USING THE SAS MACRO FACILITY TO TABULATE COUNTS AND PERCENTS FOR INDIVIDUAL VARIABLE VALUES

Thomas J. Higgins, Blue Cross/Blue Shield of Mass.
Richard H. Potter, Mass Ratesetting Commission

When dealing with a file of records it is often useful to know the values that certain fields may take on. It may be that only certain values are permissible, or that we may want to use PROC FORMAT to define value labeling formats for all possible variable values. It may be useful to know the frequency for each variable value.

For example, suppose we have a file of records which consists of a certificate number, sex code, and location code for each record. The certificate number will be unique for each family, the sex code should have been entered as a 'F', 'M', or 'U'. The location code will consist of a single character: a digit 0-9. We would like to know what the frequency of variable values are for each certificate number.

If the following file is the input.

<table>
<thead>
<tr>
<th>CERTIFICATE NUMBER</th>
<th>SEX_CODE</th>
<th>LOCATION_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>123</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td>123</td>
<td>U</td>
<td>2</td>
</tr>
<tr>
<td>124</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>124</td>
<td>U</td>
<td>2</td>
</tr>
<tr>
<td>125</td>
<td>M</td>
<td>1</td>
</tr>
</tbody>
</table>

We would like the output to be:

<table>
<thead>
<tr>
<th>CERT NO</th>
<th>SEX_CF</th>
<th>SEX_CU</th>
<th>SEX_PF</th>
<th>SEX_PI</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>124</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>125</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

where SEX CF represents the frequency (count) of the SEX variable 'F', LOC PI represents the percent of the LOC values included in LOC variables with a value of "3".

How do we accomplish this in SAS? One possible solution involves using PROC SORT to create a BY variable (CERT NO in this case), PROC FREQ to determine the number of unique values for each field in question, and to tabulate the counts and percents for each of the unique variable values for each BY variable. This process requires the creation of variables to hold the counts and percents for each cell of the frequency table. We would repeat this process for each field of interest and then merge the results together creating one record for each value of the BY variable.

The following is an example of this process.

```
PROC SORT DATA=SASIN.; BY CERT_NO;
PROC FREQ;
BY CERT_NO;
TABLES CERT_NO*SEX / OUT=FOXSEX;
TABLES CERT_NO*LOC / OUT=FOLOC;
DATA FOXSEX;
SET FOXSEX;
BY CERT_NO;
IF FIRST.CERT_NO THEN DO;
  SEX_C1=0; SEX_C0=0; SEX_CU=0;
  SEX_F1=0; SEX_PF=0; SEX_PU=0;
END;
IF SEX ="M" THEN SEX_C1=COUNT;
IF SEX ="F" THEN SEX_C0=COUNT;
IF SEX ="U" THEN SEX_CU=COUNT;
IF SEX ="M" THEN SEX_PF=PERCENT;
IF SEX ="F" THEN SEX_PF=PERCENT;
IF SEX ="U" THEN SEX_PU=PERCENT;
RETAIN SEX_CH SEX_CF SEX_CU SEX_PF SEXP; SEX_PU;
DROP COUNT PERCENT SEX;
IF LAST.CERT_NO THEN OUTPUT;
DATA FOLOC;
SET FOLOC;
BY CERT_NO;
IF FIRST.CERT_NO THEN DO;
  LOC_C1=0; LOC_C2=0; LOC_C3=0;
  LOC_P1=0; LOC_P2=0; LOC_P3=0;
END;
IF LOC ="1" THEN LOC_C1=COUNT;
IF LOC ="2" THEN LOC_C2=COUNT;
IF LOC ="3" THEN LOC_C3=COUNT;
IF LOC ="1" THEN LOC_P1=PERCENT;
IF LOC ="2" THEN LOC_P2=PERCENT;
IF LOC ="3" THEN LOC_P3=PERCENT;
RETAIN LOC_P1 LOC_P2 LOC_P3;
DROP COUNT PERCENT LOC;
```

As we were working on this problem it became evident that we would have to recode for each variable we were interested in. This process involved using the initial PROC FREQ to determine the SAS variable values which are used in writing the DATA step that defines the variable names for the counts and percents. It seemed to be a candidate for using the SAS Macro Facility.

This paper will describe a possible solution which will use the SAS Macro Facility to create a SAS data set which will contain counts and percentages for each variable for each value of the BY variable.
SAS MACRO facility

SAS is a 'compile-and-go' language. The SAS supervisor uses the wordscanner to read each word, checking for syntax errors until it encounters a semicolon. It collects statements until it encounters a DATA, PROC, RUN, CARDS, or CARDS4 statement. These statements signal a step boundary. The collected statements are then compiled and executed. No other SAS statements are read until this process is complete.

A Macro is stored text combining SAS code and Macro code which can be referenced by name. The Macro is translated into SAS code by the Macro Processor. Macro code consists of normal SAS code, Macro Variable references, Macro function references, or Macro program statements.

The Macro Processor is between the wordscanner and the compiler. When the wordscanner encounters a '%' or a '&' character not followed by a space the Macro Processor is called.

