Paper 87

Two Macros to Compute the Median for A List of Variables

Fan Xu, SpectRx Inc., Norcross, GA

Abstract

SAS® does not have a built-in function to compute the median of a list of variables as it does for the mean. The median has some unique statistical properties such as insensitivity to outliers. It is also more preferable as a measure of the center location for a skewed distribution. Two macros are presented in this paper to compute the median of a list of variables. Examples are given to illustrate the applications. The readers are assumed to have a good understanding of SAS BASE procedures as well as advanced knowledge of Macro processing.

The first macro makes use of the TRANSPOSE procedure and the median statistics in the UNIVARIATE procedure [1]. The second macro is designed to compute the median at the level of each observation within a data step. Array is used for sorting.

Key Words

Median Statistics, TRANSPOSE Procedure, UNIVARIATE Procedure, Array, Macro.

Introduction

SAS has built-in statistical functions to calculate the mean, standard deviation, and coefficient of variation. There is no standard function in SAS for the median. When the distribution is highly skewed or there are outstanding outliers, the median is always preferred to the mean as a measure of the average or the center of a distribution. The UNIVARIATE procedure does calculate the median statistics, but it is for one variable instead of a list of variables.

Two macros are presented to compute the median of a list of variables. The first one utilizes the TRANSPOSE procedure. One observations in a list of variables is transposed to form one variable. The UNIVARIATE procedure is then applied to compute the median. The second macro is meant for use inside a data step for each observation. It copies the list of variables to a temporary array. The temporary array is then sorted to find the median.

The Programs

The first macro starts with the following macro call, where ‘datain’ is the name of the data set, ‘varlist’ is a list of variable names separated by a space and ‘medname’ is the name given to the resulting median. The default name for the resulting median is ‘median’.

```
%macro median(datain =, varlist =,
        medname = median);
```

When a data set is transposed, the rows become the columns, and vice versa. So one observation in a list of variables becomes one variable. The number of observations in the original data set is the number of variables in the transposed data. The following is a classic macro to find the number of observations in a data set [2]. Note since the IF condition is always false, the SET statement does not execute. But the NOBS option is executed when the program is complied. The SYMPUT function is used to create a macro variable of the same name containing the number of observations in the data set.

```
data _null_;
    if 0 then set &datain nobs = nobs;
    call symput('nobs', nobs);
run;
```

The data set is transposed by the TRANSPOSE procedure. By default the first observation in the list of variables is named ‘col1’, the second ‘col2’ and so on. The UNIVARIATE procedure is then applied to find the medians of ‘col1’ to ‘col&nobs’. The medians are output to a data set called ‘mout’ with the same names.

```
proc transpose data = &datain out = tout1;
    var &varlist;
run;

proc univariate data = tout1 noprint;
    var col1-col&nobs; output out = mout
        median = col1-col&nobs;
run;
```

Remember each median corresponds to one observation. The next step is to transpose the output file back. The medians are merged with the original data and aligned correctly with the original observations. Again the single row of medians in ‘mout’ is transposed to a single column called ‘col1’ by default. The transposed data ‘tout2’ is merged with the original data observation by observation. Variable ‘col1’ is renamed to the user-selected name (or ‘median’ by default).

```
proc transpose data = mout out = tout2;

data &datain;
    merge &datain tout2(keep = col1
        rename = (col1 = &medname));
    label &medname = "Median of
        &varlist";
run;

%mend;
```

The second macro entails more programming. This macro is invoked inside a data step. It starts with the following macro call with the list of variable names (‘varlist’) and the name given to the resulting median (‘medname’). The default name is ‘median’.

```
%macro median(varlist = , medname =
        median);
```

The first step is to find out how many variables there are in the list. It is accomplished by the following macro segment. ‘n’ is a macro variable initialized to zero. It is incremented by one when the %SCAN function scans another non-null string. The %SCAN function scans for the next (&n + 1) string in the list of variables delimited by space. If the string is null (%str()), the UNTIL condition becomes false. At the end of the loop &n is the number of variables in the list.

```
%let n = 0;
%do %until (%scan(&varlist, %eval(&n +
        1), %str( )) = %str() );
    %let n = %eval(&n + 1);
%end;
```

When sorting the list of variables, we do not want to change the actual values of the variables. So a copy of the variables is made by a temporary array called ‘temp1’ to ‘temp&n’.

```
array y(*) &varlist;
array t(*) temp1-temp&n;
do i = 1 to &n; t(i) = y(i); end;
```

Sorting is done on the temporary array. Note that the array is sorted in the descending order. The variable ‘temp’ serves as a swapping place when two values are switched.

```
do i = 1 to &n-1;
    do j = i + 1 to &n;
        if t(i) < t(j) then do;
            temp = t(i);
            t(i) = t(j);
            t(j) = temp;
        end;
    end;
```

```
end;
```

