). <u>A MACRO Using SAS ODS to Summarize Client Information from Multiple Procedures.</u> *Proceedings of the 28th Annual SAS Users Group International Conference.*

CONTACT INFORMATION

Stuart Long (long3@niehs.nih.gov)
Westat
1009 Slater Road, Suite 110
Durham, NC 27703

APPENDIX 1: The Macro Variable Array method.

```
%MACRO runfreq(dset);
  %IF %SYSFUNC(EXIST(&out)) %THEN %DO;
    PROC DATASETS;
      DELETE &out;
    QUIT;  RUN;
  %END;
  %DO j = 1 %TO &dimcol;
   %DO i = 1 %TO &dimrow;
     PROC DATASETS; DELETE __ctab __chi __rrisk;
     QUIT; RUN;
     ODS LISTING CLOSE;
     ODS OUTPUT CrossTabFreqs = __ctab
                ChiSq        = __chi
                RelativeRisks = __rrisk;
     PROC FREQ DATA=&dset ORDER=INTERNAL;
       WHERE &&row_&i>.z AND &&col_&j>.z;
       TABLES &&row_&i * &&col_&j
              / CHISQ RELRISK; RUN;
     ODS OUTPUT CLOSE;
     ODS LISTING;
     %onerec(row  = &&row_&i ,
             col  = &&col_&j ,
             rlab = %NRBQUOTE(&&rlab_&i) ,
             clab = %NRBQUOTE(&&clab_&j)
            );
   %END; /*__i %LOOP*/
  %END; /*__j %LOOP*/
%MEND runfreq;

%MACRO onerec( row  = ,
               col  = ,
               rlab = ,
               clab =       );
 DATA __onerec (KEEP=row_var col_var n_col1
                     n_col2 p_col1 p_col2
                     col_label row_label
                     col1_val col2_val Prob
                     OddsRatio LowerCl UpperCl);
   LENGTH row_var col_var    $ 32
          col_label row_label $ 50;
   rowval    =  &row;
   colval    =  &col;
   row_var   = "&row";
   col_var   = "&col";
   col_label = "&clab";
   row_label = "&rlab";
   col1_val  = 0;
   col2_val  = 1;
   ARRAY n_col (0:1);
   ARRAY p_col (0:1);
   DO UNTIL ( eof_ctab );
    SET __ctab (WHERE=(&row=1 AND &col IN (0 1)))
                END = eof_ctab;
    n_col [ &col ] = Frequency;
    p_col [ &col ] = ColPercent;
```

```
   END;
   IF (EXIST("__chi")) THEN DO;
    DO UNTIL ( eof_chi );
      SET __chi (WHERE=(SUBSTR(statistic,1,6)
                     = "Mantel") )
             END = eof_chi;
    END;
   END;  ELSE Prob = .;
   OddsRatio = .;
   LowerCL   = .;
   Uppercl   = .;
   IF (EXIST("__rrisk")) THEN DO;
     DO UNTIL ( eof_rrisk );
       SET __rrisk END = eof_rrisk;
       IF (substr(studytype,1,4)="Case") THEN DO;
         OddsRatio = value;
         lcl = LowerCL;
         ucl = UpperCL;
       END;
     END;
     LowerCL = lcl;
     UpperCL = ucl;
   END;
   FORMAT col1_val col2_val _ny.;
   LABEL
    col_var   = "Name of Column Variable"
    col_label = "Label of Column Variable"
    col1_val  = "1st Value of Col Variable"
    col2_val  = "2nd Value of Col Variable"
    row_var   = "Name of Row Variable"
    row_label = "Label of Row Variable"
    n_col1    = "Frequency of Column 1 cell"
    n_col2    = "Frequency of Column 2 cell"
    p_col1    = "Percent of Column 1 cell"
    p_col2    = "Percent of Column 2 cell"
    Prob      = "Mantel-Haenszel Chi-Square"
    OddsRatio = "Odds Ratio"
    LowerCL   = "95% Lower Confidence Limit"
    UpperCL   = "95% Upper Confidence Limit";
 RUN;
 PROC APPEND BASE = &out DATA = __onerec FORCE;
 RUN;
%MEND onerec;

%MACRO freqinfo ( data = &syslast
               , out =
               , rowvars =
               , colvars =
               , report =          );
  %LET data  = &data; *force evaluation of DATA;
  %LET report = %UPCASE(&report) ;
  DATA _NULL_;
    SET &data;
    ARRAY __row {*} &rowvars;
    ARRAY __col {*} &colvars;
    CALL SYMPUT("dimrow",DIM(__row));
    CALL SYMPUT("dimcol",DIM(__col));
    DO i = 1 TO DIM(__row);
      CALL SYMPUT("row_" || LEFT(i),
               VNAME(__row(i)));
      CALL SYMPUT("rlab_" || LEFT(i),
               VLABEL(__row(i)));
    END;
    DO j = 1 TO DIM(__col);
      CALL SYMPUT("col_" || LEFT(j),
               VNAME(__col(j)));
      CALL SYMPUT("clab_" || LEFT(j),
               VLABEL(__col(j)));
    END;
    STOP;
  RUN;
  %runfreq(%STR(&data));
  %IF &report = YES %THEN
    %print_summary_table ( out = &out ) ;
%MEND freqinfo;
```

