

SESUG Poster 10-2022: Long to Wide Format, Isaiah Omerhi, Emory University Alumni

ABSTRACT:

Reports generated from the various medical record sources may present in the long format with multiple rows for a single ID variable. To perform descriptive analysis on such data all unique ID must be in one row. One way SAS reshape the long format to a wide format -where all unique ID are in a single row is by using the function `proc TRANSPOSE`. However, datasets with multiple repeating categorical variables introduces a special challenge to the *transpose* procedure.

This is your poster abstract.

INTRODUCTION:

SAS software is equipped with several tools for organizing data. Rearranging in this context means converting variables into observations or observations into variables. Data might comprise sets of observations for different individuals who were followed over time and each results record is entered in a separate row for the same individual, i.e., each observation represents a measurement taken on a different date. In order to see the change in outcome for everyone for interest, the start and end date for each individual preferable should be on one row. `Proc Transpose` changes multiple values in rows (for a column) into columns and can also change multiple columns' values into multiple rows values for a single column. The `ID` statement names the column in the input file whose row values provide the column names in the output file. Typically, there should only be one variable in an `ID` statement, and the column used for the `ID` statement cannot have any duplicate values. The `VAR` statement in *proc transpose* specifies which variables' values are to be transposed and this can be either character and/or numeric variables if the `VAR` statement is omitted, *proc transpose* transposes all numeric variables. The `BY` statement names row-identification variable(s) whose values are not transposed, like other procedures that use the `BY` statement preliminary sorting using `Proc Sort`, is required. *Proc Transpose* also includes some default variables in the output dataset `_NAME_`, `_LABEL_`, these can be overridden by the prefix and/or drop statement. The prefix option provides a prefix to the transposed column names instead of `COL1`, `COL2`, etc. and the Name option provides the name for an output file column which tells which input variables were transposed.

```
proc transpose data=mydata out=mydata1 (drop=_NAME_ _LABEL_) prefix=columnn1;
var columnn1;
by ID;
run;
proc transpose data=mydata out=mydata2 (drop=_NAME_ _LABEL_) prefix=columnn2;
var columnn1;
by ID;
run;
proc sort data=columnn1;
by ID;
run;
proc sort data=columnn2;
by ID;
run;
data columnnbyID transpose;
merge columnn1 (in=a) columnn2(in=b);
by ID; if a=b;
run;
```

This is your poster introduction. This is not required if covered in abstract.

SUMMARY:

When only a single variable is transposed, the SAS code needed is usually quite simple. The use of an `ID` statement allows control of the placement of variable values into new variables in the transposed data set

This is your poster summary.

CONCLUSION/IMPLICATIONS:

`Proc Transpose` needs prior sorting if the `BY` statement is used this will create the necessary number of columns to accommodate all records being transposed. `Proc Transpose` may not be able to transpose several variables into one row, instead, they must be transposed one by one

This is your poster conclusion.

CONTACT INFO:

Isaiah Omerhi, skimoid7@gmail.com, 7049990397

Your comments and questions are valued and encouraged. Contact the author at: *Isaiah Omerhi, skimoid7@gmail, 7049990397*