ABSTRACT
In the life science industry, CDISC standards dictate that we keep the data for several domains in a vertical format. It is very efficient to have this format to store the data in this way as there is less waste of space. In order to create our tables, listings and figures, we need to transform or transpose the data into a horizontal format. This is a more efficient way to analyze the data. There are many ways in which this can be done. The purpose of the Transpose Procedure is to reshape the data so that it can be stored as needed and then analyzed in the easiest way possible. PROC TRANSPOSE is the easiest and most complex procedure in SAS. It has only has five options. This paper will revisit how to change the format of the data. You will be taken from the easiest way of doing the transformation without any options to a more complex manner using many options.

INTRODUCTION
The Transpose Procedure restructures the data by changing the variables into observations. How this is done and what variables are chosen to transform are determined by the options that are chosen when running the procedure. The Transpose Procedure can eliminate the need to write a complex DATA step that requires the use of one or more PROC SORT. The output from the procedure is a data set that contains the transposed data or reformatted data. The output data set can be used for analysis, reporting or further manipulation of the data. The output from the procedure can be used in other reporting procedures such as PROC PRINT or PROC REPORT.

TRANSFORMATIONS DEFINED
According to the Merriam Webster dictionary “Transformations are usually applied so that the data appear to more closely meet the assumptions of a statistical inference procedure that is to be applied, or to improve the interpretability or appearance of graphs. Nearly always, the function that is used to transform the data is invertible, and generally is continuous.” ¹ The transformation of which we speak is from the raw data into standard formats. In the life science industry, CDISC is the standard which is now required for a regulatory submission.

VERTICAL VS HORIZONTAL DATA
We classify data as either vertical (a vertical database is one in which the physical layout of the data is column-by-column rather than row-by-row) or horizontal. The format in which we store data can be very different from the format that we need to analyze the data. There are very good reasons why we need to be able to change the layout of the data based on what we need to do with the data. The best method of data storage is to keep it in a vertical format. This is also called stacked data that has a few variables and many rows. There is an identifier on each record with a little bit of information. If there is a piece of information that is missing (e.g., a test not taken, a visit not attended), there will be no row in the data.

Take a look at the listing of the vital signs data that below. You will see that there are only records if a measurement was taken for a subject’s vital signs. There aren’t any rows that are blank. And there are relatively few variables. In order to perform an analysis, it would be infinitely easier to have the data in a horizontal format.
Let’s start with the simplest process and work up to something more complex. We will use the dataset that is listed in Table 1. This data is stored in a vertical format and we need to transform it into a horizontal format. Notice that each subject has multiple records and that the identifier information is the STUDYID and USUBJID. This dataset has 43,209 rows for 902 subjects in this study. This is an SDTM Vital Signs dataset.

Let’s start with the simplest process and work up to something more complex. We will use this simple vital statistics dataset in our examples.

We will start out without any options for the PROC TRANSPOSE and add options along the way. After adding each option, I will show what the data looks like so that you can see what the effect of the option is on the data.
The first thing that I have done is to sort the data. In this case, the data is being sorted by the STUDYID, USUBJID, VSTESTCD and VISITNUM. For ease of review, I have pulled off the first 100 observations.

```
PROC SORT DATA=SDTM.vs
  (obs=100 KEEP=VSORRES USUBJID STUDYID VSTESTCD VISITNUM)
  OUT=WORK.Ssdtm
 ;
RUN;
```

```
PROC TRANSPOSE DATA=WORK.SSDTM;
RUN;
```

### Table 3: Proc Transpose: No Options

As you can see from the results, we have two records with 100 columns. By default, the procedure will only work with numeric variables. Since there are only two variables that have been transposed of the variables that we kept from the output of the sort procedure there are only 2 rows in the output. By default without any options, a transpose will only be done on numeric variables. In the CDISC datasets, most of the variables in the SDTM files are character. That is why there are only 2 variable that have been transposed.

So, what do we need to do now?

We see that we need some options to add to the transpose because this is the not result that we expected and not what we want. So, we will now explore some of the important options and statements in the transpose procedure that will help to change the output. Let’s take a look. And see what happens when we do this?

