

A Better Transpose: %TRANSP Macro Preserves Variable Attributes

Ted Clay, Clay Software & Statistics, Ashland, OR

ABSTRACT

The Transpose procedure will “flip” a rectangular table of your data, converting a R-by-C matrix into a C-by-R matrix. This talk presents a macro which **outputs all variables onto a single observation** per by-group. For example, if you have a data set with variables A, B and C and up to 4 observations per by-group, the output has one observation per by-group with variables A1-A4, B1-B4 and C1-C4. The advantage over PROC TRANSPOSE is that **variable attributes are preserved**. You keep your original variable lengths, formats and labels, and the variables can be a mixture of character and numeric. You can specify an optional ID variable, which contains the suffixes of the created variables. The macro merges the input data set with multiple copies of itself, each copy with a where-option selecting a specific value of the ID variable. The combination of simplicity and utility makes this is my favorite macro.

INTRODUCTION – THE NEED

You are trying to manage your data. Your data is running down the page, but you really want it to run across the page. Your dataset is long and skinny, and you want it to be short and fat. If you only have one variable to deal with, no problem. PROC TRANSPOSE with only one variable on the VAR statement will do the trick nicely. But this is rarely the case. When you want to transpose more than one variable, PROC TRANSPOSE will produce a separate observation for each variable (within each by-group). Often what you really want is to have a single observation per by-group with all the variables on it. And you would like the new variables to be given meaningful names related to your original names.

The second limitation with PROC TRANSPOSE is that you cannot transpose a mixture of numeric and character variables. Yes, PROC TRANSPOSE has an “answer” to this situation – it converts all the numeric variables to character – but I have yet to meet anyone who actually wanted this to happen.

Finally, PROC TRANSPOSE leaves all your variable attributes behind. Any formats, labels and lengths which you had on your input variables get lost. Formats and labels are made blank, and the lengths of numeric variables all become 8. This is a necessary consequence of the way PROC TRANSPOSE tries to “flip” a matrix of variables and observations.

THE SOLUTION

By trying to do less, we are able to preserve more. The %TRANSP macro outputs all variables onto a single observation per by-group. For example, if you have a dataset with variables A, B and C and up to 4 observations per by-group, the output will be one observation per by-group with variables A1-A4, B1-B4 and C1-C4. The variables A, B and C can be a mixture of character and numeric, and their lengths, formats and

labels carry over to the corresponding output variables.

Here are the parameters of %TRANSP:

DATA = (required) Name of input dataset.
OUT = (required) Name for output dataset.
BY = (required) By-grouping variable(s). Output data set will have one observation per unique value.
VARS = (required) List of one or more variables to be transposed. Can be any character or numeric variables. The list can be any valid SAS variable list, e.g. x1-x5, age--sex, etc.
ID = (optional) Name of variable whose values will determine the suffix of the new variable names. May be character or numeric. If it has a format assigned, the formatted values are used. If no ID variable is given, the suffixes of the new variables will be the numbers 1,2,3, etc. assigned to the observations in the order they are found.
VARSEP = (optional) Text to insert between the old variable name and the variable suffix.
LABELSEP = (optional) Text to insert between the old label and the variable suffix.

THE ID VARIABLE

The ID variable is the most “interesting” parameter. If it is not specified, the default suffixes are the numbers 1,2,3, etc. In some situations, you may not care about the variable suffixes, and would be happy to have them simply numbered 1,2,3, etc. because you plan to loop over them in an array statement. If you specify an ID variable, its (formatted) values are used for the variable suffixes. For example, the input data might have one observation per time point, and you would like your new variable names to indicate what time point the data is from. This is the most common situation because usually the original observations contained very specific content. I have found that the ID variable is needed 95% of the time, giving greater control over how the new variables are named.

Like PROC TRANSPOSE, the %TRANSP macro uses an ID variable for naming the output variables. It is useful to understand the difference between how PROC TRANSPOSE and the %TRANSP macro handles the ID variable. In PROC TRANSPOSE, the ID variable contains either a complete variable name, or a variable suffix to be added to the “PREFIX” option on the Proc statement. In the %TRANSP macro, the ID variable is always used as a suffix, and the prefix is the original variable name. The VARSEP and LABELSEP macro parameters allow you to insert arbitrary characters into the new variable names and labels, to “separate” the old name or label from the new suffix.

The %TRANSP macro allows the ID variable to be either character or numeric. So if you already have a numeric variable called “Year” with values 1989, 1990, etc. you can use it as-is, without adding a data step to convert Year to a character variable. Secondly, if the ID variable

has been given a format, the formatted values rather than the underlying data values are used for the output variable suffixes.

In summary, the output variable names consist of

1. The original variable name, concatenated with
2. The optional VARSEP text string, concatenated with
3. The formatted values of the ID variable, or else "1", "2", "3" if no ID variable is given.

Finally, any invalid or blank characters in the variable name are converted to underscores.