APPENDIX 2: The CALL EXECUTE method.

```
%MACRO runfreq (data =   ,
                row  = rowvar ,
                col  = colvar ,
                clab = clabel ,
                rlab = rlabel ,
                base =         );
  PROC DATASETS; DELETE __ctab __chi __rrisk;
  QUIT; RUN;
  ODS LISTING CLOSE;
  ODS OUTPUT CrossTabFreqs = __ctab
             ChiSq         = __chi
             RelativeRisks = __rrisk;
  PROC FREQ DATA = &data ORDER=INTERNAL;
     TABLES &row * &col / CHISQ RELRISK;
  RUN;
  ODS LISTING;
  ODS OUTPUT CLOSE;
  %onerec (row  = &row ,
           col  = &col ,
           rlab = &rlab,
           clab = &clab);
*NOTE: for Macro "onerec" see Appendix 1.;
%MEND runfreq;


%MACRO freqinfo ( data =
                , out =
                , rowvars =
                , colvars =
                , report = );
  %LET data = &data ; *force evaluation of DATA;
  %LET report = %UPCASE(&report);
  %IF %SYSFUNC(EXIST(&out)) %THEN %DO;
    PROC DATASETS;
      DELETE &out;
    QUIT;
    RUN;
  %END;
  DATA _NULL_;
   LENGTH rowvar colvar $ 32;
   SET &data;
   ARRAY col (*) &colvars;
   ARRAY row (*) &rowvars;
   DO i = 1 TO DIM ( col );
    colvar = VNAME ( col[i] );
    clabel = VLABEL( col[i] );
    DO j = 1 TO DIM ( row );
     rowvar = VNAME ( row[j] );
     rlabel = VLABEL( row[j] );
     CALL EXECUTE
       ('%runfreq(data = &data ' ||
       ", row = " || rowvar ||
       ", col = " || colvar ||
       ', clab = %NRBQUOTE(' || clabel || ')' ||
       ', rlab = %NRBQUOTE(' || rlabel || ')' ||
       ", base = &out)"
       );
    END;
   END;
  STOP;
 RUN;
 %IF &report = YES %THEN
     %print_summary_table(out=&out);
%MEND freqinfo;
```