### ADDING OPTIONS

There are a number of options that you can add to the statements in a PROC TRANSPOSE in order to achieve results that are clear and data that is ready for analysis. At first, it will seem like a great deal of trial and error to get your output dataset in the format that you need. The PROC TRANSPOSE statement has a number of options. Then there are other statements with options of their own that dictate how the data is to be transformed. Let take a look at a few of the most important ones.

**PROC TRANSPOSE statement: dataset options**

In this iteration of the program, we have added a number of dataset options. The first option is the OUT options which gives a specific name to the output dataset. If we didn’t use this, the default dataset name is DATA1. Each time the program is run, the dataset name changes (DATA2, DATA3, DATA4, etc.). Now, we have a way to control what the output dataset name can be. With the dataset name under our control, we can easily use it in the future for other programming rather than guess what the number is while we are running programs. The PREFIX options gives you the control for the column names. In the previous example, the output started with COL. With this options, you can start the name with whatever makes sense to you. The NAME and LABEL option tells you the name of the data source and the label of that column. Of course, you can also use KEEP and DROP statements to remove any variables in the output that you don’t want or need. Remember, you cannot use the DROP and KEEP on the same statement. Also RENAME (RENAME=(oldname=newname) works as well).
PROC TRANSPOSE DATA=WORK.ssdtm
   OUT=WORK.TVSTEST
   PREFIX=VS
   NAME=Source
   LABEL=Label
; run;

Results

NOTE: There were 100 observations read from the data set WORK.SSDTM.
NOTE: The data set WORK.TVSTEST has 2 observations and 102 variables.
NOTE: PROCEDURE TRANSPOSE used (Total process time):
   real time           0.04 seconds
   cpu time            0.01 seconds

Table 4: Proc Transpose: with dataset options

| Obs | Source | Label | VS1 | VS2 | VS3 | VS4 | VS5 | VS6 | VS7 | VS8 | VS9 | VS10 | VS11 | VS12 | VS13 | VS14 | VS15 | VS16 | VS17 | VS18 | VS19 | VS20 | VS21 | VS22 | VS23 | VS24 | VS25 | VS26 |
|-----|--------|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 1   | VISITNUM | Visit Number | 1  | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10   | 11   | 12   | 13   | 17   | 18   | 19   | 20   | 21   | 22   | 23   | 24   | 25   | 26   | 27   | 28   | 1    |
| 2   | VISITDY | Planned Study Day of Visit | 0   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11   | 12   | 13   | 14   | 15   | 16   | 17   | 18   | 19   |

PROC TRANSPOSE statement: BY statement

This example shows how we can use the BY statement to change how the data is processed. Suppose that we want to have a record for each subject for each vital sign. How would we accomplish this? We introduce the BY statement into the code. As with any SAS procedure that uses a BY statement, the data must be sorted before the procedure is invoked. The BY statement that we will use can be seen in the sample code below. Proc Transpose does not transpose the BY group. So, the end result is exactly what we want. Now the data is beginning to look more believable and be easier to analyze.

PROC SORT DATA=SDTM.vs(OBS=100 KEEP=VSORRES USUBJID STUDYID VSTESTCD VISITNUM VISITDY VSORRESU)
   OUT=WORK.Ssdtm;
   BY STUDYID USUBJID Visitnum VSTESTCD;
RUN;

PROC TRANSPOSE DATA=WORK.ssdtm
   OUT=WORK.BYTEST
   PREFIX=VS
   NAME=Source
   LABEL=Label
; run;

PROC TRANSPOSE DATA=WORK.ssdtm
   OUT=WORK.BYTEST
   PREFIX=VS
   NAME=Source
   LABEL=Label
; run;
NOTE: There were 100 observations read from the data set WORK.SSDTM.

NOTE: The data set WORK.BYTEST has 100 observations and 7 variables.

