Paper 148-26

The Power and Simplicity of the TABULATE Procedure

Dan Bruns, Tennessee Valley Authority, Chattanooga, TN

ABSTRACT

When I first started using SAS in a university environment in 1972, I was very excited about how much SAS could help me. As I learned more and more and moved to the real world of the employeed I needed more flexibility in my reporting - PROC PRINT had some and PROC MEANS and FREQ had basically none. And SUMMARY needed too much DATA step manipulation. What I needed was a marriage of all these with the flexibility to control what I wanted where and how! Thank the Gods for PROC TABULATE!! I was able to produce reports from massive amounts of data in practically any way I needed! The power of being able to simply rearrange a few variable names and change the complete look of the report was great. But then I also found a few things that were not so simple-like percentages.

BEFORE TABULATE

So many times when you first learn some new software product, you simply do what you are told or can findout about it. I learned base SAS software the same way at a university from a friend. It was for some math classes and I learned to do some simple inputting and frequency tables and summaries using PROC FREQ and SUMMARY. It wasn't until years later I discovered the power of PROC TABULATE.

If all you ever need to do is some simple cross-tabulations or simply print a list of summaries, PROC FREQ or MEANS or SUMMARY will do the job. But more often, you need to be able to control WHAT goes in the cross-tabulation cell and HOW it is printed; PROC TABULATE gives you this control and versatility.

In this tutorial you will see how - with very little coding - you can produce some simple or very complex output. The output from a CONTENTS procedure below is just so you know a little about the data set we will be working with.

CONTENTS PROCEDURE

Data Set Name:	SASDATA.CLASS	Observations:	27
Member Type:	DATA	Variables:	5
Engine:	V60x	Indexes:	0
Created:	9:14 Wednesday, Sep 19	Observation Length:	48
Last Modified:	11:40 Tuesday, Feb 5	Deleted Observations:	0
Data Set Type:		Compressed:	NO
		Reuse Space:	NO

-----Alphabetic List of Variables and Attributes-----

#	Variable	Type	Len	Pos	Format	Label
4	DATE	Num	8	32	DATE5.	Class Date
2	LOC	Char	1	25		Location
1	NAME	Char	25	0		
3	ORG	Char	6	26		Org
5	SCORE	Num	8	40	5.1	Final Exam Score

THE OLD WAY

The FREQ procedure is very simple and easy to use to get either a one-way or two-way table by coding TABLE statements as below.

```
PROC FREQ      DATA=CLASS;
     TABLES  DATE;
     TABLES  ORG*DATE;
```

The output below shows that basically all you get is a count of observations and percentages.

Class Date

DATE	Frequency	Percent	Cumulative Frequency	Cumulative Percent
07APR	5	18.5	5	18.5
03MAY	4	14.8	9	33.3
22JUN	9	33.3	18	66.7
12OCT	9	33.3	27	100.0

TABLE OF ORG BY DATE

ORG(Org) DATE(Class Date)

Frequency / Percent / Row Pct / Col Pct	07APR	03MAY	22JUN	12OCT	Total
Energy	1	2	1	4	8
	3.70	7.41	3.70	14.81	29.63
	12.50	25.00	12.50	50.00	
	20.00	50.00	11.11	44.44	
Mgt S	1	1	4	4	10
	3.70	3.70	14.81	14.81	37.04
	10.00	10.00	40.00	40.00	
	20.00	25.00	44.44	44.44	
Power	3	1	4	1	9
	11.11	3.70	14.81	3.70	33.33
	33.33	11.11	44.44	11.11	
	60.00	25.00	44.44	11.11	
Total	5	4	9	9	27
	18.52	14.81	33.33	33.33	100.00

The MEANS procedure is also very simple to use and gives you more information about your data, i.e. MEAN and SUM, if requested as below.

```
PROC MEANS  DATA=CLASS  N  NMISS  MEAN  SUM ;
     CLASS LOC;
     VAR   SCORE;
```

But the output produced is only a simple list as below.

