The SUMMSTAT Macro: Getting More Mileage Out of PROC SUMMARY

Michael D. Rhoads, Westat

A. ABSTRACT

SUMMSTAT is a SAS® macro that enables PROC SUMMARY to be utilized more effectively for producing summary statistics. SUMMSTAT allows the user to specify a list of variables to be analyzed, a list of the classification variable or variables to be used to divide the data into subgroups, and a list of the output statistics desired. The macro, after generating a PROC SUMMARY step to perform the actual data summarization, uses DATA steps and PROC TRANSPOSE to restructure the output data set from PROC SUMMARY. A PROC PRINT is then generated to print the desired summary statistics. The user may choose one of six possible formats for transposing and printing the summary data.

This paper describes how to call SUMMSTAT, discusses some of the coding techniques used within the macro, and illustrates the output which it produces. Since SUMMSTAT is a generalized macro, it may be used in a wide variety of applications. In addition, some of the coding techniques used could be adapted for use in other macros.

B. INTRODUCTION

PROC SUMMARY is an efficient and flexible tool for producing various types of summary statistics. In particular, SUMMARY is useful when statistics must be produced for various subgroups of a file. The CLASS statement enables such subgroup statistics to be produced without requiring the file to be sorted by the variable (or variables) defining the subgroups.

The primary disadvantage of PROC SUMMARY is that it does not produce any printed output. Rather, it produces an output data set containing summary statistics, which can then be printed. In simple cases—as, for instance, when only one summary statistic (such as the mean) is desired and only a few variables are being analyzed—this does not represent a major difficulty. A simple PROC PRINT can produce perfectly acceptable results. In other cases, however, this approach is much less satisfactory. For instance, if twelve variables are to be analyzed, with five separate types of statistics (for instance, N, NMISS, MEAN, MIN, and MAX) produced for each, the output data set from PROC SUMMARY will contain more than 60 variables. This requires the user, when coding the OUTPUT statement, to dream up names for each of these variables. Moreover, the output from a PROC PRINT of such a data set will not be very easy to follow.

The key to solving this dilemma is the recognition that, in many instances, it is highly desirable to restructure the output data set produced by PROC SUMMARY. For instance, one possible arrangement might have each variable on the restructured data set contain data for one classification subgroup. Each record of this data set would contain one particular statistic for one analysis variable; the records might be grouped either by analysis variable or by statistic. Once this restructuring is accomplished, a PROC PRINT of the reorganized data set will produce concise and meaningful output. The SUMMSTAT macro allows this restructuring to be accomplished automatically, without requiring the user to worry about generating temporary variable names or coding DATA steps.

In many situations, PROC TABULATE can serve as a useful alternative to PROC SUMMARY. Although SUMMSTAT involves more overhead than TABULATE, it is more efficient on large data sets. In addition, it produces an output data set, which allows the production of completely customized reports.

C. CALLING SUMMSTAT

Eleven keyword parameters are available for the invocation of the SUMMSTAT macro. In most instances, not all of these parameters will need to be supplied, since many of them have default values.

DATA= and OUT= identify the input and output files to be used by SUMMSTAT. If DATA= is not specified, the most recently created data set is used for input. If OUT= is not specified, the working data sets created by SUMMSTAT will be named according to the standard SAS DATAn convention. If the final output data set produced by SUMMSTAT is to be retained for future use, a two-level data set name should be specified for OUT=.

Three parameters specify the analyses to be performed by SUMMSTAT: VAR=, which must be specified, identifies the variable or variables for which summary statistics are to be produced; STAT=, which defaults to MEAN if not specified, lists the statistic(s) to be produced; CLASS= lists the variable or variables, if any, that define subgroups of the data.

The parameters COLUMNS= and ROWSORT= specify the desired format for the restructured data set and printed output to be produced by SUMMSTAT. The permissible values for these parameters are VAR, STAT, and CLASS, although the same value may not be used for both parameters. (A total of six combinations are possible.) For instance, combining COLUMNS=VAR with ROWSORT=STAT produces the type of output shown in Figure 1. The columns of the printed report (and the variables in the restructured output data set) represent the variables that are being analyzed (VAR), while the primary sort key for the report and data set is the particular statistic being produced (STAT=).

Figure 2 illustrates the output produced when the columns of the report are values of the classification variable (COLUMNS=CLASS) and the rows of the report are sorted by analysis variable (ROWSORT=VAR).

