Paper 234-2007

An Animated Guide: Proc Transpose

Russ Lavery: Contractor for Numeric Resources

ABSTRACT

If one can think about a SAS® data set as being made up of columns and rows, one can say Proc Transpose "flips" the columns of data into rows of data -and also does the reverse (flips rows into columns). The ability to "flip" data is necessary because some SAS Procs want your data to "be in rows" and some Procs want your data to "be in columns". Complex manipulations of the shape of the data (tall & thin data sets into wide and short data sets and the reverse) can be quickly performed with Proc Transpose. The trick in using Proc Transpose is to recognize that the statements in your code do not completely determine the output you get. Output is determined by your individual coded statements, the interaction between coded statements, the shape of the input data set and also by actions that Proc Transpose takes by default. This paper contains many slides. It should be noted that most slides are a complete "worked example" and can be used as a reference. All examples shown are included in the paper.

INTRODUCTION

Many programmers, when they have to "reshape" data sets, use a data step and justify their choice by saying that they want control of the process. Proc Transpose, properly understood, also gives control of the process. Anyone doing reporting on multiple levels (e.g. reporting on region *and* state *and* county *and* city *and* zip) will find the combination of Proc Summary output files "pushed through" a Proc Transpose to be a great time saver.

We must remember four things when learning Proc Transpose. The first is that the several coded statements/options in Proc Transpose interact to produce the output. This means that understanding of statement combinations- not understanding of statements- is required. The second thing to remember is that, *for the same Proc Transpose code*, input data files with different "shapes" can produce output data files with different shapes. The third thing to remember is that Proc Transpose has several actions that it takes on its own - in the absence coded statements. This means that if the programmer does not code instructions telling Proc Transpose what to do, Proc Transpose will take action on its own. These automatic actions can interact with your coded instructions. The fourth thing to remember is the set of rules you apply to make the columns created by Proc Transpose come out in a desired order. Note: code for all the examples in this paper, and more, is in small font at the end of the paper.

The basic syntax (no Let or Copy statements – statement I never use) of Proc Transpose is shown in Figure 1 and the basic process is shown in Figure 2. This paper will endeavor to cover, in detail, the statements in Figure 1. Let and Copy statements (not shown in Figure 1) will be briefly covered in Section 6.

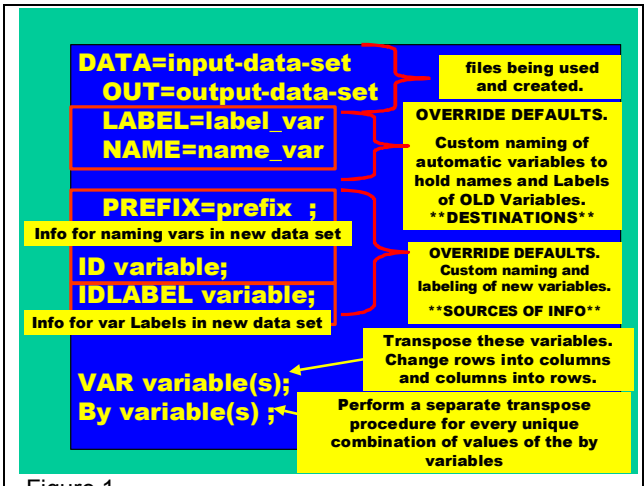

Figure 1

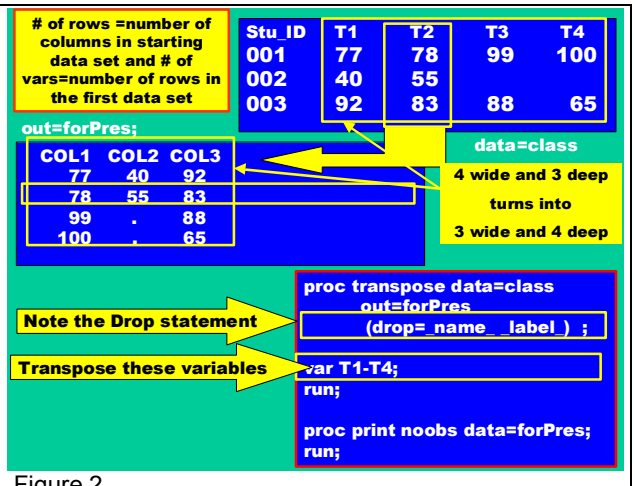

Figure 2

GOALS/ISSUES

Mastering Proc Transpose can be broken into six sections/issues/goals.
Section 1) Understanding which variables will be transposed
Section 2) Understanding the storing of old variable names in columns in the new dataset – automatically & manually
Section 3) Understanding the creation of names & labels for variables being created – maintaining a link to reality
Section 4) Understanding how to control the shape of the output data set (rows and columns)
Section 5) Making the columns come out in the correct order and "filling in" missing columns
Section 6) Understanding a few quirks: Let, Copy, By combined with ID

Note, in Figure 2, the numbers in the source data set (data=class) had "links to reality" by means of the values in the stu_id column and the names of the variables/columns. By having "links to reality" in the data set, the data set is self-documenting. One can open the data set and link the values, in the data set, to their meaning. There is one score of 83 in the data and it is the score of Stu_Id 3 on test 2. These "links to reality" were destroyed by the transposition shown in Figure 2. The 83 is in the output data set but we no longer have, *in the data set itself*, any way of linking the 83 to reality (linking it back to a student and a test). This data set is small and the observations can be "lined to reality" by visual examination. If the data set had thousands of observations, a visual check would be difficult. The loss of the ""link to reality" is a problem and we need to have Proc Transpose maintain "links to reality" for us.

SECTION 1: WHICH VARIABLES WILL BE TRANSPOSED

The var statement specifies the variables/columns to be transposed. The "transposing process" is illustrated in Figure 2. As you can see there, the columns in the source data set have been "flipped" into rows in the output data set. Column 1 in Class is turned into row 1 in For_pres. Row 1 in Class is now column 1 in For_pres. This "flipping" is the basic operation of a transpose. However, if we code just this basic operation, it is difficult to look at the output data set and link a score to a student or test. There are no "links to reality" in the output data set. While the transpose "worked", we need to learn more features of Proc Transpose and to become better transposers of data.

SECTION 2) MAINTAINING LINKS TO REALITY- STORING THE OLD VARIABLE NAMES IN COLUMNS IN THE NEW DATA SET- AUTOMATICALLY AND MANUALLY

A technique for maintaining "a link to reality" through the transpose is to store old variable names in variables in the new data set. This makes the new data set self-documenting. This is so useful that Proc Transpose does it *as a default*. See Figures 3 and 4, where _name_ and _Label_ were created.

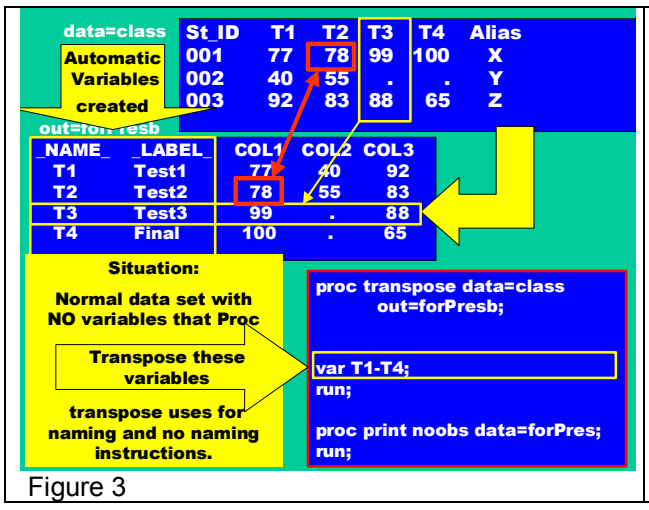

Figure 3

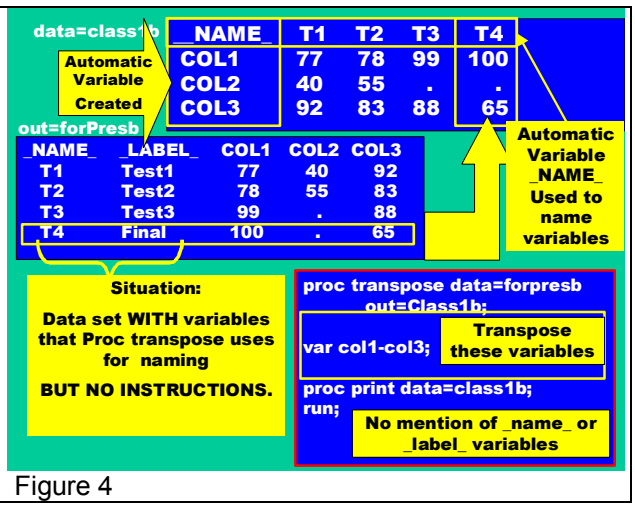

Figure 4

Figure 3 shows the automatic creation of two variables (_name_ and _label_) to hold information about the source data set. By holding information about the source data set, they hold "links to reality". By default, Proc Transpose will create 1 or 2 new columns (_name_ *always*, and _label_ if the variables being transposed have labels), in the output data set. By default, the names of these variables are _name_ and _label_. Rows in these new variables/columns contain the names and labels of the columns in the original data set as "links to reality".

Remember a column in the old data set turns into a row in the new data set, so it makes sense to store the name of the old column in a row of the new data set. Note that the code in Figure 3 only does half the job of linking the data to reality. While we have test information in the output file, there is no student information in the output file. Figure 4 is the transpose of the file we created in Figure 3 and shows the second half of a "double transpose". Figure 3 shows a transpose of the data set class into a dataset named ForPressb. Figure 4 shows the transpose of the output data set created in Figure 3 (ForPressb) into a dataset called Class1B. This is a double transpose… or a transpose of a transpose. Note that Proc Transpose *automatically* senses the values in _name_ and _label_, in ForPresb, and uses them to create names and labels for variables in the output data set class1B. This is an important rule of Proc Transpose. Without any instructions, Proc Transpose uses variables called _name_ and _label_ if they exist in the input data set.

The Proc Print, in Figure 4, uses values of _name_ as column headers for the dataset being created. If the Proc Print statement had used the "label" option, Figure 4 would show that values in the _label_ variable had been assigned as labels for the columns in the output data set.

Please pause and consider three issues. Firstly; most people would hope that a transpose - of a transpose - would get them back to the original data file; back to where they started from. In Figure 4 we see that it did not. The columns have been names T1 to t4 (test1 to test4) but student information has been lost. It is possible for a double transpose to keep track of student information ("links to reality") and test information. Keeping "links to reality" is both good programming practice and easy to do. It simply requires use of some commands to be seen later in the paper. Secondly; as all transposes like to do, the second transpose created _name_ as it created data set Class1b. Thirdly, as we see in Figure 4, the value T1 in a row in _name_, in the file ForPresb, will be used as a column name in the output data set class1b. You must be sure, in your more complicated transposes, that code attempts to "put" that value of _name_ over *only* one column, or the transpose will fail.