Analysis Variable : SCORE Final Exam Score

LOC	N Obs	N	Nmiss	Mean	Sum
A	12	10	2	84.0500000	840.5000000
B	5	5	0	83.9000000	419.5000000
C	10	10	0	83.2900000	832.9000000

A BETTER WAY

The TABULATE procedure is a marriage of the FREQ and MEANS procedures to produce an even more powerful and flexible procedure. TABULATE makes use of the CLASS and VAR statements of PROC MEANS and the TABLE statement of PROC FREQ. The CLASS and VAR statements haven't changed, they still define which variables are used for classification (or categorization) and analysis (for statistics). The TABLE statement has been greatly enhanced to allow you to specify not only ***what*** you want each cell to be but also ***how*** to format it and label it. It has also been expanded to allow you to group and/or concatenate more than one table. Let's look at some simple examples.

```
PROC TABULATE  DATA=CLASS;
     CLASS    ORG LOC DATE;
     VAR      SCORE;
     TABLE    ORG, DATE;
```

The PROC statement simply identifies the data set and other options to control various output features. The CLASS and VAR statements simply define the variables used for classification and analysis. The TABLE statement specifies the table expressions used to define the pages, rows, and columns to be produced.

The key to using TABULATE are the **table expressions** which consist of a combination of operands and operators much like an arithmetic expression:

TABLE
page-expression,
row-expression,
column-expression ;

Operators (discussed later)
- * for nesting
- space for concatenation

Operands
- class variables(from CLASS stmt) or ALL
- analysis variables(from VAR stmt)
- () for grouping
- statistics keywords (i.e. N, NMISS, MEAN, STD, MIN, MAX, SUM)
- title specification (='text')
- format specification (F=w.d)

When only two expressions are coded (like ORG, DATE), they are the row-expression and the column-expression. What if only ONE expression is coded; will it be the row or column expression?

The tables below show a very simple yet powerful use of TABULATE. Although our data set only contains a few observations, it could have contained half a million and the code would not change.

	Class Date			
	07APR	03MAY	22JUN	12OCT
	N	N	N	N
Org				
Energy	1.00	2.00	1.00	4.00
Mgt S	1.00	1.00	4.00	4.00
Power	3.00	1.00	4.00	1.00

The above output shows that when CLASS variables are used, TABULATE will give you a count (N) by default.

```
TABLE  ORG,  SCORE;
```

	Final Exam Score
	SUM
Org	
Energy	530.40
Mgt S	825.60
Power	736.90

The above output shows that when VAR variables are used, TABULATE will give you a total (SUM) of that variable by default.

```
TABLE  LOC,  ORG,  DATE;
```

Location A

	Class Date			
	07APR	03MAY	22JUN	12OCT
	N	N	N	N
Org				
Energy	1.00	1.00	1.00	3.00
Mgt S	1.00	.	.	1.00
Power	1.00	1.00	1.00	1.00

Location B

	Class Date		
	07APR	03MAY	12OCT
	N	N	N
Org			
Energy	.	1.00	1.00
Mgt S	.	1.00	.
Power	2.00	.	.

Location C

	Class Date	
	22JUN	12OCT
	N	N
Org		
Mgt S	4.00	3.00
Power	3.00	.

The above outputs show the results of using a page-expression.

FORMATTING

The above examples show a very simple use of tables, and if all you needed was the numbers to plug in some report for your boss, this would suffice. But I have found that sometimes these tables go beyond your eyes or you need to make the cells smaller (or larger). As you can see, the default format for the cells is 12.2, but you can control the size of each cell by attaching a format specification to the variable in the table expression.

```
TABLE  ORG,  DATE*(N*F=6.0);
```

	Class Date			
	07APR	03MAY	22JUN	12OCT
	N	N	N	N
Org				
Energy	1	2	1	4
Mgt S	1	1	4	4
Power	3	1	4	1

You could also specify FORMAT=6.0 as an option on the PROC statement, but that changes the default for all cells. By using the FORMAT= (or F=) in the table expression you can control the format of individual rows or columns based on which variable you attach it to.

