Paper CC10

Presenting Descriptive Statistics by the Rapid Processing of Data Sets from Proc Means

Rod Norman, i3statprobe, San Diego, CA

ABSTRACT

Calculation and presentation of descriptive statistics for clinical data are frequently under the domain of complex macros. Less commonly used features of SAS Procs, however, can rapidly output data sets containing statistics for multiple variables and data subgroups. Post Proc processing of these data sets with simple but little utilized SAS features generate SAS data sets for outputs with very transparent and adaptable code. In this paper, I use the MEANS procedure with keywords, formats, and multiple output statements to generate individual data sets for each statistic. Processing of these data sets is aided by use of the automatic _TYPE_ variable. A column is created to capture overall subgroup totals without need to double SET the input data. An uncommonly used method of combining data sets employing the SET statement with a BY statement is used to reconstitute the statistics into one data set. Finally, I show an example use of the PUTN and INPUTN functions to capture the specified decimal precision for each statistic.

INTRODUCTION

The presentation of descriptive statistics for clinical data is a common task and frequently is performed by complex macros. Certain features of Means procedure, however, permit rapid output of summary statistics from code that is highly transparent and easily modified. Theses features include statistics for multiple variables and subgroups, output of zeros for counts of groupings that are not found within the data but expected, and summaries for population totals without resorting to double setting of the input data.

GENERATION OF STATISTICS WITH PROC MEANS EMPLOYING MULTIPLE OUTPUT STATEMENTS

OUTPUT OF STATISTICS FOR SPECIFIED GROUPINGS

For this simple example consider an input data set of numeric laboratory values. These data are for several different treatment-dose level groups or cohorts and are recorded weekly during the study. The request is for descriptive statistics at each study week for each cohort and for a total summary of all cohorts.

The code below uses PROC MEANS and multiple OUTPUT statements to create output data sets for each requested statistic. In this case, the statistics are the sample size, range, means, median, standard deviation and quartiles, respectively. A FORMAT statement has been applied to the time points (labvisitf) and to the cohorts (doselvlf). The keywords *completetypes* and *preloadfmt* are employed to allow reporting of 0 observations for variable combinations that may be missing from the data at present. CLASS statements name the variables for analysis and a TYPES statement restricts the output to just the data groupings desired.

```
proc format ;
     value  labvisitf  1='Week 1' 4='Week 4' 12='Week 12'  999='Change' ;
     value  doselvlf 1=100 2=200 3=400 4=800 ;

proc means data=lb2  noprint completetypes ;
     format week labvisitf.  doselvlCode doselvlf. ;
     class labcode ;
     class week doselvlCode / preloadfmt ;
     types labcode*week  labcode*week*doselvlCode ;
     var labvalue  ;
     output  out=mnN   N=var ;
     output  out=mnrng  min=var1 max=var2 ;
     output  out=mnmean  mean=var ;
     output  out=mnmed  median=var;
     output  out=mnstd  std=var varcbl;
     output  out=mnq3  q1=var1 q3=var2 ;
     run;
```

The statistic containing data sets are then combined using an interlaying SET statement with the BY statement using variables from the CLASS statement listed in the PROC MEANS. A variable *stat* is created to name and order the statistics acquired from the input data sets. In this example, selecting *_type_=7* gives results for each cohort by laboratory test by visit week.

```
data statGroups;
set mnN    (in=N where=(_type_=7))
    mnmean (in=M where=(_type_=7))
    mnstd  (in=S where=(_type_=7))
    mnmed  (in=D where=(_type_=7))
    mnq3   (in=Q where=(_type_=7))
    mnrng  (in=R where=(_type_=7)) ;
    by labcode week doselvlCode;
    if N then stat='1_N       ';
    if M then stat='2_Mean    ';
    if S then stat='3_SD      ';
    if D then stat='4_Median  ';
    if Q then stat='5_Q1, Q3  ';
    if R then stat='6_Min, Max';
    drop _freq_;
    run;
```

OUTPUT OF STATISTICS FOR TOTALS

To get statistics for a Totals group of all cohorts, the following virtually identical code is employed to create a Totals data set by restricting the data to *_type_*=6.

```
data statTotals;
set   mnN    (in=N where=(_type_=6))
      mnmean (in=M where=(_type_=6))
      mnstd  (in=S where=(_type_=6))
      mnmed  (in=D where=(_type_=6))
      mnq3   (in=Q where=(_type_=6))
      mnrng  (in=R where=(_type_=6));
       by labcode week doselvlCode;
       if N then stat='1_N       ';
       if M then stat='2_Mean    ';
       if S then stat='3_SD      ';
       if D then stat='4_Median  ';
       if Q then stat='5_Q1, Q3  ';
       if R then stat='6_Min, Max';
       drop _freq_;
       doselvlCode=6;            *** give Totals this dose level code *****;
       run;
```