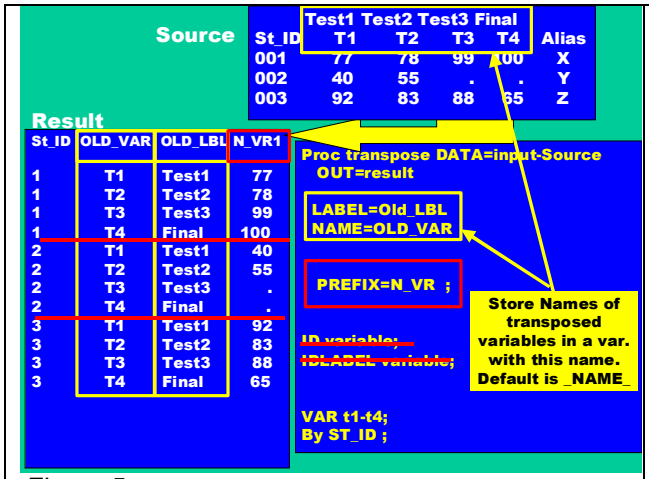

Figure 5

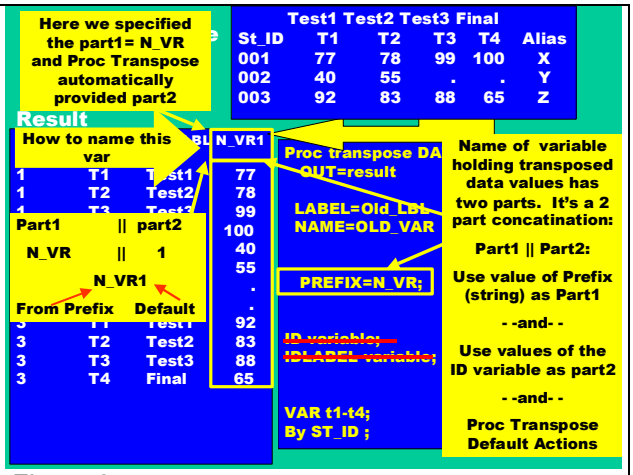

Figure 6

Figure 5 illustrates a technique for manual control of "links to reality" - the process of storing information about the "old" name and labels. A programmer can influence/control the names assigned to columns that are created. In Figure 5, a yellow box surrounds the code that allows a programmer to rename the columns that are being created to hold variable name and label information. The default names _name_ and _label_ have been changed to OLD_VAR and Old_LBL by using the Name= and Label= options . Also note, in Figure 5, the position of the semicolon that ends the Proc Transpose statement. The first semicolon is several lines "down from" the start of the procedure. Data=, Out=, Label=, Name= and Prefix= are all options on the Proc Transpose statement and occur before the first semicolon.

SECTION 3) MAINTAINING LINKS TO REALITY- CREATING NAMES AND LABELS FOR THE VARIABLES BEING CREATED

Figures 4 and 5 showed part of the technique for maintaining links to reality through a transpose. These figures show how to store the names and labels of the old variables in variables (_name_ _label_) in the new data set. This is half of the process of maintaining links to reality. We will also need to create meaningful names for variables being created by the transpose. An examination of Figures 2 and 3 shows several new variables (with names like col1 -- col3) that hold the transposed data. The names col1 -- col3 are not very informative. If one looks at Figure

3, it can be seen that all information for Student 001 ended up in col1. It would be desirable to name that column something like Student_1. If that were the possible, a programmer would be able to maintain both row and column "links to reality" through a transpose and have a self-documenting data set. In fact, this is possible and the technique will be shown below.

As Proc Transpose creates variables to hold the transposed data, the name of the new variables is the result of concatenating two pieces of information (call them left and right) into one column name. The rule for name creation is: the value in the prefix= option is used as the left hand part of the new name and the variable in the ID statement contains text used as the right hand part of the two part name. Prefix= specifies a string and that string does not change. This makes prefix= ideal for creating columns with a constant left hand side (like student_1 student_2). The right hand side of the name is specified by the ID statement and points to a variable in the source data set. This makes ID ideal for making column names with a varying right hand side (like student_1, student_2 or Territory_NY, Territory_NJ). In review, the new name is made up of a constant prefix and a varying suffix.

The eagerness of Proc Transpose to be helpful (by taking action without any instructions from the programmer) makes using Proc Transpose a bit confusing. If you specify prefix= and ID (both the left and right parts of the name), Proc Transpose follows your instructions and assembles the column names from the two components you coded. However, you can specify, *or not specify*, either of the components without SAS throwing any errors. When either right, or left, is omitted Proc Transpose will take the initiative to "help (or maybe confuse)" the programmer by creating logical and (sometimes) useful names.

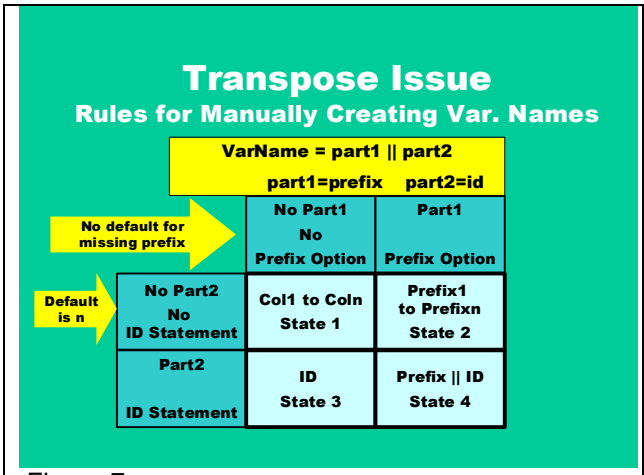

Figure 7

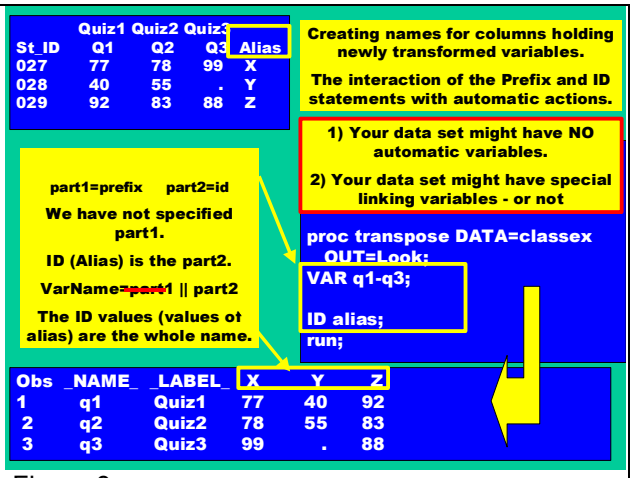

Figure 8

Figure 7 shows the states of nature that exist and the rules that Proc Transpose will apply for each of the four states. We must understand how to code or all these states of nature. A programmer can code the prefix= option, or not. A programmer can code the ID statement, or not. Combining the ways of coding these two instructions gives the four "states of nature" shown in Figure 7. Figure 7 says that if a programmer does not specify either Prefix or ID (State of nature 1: upper LH cell), the column names will be assembled as Col1 to Coln (Column||N). Transpose automatically makes the string "col" the value of the left-hand part of the new column names and an increasing series (1,2,3) the value of the right hand part of the new column names. This can be seen in Figure 3.

If only the Prefix=prefixvalue option is specified (state of nature 2: Upper RH cell), Proc Transpose will create column names prefixvalue1 to prefixvalue*n*. The left hand part of the new name is the value of the Prefix= option. Since the right hand side was not specified, Proc Transpose automatically makes an increasing series (1,2,3) the value of the right hand part of the name of the new columns/variables. This is what has happened in Figures 5 and 6, though, the pattern is not obvious since only one column is created in those slides.

Remember that ID "provides" the right hand part of the new variable name. If the ID statement is specified and the prefix= option is not specified (state of nature 3 in Figure 7), Transpose assumes that there is no need for a "left hand" part of the new name and makes column names from the values of the ID variable in the old table. Proc Transpose does not assume any value for the left hand part of the new name and names of new columns are simply the values of ID. There will be no 1,2,3 etc. in the new column names because ID, itself, is used to provide the right hand side of new variable names. This situation is shown in Figure 8 and can be quite useful if the ID variable is character. Variable names, created using this process are often very effective "links to reality".

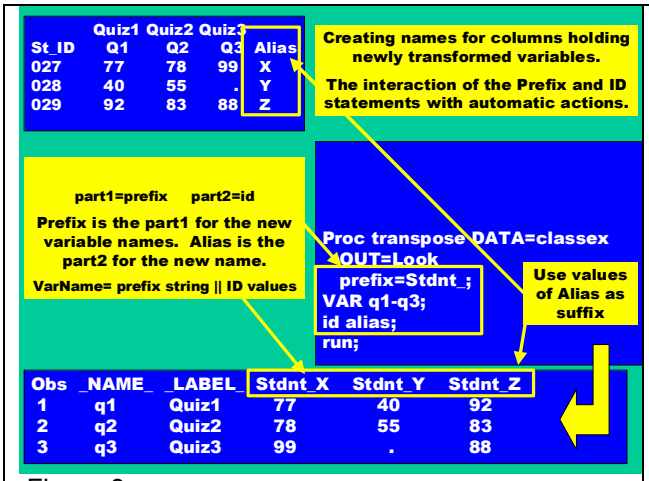

Figure 9

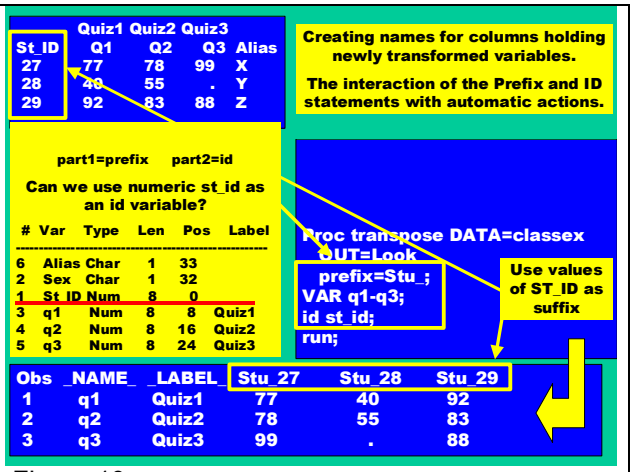

Figure 10

State of nature 4 (in Figure 7) offers great flexibility in creating names. Figures 9 and 10 illustrate two uses of Prefix= and ID combination. If the ID variable is numeric, as is shown in Figure 10, use of a prefix allows the programmer to create names for the new columns that are both valid SAS names (not starting with a number) and preserve the "link to reality". Having Proc Transpose maintain the "link to reality", as opposed to "fixing up the names" in a post-Transpose data step, is efficient and reduces mistakes. Sometimes the values to be used to create labels and names are not in a variable named _LABEL_ (the standard source for label information).