The last step is to compute the median. It is important to remember that there could be missing values. Missing values are smaller than any numerical values in SAS by definition. When the array is sorted in the descending order, the missing values (if any) are put in the right end. Only the first ‘m’ variables with non-missing values are used to compute the median. The median is defined as the middle observation of a sorted sequence if the number of observations is odd. If the number of observations is even, the median is defined as the mean of the two middle observations. All temporary variables are dropped in the end of the session.

```
m = &n - nmiss(of temp1-temp&n);
if mod(m, 2) = 0 then &medname =
        (t(m/2) + t(m/2 + 1))/2;
else &medname = t((m+1)/2);
drop i j m temp1-temp&n temp;
label &medname = "Median of &varlist";

%mend;
```

The examples

To illustrate how the first macro works, suppose we have the following hypothetical data set ‘example1’:

```
OBS   A     B     C     D

1     10     6    45     7
2     11    20     .    34
3      .     3    65    21
```

The first macro is used to compute the median of variables A, B, C and D by invoking %median(datain = example1, varlist = a b c d, medname = median); The transposed data of the original data follows. Note each observation becomes a variable.

```
OBS   _NAME_    COL1    COL2    COL3

1       A        10      11       .
2       B         6      20       3
3       C        45       .      65
4       D         7      34      21
```

The output file from the UNIVARIATE procedure is a row of medians of each variable in the transposed data set:

```
OBS   COL1    COL2    COL3

1      8.5     20      21
```

This data set is transposed and merged back with the original data to create the final data set:

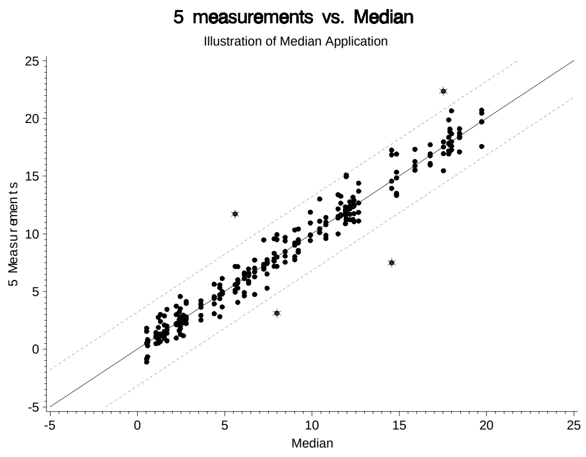
```
OBS    A     B     C     D    MEDIAN

1      10     6    45     7     8.5
2      11    20     .    34    20.0
3       .     3    65    21    21.0
```

The second example is a clinical trial where five repeated measurements were made on each subject. Some kind of average of the 5 measurements is the desired result. It is possible that there are bad measurements (outliers). Since the mean is highly sensitive to the outliers, the median is the right statistics for average in this case. Outliers need to be identified for the purpose of further investigation. For this application, outliers are defined as the observations whose distances from the center location of the five measurements are greater than a preset threshold. Because of its insensitivity to outliers, the median is used to estimate the center location. Using the second macro, this algorithm can easily be carried out. See the following program. ‘m1’ to ‘m5’ are the five measurements. Note after the macro call, the resulting median of ‘m1’ to ‘m5’ is stored in a variable named ‘median’ by default. If the criterion for outliers is met, a message is printed in the log window.

```
data mydata;
    set mydata;
    %median(varlist = m1 m2 m3 m4 m5);
    array measure m1-m5;
    do i = 1 to 5;
        if abs(measure(i) - median) >
                &thrshld then do;
            put "Outlier Detected! " _n_
                 "Observation, " i
                 "Measurement";
        end;
    end;
run;
```

The following plot illustrates the result with a simulated data set. The solid line is the median, the dashed-lined bands indicate the preset threshold, and the stars are the detected outliers.



Conclusion

The median statistics are very useful in many situations. While SAS does not have a built-in function to compute the median for a list of variables, the two macros presented can be used easily for this purpose. The users can choose to compute the median either within or outside a data step.

References

1. SAS Institute Inc. (1995), SAS Procedures Guide, Version 6, Third Edition, Cary, N.C:SAS Institute Inc.
2. SAS Institute Inc. (1995), SAS Guide to Macro Processing, Version 6, Second Edition, pp. 263, Cary, N.C.: SAS Institute Inc.

Trademarks

Contact

Fan Xu
SpectRx Inc.
6025 A Unity Drive
Norcross, GA 30071

(770)242-8723
fxu@spectrx.com