APPENDIX 3: The Single PROC FREQ method.

```
%MACRO buildset(dset);
  DATA q ( KEEP = col_var row_var freq colval
                  rowval col_label row_label) ;
    LENGTH col_var row_var $ 32 ;
    SET &dset  END = eof ;
    ARRAY __col (*) &colvars ;
    ARRAY __row (*) &rowvars ;
    ARRAY count ( &dimcol , &dimrow , 0:3 ) ;
    RETAIN count 0 ;
    DO i = 1 TO DIM ( __col ) ;
      DO j = 1 TO DIM ( __row ) ;
        count[i, j, 2*__col[i]+__row[j]] + 1;
      END ;
    END ;
    IF eof THEN DO  ;
      DO i = 1 TO DIM ( __col ) ;
        col_var = VNAME ( __col [i] ) ;
        col_label = VLABEL ( __col [i] ) ;
        DO j = 1 TO DIM ( __row ) ;
          row_var = VNAME ( __row[j] ) ;
          row_label = VLABEL ( __row [j] ) ;
          DO k = 0 TO 3 ;
            freq = count [ i , j , k ] ;
            colval = INT ( k / 2 ) ;
            rowval = MOD ( k , 2 ) ;
            OUTPUT ;
          END ;
        END ;
      END ;
    END ;
  RUN ;
  %odsmodule
%MEND buildset;

%MACRO odsmodule;
  QUIT;
  ODS LISTING CLOSE ;
  ODS OUTPUT CrossTabFreqs =__ctab
             ChiSq         =__chi
             RelativeRisks =__rrisk;
  PROC FREQ DATA = q ;
    BY col_var row_var col_label row_label
       NOTSORTED;
    TABLE rowval * colval / CHISQ RELRISK ;
    WEIGHT freq ;
  RUN ;
  ODS OUTPUT CLOSE ;
  ODS LISTING ;
  %summaryset
%MEND odsmodule;

%MACRO summaryset;
  DATA __c (KEEP=row_var col_var n_col1 n_col2
               p_col1 p_col2 col_label
               row_label col1_val col2_val );
    ARRAY n_col (0:1) ;
    ARRAY p_col (0:1) ;
    SET __ctab ( WHERE = ( rowval = 1 AND
                           colval IN (0 1) ) )
             END=eof;
      BY col_var row_var NOTSORTED;
    DO i = 0 TO 1;
      IF colval=i THEN DO;
        n_col [ colval ] = Frequency ;
        p_col [ colval ] = ColPercent ;
      END;
    END;
    col1_val=0;
    col2_val=1;
    IF LAST.row_var THEN DO;
      OUTPUT __c;
      ncol1=.;n_col2=.;p_col1=.;p_col2=.;
    END;
```

```
      RETAIN n_col1 n_col2 p_col1 p_col2 ;  RUN;
   DATA __chi (KEEP = col_var row_var prob);
      SET __chi (WHERE=(SUBSTR(statistic,1,6) =
                       "Mantel" ) );  RUN;
   DATA __rrisk (KEEP = col_var row_var OddsRatio
                      UpperCl LowerCl);
      SET __rrisk (WHERE=(SUBSTR(studytype,1,4) =
                         "Case" ) ) ;
      RENAME Value = OddsRatio;  RUN ;
   PROC SORT DATA=__c;
      BY col_var row_var;  RUN;
   PROC SORT DATA=__chi;
      BY col_var row_var;  RUN;
   PROC SORT DATA=__rrisk;
      BY col_var row_var;  RUN;
   DATA &out;
      MERGE __c __chi __rrisk;
          BY col_var row_var;
      FORMAT col1_val col2_val _ny.;
      LABEL
        col_var   = "Name of Column Variable"
        col_label = "Label of Column Variable"
        col1_val  = "1st Value of Col Variable"
        col2_val  = "2nd Value of Col Variable"
        row_var   = "Name of Row Variable"
        row_label = "Label of Row Variable"
        n_col1    = "Frequency of Column 1 cell"
        n_col2    = "Frequency of Column 2 cell"
        p_col1    = "Percent of Column 1 cell"
        p_col2    = "Percent of Column 2 cell"
        Prob      = "Mantel-Haenszel Chi-Square"
        OddsRatio = "Odds Ratio"
        LowerCL   = "95% Lower Confidence Limit"
        UpperCL   = "95% Upper Confidence Limit";
   RUN;
%MEND summaryset;


%MACRO freqinfo ( data = syslast
                , out =
                , rowvars =
                , colvars =
                , report =         );
 %LET data = &data ; *force evaluation of DATA;
 %LET report = %upcase(&report) ;
 %IF %SYSFUNC(EXIST(&out)) %THEN %DO;
   PROC DATASETS;
     DELETE &out;
   QUIT; RUN;
 %END;
 DATA _NULL_;
   SET &data;
   ARRAY row {*} &rowvars;
   ARRAY col {*} &colvars;
   CALL SYMPUT("dimrow",DIM(row));
   CALL SYMPUT("dimcol",DIM(col));
 STOP; RUN;
 %buildset(%STR(&data));
 %IF &report = YES %THEN
   %print_summary_table ( out = &out ) ;
%MEND freqinfo;
```

```
TITLE "Medical Conditions crossed with Symptoms";
PROC FORMAT;
   VALUE _ny 0 = "0) No"
             1 = "1) Yes";
RUN;
%let METHOD = Macro_Array;   *See Appendix 1;
*%let METHOD = Call_Execute; *See Appendix 2;
*%let METHOD = One_Freq;      *See Appendix 3;
%INCLUDE "d:\sas\macro_library\&METHOD..SASINC";
%freqinfo ( data = mydata
          , out = summary
          , rowvars = dizziness--absentminded
          , colvars = medcond1-medcond40
          , report = yes );
```

APPENDIX 5: Example Macro to print a summary report
using DATA _NULL_ with PUT statements.

```
%MACRO print_summary_table(out = );
 DATA _NULL_ ;
   SET &out;
   LENGTH ret_var $ 32;
   FILE PRINT;
   IF _N_=1 THEN pagebreak=0;
   IF ret_var^=col_var THEN DO;
     IF page_break=2 THEN PUT _PAGE_;
     IF page_break=2 THEN page_break=0;
     page_break+1;
     PUT;
     PUT @29 col_label;
     PUT @32 col1_val _ny. @44 col1_val _ny.
         @57 ' CHI      Odds  95%'
         'Confidence Limit ';
     PUT @32 ' n      %     n       %     '
         'Square    Ratio     Lower    '
         'Upper';
     PUT %DO i = 1 %TO 89; '-' %END;;
   END;
   PUT row_label  @29 n_col1 5.0 @36 p_col1
       5.2 @42 n_col2 4.0 @48 p_col2 5.2
       @57 prob @66 oddsratio 5.2 @75
       lowercl 5.2 @84 uppercl 5.2;
   ret_var=col_var;
   RETAIN ret_var page_break;
 RUN;
%MEND print_summary_table;
```