The two data sets are then combined, again using an interlaying BY statement, and a format value for the Totals cohort is added.

```
    proc format;
         value doselvl2f  Other=[doselvlf.] 6='Total' ;
    data alldis;
    set statGroups statTotals;
         by labcode week doselvlCode;
         format doselvlCode doselvl2f. ;
        run;
```

REINSERTION OF IDENTIFICATION VARIABLES INTO OUTPUT DATA SETS

There are supplementary variables in the input data set that are desired in the outputs but are difficult to preserve in the PROC MEANS outputs. With these laboratory data, it is desirable to maintain the units and laboratory test names. The following code is used to reinsert these variables into the output data sets just created.

```
proc sql ;
     create table alldis2 as
     select A.*,upcase(L.lbstunit) as lbstunit, L.lab
     from alldis as A  left join
          (select distinct labcode,lab,lbstunit from lb2)  as L
     on A.labcode=L.labcode
     order by labcode,lab,week,stat,doselvlCode
;quit;
```

CREATING VARIABLE FOR OUTPUT IN PROPER DATA PRECISION

The decimal precision of the laboratory data is determined by how accurately the data are reported and it will need to vary from test to test. The precision for each test is commonly specified in either a SAS data set or provided by a SAS macro. In the case presented here, a precision determining variable (LabFmt) is acquired from a previously created SAS data set and returned by invoking a SAS macro (code not shown). Once supplied with a format precision, the following code is used to create a character variable (*display*) that is used to write the statistics in the decimal precision stated in guidelines.

```
data alldis3;
    length display $16 lbstunit $18;
set alldis2;
    if stat='1_N' then
    display=trim(left(put(var,best.)));

    *** use one more decimal place than specified for raw data  **************;
    else if substr(stat,1,1) in ('2','3','4') then
    display=trim(left(putN(var, put(inputN(LabFmt,LabFmt)+.1,best.))));
    else if substr(stat,1,1)='5' then
    display=trim(left(putN(var1,put(inputN(LabFmt,LabFmt)+.1,best.)))||',
        '||trim(left(putN(var2,put(inputN(LabFmt,LabFmt)+.1,best.))));

     *** use same decimal place as specified for raw data  *******************;
     else if substr(stat,1,1)='6' then
        display=trim(left(putN(var1,LabFmt)))||','||
        trim(left(putN(var2,LabFmt)));

     ***  if missing  ********************************************************;
     if display in (".","., .") then display='-';   run;
```

Of special note, this code contains the rarely used SAS functions INPUTN and PUTN. These functions allow variable values to be used as formats during execution without being specified in a previous FORMAT procedure. INPUTN allows the input character informat to be converted to a numeric value so that an arithmetic expression can be applied, namely, the addition of decimal place for formatting the statistics such as the mean. A conventional PUT statement converts this value back to a character value. Finally, a PUTN statement is applied to produce the character variable *display* used for the output.

OUTPUT OF STATISTCS

The specifications for the output call for the dosing level cohorts to be represented in vertical columns and the weekly visits to be listed sequentially with all of the statistics reported. To accomplish this layout, the resulting data set is transposed for the variable *display* using the cohorts with the ID statement to list them horizontally across the page.

```
proc transpose  data=alldis4 out=alldisT;
     by labcode lab week stat lbstunit ;
     id doselvlCode ;
     var display ;  run;
```

Once transposed the data are properly arranged and ready for export looking something like the following abbreviated table.

Absolute Neutrophil Count (10^9/L)

	100 µg	200 µg	…	400 µg	Total
Week 1					
N	6	11	…	10	43
Mean	3.957	2.540	…	3.928	2.964
SD	1.903	1.779	…	2.369	2.145
Median	4.208	1.911	…	3.004	2.108
Min, Max	1.13, 6.06	0.23, 6.16	…	1.52, 7.68	0.08, 8.05
Week 4					
N	6	0	…	11	29
Mean	5.534	-	…	8.468	5.602
…….	…..	……		……	…….
……	…..	…….	…..	…..	……

CONCLUSION

This brief paper exemplifies the use of multiple OUTPUT statements with PROC MEANS to produce descriptive statistics for quantitative data. The approach can be expanded to handle any number of analysis variables (laboratory tests in this case) and sub groupings using code that is free of macro variables, very transparent and easy to modify. To combine the output data sets for data presentation, this paper also demonstrates the interlaying SET with BY statements and uses of the functions INPUTN and PUTN.

.

ACKNOWLEDGMENTS

The author also thanks Paul LaBrec for his help and encouragement.

CONTACT INFORMATION

Rod Norman
i3statprobe
10052 Mesa Ridge Court
San Diego, CA 92121
E-mail: rodney.norman@i3statprobe.com