A SAS® Procedure (PROC DICHOT) is presented which (a) computes the median for a given set of numeric variables, (b) performs a median split on those variables, (c) determines whether or not the numeric variable is above or below the median and creates a new variable (DICOTn) whose value is either 1 or 0, and (d) outputs the variable(s) to a SAS DATASET. PROC DICHOT provides an option which allows processing the data in either sorted or nonsorted form as well as an option which allows users to determine the logistic values of scores which are equal to the median. These new dichotomous variables may then be used as input to procedures which perform analysis on categorical variables. Examples of using PROC DICHOT are presented along with data regarding time and memory utilization using input SAS datasets with varying numbers of observations.

INTRODUCTION

Although there are SAS Procedures that will perform statistical analysis on discrete (categorical) variables, input SAS datasets are required to have such variables coded in a discrete manner. SAS datasets which contain continuous numeric variables must be processed in a prior DATA STEP which either recodes those variables, or creates new variables which take on the desired discrete values. One strategy that is frequently employed to create dichotomous variables from a list of continuous variables is to compute the median, determine whether or not a score falls above, below, or at (equal to) the median, and then assign a logistic value (1, or 0) to a new variable. These new "logistic variables" may be used as input to an appropriate procedure for statistical analysis. These computations may be accomplished using the SAS system. One approach would be to (a) create a SAS dataset, (b) use Proc Univariate to compute the median for the variables of interest, (c) create an output SAS dataset from Proc Univariate which contains the median(s) for the continuous variables, (d) merge the output SAS dataset with the input SAS dataset, and (e) create the new logistic variables based on the continuous variables relationship to the median. We present a SAS Procedure (PROC DICHOT) which will compute the median for a set of continuous numeric values and subsequently create new logistic variables based on the continuous variables relationship to the median.

PROC DICHOT

PROC DICHOT options;
VAR varlist;

Proc DICHOT will compute the median and create new dichotomous variables from an input SAS DATASET. The following restrictions apply:

(a) The variables must be numeric.
(b) The number of variables is limited to 999 per execution of PROC DICHOT.
(c) Currently no provisions are made for BY group processing.

The following options are available for PROC DICHOT:

(a) SORTED - The input SAS DATASET has been sorted on the numeric variable(s) on which PROC DICHOT will perform the analysis. The variables may be sorted in either ascending or descending order. The default is NOTSORTED.

(b) LOW - This option allows the User to assign a value of 0 (zero) for variables whose numeric counterparts are equal to the median. The default is to assign a value of 1 (one) to new variables whose numeric counterparts are equal to the median.

(c) OUT=SASDATASET - When this option is specified the new dichotomous variables along with all variables specified in variable list statement are included in the OUT SASDATASET. The default is to
include the new variables in the latest
(WORK.DATAn) DATASET.

(d)DATA=SASDATASET - Identifies the
input SAS DATASET to PROC
DICHOT. The default is the current
SAS DATASET.

PROCEDURE OUTPUT

The computed medians for each of the numeric
variables are provided as standard procedure
output.

MISSING VALUES

PROC DICHOT will use only valid (non-missing)
values in computing the median. If all values for a
given variable are missing or if there is only one non
missing value for a variable, PROC DICHOT will
print an appropriate warning message in the
SASLOG and the dichotomous counterparts of the
missing numeric variables will be set to the
standard SAS missing value (.)

USING PROC DICHOT

The following examples illustrate the use of PROC
DICHOT under VM/CMS.

EXAMPLE 1:

(1) CMS FI $SYSLIB DISK DICHOT
LOADLIB A (CONCAT;

(2) DATA TEST;
   INPUT NUM1 NUM2@@;
   CARDS;
   6 9 1 10 5 22 5 12
   4 11 3 24 7 35 2 44
   ;

(3) PROC DICHOT DATA=TEST;

(4) VAR NUM1 NUM2;
   TITLE 'OUTPUT FROM PROC DICHOT';

(5) PROC PRINT DATA=TEST;
   TITLE 'CONTENTS OF DATASET TEST';

(6) PROC PRINT;
   TITLE1 'OUTPUT DATASET';
   TITLE2 FROM PROC DICHOT';

The above example (1) defines the LOADLIB
containing PROC DICHOT to the SAS System, (2)
creates an input SAS dataset containing numeric
variables, (3) invokes PROC DICHOT specifying
the input SAS dataset, (4) identifies the numeric
variables for PROC DICHOT, (5) prints the contents
of the input SAS dataset TEST, and (6) prints the
contents of the output SAS dataset
(WORK.DATAn) produced by PROC DICHOT.
Below is the output (Listing) file produced by the
SAS System under VM/CMS.