Figure 11 shows how to tell Proc Transpose to get label information from a custom named variable. The IDLABEL statement allows a programmer to tell Proc Transpose to get variable label information from a column s/he chooses and not from a column/variable named _Label_ (_label_ is the default source for label information).

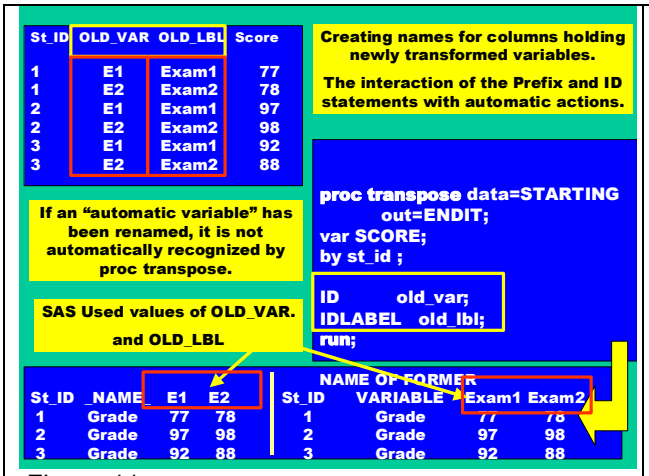

Figure 11

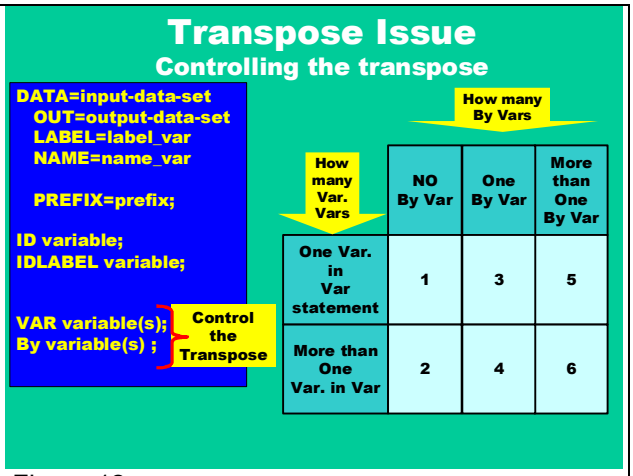

Figure 12

SECTION 4) CONTROLLING THE SHAPE OF THE OUTPUT DATA SET (ROWS AND COLUMNS)

As a Q.C. measure, and a guide to proper coding, it is useful to be able to predict how Proc Transpose statements will change the "shape" of a data set. This ability to predict the output shape helps a programmer at two times – in deciding what statements to code and in checking the results. This section examines how the data set changes shape in response to the BY and VAR statements.

Some rules for predicting the shape of the transpose are:

For a *simple transpose* (an individual transpose):

Rows (in new data set) = columns (in old) & Columns (in new data set) = Rows (in old).

AND: You get a *simple transpose* for each "combination of levels" in variables in the BY statement.

The shape of the transpose is controlled by the interaction of the VAR and BY statements. A programmer can specify either one, or more than one variable in the VAR statement. A programmer can specify none, one, or more than one variable in the BY statement. Combining the ways of specifying the two instructions gives the six states of nature shown in Figure 12. Figure 12 shows the states of nature we must understand. The rules for each of the six states will be developed below. Again; we must examine several states of nature to fully understand the problem.

State of nature one is shown in Figure 13 (and others).

A mental crutch is to imagine laying rectangles over the data set. In the Figures illustrating these six states of nature, rectangles have been drawn over the input data sets to aid in understanding the process of transposing. A rectangle will contain the source data for a simple (an individual) transpose and the "complete" transpose of the data set can be thought of as being comprised of one/several/many of these simple (individual) transposes. Studying the examples below will make these hard-to-read rules more understandable.

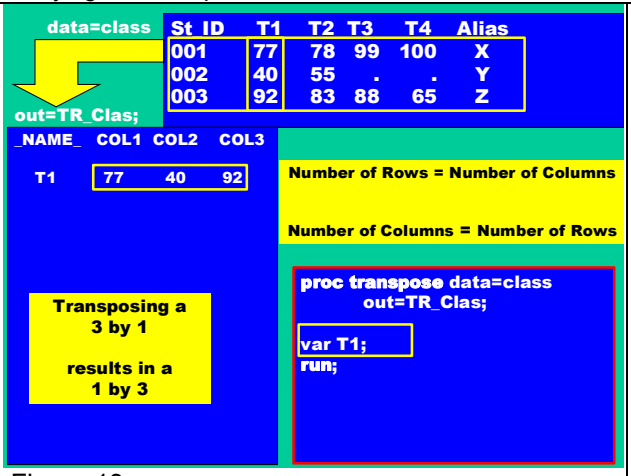

Figure 13

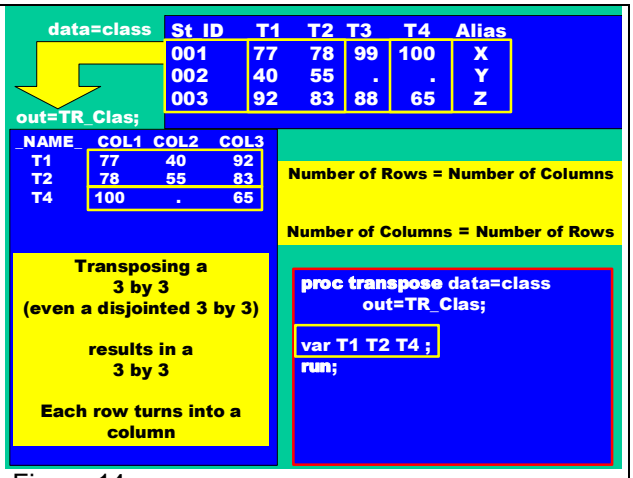

Figure 14

As a mental aid, look at the input data set and draw a rectangle around the variables (colum(s)) in the var statement. Then, draw a rectangle around *one combination of values* of the variable(s) in the BY statement (this selects row(s)). The box where the two rectangles intersect will show the source data for the simple/individual transpose. You get a transpose for every combination of levels of variables in the BY statement (Or, more informally, the stuff in the box gets transposed). The Rows, in the box become columns. Columns in the box, become rows. A mini-transpose happens on the data in the box. The same process happens for every combination of levels in the BY statement (other boxes). If there is no BY statement, make the rectangle enclose all the rows in the data set, as is shown in Figures 16 and 17.

State of nature 1 is shown in Figure 13. Figure 13 gives odd looking results until one realizes that the code is transposing a 3 by 1 matrix, so the output is naturally a 1 by 3 matrix. Remember our rule that there is a transpose within each level of the BY variable. Since there is no BY variable coded, SAS thinks that the whole data set is one level of a BY. In this transpose, a whole column has become one row.

State of nature two is illustrated in Figure 14. In Figure 14 we see a request to transpose a 3 by 3 matrix (t1 t2 t4 can be considered a disjointed 3 by 3 matrix). This will produce a 3 by 3 matrix. There is no By statement in this example, so SAS thinks that the whole data set is one level of the BY variable.

State of nature three is illustrated in Figure 15. Study the boxes layered on top of the input data set. One box is "defined by a level of the BY variable (one value of student ID) and the other box by the variables in the var statement. Where the boxes overlap is the area that will be transposed. In Figure 18 we see a request to transpose a 1 by 1 matrix. This will produce a 1 by 1 matrix for each level of the by variable. (The author found this example quite odd-looking, until realizing that a 1 by 1 trasnsposes into a 1 by 1).
Figure 16 illustrates state of nature four and the use of Proc Transpose on data from Proc Summary (the Proc Summary is not shown). Figure 16 illustrates the effect of multiple variables in the var statement. The code asks for the transpose of a 5 by 2 matrix for each level of the by variable. The output is several 2 by 5 arrays. Proc

Summary and Proc Transpose are a powerful combination.

State of nature five is illustrated in Figure 17. Figure 17 illustrates the effect of multiple variables in the BY statement and one variable in the var statement. A transpose is done for each "combination of levels" of the variables in the BY (a transpose is done for F-11, F-12…..M-15, M16). Each combination of levels of the BY gets a transpose and you can see that the overlapping rectangles define a 1 by 1. Results look odd because the code asks for the transpose of a 1 by 1 matrix, for each combination of levels of variables in the BY. Two examples of using rectangles to define the transpose are shown on Figure 17.

