A Macro Which Transposes Data
Beyond What PROC TRANSPOSE Can Do

Yen L. Ouyang
Hoffmann-La Roche, Inc.

Introduction

In order to write a nice report in SAS/BASE, sometimes you require the data set to be vertical orientation, instead of horizontal orientation, especially for the large size of variables data sets. The %MH2V macro was developed to transpose data from horizontal to vertical without knowing the variable name and type, regardless of the type and number of variables desired, providing functionality which PROC TRANSPOSE does not.

A main task of Macro %MH2V is to transpose all the variables (no matter how many you have, up to SAS limits) to two vertical columns, one is numeric value, and another is character value. In addition, the original label also will be saved in the variable called DESC. Of cause, you can select any variables (we call KEYVARS or BY VAR) that you do not want to transpose, even for multi duplicated value in KEYVARS.

Can PROC TRANSPOSE do this? It is not so easy. All transposed variables are the same type and length. If all input variables are numeric, the transposed variables are numeric. If any input variable is character, all transposed variables are character. Especially, if you have multi-duplicated observations in BY VAR, then you will get multi transposed columns (_COLn_).

Description

Technically, the macro uses _character_ and _numeric_ to determine the variable type from the input data set(s). The macro then defines two arrays which can handle an unlimited number of variables of mixed type. After macro processing, the transposed data is output to the user-specified output data set. Use of %MV2H preserves the integrity of the data much more than PROC TRANSPOSE.

Some functions and subroutines available in SAS Version 6.07 (or later) are used to keep the macro short and efficient. These include:

- CALL LABEL: A subroutine to get the label of a variable.
- CALL VNAME: A subroutine to get the name of a variable.
- COMPBL: A function to compress multiple blanks to a single blank.
- TRANWRD: A function to replace or remove occurrences of words in character string(s).

The following example illustrates how this macro works.

%mh2v(ind=datain,outd=dataout,
keyvars=orgclass);

(Detail of the macro codes see Appendix)

Data Set: DATAIN

<table>
<thead>
<tr>
<th>OBS</th>
<th>TRT2</th>
<th>TRT3</th>
<th>TRT4</th>
<th>ORGCLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>13</td>
<td>21</td>
<td>Any *</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>Renal</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>Ards respiratory</td>
</tr>
</tbody>
</table>


DATAIN is an input dataset in macro %mh2v, where the ORGCLASS is the organ class, and TRT2, TRT3, and TRT4 are the number of patients with serious AE leading to death who took test medication 5mg, 10mg, and 20mg respectively.

Data Set: DATAOUT

<table>
<thead>
<tr>
<th>OBS</th>
<th>ORGCLASS</th>
<th>DESC</th>
<th>INVAR</th>
<th>VALUE1</th>
<th>VALUE2</th>
<th>INTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Any **</td>
<td>5</td>
<td>TRT2</td>
<td>16</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Any **</td>
<td>10</td>
<td>TRT3</td>
<td>13</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Any **</td>
<td>20</td>
<td>TRT4</td>
<td>21</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Renal</td>
<td>5</td>
<td>TRT2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Renal</td>
<td>10</td>
<td>TRT3</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Renal</td>
<td>20</td>
<td>TRT4</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Ards respiratory</td>
<td>5</td>
<td>TRT2</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Ards respiratory</td>
<td>10</td>
<td>TRT3</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Ards respiratory</td>
<td>20</td>
<td>TRT4</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>


DATAOUT is an output dataset after we run macro %mh2v, where ORGCLASS assigned to &keyvars that is a variable not being transposed to vertical; DESC contains the original label of variables; INVAR contains the names of input variables being transposed to vertical; VALUE1 contains numeric values of the variables being transposed; VALUE2 contains character values of the variables being...
transposed; INTYPE contains the INVAR type (1-numeric, 2-character). Of cause, the &keyvars could be assigned more than one variable as well as the duplicated value for &keyvars. Furthermore, mixed type of variables are allowed to be transposed under the macro, their value could go to either VALUE1 or VALUE2 columns.

The following shows the report used above with transposed dataset easily generated by PROC REPORT with across variable INVAR.

<table>
<thead>
<tr>
<th>Test Medication</th>
<th>5 mg</th>
<th>10 mg</th>
<th>20 mg</th>
<th>120</th>
<th>128</th>
<th>110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serious AE leading to Death</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>(%)</td>
<td>No</td>
<td>(%)</td>
<td>No</td>
<td>(%)</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>Any**</td>
<td>16 (13%)</td>
<td>13 (10%)</td>
<td>21 (19%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aids respiratory</td>
<td>5 (4%)</td>
<td>3 (2%)</td>
<td>2 (1%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arrhythmias</td>
<td>1 (0%)</td>
<td>1 (0%)</td>
<td>1 (0%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNS</td>
<td>1 (0%)</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>58 (48%)</td>
<td>46 (35%)</td>
<td>73 (66%)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

References

SAS™ is a registered trademark of SAS Institute, Inc, Cary, NC.

Appendix
%MACRO MH2V(inds=,outs=,keyvars=)
data ina_;
set &inds;
run;
%local nvar kpvars;
/*****************************
This DATA_NULL block generates two macro variables:
&KPVARS: quoted &KEYVARS, for example:
if we set &KEYVARS= STUDY PNO, then
&KPVARS="STUDY","PNO"
&NVAR: Number of the variables in the KEYVARS.
/*****************************/
data _null_;
length newvar keyvars $200;
/*****************************/
Determine the number of elements in the array. Store this count in a macro variable called NVAR
/*****************************/
array count(*) &keyvars;
call symput('invar',trim(left(dim(count))));
%MACRO END: ""
End of the macro ***/

Compress multiple blanks from the macro variables passed as a parameter. Translate the single blank between words to a '.'. This will make the string appropriate as initial values for the array elements. Store this string in a macro variable called KPVARS.

**/compress multiple blanks to single blanks */

%MACRO MH2V(inds=,outs=,keyvars=)
data ina_;
array char[*] _character ; /* all character variables */
array stub[*] _numeric ; /* all numeric variables */
array kvar &kpvars $ _temporary (_ &kpvars);
length desc $40 /* desc is not in either array */
invar $8 /* invar is not in either array */
value1 $ value2 $80;
do i = 1 to dim(numb);
value1 = ;
value2 =;
call vname(numb{i},invar); /*get a name of numeric variable */
/* if the variable is in the &KEYVARS then it will not be transposed */
do j = 1 to dim(kvar);
d = dim(kvar);
if invar = upcase(kvar{j}) then goto EN; end;
call label(numb{i},desc); /*get a label of numeric variable */
/* get a value from numeric variable */
if value1 = then output;
EN:
do i = 1 to dim(char);
value1 = ;
value2 =;
call vname(char{i},invar); /* get a name of character variable */
do j = 1 to dim(kvar); /* if the variable is in the &KEYVARS then it will not be transposed */
if invar = upcase(kvar{j}) then goto EC; end;
call label(char{i},desc); /* get a label of character variable */
/* get a value from character variable */
if value2 = INIT then output;
EC:
label invar = 'Original input variable name'
intype = 'Original var value type. 1: num, 2: char'
desc = 'Original variable label'
value1 = 'Numeric value of original variable'
value2 = 'Character value of original variable'
run;
%MACRO END: ""
End of the macro ***/