The Macro Processor does whatever is required (execute a Macro, create a Macro Variable, provide a value for a Macro Variable). The result is sent back to the SAS supervisor to continue the process.

The special symbols used by the Macro Processor are:

1. % indicates a Macro definition or a Macro call.
2. & Used to specify a Macro Variable (symbolic variable).

The SAS Macro presented in this paper was required to do the following:

1. Read a file of names which are the fields on which we want to calculate the counts and percents.
2. Create the TABLES statements.
3. Create the SAS variable names for the various counts and percents.
4. Create the SAS code which will accomplish the task at hand.

We wanted to call the Macro and supply the name of the field which becomes the BY variable, the number of fields that we would like to perform the analysis on, and the internal dataset name of the SAS dataset to be used as input.

For example, the Macro call

IRUNIT (CERT_NO,3,SUGIDB2);

Will perform the analysis on the first three variables read from an external file after sorting the SAS dataset SASIN.SUGIDB2 by CERT_NO.

There are two SAS syntaxes in this Macro that took some time to figure out. After the Macro was written and after some reflection we recognized two corresponding processes. What follows is a brief description of these processes and the syntax that performs them. The first process is reading data from a file and making it accessible to the Macro Processor. This is done twice in our application. The first time a list of SAS variables names are read.

The second process is getting access to these Macro Variables. There may be five data elements stored in five Macro Variables or there may be 10. We found it convenient to form the Macro Variable names so they ended in sequential numbers. In this way we can access them by forming Macro Variables names such as _VNAM1 thru _VNAM10 getting just the right number.

1. SYMPUT

The first process reads data elements from a file and stores these in Macro Variables. In other words we construct Macro Variable names and assign them values. For this we need the SAS Data step function SYMPUT. To explain SYMPUT lets introduce a simple version of the statement:

```
SYMPUT _VNAM1 "field1"
SYMPUT _VNAM2 "field2"
SYMPUT _VNAM3 "field3"
```
CALL SYMPUT('VNAH1', INNAME);

This will take the value of INNAME, a SAS variable available in the DATA step, and create the Macro Variable _VNAH1 and assign it the value of INNAME. Suppose INNAME had the value 'SEX' then the character string 'SEX' would be assigned to the Macro Variable _VNAH1. A similar effect would be accomplished by using the statement:

%LET _VNAH1 = SEX;

In this process a data element is read from a file, a new Macro Variable is created and assigned a value. Somehow we must create variable names _VNAH1 ... _VNAH10, or whatever. The SYMPUT syntax has a more generalized version:

CALL SYMPUT (MACNAME, INNAME);

Where the value of MACNAME is used to create a Macro Variable and it is assigned the value associated with INNAME. So we want MACNAME to have the value _VNAH1 on the first reading and _VNAH2 on the second reading etc. These Macro Variables will be assigned the successive values that INNAME takes on. Thus:

DATA _NULL_;
SET INPUT DEFINE;
INPUT NAME #1;
LENGTH SUFFIX 20;
DATA SYMBOLIC MACRO SUBSTITUTION

The second process accessed data element stored in these Macro Variable names. However, because the number of Macro Variables will vary, the names of the Macro Variables must be constructed before they are accessed. For example, in the application we wish to generate frequencies for each of the variable names read in.

So, suppose two variable names 'SEX' and 'LOC' are stored in the Macro Variables _VNAH1 and _VNAH2 respectively. Then the code generated by the Macro should look like this:

PROC FREQ;
BY CERT NO;
TABLES CERT NO*SEX /OUT=FQSEX;
TABLES CERT NO*LOC /OUT=FQLOC;

If there are more than two variable names, then there should be more that two TABLES statements and we would need to access more than two Macro Variables. What is needed is not only a way of generating two or more TABLES statements but also gaining access to a different Macro Variable name each time a TABLES statement is generated. Consider the following rules governing the Macro Processor:

The Macro Processor resolves a Macro Variable reference such as _VNAH by substituting the value of the Macro Variable _VNAH for the reference. For example,

%LET _VNAH1 = SEX;
%LET NEWNAME = NO_&_VNAH1;

NEWNAME has the value NO_SEX.

The Macro Processor stops building the Macro Variable name when it encountered a period. In the following,

% LET VNAM2 - LOC;
% LET RESOLVED - &VNAM1;
% LET DOUBLE - &_VNAH2;

The value of RESOLVED becomes SEX1, the value of DOUBLE becomes SEX.1.

The Macro Processor resolves '&&' to '&&' and continues to resolve the statement until completion. In the following example,

%LET VNAM2 = LOC;
%LET X = 2;
%LET RESOLVED - &&_VNAH&X;

RESOLVED becomes & VNAM2 on the first pass through the Macro Processor, then RESOLVED becomes LOC.

Thus the TABLES statement above can be generated with a statement that looks like this:

TABLES CERT NO*&&_VNAH&X / OUT = FQ&&_VNAH&X;

Assume that the Macro Variable X takes on the values 1,2,..., and the Macro Variable _VNAH1 and _VNAH2 have values SEX and LOC respectively.