Four other keyword parameters are also available. FREQ= allows the user to specify a variable to be used in the FREQ statement of PROC SUMMARY. SUMMOPT= allows the user to specify options (such as NWAY or MISSIN) to appear on the PROC SUMMARY statement. Similarly, PRINTOPT= allows such PROC PRINT options as DOUBLE and UNIFORM to be specified if desired. Finally, the user may suppress all printed output from SUMMSTAT by specifying PRINT=NO. (The restructured data set containing summary statistics would still be available for additional manipulation.)
SUMMSTAT is divided into four main sections (see Figure 3 for source code). Lines 14-73 contain preliminaries such as creating working macro variables and checking for user errors. Lines 75-89 generate code to invoke PROC SUMMARY, while lines 91-155 reformat the data set created by PROC SUMMARY. Finally, lines 157-191 generate a PROC PRINT.

The working variables used by SUMMSTAT are listed in line 15. NUMSTAT, NUMCLASS, and NUMVAR are used to hold counts of the number of statistics, class variables, and analysis variables specified by the user. ERROR is an error flag, while TEMP and X are scratch variables. The only other variables used (besides the eleven parameter variables) are a set of STATn variables, which are generated in lines 34-42. One of these is created for each user-specified statistic (i.e., the nth STATn variable contains the name of the nth user-specified statistic).

E. COUNTING VARIABLES

Two of the specific techniques used in SUMMSTAT are generalized enough to be useful in other macros as well. The first of these (lines 44-51) is a method for counting the number of variables in any valid SAS variable list. In particular, lists such as VAR1-VAR3, A-Z, and NUMERIC are handled properly.

This technique works by using the SAS DATA step function NMISS, which returns the number of missing values in its argument list. The SET statement in line 46, although it is never executed, identifies the input data file and its variables to the DATA step compiler. Lines 48-49 illustrate the invocation of NMISS, using the "OF" argument list convention so that lists of variables may be passed. Since no cases are read from the input data set, all of the variables in the &CLASS and &VAR lists will have missing values. (The dummy variables _1 and _2, created in line 47, are also included in the argument lists to make sure that at least two arguments are passed, since otherwise a syntax error is produced.) The micro language routine SYMPUT is used to place the values returned by NMISS into the macro variables NUMCLASS and NUMVAR. The LEFT and PUT functions in the arguments to SYMPUT are used to ensure that the resulting macro variables do not contain leading blanks.

F. VALIDATING USER PARAMETERS

The VALIDATE macro used within SUMMSTAT could also be useful in other contexts. As is illustrated in lines 63-65, VALIDATE checks whether or not the macro variable whose name is passed as its first argument consists of one of the keywords passed as its second argument. The name of the macro variable, rather than its value, is passed so that both the name of the variable and its value can be included in the error message generated by VALIDATE. For instance:

ERROR: "MAYBE" IS INVALID VALUE FOR "PRINT".

VALID VALUES ARE "YES", "NO".

The source code of VALIDATE is contained in lines 23-31. Since the name of the macro variable to be checked, rather than its value, is passed, three ampersands are necessary to retrieve the value of the macro variable. For instance, assume that PRINT is passed as the second argument to VALIDATE. Assume also that PRINT is a macro variable with a value of MAYBE. The reference &VARNAME simply produces PRINT after scanning, while the reference &VARNAME produces &PRINT as the result of the first scan and then MAYBE as a result of re scanning. It is important to note that, in order for this technique to work, VALIDATE must be within the "scope of reference" of the macro variable PRINT. This is accomplished here by nesting the definition of VALIDATE within SUMMSTAT.

VALIDATE uses the %INDEX macro function to determine whether or not the value of the macro variable being checked is contained within the list of keywords. Caution is required to guard against partial matches: for instance, if YES and NO are the valid responses, we do not want ES to be accepted, even though it is contained within the string "YES NO". This is accomplished by using the %QUOTE quoting function to quote both the keyword list and the macro variable value with leading and trailing blanks for the call to %INDEX (line 24). A second call to %INDEX is used to make sure that the macro variable does not contain embedded blanks.

G. OTHER POINTS OF INTEREST

The actual PROC SUMMARY code is generated in lines 75-89. Note that the CLASS and FREQ statements are only generated if the corresponding parameters are nonblank. In the OUTPUT statement (lines 84-88), the names of the original analysis variables are used for the first user-specified statistic, while variable names are generated for the second and subsequent statistics. The reference &STATn&X provides the value of one of the STATn variables, depending on X. For example, assume that the user has specified STAT=N MIN MAX. When X=3, the first scan of &STATn&X produces &STAT3, which is a previously-generated macro variable (lines 34-42) containing the value MAX. This referencing technique allows arrays to be simulated in the macro language.