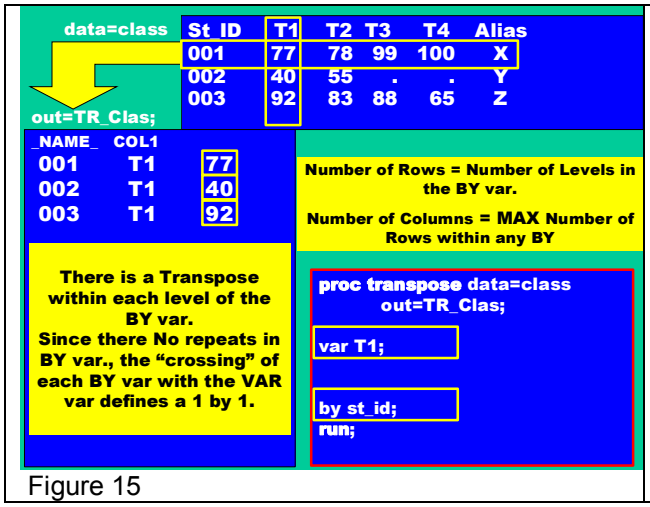

Figure 15

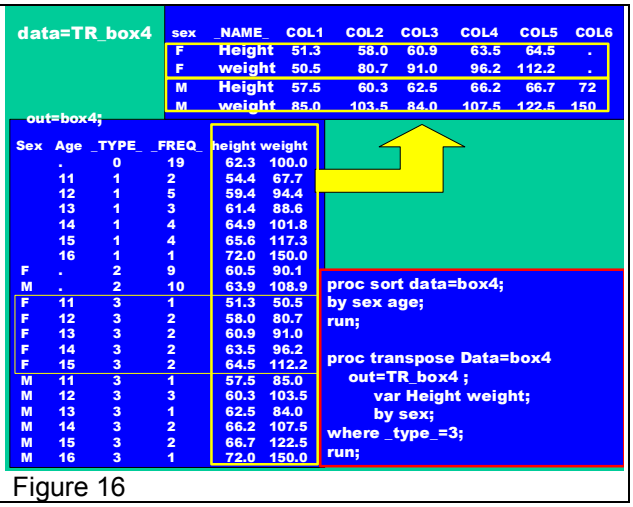

Figure 16

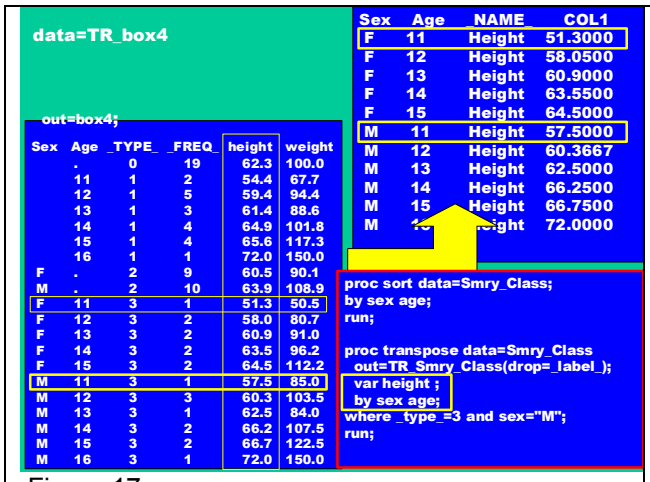

Figure 17

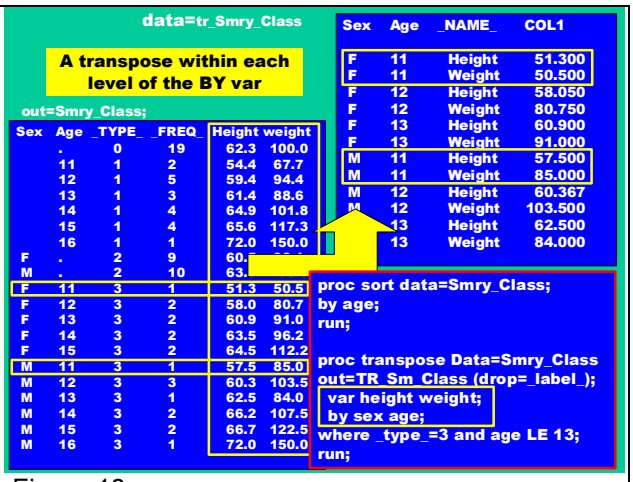

Figure 18

State of nature six is illustrated in Figure 18. Figure 18 illustrates the effect of multiple variables in the BY statement and multiple variables in the var statement. A simple transpose is done for each "combination of levels of the variables in the BY". Because of the data in the input data set, coding "By Sex age;" results in a combination of the By being 1 row. With a different input data set, this same code might result in a combination of the By being more than one row. Each combination of levels in the by gets a transpose and you can see that intersection of boxes in Figure 18 defines a 1 by 2. Thus the code asks for the several transposes of a 1 by 2 matrix – a simple transpose for each "combination of levels" of the variables in the BY. Two examples of using rectangles to define the transpose are shown on Figure 18.

SECTION 5) GETTING COLUMNS IN CORRECT ORDER & FILLING IN MISSING COLUMNS

As Figure 19 shows, when we use ID to name columns, Proc Transpose creates columns *as it encounters values* of the ID variable *as it reads the input data set*. This makes sense because Transpose can not know what columns

should be in the output data set until it reads the input data set. SAS leaves it to us to tell it what to do - by our ordering the data set previous to doing the transpose. In Figure 22, as the data set bad_shape is read, columns are created in this order: Week_5, Week_2, Week_3 and Week_1. If we want the weeks ordered from 1 to 5, it would seem that sorting the data by subject (required for the BY in the transpose) and then also by week would be a logical solution.

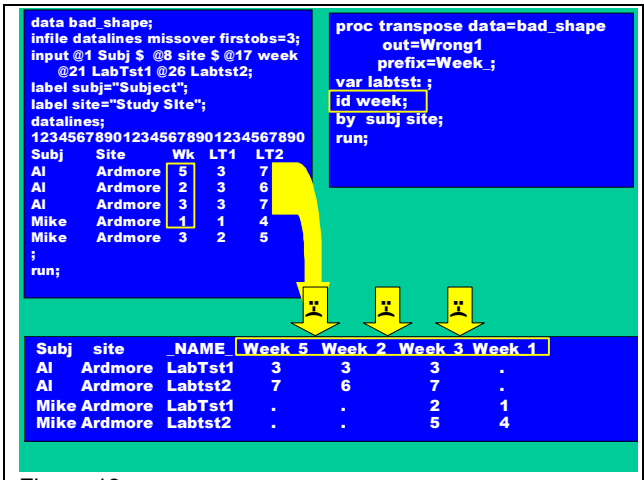

Figure 19

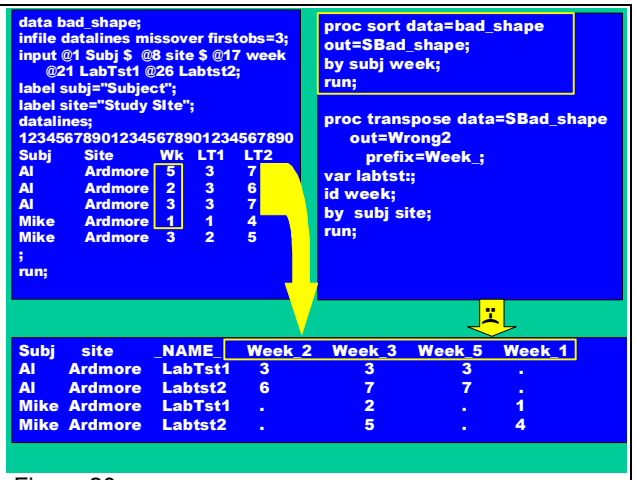

Figure 20

Figure 23 shows that sorting does not solve the problem. In Figure23, Week_1 is not in its proper place and there is no column for week_4. This happens because the first subject does not have all the values of the ID variable and week_1 is only encountered *as data is read from the second subject*. Week_4 is not in the data set at all so there is no week_4 column in the transposed data set. In summary, to get ordered columns from Proc Transpose, the first subject must contain all desired levels of week and contain them in the order in which we want the columns created.

There are several ways to solve this problem and two are shown below.
If we are only interested in having the columns in proper order, the code in Figure 24 would be sufficient. If the data is pushed through a Proc Summary with “completetypes” and “ways” options, Proc Summary will create an output dataset structured so each subject has every level of week that is present in any subject (the logic is: set ways to 3 if there are three class variables-I don’t use NWAYS). This is a bit of overkill, since we are only concerned that the *first* level of the by variable (here the first subject) have every level of the variable used in the ID statement in the transpose that follows. While overkill, it is simple to code.

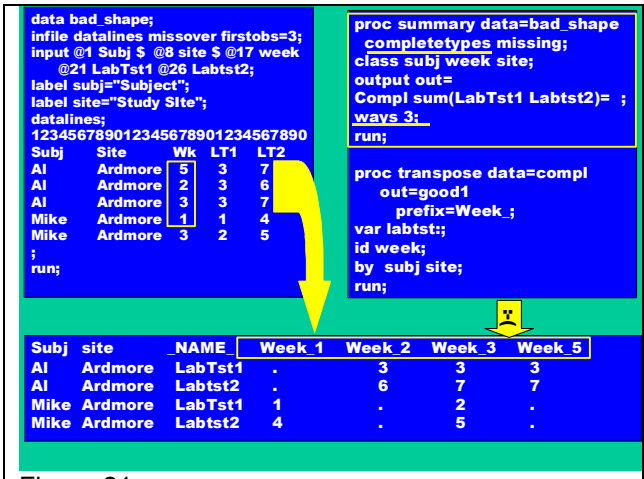

Figure 21

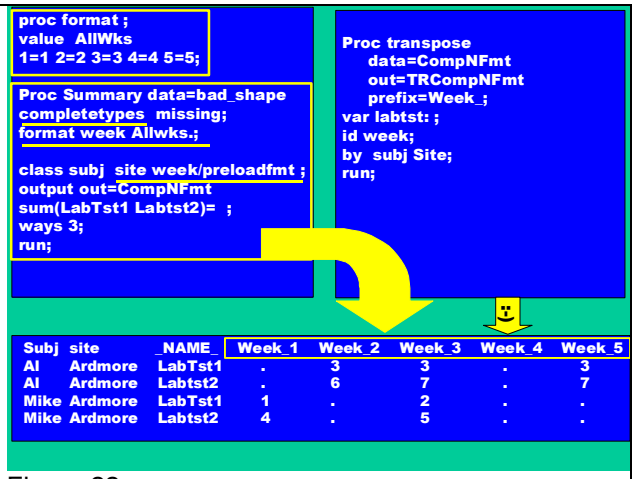

Figure 22

The ouput file in Figure 24 *does* have all the weeks that are *in the data set*, and has them in the proper order, but it does not have a column for week_4. Perhaps the situation is the data for week 4 has been delayed in the lab, but

we still want Week_4 shown in reports to management - to remind management that the data is incomplete. It is common that we desire missing values of the ID variable show up in the transpose output (and resulting reports).

One way to put missing values of the ID variable into the transpose output data set is shown in Figure 25. The first step is to create a format with all the desired values of week. The second step is to push the dataset through a Proc Summary with three options: Format, Completeypes and preloadfmt.
In the Proc Summary Statement, add the completetypes option so ever subject has every week. Add a format statement to apply the format you created to the variable you will use in the ID statement in the Proc Transpose that follows. Finally, the class statement must be changed. After week, in the class statement, add "/preloadfmt". As the author has explored this technique, it seems that the variable for which we desire to have a format preloaded, should be the rightmost variable in the class statement or two class statements should be used. Proc summary supports multiple class statements.

As Proc Summary creates its internal file, it will use the values in the format, not the data lines in the input file, to create lines of data "within" each subject. The appendix to the paper contains code pictured in Figures 24 and 25. Possibly, a reader would be interested in copying the code into SAS and doing their own experiments.

SECTION 6) QUIRKS: MIXED CHAR AND NUM VARIABLES, LET, COPY, BY COMBINED WITH ID

Figure 26 illustrates an unfortunate limitation in the logic of transposing. Using the "rectangle crutch" (not shown in this Figure) a level of the BY and the two var variables would define a 4 by 2 to be transposed. The transpose is a 2 by 4 for each level of st_id. The code asks for a transpose of col1 (a numeric var) and sex (a character var). These variables, after transpose, end up in the same column and are thus BOTH transformed to character (with odd alignments). Often this char/numeric issue forces the programmer to do two Proc Transposes (one to transpose char variables and one to transpose numeric variables) followed by a data set merge.