NOTE: PROCEDURE TRANSPOSE used (Total process time):
real time          0.04 seconds
cpu time           0.00 seconds

Table 5:Proc Transpose with BY statement

<table>
<thead>
<tr>
<th>Obs</th>
<th>USUBJID</th>
<th>STUDYID</th>
<th>VISITNUM</th>
<th>VSTESTCD</th>
<th>Source</th>
<th>Label</th>
<th>VS1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>1</td>
<td>DBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>1</td>
<td>HR</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>1</td>
<td>SBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>2</td>
<td>DBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>2</td>
<td>HR</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>2</td>
<td>SBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>3</td>
<td>DBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>3</td>
<td>HR</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>3</td>
<td>SBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>4</td>
<td>DBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>4</td>
<td>HR</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>4</td>
<td>SBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>5</td>
<td>DBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>5</td>
<td>HR</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>5</td>
<td>SBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>6</td>
<td>DBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>6</td>
<td>HR</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>6</td>
<td>SBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>7</td>
<td>DBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>7</td>
<td>HR</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>NIC001-011-001</td>
<td>NIC001</td>
<td>7</td>
<td>SBP</td>
<td>VISITDY</td>
<td>Planned Study Day of Visit</td>
<td>1</td>
</tr>
</tbody>
</table>

**PROC TRANSPOSE statement: VAR statement**

The VAR statement is the one that is needed to list the variables that need to be transposed. As we noted previously, the default for PROC TRANSPOSE is to transpose only numeric variables. The VAR statement is needed to transpose character variables. So, if you have character variables that you want to be transposed, you must list them here.

```
PROC TRANSPOSE DATA=WORK.ssdTM
    OUT=WORK.VARTEST
    PREFIX=VS
    NAME=Source
    LABEL=Label;
    BY USUBJID STUDYID VISITNUM VSTESTCD;
    VAR VSORRES VSORRESU;
run;
```
We are getting closer to what we wanted to do. The only problem is that each variable was transposed onto a different record. The easiest way to handle this is to perform each transpose separately for each variable and then merge the data back together.

**PROC TRANSPOSE statement: ID statement**

Suppose you want to use the value of a variable as the variable name in the output dataset. Using the ID statement is the method to accomplish this. The ID statement specifies a variable in the input dataset whose value is used to name a variable in the output dataset. In this case, we want the variable names to be DBP, HR and SBP rather than the variable name of VS1 as seen in Table 6. The variable that contains the name that we want to use for the columns are in VSTESTCD. So that is what we will use for the ID statement.

```plaintext
PROC TRANSPOSE DATA=WORK.ssdmtm
   OUT=WORK.IDVSORRES;
   BY USUBJID STUDYID VISITNUM;
   VAR VSORRES;
   ID VSTESTCD;
run;
```

NOTE: There were 50 observations read from the data set WORK.IDVSORRES.

NOTE: PROCEDURE PRINT used (Total process time):
real time 0.61 seconds
cpu time 0.37 seconds
So far, we have talked about the values of a character variable. What happens if the variable that you want to have as the name of the transposed variable is a numeric value? SAS will change the name to a valid SAS name by affixing an underscore as the first character. If there are any missing values in your data, SAS will not transpose the data and issue a warning message in the log. If the value in the data is longer than 32 characters, SAS will truncate the value to 32 characters. If you have the system option VALIDVARNAME set to V6 where the variable names cannot be any longer than 8 characters, SAS will truncate the value to 8 characters. If you have duplicate values in the input dataset or in a BY group, Proc Transpose will issue a warning message in the log. It will stop processing and you will not get the results that you expect in the output dataset.

CONCLUSION
We have explored some of the features and options that can be used in a Proc Transpose. From its simplest invocation without any options to some of the most used ones, this procedure can be very powerful and helpful when analyzing data. With a minimal amount of code, the data can be manipulated so that the analysis is simple and easy. Because the procedure creates an output dataset and not a report, you can use any type of printing procedure to report on the output dataset. This is a very powerful tool in the base SAS product.

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.
REFERENCES
2. www.cdisc.org
3. SAS Products, Base SAS, SAS 9.4 Data Set Options: Reference, Fourth Edition

ACKNOWLEDGEMENTS
This paper would not have been written if it hadn't been for the support given to me by my husband, Robert Stuelpner. Bob diligently read this paper to correct obvious errors and keep me on the right track. His criticisms were constructive and his support never ending.

AUTHOR CONTACT
Janet Stuelpner
SAS Institute, Inc.
1388 Kettner Blvd
Suite 602
San Diego, CA 92101
(919) 531-9758
Janet.Stuelpner@sas.com