For X = 1 the above code resolves to

TABLES CERT NO*VNAH1 / OUT = FQVNAH1;

on the first pass and then to

TABLES CERT NO*LOC / OUT = FQLOC;

For X = 2 the above statement resolves to:

TABLES CERT NO*SEX / OUT = FQSEX;
We use the Marco DO statement (X DO) to set the values of X. With the %DO statement the code looks like this:

```
%DO X • 1 %TO &NOFSEL %BY 1;
  TABLES CERT NO*&& VNAM&X I
    OUT - FQ&&_VNAM&X;
%END;
```

Another example is the generation of the IF statements in the DATA step of our solution. The &&, &, and. are used in this example.

Assume that the Macro Variables K and Z have values 2 and 1 respectively. The Macro Variables VNAHl and NVAL2 have values SEX and F respectively. The following string,

```
IF && VNAH&Z .. "&& NVAL&K" THEN &&_VNAM&Z .. _P&&_NVAL&K' PERCENT;
```

becomes

```
IF &VNAHl .. "&NVAL2" THEN &VNAMI .. _P& NVAL2 .. PERCENT;
```

after the first pass, then becomes

```
IF SEX .. F THEN SEX .. PF .. PERCENT;
```

after the second pass.

We use this type of statement to assign the counts and percents to the appropriate SAS variable names.

The following is a sample listing of the Macro and the input and output datasets.

```
PROGRAM NAME SUGIMAC1

XMACRO VAL_FILEC_VNAH)J
X .. READ THE VARIABLE VALUES AND CREATE MACRO VARIABLES WITH THESE VALUES.
X CREATE A MACRO VARIABLE WITH THE NUMBER OF UNIQUE VARIABLE VALUES.
PROC FREQ DATA=FT&_VNAM; TABLES &VNAM/OUT=XX&_VNAM(KEEP=&_VNAM) NOPRINT;
DATA XXXXXX;
  SET XX&_VNAM END=EOF;
  VALUE=&_VNAM;
  IF VALUE = '.' THEN VALUE = '_.';
  IF VALUE = ' ' THEN VALUE = '_.';
  IF VALUE = ' ' THEN VALUE = '___';
  LENGTH NUM 63;
  NUM=TRIM(LEFT(NUM));
  VALUENUM = 'NVAL'|NUM;
  CALL SYMPUT(VALUENUM,TRIM(VALUE));
  IF EOF=1 THEN CALL SYMPUT('_NUMARG',NUM);
XEND VAL_FILE;

MACRO VARLIST(_FDNAHE,_NUM);
X.. INITIALIZE THE COUNTS AND PERCENTS TO ZERO.
X.. READ THE LIST OF FIELD NAMKES TO BE ANALYZED FROM AN EXTERNAL FILE.
X .. CREATE MACRO VARIABLES WITH THESE VALUES.
PROC FREQ DATA=FT&_VNAM; TABLES &VNAM(KEEP=&_VNAM) NOPRINT;
DATA XXXXXX;
  SET XX&_VNAM END=EOF;
  VALUE=&_VNAM;
  IF VALUE = '.' THEN VALUE = '_.';
  IF VALUE = ' ' THEN VALUE = '_.';
  IF VALUE = ' ' THEN VALUE = '___';
  LENGTH NUM 63;
  NUM=TRIM(LEFT(NUM));
  VALUENUM = 'NVAL'|NUM;
  CALL SYMPUT(VALUENUM,TRIM(VALUE));
  IF EOF=1 THEN CALL SYMPUT('_NUMARG',NUM);
MACRO VARLIST(_FDNAHE,_NUM);
X.. CREATE A LIST OF COUNT AND PERCENT VARIABLES.
X .. CREATE A LIST OF SAS DATASETS TO BE MERGED.
PROC SORT DATA=SASIN.'_INTDSN OUT=MACDSIN) BY &_SRTFD;
```

168
**References**


* SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA.
<table>
<thead>
<tr>
<th>OBS</th>
<th>CERT_HD</th>
<th>SEX_CP</th>
<th>SEX_CM</th>
<th>SEX_PF</th>
<th>SEX_PN</th>
<th>LOC_C1</th>
<th>LOC_C2</th>
<th>LOC_C3</th>
<th>LOC_P1</th>
<th>LOC_P2</th>
<th>LOC_P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>123</td>
<td>2</td>
<td>4</td>
<td>5.5555</td>
<td>55.5555</td>
<td>66.6667</td>
<td>77.7777</td>
<td>88.8888</td>
<td>99.9999</td>
<td>11.1111</td>
<td>22.2222</td>
</tr>
<tr>
<td>2</td>
<td>456</td>
<td>3</td>
<td>6</td>
<td>66.6667</td>
<td>77.7777</td>
<td>88.8888</td>
<td>99.9999</td>
<td>11.1111</td>
<td>22.2222</td>
<td>33.3333</td>
<td>44.4444</td>
</tr>
<tr>
<td>3</td>
<td>789</td>
<td>4</td>
<td>8</td>
<td>88.8888</td>
<td>99.9999</td>
<td>11.1111</td>
<td>22.2222</td>
<td>33.3333</td>
<td>44.4444</td>
<td>55.5555</td>
<td>66.6666</td>
</tr>
</tbody>
</table>