EXAMPLE

INPUT DATASET: OFFICERS

Club	Posit	Name	DOB
Poker	Pres	Jack	04/03/70
Poker	VP	Fred	09/24/81
Poker	Tres	Beth	07/19/21
Yoyo	Pres	Joan	02/09/88
Yoyo	VP	Hal	12/25/70

INPUT DATASET CONTENTS

Variable	Type	Len	Format	Label
Club	Char	7		Club
Posit	Char	4		Position
Name	Char	4		First Name
DOB	Num	8	MMDDYY8.	Date of Birth

THE MACRO CALL

```
%TRANSP0(DATA=officers,OUT=clubs,
BY=club,
ID=posit,
VARS=name dob,
VARSEP=,
LABELSEP=of);
```

OUTPUT DATASET: CLUBS (edited)

Club	Name_ Pres	DOB_Pres	Name_ Tres	DOB_Tres	Name_ VP	DOB_VP
Poker	Jack	04/03/70	Beth	07/19/21	Fred	09/24/81
Yoyo	Joan	02/09/88			Hal	12/25/70

OUTPUT DATASET CONTENTS (edited)

Variable	Typ	Len	Format	Label
Club	C	7		Club
Name_Pres	C	4		First Name of Pres
DOB_Pres	N	8	MMDDYY8.	Date of Birth of Pres
Name_Tres	C	4		First Name of Tres
DOB_Tres	N	8	MMDDYY8.	Date of Birth of Tres
Name_VP	C	4		First Name of VP
DOB_VP	N	8	MMDDYY8.	Date of Birth of VP

In examining the datasets above, notice the following features: The date-of-birth variable is numeric while the name variable is character. The date format on the input variable DOB is also on the output variables DOB_Pres, DOB_VP and DOB_Tres. The values of the ID variable POSIT become the suffixes. Between the old name and the suffix is an underscore, which was specified by the VARSEP parameter. Between the old label and the suffix is the word "of", which was specified by the LABELSEP parameter. Notice also that the Yoyo club had no Treasurer, so the Name_Tres and DOB_Tres variables are missing on the Yoyo observation.

HOW IT WORKS

In the simple example above, the macro internally generated the following SAS code:

The Generated Code

```
data clubs;
merge officers(where=(Posit="Pres"))
      rename=(Name= Name_Pres
              DOB = DOB_Pres)
officers(where=(Posit="Tres"))
      rename=(Name=Name_Tres
              DOB =DOB_Tres))
officers(where=(Posit="VP"))
      rename=(Name= Name_VP
              DOB =DOB_VP));
by club;
label Name_Pres = "First Name of Pres" ;
label DOB_Pres = "Date of Birth of Pres";
label Name_Tres = "First Name of Tres";
label DOB_Tres = "Date of Birth of Tres";
label Name_VP = "First Name of VP";
label DOB_VP = "Date of Birth of VP";
run;
```

The Preparatory Steps

Leading up to the data step above a few actions are taken:

1. If no ID variable is specified a view of the input data is defined which assigns a sequential number to each observation within a by-group.
2. The type and format of the ID variable are determined and stored in macro variables IDTYPE and IDFMT, with IDFMTED=YES if the ID variable has a format.
3. Proc Freq gets the list of distinct (formatted) values of the ID variable, and we store them in a series of macro variables (ID1 ID2 ID3 etc. up to &NUMIDS).
4. Proc Contents gets the labels of the variables to be transposed, and we store the names and labels in a series of macro variables (VAR1 VAR2 VAR3 etc. and LBL1 LBL2 LBL3 etc. up to &NUMVARS). If a variable has no label, the variable name is assigned to the label macro variable, so the output variables will always have labels.
5. Then the final data step is generated using the following macro code:

The Key Macro Statements

```
data &OUT;
  merge
  %DO I=1 %TO &NUMIDS;
    &DATA(keep=&BY &IDVAR &VARS

    %IF &IDFMTED=YES %THEN
      where =
      (left (put (&IDVAR, &IDFMT&IDFMTL. .))=
      "&&ID&I");
    %ELSE %IF &IDTYPE=NUM %THEN
      where= (&IDVAR = &&ID&I );
    %ELSE %IF &IDTYPE=CHAR %THEN
      where=(left (&IDVAR) = "&&ID&I");
    rename=(
      %DO V=1 %TO &NUMVARS;
        &&VAR&V = &&VAR&V. .&VARSEP&&ID&I
      %END;
    )
  %END;
  ; by &BY;
  %* Put a blank at the end of LABELSEP;
  %IF %STR(&LABELSEP) NE %THEN %LET
    LABELSEP=%STR(&LABELSEP );
  %DO I=1 %TO &NUMIDS;
    %DO V=1 %TO &NUMVARS;
      label &&VAR&V. .&VARSEP&&ID&I
        = "&&LBL&V &LABELSEP&&ID&I";
    %END;
  %END;
  drop &IDVAR;
run;
```

CONCLUSION

Perhaps a future version of PROC TRANSPOSE will have an option on the PROC statement to do what is described in this presentation. In the meantime, you can download the %TRANSPO macro from the web page. Give it a try. Any feedback and suggestions for improvement will be appreciated.

REFERENCES

SAS PROC TRANSPOSE is documented in SAS Procedures Guide, Version 8, chapter 39, pages 1269-1290.

ABOUT THE AUTHOR

Ted Clay, M.S. is a statistical consultant and data analyst. His clients have included pharmaceutical companies, manufacturing companies, and grass-roots organizations, as well as research projects in epidemiology and health policy at the University of California San Francisco.

Your comments and questions are valued and encouraged. Contact the author at:

Ted Clay
Clay Software & Statistics
168 Meade St.
Ashland, OR 97520
Work Phone: 541-482-6435
Fax: Same
Email: tclay@ashlandhome.net
Web: www.ashlandinternet.com/~tedclay

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.