NESTING

Nesting is a very common use of TABULATE and is produced by coding an asterisk (*) between the variables you want to breakdown even further for more detailed information. So the expression DATE*LOC will produce a row/column for each date (4 in our example) broken-down even further by each location within that date. Notice that even though the table is in sorted order, we did ***NOT*** have to do a PROC SORT prior to this step--the CLASS statement takes care of this for us. In the following example of nesting columns, parentheses are used to aid in clarification. Later they are also used to group expressions just like in arithmetic.

```
TABLE  ORG,  DATE*LOC*(N*F=3.0);
```

	Class Date								
	07APR		03MAY		22JUN		12OCT		
	Location		Location		Location		Location		
	A	B	A	B	A	C	A	B	C
	N	N	N	N	N	N	N	N	N
Org									
Energy	1	.	1	1	1	.	3	1	.
Mgt S	1	.	.	1	.	4	1	.	3
Power	1	2	1	.	1	3	1	.	.

Notice how TABULATE split the "Location" label because it was too long to fit over its nested column. TABULATE makes no attempt to hyphenate it correctly; it simply puts as much as it can on the first line, inserts a hyphen, and continues on the next line. The best way to control the splitting is to use labels with blanks where you want it split, since it will split at a blank the same way the PRINT procedure does.

Now by simply moving the nesting to the row expression you have changed the whole look of the table.

```
TABLE  ORG*DATE,  LOC*(N*F=3.0);
```

		Location		
		A	B	C
		N	N	N
Org	Class Date			
Energy				
	07APR	1	.	.
	03MAY	1	1	.
	22JUN	1	.	.
	12OCT	3	1	.
Mgt S	07APR	1	.	.
	03MAY	.	1	.
	22JUN	.	.	4
	12OCT	1	.	3
Power	07APR	1	2	.
	03MAY	1	.	.
	22JUN	1	.	3
	12OCT	1	.	.

WOW ! ! WHAT HAPPENED ?!?!

The look of the table completely changed by simply moving a variable from nesting in the columns to nesting in the rows.

You'll also notice that TABULATE uses the same amount of space to label the rows, regardless of the levels of nesting. The above output is all right in this case, but could get very narrow with another level of nesting. The default amount of space is 1/4 of the linesize (LS= option), but can be specified with the RTS= table option as below. The RTS space will be divided equally among the levels of nesting. Can you guess what RTS stands for?

```
TABLE  ORG*DATE,  LOC*(N*F=3.0) / RTS=20;
```

		Location		
		A	B	C
		N	N	N
Org	Class Date			
Energy				
	07APR	1	.	.
	03MAY	1	1	.
	22JUN	1	.	.
	12OCT	3	1	.
Mgt S	07APR	1	.	.
	03MAY	.	1	.
	22JUN	.	.	4
	12OCT	1	.	3
Power	07APR	1	2	.
	03MAY	1	.	.
	22JUN	1	.	3
	12OCT	1	.	.

As with most base SAS software statements used in procedures (i.e. PLOT and CHART), you can follow the table request with a slash followed by table options. We'll see more of these options later.

CONCATENATION

Concatenation is like putting tables side-by-side or stacked on top of each other. By simply using a blank between the variables you get tables for each variable concatenated from left-to-right or top-to-bottom.

```
TABLE  ORG  DATE,  LOC*(N*F=3.0) ;
```

	Location		
	A	B	C
	N	N	N
Org			
Energy	6	2	.
Mgt S	2	1	7
Power	4	2	3
Class Date			
07APR	3	2	.
03MAY	2	2	.
22JUN	2	.	7
12OCT	5	1	3

The above output shows the results of concatenating in the row expression; basically stacked tables.

```
TABLE  ORG,  (LOC  DATE)*(N*F=3.0) ;
```

	Class Date					
Location	07-	03-	22-	12-		
	A	B	C	APR	MAY	JUN	OCT
	N	N	N	N	N	N	N
Org							
Energy	6	2	.	1	2	1	4
Mgt S	2	1	7	1	1	4	4
Power	4	2	3	3	1	4	1

The above output shows the results of concatenating in the column expression; basically side-by-side tables.

```
TABLE  ORG,  (LOC*N*F=2.0  DATE*N*F=6.0);
```

Location	Class Date					
	A	B	C	07APR	03MAY	22JUN	12OCT
	N	N	N	N	N	N	N
Org							
Energy	6	2	.	1	2	1	4
Mgt S	2	1	7	1	1	4	4
Power	4	2	3	3	1	4	1

This output shows the results of concatenating in the column expression and specifying which statistic (N) and how to format it (F=) for each column.