Figure 27 shows the use of the copy statement. Copy copies one, or more, variables from an input data set to an output data set without transposing them. In Figure 28, a let statement causes the transposing of the observation that contains the LAST OCCURRANCE of a particular ID value within the DATA SET or the BY GROUP.

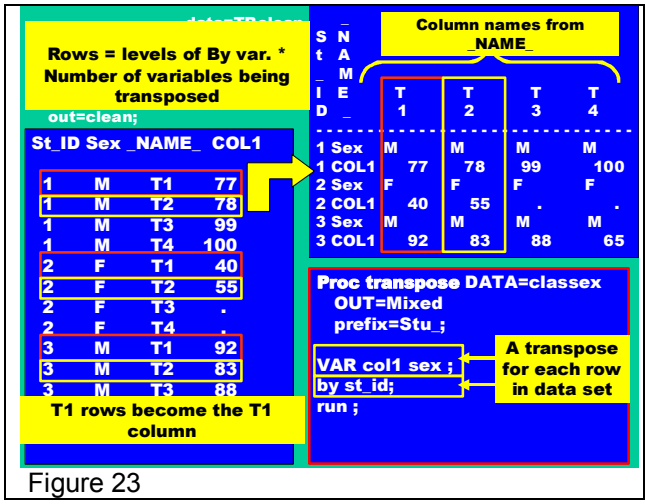

Figure 23

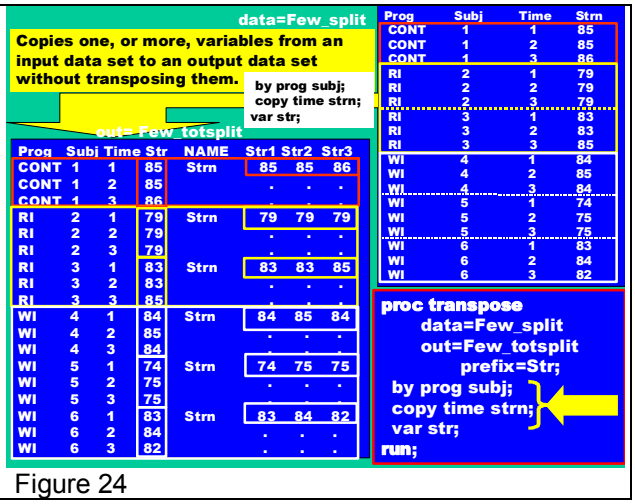

Figure 24

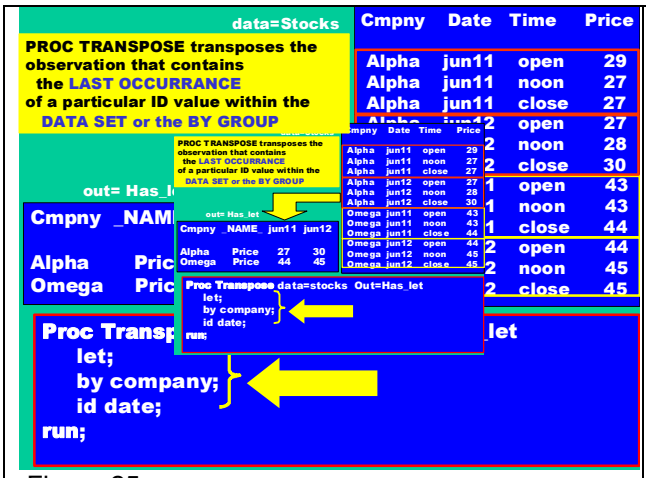

Figure 25

CONCLUSION

Programmers should maintain links to reality when transposing. Proc Transpose has simple rules and is very useful to SAS programmers, especially when used on data sets created by a Proc Summary.

CONTACT INFORMATION

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

Russell Lavery Email: russ@Russ-lavery.com
or c/o Numeric, LLC 5 Christy Drive Bldg.2 Suite 107
Chadds Ford, PA, 19317 (610)642-0700

