PharmaSUG 2020 – Paper AP-136

Transpose Procedure: Turning it Around Again

Janet Stuelpner, SAS Institute, Inc.

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.

Listing of Vital Signs

Obs	STUDYID	DOMAIN	USUBJID	VSSEQ	VSTESTCD	VSORRES	VSORRESU	VSSTRESN	VISITNUM	VISIT	VISITNUM	VISITDY
1	NIC001	VS	NIC001-011-001	1	DBP	68	MMHG	68	1	B	1	0
2	NIC001	VS	NIC001-011-001	2	DBP	68	MMHG	68	2	M15	2	1
3	NIC001	VS	NIC001-011-001	3	DBP	66	MMHG	66	3	M30	3	1
4	NIC001	VS	NIC001-011-001	4	DBP	67	MMHG	67	4	M45	4	1
5	NIC001	VS	NIC001-011-001	5	DBP	68	MMHG	68	5	H1	5	1
6	NIC001	VS	NIC001-011-001	6	DBP	71	MMHG	71	6	H2	6	1
7	NIC001	VS	NIC001-011-001	7	DBP	70	MMHG	70	7	H3	7	1
8	NIC001	VS	NIC001-011-001	8	DBP	80	MMHG	80	8	H4	8	1
9	NIC001	VS	NIC001-011-001	9	DBP	70	MMHG	70	9	H5	9	1
10	NIC001	VS	NIC001-011-001	10	DBP	75	MMHG	75	10	H6	10	1
11	NIC001	VS	NIC001-011-001	11	DBP	111	MMHG	111	11	H12	11	1
12	NIC001	VS	NIC001-011-001	12	DBP	55	MMHG	55	12	H18	12	1
13	NIC001	VS	NIC001-011-001	13	DBP	66	MMHG	66	13	H24	13	1
14	NIC001	VS	NIC001-011-001	14	DBP	86	MMHG	86	17	D4	17	4
15	NIC001	VS	NIC001-011-001	15	DBP	70	MMHG	70	18	D5	18	5

Table 1: Vertical Data Example

Listing of Vital Signs

Obs	VSTPT	sys1	dia1	pulse1	sys2	dia2	dia2	sys3	dia3	pulse3	VSDAT
1	B	143	68	68	145	64	64	139	72	69	12OCT2007
2	M15	136	68	66	138	64	64	132	72	67	12OCT2007
3	M30	145	66	54	147	62	62	141	70	55	12OCT2007
4	M45	141	67	56	143	63	63	137	71	57	13OCT2007
5	D6	A	A	A	.	.	.	.	.	.	16OCT2007
6	D14	118	74	72	120	70	70	114	78	73	24OCT2007
7	D14	136	70	78	138	66	66	132	74	79	24OCT2007
8	D14	140	70	.	142	66	66	136	74	.	24OCT2007
9	D8	.	.	.	.	.	.	.	.	.	06JAN2008
10	D8	.	.	.	.	.	.	.	.	.	06JAN2008
11	D6	120	80	.	122	76	76	116	84	.	16MAR2008
12	D6	122	80	79	124	76	76	118	84	80	20MAR2008
13	D6	130	80	72	132	76	76	126	84	73	20MAR2008
14	D5	180	80	.	182	76	76	176	84	.	09APR2008
15	D5	132	62	76	134	58	58	128	66	77	09APR2008

Table 2: Horizontal Data Example

PROC TRANSPOSE

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
      ;
      BY STUDYID USUBJID VISITNUM VSTESTCD;
RUN;

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

Results of Transpose Without Options

Obs	_NAME_	_LABEL_	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9	COL10	COL11	COL12	COL13	COL14	COL15	COL16	COL17	COL18	COL19	COL20	COL21
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
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

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
```

Results of 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	0

Table 4: Proc Transpose: with dataset options

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
    ;
      BY USUBJID STUDYID VISITNUM VSTESTCD;
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
```

