



# Getting Your Data in Shape with PROC TRANSPOSE

Nancy Brucken  
PharmaNet/i3

# Agenda

- ▶ Wide to narrow
- ▶ Narrow to wide
- ▶ Multiple variables
- ▶ Review

## Wide to Narrow

► Have:

<b>X1</b>	<b>X2</b>	<b>X3</b>	<b>X4</b>
1	3	4	8

## Wide to Narrow

► Want:

<b>X1</b>	1
<b>X2</b>	3
<b>X3</b>	4
<b>X4</b>	8

# PROC TRANSPOSE

Simplest form:

```
proc transpose data=a out=b;  
run;
```

Transposes all numeric variables in the input dataset that do not appear in another statement in the PROC TRANSPOSE step.

## Wide to Narrow

► Have:

<b>X1</b>	<b>X2</b>	<b>X3</b>	<b>X4</b>
1	3	4	8

► All numeric variables will be transposed and stacked

## Resulting Dataset

_NAME_	COL1
x1	1
x2	3
x3	4
x4	8



What if I don't like the variable names?

## Controlling Variable Names

- ▶ NAME = option
  - ▶ Changes the name of `_NAME_`
- ▶ PREFIX = option
  - ▶ Specifies a prefix for the name of the transposed variables

## Controlling Variable Names

```
proc transpose data=a out=b name=test prefix=var;  
run;
```

## Resulting Dataset

test	var1
x1	1
x2	3
x3	4
x4	8



What about BY variables?

## Have:

Region	x1	x2	x3	x4
North	1	3	4	8
South	2	6	9	7

## BY-Group Processing

```
proc transpose data=a out=b (drop=_label_) name=test prefix=var;  
  by region;  
run;
```

Variable names and labels are identical in this example, so no need for both in output dataset.

Variables are transposed within BY-group

## Resulting Dataset

Region	test	var1
North	x1	1
North	x2	3
North	x3	4
North	x4	8
South	x1	2
South	x2	6
South	x3	9
South	x4	7



What about a more complicated transpose?

## Have:

Region	x1	x2	x3	x4
North	1	3	4	8
South	2	6	9	7

**Want:**

Region	test	var1	var2
North	x1	1	3
North	x2	4	8
South	x1	2	6
South	x2	9	7

## DATA Step Transpose

```
data b (keep=region var1-var2);
```

```
  set a;
```

```
  by region;
```

```
  array xs(*) x1-x4;
```

Input variables

```
  array vars(*) var1-var2;
```

Output variables

```
  do i=1 to dim(xs);
```

```
    vars(2 - mod(i, 2)) = xs(i);
```

```
    if mod(i, 2) = 0 then output;
```

```
  end;
```

Send pairs of  
values to output  
dataset

```
run;
```



How do I go from narrow to wide?

## Narrow to Wide

► Have:

x
1
3
4
8

## Narrow to Wide

► Want:

col1	col2	col3	col4
1	3	4	8

# PROC TRANSPOSE

Simplest form:

```
proc transpose data=a out=b;  
run;
```

Transposes all numeric variables in the input dataset that do not appear in another statement in the PROC TRANSPOSE step.

Seen this before?

## Resulting Dataset

_NAME_	COL1	COL2	COL3	COL4
x	1	3	4	8

## Controlling Variable Names

```
proc transpose data=a out=b name=test prefix=var;  
run;
```

## Resulting Dataset

test	var1	var2	var3	var4
x	1	3	4	8



How do I let the data drive my variable names?

## Data-Driven Variable Names

► Have:

Group	x
1	1
3	3
2	4
4	8

## Data-Driven Variable Names

► Want:

test	var1	var2	var3	var4
x	1	4	3	8

► Use value of GROUP to determine names of transposed variables

## Data-Driven Variable Names

- ▶ ID statement
  - ▶ Value of the ID variable is used as the name of the transposed variable
  - ▶ If ID variable is numeric, resulting variable names start with '-'
  - ▶ If PREFIX is specified, resulting variable names are concatenation of PREFIX and ID variable value

## ID Statement

```
proc transpose data=a out=b name=test prefix=var;
```

```
  id group;
```

```
run;
```

## Resulting Dataset

test	var1	var2	var3	var4
x	1	4	3	8



What about BY groups?

## BY-Group Processing

► Have:

Region	Group	x
North	1	1
North	2	3
North	3	5
South	1	4
South	2	8
South	3	6

## BY-Group Processing

► Want:

Region	var1	var2	var3
North	1	3	5
South	4	8	6

## BY-Group Processing with ID Variable

```
proc transpose data=a out=b (drop=_label_) name=test prefix=var;
```

```
  by region;
```

```
  id group;
```

```
run;
```

## Resulting Dataset

Region	test	var1	var2	var3
North	x	1	3	5
South	x	4	8	6



How can I transpose multiple variables?

## Handling Multiple Variables

► Have:

Group	x	y
1	1	2
3	3	5
2	4	7
4	8	9

## What if....?

```
proc transpose data=a out=b name=test prefix=var;
```

```
id group;
```

```
var x y;
```

```
run;
```

## Resulting Dataset

test	var1	var3	var2	var4
x	1	3	4	8
y	2	5	7	9

Row created for each column  
Column created for each row



How about another type of multi-column transpose?