*****Below, in small font, is an compilation of transpose examples****
All examples in the paper, and more, are illustrated.
A student of transpose can use this code for self study.

```
/*************************************************
Program:              Parmasug_transpose_W_Let_N_Copy
Programmer:           Russ Lavery     Russ@russ-lavery.com
Date Started:
Date Finished:  XXXXXXXXXXXXXXX
Purpose:        Illustrates all of the examples in the SGF paper (243-2007), and more
Modifications:  XXXXXXXXXXXXXXX

*************************************************/
*Example 1;
title "Example 1 the proc means example - why we want to transpose";
title2 "BEFORE TRANSPOSE";
options nocenter nodate pageno=1 linesize=64 pagesize=40;

data class;
infile datalines missover firstobs=2;;
INPUT @1 St_ID  1.
      @8 _name_  $char2.
      @14 _label_ $char5.
      @21 col1 3.;
datalines;
St_ID _name_ _label_ Col1
1      T1    Test1   77
1      T2    Test2   78
1      T3    Test3   99
1      T4    Final  100
2      T1    Test1   40
2      T2    Test2   55
2      T3    Test3    .
2      T4    Final    .
3      T1    Test1   92
3      T2    Test2   83
3      T3    Test3   88
3      T4    Final   65
;
run;

proc print data=class;
title3 "WITHOUT Label option";
run;
proc print data=class LABEL;
title3 "Label option selected"; run;

proc transpose data=class
 out=trclass  /*Output*/
  ;
var Col1     /*vars. to flip*/;
By st_id;    /*grouping var.*/
run;

title2 "AFTER Transpose";
proc print data=TRclass ;
title3 "WITHOUT Label option";run;

title2 "AFTER Transpose";
proc print data=TRclass label;
title3 "Label option selected";run;

Proc Means
data= trclass n nmiss sum;
var T1-T4;
run;
**END of Ex 1*****;

*********Example 2*************************;
* by the way, this is a Double transpose, (hence the D in the file name) ... a transpose of a transpose;
title "Example 2 the proc chart example - why we want to transpose";
title2 "BEFORE TRANSPOSE";
proc print data=Trclass;
title3 "WITHOUT Label option";run;
proc print data=Trclass label;
title3 "Label option selected";run;


proc transpose data=TRclass
 out=D_TRclass  /*Output D stands for double*/
 ;
var T1-T4;                /*vars. to flip*/
By St_ID;           /*grouping var.*/
run;
```

```
title2 "AFTER Transpose";
proc print data=D_TRclass;
title3 "WITHOUT Label option";run;
proc print data=D_TRclass label;
title3 "Label option selected";run;

proc chart data=D_TRclass;
VBAR St_id /group=_name_
                        discrete
                        sumvar=col1;
run;
QUIT;
**END of Ex 2*****;

*************************************************************************;
*Code to support the Aminations of what a transpose does;
data what_do;
infile datalines missover firstobs=3;
input  @1 name $char4. @5 diast @10 syst;
datalines;
NAME Diast      Syst
12345678901234567890123456789O
Bob 68   105
Bob 67   105
Bob 66   104
Sue 77   110
Sue 79   110
Sue 80   112
Al  84   123
Al  86   121
Al  83   123
;
run;

***************;
title "this is a simple 9 by 1 being put into a 1 by 9";
proc transpose data=what_do out=nineby1;
var diast;
run;
proc print data=nineby1;run;
title "";

***************;
title "this is a  9 by 2 being put into a 2 by 9";
title2 "The order of the variables in the var statement is iomportant";
title3 "The first var in the var statement ends up on top";
proc transpose data=what_do out=nineby2;
var diast syst;
run;
proc print data=nineby2;run;
title "";

title "this is a  9 by 2 being put into a 2 by 9";
title2 "The order of the variables in the var statement is iomportant";
title3 "The first var in the var statement ends up on top";
proc transpose data=what_do out=nineby2rev;
var  syst diast ;
run;

proc print data=nineby2rev;run; title "";

*********************************;
proc sort data=what_do; by name; run;

title "this is a  9 by 1 being put into a three 1 by 3s";
title2 "The order of the variables in the var statement is iomportant";
title3 "The first var in the var statement ends up on top";
proc transpose data=what_do out=three3by1;
var  diast ;
by name;
run;

proc print data=three3by1;
run;
title "";

*********************************;

title "this is a  9 by 2 being put into a three 2 by 3s";
title2 "The order of the variables in the var statement is iomportant";
title3 "The first var in the var statement ends up on top";
proc transpose data=what_do out=three3by2;
var  syst diast ;
by name;
run;

proc print data=three3by2;
run;
title "";


**********Example 3*************************;
title "Example 3";
title2 "BEFORE TRANSPOSE";
proc sql;
alter table TRclass
drop _name_;
quit;

proc print data=TRClass;
title3 "WITHOUT Label option";
run;
proc print data=TRClass label;
title3 "Label option selected";
run;

proc transpose data=TRclass
 out=forPres(drop=_name_ _label_)
      ;

var T1-T4;
run;

title2 "AFTER Transpose";
proc print noobs data=forPres;
title3 "WITHOUT Label option";
run;
proc print noobs data=forPres label;
title3 "Label option selected";
run;
**END of Ex 3*****;

**********Example 4*************************;
title "Example 4";
title2 "BEFORE TRANSPOSE";

proc print data=forpres;
title3 "WITHOUT Label option";
run;
proc print data=forpres label;
title3 "Label option selected";
run;

proc transpose data=forpres
```

```
      out=Class1
    ;
var col1-col3;
run;

title2 "AFTER Transpose";
proc print data=class1 noobs;
title3 "WITHOUT Label option";
run;

proc print data=class1 noobs label;
title3 "Label option selected";
run;

**END of Ex 4*****;

**********Example 5************************;
Title "Example 5";
title2 "BEFORE TRANSPOSE";
data Post;
infile datalines firstobs=2;
input @1 St_ID $char3.
      @6  T1 3.
      @10 T2 3.
      @14 T3 3.
      @20 T4 3.
      @25 Alias $char1.;
label t1="Test 1";
label t2="Test 2";
label t3="Test 3";
label t4="Final";
label Alias="posting name";
datalines;
St_ID T1  T2  T3    T4 Alias
001   77  78  99   100  X
002   40  55   .     .   Y
003   92  83  88    65  Z
;
RUN;
proc print data=post ;
title3 "WITHOUT Label option"; run ;
proc print data=post Label;
title3 "Label option selected"; run ;

proc transpose data=Post
       out=TRPost
    ;
var T1-T4;
run;

title2 "AFTER Transpose";
proc print noobs data= TRPost ;
title3 "WITHOUT Label option";
run;
proc print noobs data= TRPost label;
title3 "Label option selected";run;

**END of Ex 5*****;

**********Example 6************************;
title "Example 6";
title2 "BEFORE TRANSPOSE";

proc print data=TRPost;
title3 "WITHOUT Label option";
run;
proc print data=TRPost label;
title3 "Label option selected";
run;

proc transpose data=TRPost
       out=DTRPost
    ;
var col1-col3; run;



TITLE2 "AFTER Transpose";
proc print data=DTRPost;
title3 "WITHOUT label option";
run;
proc print data=DTRPost Label;
title3 "Label option selected";
run;

**END of Ex 6*****;

*******************************************************************************;
      End of intro section
*******************************************************************************;

**********Example 7************************;
title "Example 7";
title2 "BEFORE TRANSPOSE";
proc print data=post ;
title3 "WITHOUT Label option";
run;
proc print data=post label;
title3 "Label option selected";
run;

Proc transpose DATA=Post
   OUT=TR2Post
  LABEL=Old_LBL
  NAME=OLD_VAR

   PREFIX=N_VR;

*ID variable;
*IDLABEL variable;

VAR t1-t4;
By ST_ID ;
run;

title2 "AFTER Transpose";
proc print data=TR2Post ;
title3 "WITHOUT label option";
run;
proc print data=TR2Post label;
title3 "Label option selected";
run;

**END of Ex 7*****;
*******************************************************************************;
**Start of  Data Prep A*****;
data NoAutNamVar ; /*no automatic naming variables*/
infile datalines firstobs=3;
input @1 St_ID 3.0  @1 CS $char3.
      @8   Q1 3.
      @16  Q2 3.
      @24  Q3 3.
      @31  Alias $char1.;
Label Q1="Quiz 1";
Label Q2="Quiz 2";
```

```
Label Q3="Quiz 3";
format st_id Z3.;
datalines;
      Quiz1  Quiz2   Quiz3
St_ID   Q1     Q2      Q3  Alias
027     77     78      99    X
028     40     55       .    Y
029     92     83      88    Z
;
run;
proc print data=NoAutNamVar;
title "Data Prep A - data set NoAutNamVar";
title2 "NO Automatic Naming variables";
title3 "WITHOUT label option";
run;

proc print data=NoAutNamVar label;
title "Data Prep A - data set NoAutNamVar";
title2 "NO Automatic Naming variables";
title3 "Label option selected";
run;

data HasAutNamVar ; /*HAS automatic naming variables*/
infile datalines firstobs=2;
input @1 St_ID      3.
      @12  _name_   $char3.
      @19  _label_  $char6.
      @32  Score    3.
;
format st_id Z3.;
datalines;
St_ID    _NAME_  _LABEL_     Score
1          HW1    HmWrk1        77
1          HW2    HmWrk2        78
2          HW1    HmWrk1        97
2          HW2    HmWrk2        98
3          HW1    HmWrk1        92
3          HW2    HmWrk2        88
;
proc print data=HasAutNamVar;
title "Data Prep A - data set HasAutNamVar";
title2 "HAS Automatic Naming variables _name_ and _Label_";
title3 "WITHOUT Label option";
run;
proc print data=HasAutNamVar label;
title "Data Prep A - data set HasAutNamVar";
title2 "HAS Automatic Naming variables _name_ and _Label_";
title3 "Label option selected";
run;

data HasCustNamVar ; /*has Custom named naming variables*/
infile datalines firstobs=2;
input @1 St_ID       3.
      @11  Old_var   $char2.
      @18  Old_LBL   $char5.
          @27  Grade    3.
;
format st_id Z3.;
datalines;
St_ID  OLD_VAR  OLD_LBL  Grade
1         E1      Exam1     77
1         E2      Exam2     78
2         E1      Exam1     97
2         E2      Exam2     98
3         E1      Exam1     92
3         E2      Exam2     88
;
run;
proc print data=HasCustNamVar;
title "Data Prep A - data set HasCustNamVar";
title2 "Has CUSTOM NAMED Naming variables ";
title3 "Info for new columns is in file but not in _name_ or _Label_";
title4 "WITHOUT Label option";
run;

proc print data=HasCustNamVar;
title "Data Prep A - data set HasCustNamVar";
title2 "Has CUSTOM NAMED Naming variables ";
title3 "Info for new columns is in file but not in _name_ or _Label_";
title4 "Label option selected";
run;

**END of Data Prep A*****;


*********Example 8************************;
title "Example 8";
title2 "BEFORE TRANSPOSE";

proc print data=NoAutNamVar ;
title3 "WITHOUT Label option";
run;
proc print data=NoAutNamVar label;
title3 "Label option selected";
run;

proc transpose  DATA=NoAutNamVar
   OUT=TRNoAuNmVar;
VAR q1-q3;
run;

title2 "AFTER Transpose";
proc print data=TRNoAuNmVar;
title3 "WITHOUT label option";
run;

proc print data=TRNoAuNmVar Label;
title3 "Label option selected";
run;
**END of Example 8*****;
*********Example 9************************;
title "Example 9";
title2 "BEFORE TRANSPOSE";

proc print data=NoAutNamVar;
title3 "WITHOUT Label option";
run;
proc print data=NoAutNamVar label;
title3 "Label option selected";
run;

proc transpose
    DATA= NoAutNamVar
   OUT= TRNoAuNmVar2;
VAR q1-q3;
ID alias;
run;
title2 "AFTER Transpose";

proc print data=TRNoAuNmVar2;
title3 "WITHOUT Label opton";
run;

proc print data=TRNoAuNmVar2;
title3 "Label option selected";
```

```
run;
**END of Example 9*****;
**********Example 10************************;
title "Example 10";
title2 "BEFORE TRANSPOSE";
proc print data=NoAutNamVar;
title3 "WITHOUT label option";
run;
proc print data=NoAutNamVar label;
title3 "Label option selected";
run;

Proc transpose
    DATA= NoAutNamVar
      OUT= TRNoAuNmVar3
prefix=st_no_;
VAR q1-q3;
run;

title2 "AFTER Transpose";
proc print data=TRNoAuNmVar3;
title3 "WITHOUT label option";
run;
proc print data=TRNoAuNmVar3 label;