STATISTICS

TABULATE gives you the same statistics as the MEANS and SUMMARY procedures, plus two new ones: PCTN and PCTSUM. With the flexibility of TABULATE you can now specify what statistic you want where and how. One requirement is you can only perform statistics on analysis variables (from the VAR statement), which only makes sense. You've already seen this in some earlier examples where you nest (or attach) the statistic to either a row or column table expression. Here are some examples.

```
TABLE  ORG,  SCORE*MEAN;
```

	Final Exam Score
	MEAN
Org	
Energy	88.40
Mgt S	82.56
Power	81.88

This is the mean of the scores for each organization. Remember the default statistic is SUM if not specified.

```
TABLE  ORG,  SCORE*(N  MEAN  MAX  PCTN);
```

	Final Exam Score			
	N	MEAN	MAX	PCTN
Org				
Energy	6.00	88.40	100.00	24.00
Mgt S	10.00	82.56	99.40	40.00
Power	9.00	81.88	99.10	36.00

Notice here that by grouping with parenthesis you can get more than one statistic for a given variable. The way to look at nesting a group is like the distributive law of mathematics. Whatever is outside the parenthesis is distributed to each item inside the parenthesis. Thus you could code the above expression as:

```
TABLE  ORG,  SCORE*N    SCORE*MEAN
             SCORE*MAX  SCORE*PCTN ;
```

But that seems a little much.

```
TABLE  ORG, SCORE*(N  MEAN  MAX  PCTN)*F=5.1;
```

	Final Exam Score			
	N	MEAN	MAX	PCTN
Org				
Energy	6.0	88.4	100.0	24.0
Mgt S	10.0	82.6	99.4	40.0
Power	9.0	81.9	99.1	36.0

Here you see how to nest (or attach) a format to a group. The same distributive principle applies as earlier. The PCTN statistic you've seen in the last couple examples is a percentage of a certain number of observations. By default it computes the percentage based on the total number of observations. This can be controlled with a denominator specification and is ***WAY*** beyond the scope of this tutorial. Look for a sequel in the near future.

```
TABLE  ORG*(N  MEAN  MAX  PCTN),  SCORE;
```

		Final Exam Score
Org		
Energy	N	6.00
	MEAN	88.40
	MAX	100.00
	PCTN	24.00
Mgt S	N	10.00
	MEAN	82.56
	MAX	99.40
	PCTN	40.00
Power	N	9.00
	MEAN	81.88
	MAX	99.10
	PCTN	36.00

Here again you can completely change the look of the table by simply moving the grouping to the row expression. Notice the statistics are not of ORG but simple specify in which dimension to put them.

Here are few more examples and their totally different looking outputs by simply changing where and how the variables are coded.

```
TABLE  ORG,  LOC*SCORE*(N  MEAN)*F=5.1;
```

	Location		
	A	B	C
	Final Exam Score	Final Exam Score	Final Exam Score

	N	MEAN	N	MEAN	N	MEAN
Org						
Energy	4.0	84.4	2.0	96.4	.	.
Mgt S	2.0	73.4	1.0	85.4	7.0	84.8
Power	4.0	89.0	2.0	70.7	3.0	79.8

Here you see a column for the count (N) and mean of SCORE for each location.

```
TABLE ORG*LOC, SCORE*(N MEAN MAX PCTN)*F=5.1;
```

		Final Exam Score			
		N	MEAN	MAX	PCTN
Org	Location				
Energy	A	4.0	84.4	93.0	16.0
	B	2.0	96.4	100.0	8.0
Mgt S	A	2.0	73.4	99.4	8.0
	B	1.0	85.4	85.4	4.0
	C	7.0	84.8	98.3	28.0
Power	A	4.0	89.0	99.1	16.0
	B	2.0	70.7	90.0	8.0
	C	3.0	79.8	93.6	12.0

Here are the same numbers from the previous output. Location was simply moved from the column expression to the row expression.

```
TABLE  ORG  LOC,
       SCORE*(N*F=5.  MEAN  MAX  PCTN)*F=5.1;
```

	Final Exam Score			
	N	MEAN	MAX	PCTN
Org				
Energy	6	88.4	100.0	24.0
Mgt S	10	82.6	99.4	40.0
Power	9	81.9	99.1	36.0
Location				
A	10	84.0	99.4	40.0
B	5	83.9	100.0	20.0
C	10	83.3	98.3	40.0