## Problem

- ▶ Original dataset (sorted by ANALYTEC and LISTNUM):

ANALYTEC	LISTTYPE	LISTNUM	RXGRP	PATCT	DENOM
CALCIUM	Subjects With One or More Abnormal Screen/Baseline Values	1	1	6	260
CALCIUM	Subjects With One or More Abnormal Screen/Baseline Values	1	2	4	227
CALCIUM	Subjects With Normal Screen/Baseline Values and One Abnormal Value After Initiation of Treatment	2	1	5	260
CALCIUM	Subjects With Normal Screen/Baseline Values and One Abnormal Value After Initiation of Treatment	2	2	6	227

## Goal

► Desired dataset:

ANALYTEC	LISTTYPE	LISTNUM	PATCT1	PATCT2	DENOM1	DENOM2
CALCIUM	Subjects With One or More Abnormal Screen/Baseline Values	1	6	4	260	227
CALCIUM	Subjects With Normal Screen/Baseline Values and One Abnormal Value After Initiation of Treatment	2	5	6	260	227

## PROC TRANSPOSE Limitation

- ▶ Multiple variables in VAR statement → one record per variable in output dataset

## Solution

- ▶ Run a separate PROC TRANSPOSE for each variable, and then merge resulting datasets.

## Drawbacks

- ▶ Requires a separate pass through the dataset for each PROC TRANSPOSE
- ▶ Requires another pass through each output dataset for the merge

## Alternate Solution

- ▶ DOW-Loop
  - ▶ Also known as Whitlock DO-Loop or Dorfman-Whitlock DO-Loop

## Advantages

- ▶ Moves DATA step SET statement inside of a DO-Loop
  - ▶ Gives complete control over retention of variable values and timing of the population of Program Data Vector (PDV)
  - ▶ Requires just one pass through the dataset

## Resulting Code

```
data out_ct (drop=patct denom rxgrp);  
  array patcts (*) patct1-patct2;  
  array denoms (*) denom1-denom2;  
  
  do until (last.listnum or eof);  
    set out_ct1 end=eof;  
    by analytec listnum;  
  
    patcts(rxgrp) = patct;  
    denoms(rxgrp) = denom;  
  end;  
  
  output;  
run;
```

## How It Works (Part 1)

- ▶ Initialization
  - ▶ DATA statement initializes PDV
  - ▶ ARRAY statements are only executed the first time through the DATA step

## Resulting Code

```
data out_ct (drop=patct denom rxgrp);  
  array patcts (*) patct1-patct2;  
  array denoms (*) denom1-denom2;
```

```
do until (last.listnum or eof);  
  set out_ct1 end=eof;  
  by analytec listnum;
```

```
  patcts(rxgrp) = patct;  
  denoms(rxgrp) = denom;  
end;
```

```
output;  
run;
```

## How It Works (Part 2)

- ▶ Individual record processing
  - ▶ DO-loop executes over all records with the same ANALYTEC/LISTNUM value
  - ▶ SET statement populates PDV
  - ▶ Appropriate array elements are assigned values based on the value of RXGRP
  - ▶ Note that array elements are retained until all BY LISTNUM processing is done and control returned to DATA statement.

## Resulting Code

```
data out_ct (drop=patct denom rxgrp);  
  array patcts (*) patct1-patct2;  
  array denoms (*) denom1-denom2;  
  
do until (last.listnum or eof);  
  set out_ct1 end=eof;  
  by analytec listnum;  
  
  patcts(rxgrp) = patct;  
  denoms(rxgrp) = denom;  
end;  
  
output;  
run;
```

## How It Works (Part 3)

- ▶ Output
  - ▶ OUTPUT statement is executed after the DO-loop completes
  - ▶ Output dataset is 1 record per ANALYTEC/LISTNUM

## Resulting Code

```
data out_ct (drop=patct denom rxgrp);  
  array patcts (*) patct1-patct2;  
  array denoms (*) denom1-denom2;  
  
do until (last.listnum or eof);  
  set out_ct1 end=eof;  
  by analytec listnum;  
  
  patcts(rxgrp) = patct;  
  denoms(rxgrp) = denom;  
end;  
  
output;  
run;
```

## Notes

- ▶ If VAR statement contains both character and numeric variables, resulting transposed variables are ALL character
- ▶ ID statement uses the value of the ID variable in the name of the transposed variable; IDLABEL statement uses the value of the ID variable in the label of the transposed variable.

## Review

- ▶ PROC TRANSPOSE can change the shape of your dataset
  - ▶ Wide to narrow
  - ▶ Narrow to wide
- ▶ PROC TRANSPOSE exchanges rows and columns of your dataset
- ▶ Use DATA step for more complex data transpose logic

## References

- ▶ This has been a quick look at some of the most commonly-used features of PROC TRANSPOSE.
- ▶ See the SAS documentation for other procedure options.
- ▶ <http://www.lexjansen.com> and search for TRANSPOSE

## Additional DOW-Loop References

- ▶ Dorfman, Paul. (2008) “The DOW-Loop Unrolled”. PharmaSUG 2008 Conference Proceedings.
- ▶ Chakravarthy, Venky, (2005). “RETAIN or NOT? Is LAG Far Behind?”. PharmaSUG 2005 Conference Proceedings
- ▶ Brucken, Nancy. (2009) “One-Step Change from Baseline Calculations”. PharmaSUG 2009 Conference Proceedings
- ▶ SAS-L listserv archives at <http://www.listserv.uga.edu/archives/sas-l.html>, and search for postings by Ian Whitlock and Paul Dorfman, among others.

## Other Useful SAS References

- ▶ SAS-L listserv archives (<http://www.listserv.uga.edu/archives/sas-l.html>- also available from Google Groups as comp.soft-sys.sas)
- ▶ sasCommunity.org (<http://www.sascommunity.org>)
- ▶ SAS Discussion Forums (<http://support.sas.com/forums/index.jspa>)
- ▶ SAS Technical Support (<http://support.sas.com>)



## Contact Information

Nancy Brucken

PharmaNet/i3

[Nancy.Brucken@i3global.com](mailto:Nancy.Brucken@i3global.com)

(734) 757-9045



Questions?