title3 "Label option selected";
run;

**END of Example 10 *****;


**********Example 11************************;
title "Example 11";
title2 "BEFORE TRANSPOSE";

proc print data=NoAutNamVar;
title3 "WITHOUT Label option";
run;
proc print data=NoAutNamVar label;
title3 "Label option selected";
run;

Proc transpose
  DATA= NoAutNamVar
  OUT= TRNoAuNmVar4
  prefix=Stdnt_;
       VAR q1-q3;
id alias;
run;

title2 "AFTER Transpose";
proc print data=TRNoAuNmVar4;
title3 "WITHOUT label option";
run;

proc print data=TRNoAuNmVar4 label;
title3 "Label option selected";
run;

**END of Example 11 *****;
**********Example 12************************;
proc contents data=NoAutNamVar;
run;

title "Example 12";
title2 "BEFORE TRANSPOSE";
proc print data=NoAutNamVar;
title3 "WITHOUT label option";
run;
proc print data=NoAutNamVar label;
title3 "Label option selected";
run;

Proc transpose DATA=NoAutNamVar
   OUT=TRNoAuNmVar5
   prefix=Stu_;
VAR q1-q3;
id st_id;
run;

title2 "AFTER Transpose";
proc print data=TRNoAuNmVar5;
title3 "WITHOUT Label option";
run;
proc print data=TRNoAuNmVar5;
title3 "Label option selected";
run;

**END of Example 12 *****;
**********Example 13************************;
title "Example 13";
title2 "BEFORE TRANSPOSE";

proc print data=HasAutNamVar;
title3 "WITHOUT Label option";
TITLE4 "-----THIs WILL PRODUCE AN ERROR MESSAGE------";
run;
proc print data=HasAutNamVar;
title3 "Label option selected";
TITLE4 "-----THIs WILL PRODUCE AN ERROR MESSAGE------";
run;

*this will produce an error message;
proc transpose data=HasAutNamVar
       out=EX13;
var SCORE;
run;
title2 "AFTER Transpose";
proc print data=EX13;
run;

**END of Example 13 *****;


**********Example 14************************;
title "Example 14";
title2 "BEFORE TRANSPOSE";
proc print data=HasAutNamVar;
title3 "WITHOUT Label option";
run;

proc print data=HasAutNamVar;
title3 "Label option selected";
run;

proc transpose data=HasAutNamVar
       out=EX14;
var SCORE;
by st_id;
run;

title2 "AFTER Transpose";
proc print data=EX14;
title3 "WITHOUT Label option";
run;
proc print data=EX14 Label;
```

```
title3 "Label option selected";
run;

**END of Example 14 *****;
**********Example 15************************;
title "Example 15";
title2 "BEFORE TRANSPOSE";
proc print data=HasCustNamVar;
title3 "WITHOUT Label option";
run;
proc print data=HasCustNamVar label;
title3 "Label option selected";
run;

proc transpose data=HasCustNamVar
      out=EX15;
var GRADE;
by st_id ;
run;

title2 "AFTER Transpose";
proc print data=EX15 ;
title3 "WITHOUT Label option";
run;

proc print data=EX15 label;
title3 "Label option selected";
run;
**END of Example 15 *****;
**********Example 16************************;
title "Example 16";
title2 "BEFORE TRANSPOSE";
proc print data=HasCustNamVar;
title3 "WITHOUT Label option";
run;
proc print data=HasCustNamVar Label;
title3 "Label option selected";
run;

proc transpose data=HasCustNamVar
      out=EX16;
var GRADE;
ID          old_var;
IDLABEL   old_lbl;
by st_id ;
run;

title2 "AFTER Transpose";
proc print data=EX16 ;
title3 "WITHOUT Label option";
run;
proc print data=EX16 label;
title3 "Label option selected";
run;
**END of Example 16 *****;

*************************************************************************;
**Start of  Data Prep B*****;
title "Data Prep B";
data CtrThTr; /* Controlling the transform*/
infile datalines missover firstobs=2;
input  @1 St_ID
       @10 T1 3.
       @17 T2 3.
       @22 T3 3.
       @28 T4 3.
       @37 Alias $char1.
 ;
datalines;
St_ID      T1      T2    T3    T4     Alias
001        77      78    99    100       X
002        40      55     .     .        Y
003        92      83    88    65        Z
;
run;
title2 "Data set for later use";
proc print data=CtrThTr;
title3 "WITHOUT Label option";
run;
proc print data=CtrThTr label;
title3 "Label option selected";
run;
title ;
**END  Data Prep B*****;

**********Example 17************************;
title "Example 17";
title2 "BEFORE TRANSPOSE";
proc print data=CtrThTr;
title3 "WITHOUT Label option";
run;
proc print data=CtrThTr Label;
title3 "Label option selected";
run;

proc transpose data=CtrThTr
      out=Ex17;
  by st_id;
 var T1;
run;

title2 "AFTER Transpose";
proc print data=Ex17;
title3 "WITHOUT Label option";
run;
proc print data=Ex17 label label;
title3 "Label option selected";
run;
**END of Example 17 *****;
**********Example 18************************;
title "Example 18";
title2 "BEFORE TRANSPOSE";
proc print data=CtrThTr;
title3 "WITHOUT Label option";
run;
proc print data=CtrThTr label;
title3 "Label option selected";
run;

proc transpose data=CtrThTr
      out=EX18;
by st_id;
var T1-T4;
run;

title2 "AFTER Transpose";
proc print data=Ex18;
titLE3 "WITHOUT Label option";
run;

proc print data=Ex18 label;
titLE3 "Label option selected";
run;
**END of Example 18 *****;
**********Example 19************************;
```

```
title "Example 19";
title2 "BEFORE TRANSPOSE";
data mult; /*multiple lines per subject*/
infile datalines missover firstobs=2;
input @1 Pat $char3.
      @13 Visit 1.
      @17 P1 3.
      @23 P2 3.
      @29 P3 3.
      @36 P4 3.
      @42 P5 3.   ;
datalines;
Pat      Visit   P1    P2    P3    P4     P5
001        1     77    78    99    100    90
001        2     47    55     .     53    60
002        1     92    83    88     75    74
002        2     90    80    80     70    75
;
run;

proc print data=mult;
title3 "WITHOUT Label Option";
run;
proc print data=mult label;
title3 "Label option selected";
run;

proc transpose data=mult
       out=EX19;
by pat visit;
var P1;
run;

title2 "AFTER Transpose";
proc print data=EX19;
title3 "WITHOUT label option";
run;
proc print data=EX19 label;
title3 "Label option selected";
run;
**END of Example 19 *****;
**********Example 20************************;
title "Example 20";
title2 "BEFORE TRANSPOSE";
proc print data=mult;
title3 "WITHOUT Label option";
run;
proc print data=mult label;
title3 "Label option selected";
run;

proc transpose data=mult
       out=EX20;

by pat visit;

var P1-P4;
run;

title2 "AFTER Transpose";
proc print data=EX20;
title3 "WITHOUT Label option";
run;

proc print data=EX20 label;
title3 "Label option selected";
run;

**END of Example 20 *****;

*******************************************************************************;
**********Example 21************************;
title "Example 21";
title2 "BEFORE TRANSPOSE - not student 2 has only two tests";
data for_Ex21;
infile datalines missover firstobs=2;
input @1  St_ID 3. @11 Sex $char1.
      @19 _NAME_ $char2. @27 COL1 3.  ;
datalines;
St_ID   Sex     _NAME_  COL1
 1       M       T1      77
 1       M       T2      78
 1       M       T3      99
 1       M       T4     100
 2       F       T1      40
 2       F       T2      55
 3       M       T1      92
 3       M       T2      83
 3       M       T3      88
 3       M       T4      65
;
run;

proc print data=for_Ex21;
title3 "WITHOUT Label option";
run;
proc print data=for_Ex21 label;
title3 "Label option selected";
run;

proc transpose data=For_ex21
       out=Ex21;
by  st_id SEX;
var col1 ;
run;

title2 "AFTER First Transpose";
proc print data=Ex21;
title3 "WITHOUT Label option"; run;
proc print data=Ex21 label;
title3 "Label option selected";run;

proc transpose data=Ex21
       out=D_Ex21;
by  st_id SEX;
var t1-t4 ;
run;

title2 "AFTER second Transpose note student 2 has 4 obs";
proc print data=D_Ex21;
title3 "WITHOUT Label option";
run;
proc print data=D_Ex21 label;
title3 "Label option selected"; run;

**END of Example 21 *****;


**********Example 22************************;
title "Example 22";
title2 "BEFORE TRANSPOSE";
data for_Ex22;
infile datalines missover firstobs=2;
input @1  St_ID 3. @11 Sex $char1.
      @19 _NAME_ $char2. @27 COL1 3.  ;
```

```
datalines;
 St_ID   Sex      _NAME_   COL1
  1       M       T1        77
  1       M       T2        78
  1       M       T3        99
  1       M       T4       100
  2       F       T1        40
  2       F       T3         .
  2       F       T4         .
  2       F       T2        55
  3       M       T1        92
  3       M       T2        83
  3       M       T3        88
  3       M       T4        65
;
run;

proc print data=for_Ex22;
title3 "WITHOUT Label option"; run;
proc print data=for_Ex22 label;
title3 "Label option selected"; run;

Proc transpose DATA=For_ex22
   OUT=Ex22;

VAR  sex col1 ;
by st_id;
run ;

title2 "AFTER Transpose";
proc print data=Ex22;
title3 "WITHOUT Label option"; run;
proc print data=Ex22 label;
title3 "Label option selected"; run;

**END of Example 22 *****;
**********Example 23************************;
/*COPY variable(s);
names one or more variables that the COPY statement copies directly
from the input data set to the output data set without transposing them.
-------------------------------------------------------------------------------
Details
Because the COPY statement copies variables directly to the output data set,
  the number of observations in the output data set
  is equal to the number of observations in the input data set.
The procedure pads the output data set with missing values
  if the number of observations in the input data set
  is not equal to the number of variables that it transposes.
*/
title "Example 23 - the copy statement";
title2 "BEFORE TRANSPOSE";

data For_Ex23;
infile datalines missover firstobs=2;
input   @1 Prog $char3.
        @8  Subj  1.
        @14 Time  1.
        @19 Str  2.   ;
datalines;
Prog  Subj  Time  Str
CON    1     1    85
CON    1     2    85
CON    1     3    86
RI     2     1    79
RI     2     2    79
RI     2     3    79
RI     3     1    83
RI     3     2    83
RI     3     3    85
WI     4     1    84
WI     4     2    85
WI     4     3    84
WI     5     1    74
WI     5     2    75
WI     5     3    75
WI     6     1    83
WI     6     2    84
WI     6     3    82
;
run;

proc print data=For_Ex23;
title3 "WITHOUT Label option";
run;
proc print data=For_Ex23 label;
title3 "Label option selected";
run;

proc transpose
data=For_ex23
  out=Ex23
  prefix=St;
  by prog subj;
  copy time str;
  var str;
run;

title2 "AFTER Transpose";
proc print data=Ex23;
title3 "WITHOUT Label option";
run;
proc print data=Ex23 label;
title3 "Label option selected";
run;

**END of Example 23 *****;
**********Example 24************************;
/*LET
allows duplicate values of an ID variable.
PROC TRANSPOSE transposes the observation that contains
the last occurrence of a particular ID value within the data set or BY group
*/

title "Example 24 - the let statement";
title2 "BEFORE TRANSPOSE";
data for_ex24; /*stock prices opening noon and close*/
infile datalines missover firstobs=2;
input @1 Cmpny $char5.  @8 Date $char5.
      @16 Time  $char5. @26  Price 3.;
;
datalines;
Cmpny  Date    Time     Price
Alpha  jun11   open       29
Alpha  jun11   noon       27
Alpha  jun11   close      27
Alpha  jun12   open       27
Alpha  jun12   noon       28
Alpha  jun12   close      30
Omega  jun11   open       43
Omega  jun11   noon       43
Omega  jun11   close      44
Omega  jun12   open       44
Omega  jun12   noon       45
Omega  jun12   close      45
;
```

```
run;
proc print data=for_ex24;
title3 "WITHOUT Label option";
run;
proc print data=for_ex24 label;
title3 "Label option selected";
run;

Proc Transpose data=for_ex24
    Out=Ex24
     let; /********the let *****/
     by cmpny;
     id date;
run;

title2 "AFTER Transpose";
proc print data=ex24;
title3 "WITHOUT label option";
run;
proc print data=ex24 label;
title3 "Label option selected";
run;
**END of Example 24 *****;


********** Example 24 ½  Filling in Missing values with proc summary;
```