The following sections of SUMMSTAT use DATA steps, PROC SORT, and PROC TRANSPOSE to restructure the output data set from PROC SUMMARY. Note the use of the variable _.,,ORD in lines 91-105. This variable is used as a sort key so that statistics will be maintained (and printed) in the order specified by the user, rather than in alphabetical order (as would be the case if sorting were by STAT). Similarly, a variable _.,,ORD is created (lines 121-131) to maintain the analysis variables in the order specified by the user.

The final section of SUMMSTAT generates PROC PRINT code to print the restructured summary-statistics data set. Separate BY, ID, and VAR statements are generated for the six possible combinations of COLUMNS and ROWSORT. (If COLUMNS=CLASS, as in lines 181-190, it is easier to use a DROP statement than to figure out the names of the column variables for the VAR statement.)

ACKNOWLEDGEMENTS

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA.

CONTACT AUTHOR

Michael D. Rhoads
Westat
1650 Research Blvd.
Rockville, MD 20850
Figure 1. SUMMSTAT output with COLUMNS=VAR and ROWSORT=STAT

<table>
<thead>
<tr>
<th>RACE</th>
<th>APPINC6M</th>
<th>APPINCAN</th>
<th>FAMINC6M</th>
<th>FAMINCAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHITE</td>
<td>1397</td>
<td>2793</td>
<td>2725</td>
<td>5450</td>
</tr>
<tr>
<td>BLACK</td>
<td>566</td>
<td>1131</td>
<td>1782</td>
<td>3565</td>
</tr>
<tr>
<td>ASIAN PAC ISLAND</td>
<td>2072</td>
<td>4144</td>
<td>4002</td>
<td>8004</td>
</tr>
<tr>
<td>AMER IND ALASKAN</td>
<td>1105</td>
<td>2209</td>
<td>2040</td>
<td>4080</td>
</tr>
<tr>
<td>HISPANIC</td>
<td>870</td>
<td>1739</td>
<td>2631</td>
<td>5392</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RACE</th>
<th>APPINC6M</th>
<th>APPINCAN</th>
<th>FAMINC6M</th>
<th>FAMINCAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHITE</td>
<td>1397</td>
<td>2793</td>
<td>2725</td>
<td>5450</td>
</tr>
<tr>
<td>BLACK</td>
<td>566</td>
<td>1131</td>
<td>1782</td>
<td>3565</td>
</tr>
<tr>
<td>ASIAN PAC ISLAND</td>
<td>2072</td>
<td>4144</td>
<td>4002</td>
<td>8004</td>
</tr>
<tr>
<td>AMER IND ALASKAN</td>
<td>1105</td>
<td>2209</td>
<td>2040</td>
<td>4080</td>
</tr>
<tr>
<td>HISPANIC</td>
<td>870</td>
<td>1739</td>
<td>2631</td>
<td>5392</td>
</tr>
</tbody>
</table>

Figure 2. SUMMSTAT output with COLUMNS=CLASS and ROWSORT=VAR

<table>
<thead>
<tr>
<th>VARIABLE=APPINC6M LABEL=APPLICANTS INCOME 6 MONTHS PRIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT</td>
</tr>
<tr>
<td>MEAN</td>
</tr>
<tr>
<td>MIN</td>
</tr>
<tr>
<td>MAX</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VARIABLE=APPINCAN LABEL=APPLICANTS ANNUALIZED INCOME</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT</td>
</tr>
<tr>
<td>MEAN</td>
</tr>
<tr>
<td>MIN</td>
</tr>
<tr>
<td>MAX</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VARIABLE=FAMINC6M LABEL=TOTAL FAMILY INCOME 6 MONTHS PRIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT</td>
</tr>
<tr>
<td>MEAN</td>
</tr>
<tr>
<td>MIN</td>
</tr>
<tr>
<td>MAX</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VARIABLE=FAMINCAN LABEL=ANNUALIZED FAMILY INCOME</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT</td>
</tr>
<tr>
<td>MEAN</td>
</tr>
<tr>
<td>MIN</td>
</tr>
<tr>
<td>MAX</td>
</tr>
</tbody>
</table>
%MACRO SUMMSTAT(
    DATA=_LAST_,
    OUT=_DATA_,
    CLASS:, VAR

    STAT=MEAN.
    FREQ=,
    SMIMOPT=,
    PRINTOPT=

    PRINT=YES.
    COlUMNS=STAT.
    ROWSORT=CLASS

);  