FIGURE 4: shows the PROC CONTENTS output of the summary dataset. Each of the three methods described in this paper will produce the exact same summary dataset as shown below. It is important to note that the summary dataset is the end product of the design of the macros. The summary table provided in Figure 5 is an example of how to report the statistics in the summary dataset.

```
 Dataset: Work.summary                                    Observations:       920
                   -----List of Variables and Attributes-----
  #     Variable      Type    Len    Pos    Format     Label
 -------------------------------------------------------------------------------
  2     col_var       Char    32     112               Name of Column Variable
  3     col_label     Char    50     144               Label of Column Variable
  5     col1_val      Num      8       0    _NY.        1st Value of Col Variable
  6     col2_val      Num      8       8    _NY.        2nd Value of Col Variable
  1     row_var       Char    32      80               Name of Row Variable
  4     row_label     Char    50     194               Label of Row Variable
  7     n_col1        Num      8      16               Frequency of Column 1 cell
  8     n_col2        Num      8      24               Frequency of Column 2 cell
  9     p_col1        Num      8      32               Percent of Column 1 cell
 10     p_col2        Num      8      40               Percent of Column 2 cell
 11     Prob          Num      8      48    PVALUE6.4   Mantel-Haenszel Chi-Square
 14     oddsRatio     Num      8      72               Odds Ratio
 12     LowerCL       Num      8      56    10.4        95% Lower Confidence Limit
 13     Uppercl       Num      8      64    10.4        95% Upper Confidence Limit
```

FIGURE 5: shows an example summary table that has been generated using PUT statements in DATA _NULL_. The original SAS Output listing required 1 page per 2x2 crosstab. We can see, in this figure, that one table can be created for each medical condition that is crossed with all 23 symptoms. This will result in 40 tables. Two of the 40 tables can be printed on each page of the summary report, which reduces 46 pre-summarization pages of output down to just one page of summary statistics. A summary report, such as this, can be much easier to review

```
 Medical conditions crossed with symptoms


                         Diagnosed with Arthritis
                          0) No        1) Yes      CHI      Odds   95% Confidence Limit
                           n      %      n      %    Square   Ratio    Lower    Upper
 --------------------------------------------------------------------------------------
 Dizziness episodes        690  28.89    62  43.97   <.0001   1.93     1.36     2.72
 Nervous episodes         1374  57.54    80  56.74   0.8519   0.97     0.69     1.36
 Speech problems           149   6.24    17  12.06   0.0067   2.06     1.21     3.51
 Nausea episodes           759  31.78    46  32.62   0.8352   1.04     0.72     1.49
 Depression episodes       757  31.70    47  33.33   0.6858   1.08     0.75     1.55
 Low energy episodes      1491  62.44    91  64.54   0.6164   1.09     0.77     1.56
 Smell/taste change episodes 184 7.71    15  10.64   0.2089   1.43     0.82     2.49
 Sweating episodes         480  20.10    39  27.66   0.0308   1.52     1.04     2.23
 Appetite loss episodes    509  21.31    34  24.11   0.4317   1.17     0.79     1.75
 Headache episodes        1675  70.14    93  65.96   0.2925   0.82     0.58     1.18
 Fast heart rate episodes  412  17.25    30  21.28   0.2216   1.30     0.85     1.97
 Numb hands/feet episodes  667  27.93    64  45.39   <.0001   2.14     1.52     3.02
 Balance problems          317  13.27    37  26.24   <.0001   2.32     1.57     3.44
 Irritable/angry episodes  997  41.75    61  43.26   0.7236   1.06     0.75     1.50
 Lose consciousness episodes 73  3.06    10   7.09   0.0090   2.42     1.22     4.80
 Trembling episodes        329  13.78    26  18.44   0.1215   1.41     0.91     2.20
 Blurred vision episodes   287  12.02    27  19.15   0.0126   1.73     1.12     2.68
 Insomnia episodes        1091  45.69    73  51.77   0.1589   1.28     0.91     1.79
 Concentration problems    577  24.16    44  31.21   0.0591   1.42     0.99     2.06
 Arm/leg weakness episodes 384  16.08    50  35.46   <.0001   2.87     2.00     4.12
 Night vision problems     281  11.77    26  18.44   0.0184   1.70     1.09     2.64
 Twitching episodes        426  17.84    41  29.08   0.0008   1.89     1.29     2.76
 Absentminded episodes     651  27.26    51  36.17   0.0217   1.51     1.06     2.16
```