*Imagine that this dataset is for a clinical trial that will last 6 weeks;
*data is enteres as it arrives and, for a number of reasons, does not arrive in time order;

```
*data can be delayed several weeks;
options nocenter;
data bad_shape;
infile datalines missover firstobs=2;
input @1 Subj $  @8 site $ @17 week
      @21 LabTst1 @26 Labtst2;
label subj="Subject";
label site="Study SIte";
datalines;
12345678901234567890123456789O
Al     Ardmore  5   3    7
Al     Ardmore  2   3    6
Al     Ardmore  3   3    7
Mike   Ardmore  1   1    4
Mike   Ardmore  3   2    5
;
run;

proc print data=bad_shape; run;

proc transpose data=bad_shape  out=Wrong1
       prefix=Week_;
var labtst: ;
id week;
by  subj site;
run;

proc print data=wrong1;
title "Columns out of order"; run;
*************************************************************;
proc sort data=bad_shape out=SBad_shape;
by subj week;       run;

proc transpose data=SBad_shape out=Wrong2
       prefix=Week_;
var labtst:;
id week;
by  subj site;
run;

proc print data=wrong2;
title "Sorting does NOT solve the problem";  run;
*************************************************************;
proc summary data=bad_shape
  completetypes missing;
class subj week site;
output out=Compl sum(LabTst1 Labtst2)=  ;
ways 3;
run;

proc transpose data=compl
      out=good1
        prefix=Week_;
var labtst:;
id week;
by  subj site;
run;


proc print data=good1;
title "columns in order but weeks are missing";  run;
*************************************************************;
proc format ;
value  AllWks   1=1
                2=2
                3=3
                4=4
                5=5;
run;

proc format ;
value  Readabl  1="Screen"
                2="Baseline"
                3="test1"
                4="test2"
                5="followup";
run;

proc summary data=bad_shape completetypes  missing;
format week AllWks.;
class subj  site week/preloadfmt ;
output out=CompNFmt1 sum(LabTst1 Labtst2)=  ;
ways 3;
run;

proc print data=CompNFmt1;
title 'filling in the missing week'; run;

 proc transpose data=CompNFmt1
     out=TRCompNFmt1;
var labtst:;
id week;
by  subj Site;
run;

proc print data=TRCompNFmt1;
title 'missing weeks filled in and then transposed';   run;
   /*Example 2 apply another format to the same problem*/
proc summary data=bad_shape completetypes  missing;
format week Readabl.;
class subj  site week/preloadfmt ;
output out=CompNFmt2 sum(LabTst1 Labtst2)=  ;
ways 3;
run;
```

```
proc print data=CompNFmt2;   run;

proc transpose data=CompNFmt2
     out=TRCompNFmt2;
var labtst:;
id week;
by  subj Site;
run;

proc print data=TRCompNFmt; run;
***Example 24 ½ ;-)  filling in missing columns with proc summary;
*data can be delayed several weeks;
options nocenter;
data bad_shape;
infile datalines missover firstobs=2;
input @1 Subj $  @8 site $ @17 week
      @21 LabTst1 @26 Labtst2;
label subj="Subject";
label site="Study SIte";
datalines;
12345678901234567890123456789O
Al     Ardmore  5   3    7
Al     Ardmore  2   3    6
Al     Ardmore  3   3    7
Mike   Ardmore  1   1    4
Mike   Ardmore  3   2    5
;
run;

proc print data=bad_shape;  run;

proc transpose data=bad_shape  out=Wrong1
       prefix=Week_;
var labtst: ;
id week;
by  subj site;
run;

proc print data=wrong1;
title "Columns out of order";    run;
*************************************************************;
proc sort data=bad_shape out=SBad_shape;
by subj week;      run;

proc transpose data=SBad_shape out=Wrong2
       prefix=Week_;
var labtst:;
id week;
by  subj site;
run;

proc print data=wrong2;
title "Sorting does NOT solve the problem";  run;
*************************************************************;
proc summary data=bad_shape
  completetypes missing;
class subj week site;
output out=Compl sum(LabTst1 Labtst2)=  ;
ways 3;
run;

proc transpose data=compl
     out=good1
       prefix=Week_;
var labtst:;
id week;
by  subj site;
run;


proc print data=good1;
title "columns in order but weeks are missing"; run;
*************************************************************;
proc format ;
value  AllWks   1=1
                2=2
                3=3
                4=4
                5=5;
run;

proc format ;
value  Readabl  1="Screen"
                2="Baseline"
                3="test1"
                4="test2"
                5="followup";
run;

proc summary data=bad_shape completetypes  missing;
format week AllWks.;
class subj  site week/preloadfmt ;
output out=CompNFmt1 sum(LabTst1 Labtst2)=  ;
ways 3;
run;

proc print data=CompNFmt1;
title 'filling in the missing week'; run;

 proc transpose data=CompNFmt1
     out=TRCompNFmt1;
var labtst:;
id week;
by  subj Site;
run;

proc print data=TRCompNFmt1;
title 'missing weeks filled in and then transposed';
run;
  /*Example 2 apply another format to the same problem*/
proc summary data=bad_shape completetypes  missing;
format week Readabl.;
class subj  site week/preloadfmt ;
output out=CompNFmt2 sum(LabTst1 Labtst2)=  ;
ways 3;
run;

proc print data=CompNFmt2; run;

proc transpose data=CompNFmt2
     out=TRCompNFmt2;
var labtst:;
id week;
by  subj Site;
run;

proc print data=TRCompNFmt;  run;


*************************************************************************************;
********** data step transposing ****************************************************;
***Example 25********;
proc contents data=What_do;run;

Data Nineby10;
```

```
set What_do;
length Vname $ 5;
array DstAry d1-d9; /* array of diastolic values*/
retain VName "Diast"  ;
DstAry(_N_)=diast;
run;

Proc Print data=Nineby10;run;

Proc summary data= Nineby10 missing;
class VName;
output out=EX25_ONeby10 sum(d:)=;
run;

Proc Print data=EX25_ONeby10; run;

****Example 26 **************;
Data Nineby1;
set What_do end=file_done;
length Vname $ 5;
array DstAry d1-d9; /* array of diastolic values*/
retain VName "Diast" d: ;
DstAry(_N_)=diast;
if file_done then ouptut;
run;

Proc Print data=Nineby1; run;


*********Example 27 ********************;
Proc sort data=what_do;
by name;run;

proc print data=what_do;
title "";run;

Data TranspAndOrder/*(keep=Subj D1 D2 D3 S1 S2 S3)*/;
set What_do;
BY NAME;
retain d1 S1 D2 S2 D3 S3 ;
counter+1;
subj=Name;
array DstAry d1-d3;
array SysAry S1-S3;
DstAry(COUNTER)=diast;
SysAry(COUNTER)=syst;

If last.name=1 then
        DO;
        output;
        COUNTER=0;
        END;
run;

Proc Print data=TranspAndOrder;run;


*****Example 28***********************************;
data Starbucks ;
infile datalines missover firstobs=3;
input @1  store $char11.
      @13  qrtr  $char2.
      @20 food
      @25 drink;
datalines;
Store      Qrter  Food Drink
12345678901234567890123456789012345678901234567890
Bryn Mawr   Q1     68   105
Bryn Mawr   Q2     72   125
Bryn Mawr   Q3     78   135
Bryn Mawr   Q4     82   146
Haverford   Q1     66   104
Haverford   Q2     77   144
Bala Cynwyd Q1     65   110
Bala Cynwyd Q2     80   112
;
run;

title "Starbucks data";
title2 "before Transpose";
proc print data=Starbucks;run;

proc sort data=starbucks noequals;by store ;run;

data onelineperstore;
set starbucks;
by store;
array Af(4)  FQ1-FQ4 (0 0 0 0);
array Ad(4)  DQ1-DQ4 (0 0 0 0);
array At(4)  TQ1-TQ4 (0 0 0 0);
if qrtr="Q1" then
         do;
           FQ1+food;
           DQ1+Drink;
           TQ1+food+drink;
         end;
Else if qrtr="Q2" then
         do;
           FQ2+food;
           DQ2+drink;
           TQ2+food+drink;
         end;
Else if qrtr="Q3" then
         do;
           FQ3+food;
           DQ3+drink;
           TQ3+food+drink;
         end;
Else if qrtr="Q4" then
         do;
           FQ4+food;
           DQ4+drink;
           TQ4+food+drink;
         end;
if last.store then
    do;
        output;
        do i=1 to 4;
           af(i)=0;
           ad(i)=0;
           at(i)=0;
        end;
    end;
run;


run;

proc print data=onelineperstore;run;

%macro zro(arraylist= af ad at, max=);
   do i= 1 to &max;

       %let counter=1;
       %do %while (%scan(&arraylist, &counter) NE );
```

```
        %let thisarray=%scan(&arraylist, &counter);
        &thisarray.(i)=0;
        %let counter=%eval(&counter+1);
      %end;
  end;
%mend;


options mlogic mprint mfile symbolgen;
  filename mprint "C:\temp\delete_me.txt";

%zro(arraylist= af ad at, max=4);
OPTIONS NOMFILE;

*****Example 29***********************************;
data Starbucks2 ;
infile datalines missover firstobs=3;
input @1  store $char11.
      @13  qrtr  $char2.
      @20 food
      @25 drink;
datalines;
Store      Qrter  Food Drink
12345678901234567890123456789012345678901234567890
Bryn Mawr   Q1     38   105
Bryn Mawr   Q1     48   105
Bryn Mawr   Q1     58   105
Bryn Mawr   Q1     68   105
Bryn Mawr   Q2     32   125
Bryn Mawr   Q2     42   125
Bryn Mawr   Q2     52   125
Bryn Mawr   Q2     32   125
Bryn Mawr   Q3     48   135
Bryn Mawr   Q3     58   135
Bryn Mawr   Q3     68   135
Bryn Mawr   Q3     78   135
Bryn Mawr   Q4     32   146
Bryn Mawr   Q4     42   146
Bryn Mawr   Q4     52   146
Bryn Mawr   Q4     62   146
Haverford   Q1     36   104
Haverford   Q1     36   104
Haverford   Q2     47   144
Haverford   Q2     57   144
Haverford   Q2     37   144
Haverford   Q3     47   144
;
run;

title "Starbucks2 data";
title2 "before Transpose";
proc print data=Starbucks2;run;

proc sort data=starbucks2 noequals;by store ;run;

%MACRO BYSTORE(StrNme=
              ,StrAbrv=
              ,TimeAbrv=Q
              ,MaxPer=4
              );

 array F&StrAbrv.(&MaxPer.) F&StrAbrv.1-F&StrAbrv&MaxPer (%do loop=1 %to &maxPer;
                                                            %str(0 )
                                                          %end;
                                                          );
 array D&StrAbrv.(&MaxPer.) D&StrAbrv.1-D&StrAbrv&MaxPer (%do loop=1 %to &maxPer;
                                                            %str(0 )
                                                           %end;
                                                          );
 array T&StrAbrv.(&MaxPer.) T&StrAbrv.1-T&StrAbrv&MaxPer (%do loop=1 %to &maxPer;
                                                            %str(0 )
                                                            %end;
                                                         );

 if store="&StrNme" then
   do;
      if qrtr="&TimeAbrv.1" then
         do;
           F&StrAbrv&TimeAbrv.1+food;
           D&StrAbrv&TimeAbrv.1+Drink;
           T&StrAbrv&TimeAbrv.1+food+drink;
         end;
    %do i=2 %to &maxper;
      Else if qrtr="&TimeAbrv.&i" then
         do;
           F&StrAbrv&TimeAbrv.&i+food;
           D&StrAbrv&TimeAbrv.&i+Drink;
           T&StrAbrv&TimeAbrv.&i+food+drink;
         end;
    %end;
    end;
%mend Bystore;

%macro ZroStr(StrNme=
              ,StrAbrv=
              ,TimeAbrv=Q
              ,MaxPer=4
              );
    %do i=1 %to &MaxPer;
           F&StrAbrv&TimeAbrv.&i=0;
           D&StrAbrv&TimeAbrv.&i=0;
           T&StrAbrv&TimeAbrv.&i=0;
    %end;
%mend ZroStr;


data onelineperstore;
set starbucks;
by store;

%BYSTORE(StrNme=Bryn Mawr
        ,StrAbrv=Bryn
        ,TimeAbrv=Q
        ,MaxPer=4
        );
%BYSTORE(StrNme=Haverford
        ,StrAbrv=Hav
        ,TimeAbrv=Q
        ,MaxPer=4
        );



if last.store then
    do;
        output;
        do i=1 to 4;
           af(i)=0;
           ad(i)=0;
           at(i)=0;
        end;
    end;
run;
```

```
run;

proc print data=onelineperstore;
run;

%BYSTORE(StrNme=
        ,StrAbrv=
        ,TimeAbrv=Q
        ,MaxPer=4
        );

options mlogic mprint mfile symbolgen;
  filename mprint "C:\temp\delete_me.txt";
%BYSTORE(StrNme=Bryn Mawr
        ,StrAbrv=Bryn
        ,TimeAbrv=Q
        ,MaxPer=4
        );

OPTIONS NOMFILE;

options mlogic mprint mfile symbolgen;
  filename mprint "C:\temp\delete_me.txt";
%ZroStr(StrNme=Bryn Mawr
              ,StrAbrv=Bryn
              ,TimeAbrv=Q
              ,MaxPer=4
              );
options nomfile;
/*****************************************************
        THE END .... AND THANKS FOR ATTENDING
*****************************************************/
```