Results of Transpose with BY statement

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

Table 5: Proc Transpose with BY statement

PROC TRANSPOSE statememt: 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;
```

Results of Transpose with VAR statement

Obs	USUBJID	STUDYID	VISITNUM	VSTESTCD	Source	Label	VS1
1	NIC001-011-001	NIC001	1	DBP	VSORRES	Result or Finding in Original Units	68
2	NIC001-011-001	NIC001	1	DBP	VSORRESU	Original Units	MMHG
3	NIC001-011-001	NIC001	1	HR	VSORRES	Result or Finding in Original Units	143
4	NIC001-011-001	NIC001	1	HR	VSORRESU	Original Units	BPM
5	NIC001-011-001	NIC001	1	SBP	VSORRES	Result or Finding in Original Units	143
6	NIC001-011-001	NIC001	1	SBP	VSORRESU	Original Units	MMHG
7	NIC001-011-001	NIC001	2	DBP	VSORRES	Result or Finding in Original Units	68
8	NIC001-011-001	NIC001	2	DBP	VSORRESU	Original Units	MMHG
9	NIC001-011-001	NIC001	2	HR	VSORRES	Result or Finding in Original Units	136
10	NIC001-011-001	NIC001	2	HR	VSORRESU	Original Units	BPM
11	NIC001-011-001	NIC001	2	SBP	VSORRES	Result or Finding in Original Units	136
12	NIC001-011-001	NIC001	2	SBP	VSORRESU	Original Units	MMHG
13	NIC001-011-001	NIC001	3	DBP	VSORRES	Result or Finding in Original Units	66
14	NIC001-011-001	NIC001	3	DBP	VSORRESU	Original Units	MMHG
15	NIC001-011-001	NIC001	3	HR	VSORRES	Result or Finding in Original Units	145
16	NIC001-011-001	NIC001	3	HR	VSORRESU	Original Units	BPM
17	NIC001-011-001	NIC001	3	SBP	VSORRES	Result or Finding in Original Units	145
18	NIC001-011-001	NIC001	3	SBP	VSORRESU	Original Units	MMHG
19	NIC001-011-001	NIC001	4	DBP	VSORRES	Result or Finding in Original Units	67
20	NIC001-011-001	NIC001	4	DBP	VSORRESU	Original Units	MMHG
21	NIC001-011-001	NIC001	4	HR	VSORRES	Result or Finding in Original Units	141
22	NIC001-011-001	NIC001	4	HR	VSORRESU	Original Units	BPM
23	NIC001-011-001	NIC001	4	SBP	VSORRES	Result or Finding in Original Units	141

Table 6: Proc Transpose with VAR statement

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.

```
PROC TRANSPOSE DATA=WORK.ssdtm
      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
```

Results of Transpose with ID statement

Obs	USUBJID	STUDYID	VISITNUM	_NAME_	_LABEL_	DBP	HR	SBP
1	NIC001-011-001	NIC001	1	VSORRES	Result or Finding in Original Units	68	143	143
2	NIC001-011-001	NIC001	2	VSORRES	Result or Finding in Original Units	68	136	136
3	NIC001-011-001	NIC001	3	VSORRES	Result or Finding in Original Units	66	145	145
4	NIC001-011-001	NIC001	4	VSORRES	Result or Finding in Original Units	67	141	141
5	NIC001-011-001	NIC001	5	VSORRES	Result or Finding in Original Units	68	142	142
6	NIC001-011-001	NIC001	6	VSORRES	Result or Finding in Original Units	71	152	152
7	NIC001-011-001	NIC001	7	VSORRES	Result or Finding in Original Units	70	150	150
8	NIC001-011-001	NIC001	8	VSORRES	Result or Finding in Original Units	80	140	140
9	NIC001-011-001	NIC001	9	VSORRES	Result or Finding in Original Units	70	124	124
10	NIC001-011-001	NIC001	10	VSORRES	Result or Finding in Original Units	75	138	138
11	NIC001-011-001	NIC001	11	VSORRES	Result or Finding in Original Units	111	205	205
12	NIC001-011-001	NIC001	12	VSORRES	Result or Finding in Original Units	55	136	136
13	NIC001-011-001	NIC001	13	VSORRES	Result or Finding in Original Units	66	155	155
14	NIC001-011-001	NIC001	17	VSORRES	Result or Finding in Original Units	86	160	160
15	NIC001-011-001	NIC001	18	VSORRES	Result or Finding in Original Units	70	130	130
16	NIC001-011-001	NIC001	19	VSORRES	Result or Finding in Original Units	86	152	152
17	NIC001-011-001	NIC001	20	VSORRES	Result or Finding in Original Units	82	130	130
18	NIC001-011-001	NIC001	21	VSORRES	Result or Finding in Original Units	82	136	136
19	NIC001-011-001	NIC001	22	VSORRES	Result or Finding in Original Units	90	130	130
20	NIC001-011-001	NIC001	23	VSORRES	Result or Finding in Original Units	78	150	150
21	NIC001-011-001	NIC001	24	VSORRES	Result or Finding in Original Units	70	130	130
22	NIC001-011-001	NIC001	25	VSORRES	Result or Finding in Original Units	70	124	124
23	NIC001-011-001	NIC001	26	VSORRES	Result or Finding in Original Units	80	126	126
24	NIC001-011-001	NIC001	27	VSORRES	Result or Finding in Original Units	70	136	136
25	NIC001-011-001	NIC001	28	VSORRES	Result or Finding in Original Units	72	130	130

Table 6: Proc Transpose with ID statement

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.

REFERENCES

1. Merriam-Webster's collegiate dictionary (10th ed.). (1999). Springfield, MA: Merriam-Webster Incorporated.
2. www.cdisc.org
3. SAS Products, Base SAS, SAS 9.4 Data Set Options: Reference, Fourth Edition
4. SAS Products, Base SAS, Base SAS 9.4 Procedures Guide, Sixth 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