OUTPUT FROM PROC DICHOT

NUM1 MEDIAN = 4.50 NUM2 MEDIAN = 17.00

CONTENTS OF DATASET TEST

<table>
<thead>
<tr>
<th>OBS</th>
<th>NUM1</th>
<th>NUM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>35</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>44</td>
</tr>
</tbody>
</table>

OUTPUT DATASET
FROM PROC DICHOT

<table>
<thead>
<tr>
<th>OBS</th>
<th>NUM1</th>
<th>NUM2</th>
<th>DICOT1</th>
<th>DICOT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>22</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>12</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>24</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>35</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>44</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

EXAMPLE 2:

CMS FI $SYSLIB DISK DICHOT LOADLIB A
(CONCAT;

DATA TEST;
   INPUT NUM1 NUM2@@;
   CARDS;
   6 9 1 10 5 22 5 12
   4 11 3 24 7 35 2 44
   ;

PROC DICHOT DATA=TEST;

VAR NUM1 NUM2;
TITLE 'OUTPUT FROM PROC DICHOT';

PROC PRINT DATA=TEST;
TITLE 'CONTENTS OF DATASET TEST';

PROC PRINT;
TITLE1 'OUTPUT DATASET';
TITLE2 FROM PROC DICHOT';
Example 2 illustrates how PROC DICHOT handles missing values. The SAS LISTING file is presented below. The median for each continuous variable is computed using only valid (non-missing) values. The variables created by PROC DICHOT (DICOTn) are assigned the SAS system missing value (.) if their continuous counterparts are missing.

**OUTPUT FROM PROC DICHOT**

NUM1 MEDIAN = 5.00 NUM2 MEDIAN = 11.50

**OUTPUT DATASET FROM PROC DICHOT**

<table>
<thead>
<tr>
<th>OBS</th>
<th>NUM1</th>
<th>NUM2</th>
<th>DICOT1</th>
<th>DICOT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.0</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5.0</td>
<td>22</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5.0</td>
<td>12</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4.0</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>3.0</td>
<td>24</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7.0</td>
<td>35</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2.0</td>
<td>44</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>4.5</td>
<td>17</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**EXAMPLE 3:**

CMS FI $SYSLIB DISK DICHOT LOADLIB A (CONCAT;

DATA TEST;
INPUT NUM1 NUM2@@;
CARDS;
6 9 1 10 5 22 5 12
4 11 3 24 7 35 2 44 4.5 17
;

PROC DICHOT DATA=TEST LOW;
VAR NUM1 NUM2;
TITLE 'OUTPUT FROM PROC DICHOT';

PROC PRINT ;
TITLE1 'OUTPUT DATASET';
TITLE2 ' FROM PROC DICHOT';

Example 3 illustrates the use of the LOW option for PROC DICHOT. Scores whose values are equal to the median are assigned a logistic value of zero. Below is the listing file produced from Example 3.

**OUTPUT FROM PROC DICHOT**

NUM1 MEDIAN = 4.50 NUM2 MEDIAN = 17.00

**EXAMPLE 4:**

CMS FI $SYSLIB DISK DICHOT LOADLIB A (CONCAT;

DATA TEST;
INPUT NUM1 NUM2@@;
CARDS;
6 9 1 10 5 22 5 12
4 11 3 24 7 35 2 44 4.5 17
;

PROC DICHOT DATA=TEST OUT=DIKOT;
VAR NUM1 NUM2;
TITLE 'OUTPUT FROM PROC DICHOT';

PROC PRINT DATA=TEST;
TITLE 'CONTENTS OF DATASET TEST';

PROC PRINT DATA=DIKOT;
TITLE 'DATASET DIKOT';

Example 4 illustrates the use of the OUT = option with PROC DICHOT. Since the LOW option is not specified scores whose values are equal to the median are assigned a logistic value of one. Below is the output produced from Example 4.