%* LOCAL VARIABLES AND NESTED MACROS;  

%LOCAL NUMSTAT NUMCLASS NUMVAR ERROR TEMP X;  

%LET ERROR = 0;  

%MACRO ERROR (MESSAGE);
%LET ERROR = 1;
%PUT ERROR: &MESSAGE •• ;
%MEND ERROR;

%MACRO VALIDATE (VALLIST, VARNAME);
%IF %INDEX(%QUOTE( &VAlLIST },%QUOTE( &&&VARNAME }:
%THEN %00;
%LET ERROR = 1;
%PUT ERROR: "&&&VARNAME" IS INVALID VALUE FOR &VARNAME.;
%PUT "STR( VALID VALUES ARE: &VAllIST •• );
%MEND VALIDATE;

%* BREAK UP STAT: LIST (DESIRED STATISTICS);
%LET NUMSTAT = 0;
%LET TEMP = %SCAN(%QUOTE(&STAT), 1);
%00 %WHIlE (%QUOTE(&TEMP) NE );
%LET NUMSTAT = &NUMSTAT + 1;
%LET STAT&NUMSTAT = &TEMP;
%LET TEMP = %SCAN(%QUOTE(&STAT), &NUMSTAT+1);
%END;

%* GET NUMBER OF "CLASS" AND "VAR" VARIABLES;
%DATA _NULL_;  
%IF _N. = 0 THEN SET &DATA;
%LET _N. = 1;
%LET _2. = 0;
%CALL SYMPUT ("NUMCLASS", LEFT(PUT(MISS(OF _2. &CLASS),3,1)));  
%CALL SYMPUT ("NUMVAR", LEFT(PUT(MISS(OF _2. &VAR),3,1)));  
%STOP;
%RUN;

%* CHECK FOR USER ERRORS;
%IF &NUMSTAT = 0 THEN %ERROR(%STR(NULl "STAT=" LIST SPECIFIED));
%IF &NUMSTAT > 99 %ERROR(%STR(NO MORE THAN 99 STATISTICS MAY BE SPECIFIED));
%LET TEMP = 0;
%WHIlE (%QUOTE(&TEMP) NE );
%LET _2. = 1;
%LET _2. = 0;
%CALL SYMPUT ("STAT", 1-);  
%LET STAT&X = &X. 1-
%LET _2. = &NUMVAR;
%ARRAY STAll &VAR;  
%DO OVER=-STAll;  
%STAll~ _STAT&X;  
%END;
%LET _X_ = _STAT&X;  
%S ORO = _STAT&X;  
%OUTPUT;
%END;
%RUN;  

%* REFORMAT OUTPUT FROM PROC SUMMARY;
%DATA &OUT;
%SET _LAST;
%KEEP &CLASS &VAR TYPE FREQ S ORO;  
%LENGTH STAT $ 6
%S ORO-2;
%LABEL STAT= , STATISTIC' S ORD='ORDER OF REQUESTED STATISTICS';
%ARRAY STAll &VAR;  
%DO OVER=-STAll;
%STAll~ _STAT&X;  
%END;
%LET _X_ = _STAT&X;  
%S ORO = _STAT&X;  
%OUTPUT;
%END;  

%* RUN PROC SUMMARY;
%PROC SUMMARY DATA:&DATA &SUMMOPT;
%LET TEMP = %SCAN(%QUOTE(&CLASS), 1);
%WHIlE (%QUOTE(&TEMP) NE );
%LET _2. = &CLASS;
%LET _2. = 0;
%WHIlE (%QUOTE(&TEMP) NE );
%LET _2. = &FREQ;
%WHIlE (%QUOTE(&TEMP) NE );
%LET _2. = &SUMMOPT;
%ADDSTAT&X. _X. _X. _X. &&VARNAME;
%END;
%RUN;  

%* REFORMAT OUTPUT FROM PROC SUMMARY;
%DATA &OUT;
%SET _LAST;
%KEEP &CLASS &VAR TYPE FREQ _ORD;
%LENGTH STAT $ 5
_S ORD 2;
%LABEL STAT=,STATISTIC' S_ORD="ORDER OF REQUESTED STATISTICS";
%ARRAY _STAT&X. _X. _X. _X. &&VARNAME;
%DO OVER=_STAT&X. _X. _X. _X. _ORD=
%LET _2. = &VARNAME;
%LET _2. = &VARNAME;
%LET _2. = &VARNAME;
%LET _2. = ?;
%RUN;
`%* NOW TRANSPOSE AND/OR SORT IF NECESSARY, DEPENDING ON COLUMNS/ROWSORT;
%IF %COLUMNS = VAR & %ROWSORT = CLASS %THEN %00:
  PROC TRANSPOSE DATA=LAST OUT=&OUT;
  BY _NAME_=VARIABLE _LABEL_=LABEL;
  VAR &VAR;
  RUN;
%END;
%ELSE %IF %ROWSORT = CLASS & %COLUMNS = VAR %THEN %00:
  PROC SORT; BY VAR LABEL NOTSORTED;
  ID STAT;
  DROP _S_ORD _ORD;
%END;
%END;