Here are two tables in one: the N, MEAN, MAX, and PCTN statistics in the column expression allows you to use the row expression to see a summary by two different variables (ORG and LOC) in one table.

```
TABLE   ORG  ALL,
        (LOC  ALL)*SCORE*(N  MEAN)*F=5.1;
```

	Location							
	A		B		C		ALL	
	Final Exam Score		Final Exam Score		Final Exam Score		Final Exam Score	
	N	MEAN	N	MEAN	N	MEAN	N	MEAN
Org								
Energy	4	84.4	2	96.4	.	.	6	88.4
Mgt S	2	73.4	1	85.4	7	84.8	10	82.6
Power	4	89.0	2	70.7	3	79.8	9	81.9
ALL	10	84.0	5	83.9	10	83.3	25	83.7

What in the world happened in this last example?

There's no variable named ALL in the data set?

That's right but ALL is kind of like a built-in variable that can be specified to accumulate totals for the entire row and/or column. In the above example it was used in the row expression to produce a set of totals after the ORG rows. If you placed it before the ORG variable (i.e. ALL ORG) you would get the totals as the ***first*** row of the table. The use of ALL in the column expression caused it to produce a column after the LOC columns. Also notice since it was grouped with LOC and then nested, the column contains the totals for all locations using the same statistics (there's that distributive law again).

You may not be able to tell from this example, but TABULATE computes true statistics (i.e. MEAN above). That means it ***DOES NOT*** add-up the means from the tables and then divides by the number of tables entries; it accumulates ***each*** observation's value and divides by the number of observations.

TITLES AND LABELS

You have seen that to have TABULATE put descriptive titles or labels for the variables you simply need to assign meaningful labels to them. You can either do this in earlier steps that create the data set or with a LABEL statement in the PROC step. But what about the statistics and ALL? Simply attach a descriptive label to ANY variable or statistic right in the TABLE statement. Follow it with an equals sign(=) and a quoted label ('This is a Label') just like you do in a LABEL statement. Or if you want to use a certain label for every use of the statistic, use the KEYLABEL statement which looks exactly like the LABEL statement except you use statistic's name instead of a variable name. Here is an example of doing both.

```
TABLE
        ORG  ALL,
        (LOC  ALL='Row Totals')
            *SCORE*(N  MEAN)*F=5.1
        /  BOX='SUGI 24';
KEYLABEL
        N='Count'
        MEAN='Avg'
        ALL='Total' ;
```

SUGI 24	Location							
	A		B		C		Row Totals	
	Final Exam Score		Final Exam Score		Final Exam Score		Final Exam Score	
	Count	Avg	Count	Avg	Count	Avg	Count	Avg
Department								
Energy	4	84.4	2	96.4	.	.	6	88.4
Mgt S	2	73.4	1	85.4	7	84.8	10	82.6
Power	4	89.0	2	70.7	3	79.8	9	81.9
Total	10	84.0	5	83.9	10	83.3	25	83.7

The above example has another tables option specified (BOX=) that specifies what to put in the upper-left corner box of the table.

In the following example we added the MISSING and NOSEPS options to the PROC statement to have TABULATE treat missing values as a valid category (which it does not do by default) and remove the separation lines between the rows. I also specified

some table options: BOX=SCORE to label the upper-left box with the SCORE variable's label; and MISSTEXT='None' to label missing values in the tables with the text 'None' instead of the standard period.

```
PROC TABULATE  DATA=CLASS  MISSING  NOSEPS ;
    CLASS    ORG  LOC  DATE;
    VAR      SCORE;
/*   TITLES & LABELS */
    LABEL SCORE='Final Exam Averages';

    TABLE
      ORG  ALL='--- Totals ---',
      (LOC  ALL='Row Totals')
           *(SCORE*MEAN=' '*F=5.1)
      /BOX=SCORE  ROW=FLOAT  MISSTEXT='None';
```

Final Exam Averages	Location			Row Tota-ls
	A	B	C	
	Final Exam Averages	Final Exam Averages	Final Exam Averages	Final Exam Averages
Department				
Energy	None	84.4	96.4	None	88.4
Mgt S	None	73.4	85.4	84.8	82.6
Power	None	89.0	70.7	79.8	81.9
--- Totals ---	None	84.0	83.9	83.3	83.7

Notice that since the MEAN label was blank and the ROW=FLOAT was specified, that no space was wasted for it.