**OUTPUT FROM PROC DICHOT**

NUM1 MEDIAN = 4.50 NUM2 MEDIAN = 17.00

**CONTENTS OF DATASET TEST**

<table>
<thead>
<tr>
<th>OBS</th>
<th>NUM1</th>
<th>NUM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.0</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>5.0</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>5.0</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>4.0</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>3.0</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>7.0</td>
<td>35</td>
</tr>
<tr>
<td>8</td>
<td>2.0</td>
<td>44</td>
</tr>
<tr>
<td>9</td>
<td>4.5</td>
<td>17</td>
</tr>
</tbody>
</table>
DATASET DIKOT

<table>
<thead>
<tr>
<th>OBS</th>
<th>NUM1</th>
<th>NUM2</th>
<th>DICOT1</th>
<th>DICOT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.0</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>5.0</td>
<td>22</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5.0</td>
<td>12</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4.0</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>3.0</td>
<td>24</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7.0</td>
<td>35</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2.0</td>
<td>44</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>4.5</td>
<td>17</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

CPU TIME and MEMORY UTILIZATION with PROC DICHTOT

The amount of time and memory that PROC DICHTOT will require depends upon several factors among which include (a) the number of numeric variables that are specified in the VAR statement, (b) the number of observations contained within the input SAS dataset, and (c) whether or not the SORTED option is specified. We have chosen to present data on time and memory utilization using a SAS DATASET with varying numbers of observations (1000 to 10000), with 10 variables, and using either the SORTED option or the default (NOTSORTED). These examples are by no means exhaustive but are provided to illustrate some guidelines concerning the efficient use of PROC DICHTOT. The CPU is an IBM 3081-D24 and the operating system is VM/CMS.

Figure 1 represents the comparison of the amount of CPU time used by PROC DICHTOT when either the SORTED option or the default is specified. The y axis represents the log of CPU seconds while the x axis represents the number of observations in the SAS dataset. Using the SORTED option with PROC DICHTOT may result in savings in the number of CPU seconds used and we recommend sorting the variables that are input to PROC DICHTOT for large SAS datasets.

Figure 2 represents a comparison of the amount of memory used by PROC DICHTOT when either the SORTED option is specified or the default (NOTSORTED) is used. The y axis represents the amount of memory in kilobytes and the x axis represents the number of observations in the SAS dataset. There are savings in memory utilization when the SORTED option is specified but these savings appear to diminish as the number of observations increase above 8000 thus, savings in memory allocation would not be realized for very large SAS datasets. These data were obtained under VM/CMS with a virtual machine size of 3 megabytes.

SUMMARY

We present a SAS Procedure PROC DICHTOT which will compute the median for a set of numeric variables and create a new dichotomous variable (DICOTn) whose value (1 or 0) depends on the relationship of the input variable to the median. These new computed variables may be used as input to statistical procedures which perform categorical or logistic analysis. The data presented on time and memory indicate that specification of the SORTED option may result in more efficient performance by PROC DICHTOT and that this performance is dependent on the number of observations contained within the input SAS dataset. Although no data was presented concerning variation in the number of variables input to PROC DICHTOT we suspect that increasing the number of input variables would also effect time and memory usage.
REFERENCES


For additional information contact:

Paul Lambert, Ph.D.
Southern Methodist University
Academic Computer Services
6100 Ownby Drive
Dallas, Texas 75275

(214) 692-2900 Bitnet: UPLL @SMUVM1

ACKNOWLEDGMENT

I would like to thank David Garrigan for his expertise with Desktop Publishing in preparation of this manuscript.
FIGURE 1: COMPARISON OF SORTED VERSUS NONSORTED DATA ON THE NUMBER OF CPU SECONDS USED BY PROC DICHOT ON SAS DATASETS WITH VARYING NUMBERS OF OBSERVATIONS

FIGURE 2: COMPARISON OF SORTED VERSUS NONSORTED DATA ON THE AMOUNT OF MEMORY USED BY PROC DICHOT ON SAS DATASETS WITH VARYING NUMBERS OF OBSERVATIONS