%IF %ROWSORT = VAR & %COLUMNS = CLASS %THEN %00:
  PROC SORT; BY _ORD TYPE &CLASS;
%END;

%* NOW PRINT, IF DESIRED;
%IF %PRINT = YES %THEN %00:
%IF %INOEX(%SUMMOPT,TRUE) %THEN %00:
  PROC PRINT DATA=LAST &PRINTOPT;
%END:
%END;

%END;
ENDMAC: %MEND SUMMSTAT;`
Figure 4. Example of SAS log output from SUMMSTAT

208 %SUMMSTAT (DATA=TEMP, VAR=APPINC6M APPINCAN FAMINC6M FAMINCAN, CLASS=RACE, STAT=MEAN MIN MAX, COLUMN=VAR, ROWSORT=STAT, SUMMOPT=NWAY MISSING)
209 * DATA _NULL_
210 * IF NMISS RACE = 0 THEN SET TEMP;
211 + _1 = 0;
212 + _2 = 0;
213 + CALL SYMPUT ("NUMCLASS", LEFT (PUT (NMISS (OF RACE), 3), 3));
214 + CALL SYMPUT ("NMISS", LEFT (PUT (NMISS (OF APPINC6M APPINCAN FAMINC6M FAMINCAN), 3), 3));
215 + STOP;
216 + RUN;

NOTE: THE DATA STATEMENT USED 0.61 SECONDS AND 344K.

219 + PROC SUMMARY DATA=TEMP NWAY MISSING;
220 + CLASS RACE;
221 + VAR APPINC6M APPINCAN FAMINC6M FAMINCAN;
222 + OUTPXL OUT=DATA MEAN MIN 1 2 4 2 4 MAX 3 1 3 4;
223 + RUN;

NOTE: DATA SET WORK.DATAl HAS 5 OBSERVATIONS AND 15 VARIABLES. 396 OBS/STR.
NOTE: THE PROCEDURE SUMMARY USED 2.86 SECONDS AND 312K.

224 + DATA _DATA_
225 + SET LAST.
226 + KEEP RACE APPINC6M APPINCAN FAMINC6M FAMINCAN TYPE REF STAT _S_ORD;
227 + LENGTH STAT 6 _S_ORD 2;
228 + LABEL STAT=I STATISTIC _S_ORD=ORDER OF REQUESTED STATISTICS;
229 + ARRAY STAT1 APPINC6M APPINCAN FAMINC6M FAMINCAN;
230 + STAT = "MEAN";
231 + _S_ORD = 1;
232 + OUTPUT;
233 + ARRAY STAT2 2 1 2 4;
234 + DO OVER STAT;
235 + STAT = STAT;
236 + ENG;
237 + STAT = "MIN";
238 + _S_ORD = 2;
239 + OUTPUT;
240 + ARRAY STAT3 3 1 3 4;
241 + DO OVER STAT;
242 + STAT = STAT;
243 + END;
244 + STAT = "MAX";
245 + _S_ORD = 3;
246 + OUTPUT;
247 + RUN;

NOTE: DATA SET WORK.DATAl HAS 15 OBSERVATIONS AND 9 VARIABLES. 756 OBS/STR.
NOTE: THE PROCEDURE SUMMARY USED 0.21 SECONDS AND 296K.

248 + PROC SORT;
249 + BY _S_ORD TYPE RACE;
250 + RUN;

NOTE: DATA SET WORK.DATAl HAS 15 OBSERVATIONS AND 9 VARIABLES. 756 OBS/STR.
NOTE: THE PROCEDURE SORT USED 0.32 SECONDS AND 568K.

251 + PROC PRINT DATA=LAST;
252 + BY RACE;
253 + VAR APPINC6M APPINCAN FAMINC6M FAMINCAN;
254 + RUN;

NOTE: THE PROCEDURE PRINT USED 0.17 SECONDS AND 264K AND PRINTED PAGE 1.