Now as one final farewell to labeling, a table that doesn't look like a table.

```
PROC TABULATE  DATA=CLASS  MISSING  NOSEPS
        FORMCHAR='                ';
    CLASS    ORG  LOC  DATE;
    VAR      SCORE;
    LABEL SCORE='Final Exam Averages';
    TABLE
        ORG  ALL='--- Totals ---',
        (LOC  ALL='Dept Totals')
            *(SCORE=' '*MEAN=' '*F=6.1)
        /BOX=SCORE ROW=FLOAT MISSTEXT='None';
```

Final Exam Averages	Location				Dept Totals
	A	B	C		
Department					
Energy	None	84.4	96.4	None	88.4
Mgt S	None	73.4	85.4	84.8	82.6
Power	None	89.0	70.7	79.8	81.9
--- Totals ---	None	84.0	83.9	83.3	83.7

By simply adding the FORMCHAR= option to the PROC statement and specifying 16 blanks, you remove all the lines from around the table. If you have access to a laser printer you can also use characters that form "solid" lines around your table.

SUBTOTALING

Subtotaling is probably not a beginning tutorial topic, but here is an example anyway.

```
TABLE
 ORG*(LOC ALL='**Subtotal') ALL='Dept Total',
 SCORE='Average Final Scores'*MEAN=' '*F=6.1
 /  RTS=25    BOX=SCORE
    ROW=FLOAT    MISSTEXT='None';
```

Average Final Scores	
Department	Location	
	A	None
	**Subtotal	None
Energy	Location	
		None
	A	84.4
	B	96.4
	**Subtotal	88.4
Mgt S	Location	
	A	73.4
	B	85.4
	C	84.8
	**Subtotal	82.6
Power	Location	
	A	89.0
	B	70.7
	C	79.8
	**Subtotal	81.9
Dept Total		83.7

The only real trick is the nesting of ALL in the row expression.

IN SUMMARY

You have seen several reasons to use PROC TABULATE over other methods:

- **displays statistics in hierarchical tables**
- **provides a concise and powerful control language**
- **provides a greater degree of flexibility and complexity in classification hierarchies than MEANS, FREQ, or SUMMARY**
- **provides a very flexible mechanism for titling and formatting**

The base SAS statements required are:

- **PROC TABULATE DATA=sasdataset options;**
  options: MISSING NOSEPS FORMAT= ORDER= FORMCHAR=
- **CLASS variables;** classification variables: character or numeric
- **VAR variables;** analysis variables: numeric only
- **KEYLABEL keyword='text';**
- **TABLE page-expression,**
  row-expression,
  column-expression
  / options;

- Others used: LABEL, FORMAT, BY, TITLEn

The key to using TABULATE is the **table expression**, which consists of a combination of operators and operands:

Operators

- * for nesting
- space for concatenation

Operands

- class variables(from CLASS stmt) and ALL
- analysis variables(from VAR stmt)
- () for grouping
- statistics keywords
- title specification(='text')
- format specification(F=w.d)

Standard statistics names from MEANS or SUMMARY are:

- N, NMISS, MEAN, STD, MIN, MAX, SUM, etc.
- Plus two new ones: PCTN and PCTSUM

ACKNOWLEDGEMENTS

For a complete discussion of TABULATE and its uses see the "SAS Guide to TABULATE Processing, Second Edition" and the "SAS Language and Procedures, Usage, Version 6, First Edition", chapter 25, "Creating Summary Tables". There is a very good article in the first issue of "Observations, The Technical Journal for SAS Software Users", vol. 1, no. 1, entitled "Computing Percentages with PROC TABULATE" by Tina Keene.

Lauren Haworth has written a book entitled "PROC TABULATE By Example" that is full of all kinds of examples for all kinds of applications and well worth the money.

AUTHOR

If you have any questions or comments, please write or call:

Dan Bruns
Tennessee Valley Authority
1101 Market Street (MP 2B)
Chattanooga, TN 37402
423/751-6430 Fax 423/751-3163
Email: debruns@tva.gov