

The Utter Simplicity of the TABULATE Procedure

Dan Bruns, Tennessee Valley Authority, Chattanooga, TN

Before TABULATE

So many times when you first learn some new software product, you simply do what you are told or can find out 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 dataset we will be working with.

```

CONTENTS PROCEDURE

Data Set Name: SASDATA.CLASS              Observations:
Member Type:   DATA                      Variables:
Engine:        V606                       Indexes:
Created:       9:14 Wednesday, Sep 19, 1990 Observation Length:
Last Modified: 11:40 Tuesday, Feb 5, 1991 Deleted Observations:
Data Set Type:                               Compressed:
                                           Reuse Space:

-----Engine/Host Dependent Information-----
Data Set Page Size:      8704
Number of Data Set Pages: 1
Physical Name:           ATX.DSS.D8921217.SASDATA
Release Created:         6.06
Release Last Modified:   6.06
Created by:              SASVST06
Last Modified by:       SSPDEB
Subextents:             1
Total Blocks Used:      1

-----Alphabetic List of Variables and Attributes-----
#   Variable   Type   Len   Pos   Format   Informat   Label
4   DATE       Num    8     32   DATES.   MMDDYY6.   Class Date
2   LOC        Char    1     25                   Location
1   NAME       Char   25     0                   Name
3   ORG        Char    6     26                   Org
5   SCORE      Num    8     40   S.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(Orig)	DATE(Class Date)				
Frequency Percent Row Pct Col Pct	07APR	03MAY	22JUN	12OCT	Total
Energy	1 3.70 12.50 20.00	2 7.41 25.00 50.00	1 3.70 12.50 11.11	4 14.81 50.00 44.44	8 29.63
Mgt S	1 3.70 10.00 20.00	1 3.70 10.00 25.00	4 14.81 40.00 44.44	4 14.81 40.00 44.44	10 37.04
Power	5 11.11 33.33 60.00	1 3.70 11.11 25.00	4 14.81 44.44 44.44	1 3.70 11.11 11.11	9 33.33
Total	5 18.52	4 14.81	9 33.33	9 33.33	27 100.00

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 is only produced as 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 which has produced even more powerful and flexible features. 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 enhanced to allow you to specify not only what you want each cell to be but 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 dataset, and later you'll see other options to control to order of the output, define the default format, and control the characters used to "draw" the box around the cells. 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 three tables produced show a very simple yet powerful use of TABULATE. Although our dataset 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	4.0
Mgt S	1.00	1.00	4.00	4.0
Power	1.00	1.00	4.00	1.0

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	3.0
Mgt S	1.00	.	.	1.0
Power	1.00	1.00	1.00	1.0

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	5.00
Power	5.00	.

The above output shows the results of a page dimension.

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	1	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 performed by coding an asterisk (*) between the variables you want to break-down even further. Here is an example of nesting columns.

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

Notice the use of parenthesis to aid in clarification. Later they are also used to group expressions just like in arithmetic.

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

	Class Date	Location		
		A	B	C
		N	N	N
Org				
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	.	.

You'll notice that TABULATE uses the same amount of space to label the rows, regardless of the levels of nesting. The above output is alright 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.

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

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.

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

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	Class Date			
Energy	07APR	1	1	1
	03MAY	1	1	1
	22JUN	1	1	1
	12OCT	1	1	1
Mgt S	07APR	1	1	1
	03MAY	1	1	1
	22JUN	1	1	1
	12OCT	1	1	1
Power	07APR	1	1	1
	03MAY	1	1	1
	22JUN	1	1	1
	12OCT	1	1	1

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

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

		Location			Class Date			
		A	B	C	07-101-122-112	101-122-112-112	122-112-112-112	112-112-112-112
		N	N	N	N	N	N	N
Org	Class Date							
Energy	07APR	1	1	1	1	1	1	1
	03MAY	1	1	1	1	1	1	1
	22JUN	1	1	1	1	1	1	1
	12OCT	1	1	1	1	1	1	1
Mgt S	07APR	1	1	1	1	1	1	1
	03MAY	1	1	1	1	1	1	1
	22JUN	1	1	1	1	1	1	1
	12OCT	1	1	1	1	1	1	1
Power	07APR	1	1	1	1	1	1	1
	03MAY	1	1	1	1	1	1	1
	22JUN	1	1	1	1	1	1	1
	12OCT	1	1	1	1	1	1	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			
		1A	1B	1C	107APR	103MAY	122JUN	112OCT
		N	N	N	N	N	N	N
Org	Class Date							
Energy	07APR	1	1	1	1	1	1	1
	03MAY	1	1	1	1	1	1	1
	22JUN	1	1	1	1	1	1	1
	12OCT	1	1	1	1	1	1	1
Mgt S	07APR	1	1	1	1	1	1	1
	03MAY	1	1	1	1	1	1	1
	22JUN	1	1	1	1	1	1	1
	12OCT	1	1	1	1	1	1	1
Power	07APR	1	1	1	1	1	1	1
	03MAY	1	1	1	1	1	1	1
	22JUN	1	1	1	1	1	1	1
	12OCT	1	1	1	1	1	1	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	Class Date		
Energy	07APR	88.40	
	03MAY	88.40	
	22JUN	88.40	
	12OCT	88.40	
Mgt S	07APR	82.56	
	03MAY	82.56	
	22JUN	82.56	
	12OCT	82.56	
Power	07APR	81.88	
	03MAY	81.88	
	22JUN	81.88	
	12OCT	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	Class Date				
Energy	07APR	6.00	88.40	100.00	24.0
	03MAY	6.00	88.40	100.00	24.0
	22JUN	6.00	88.40	100.00	24.0
	12OCT	6.00	88.40	100.00	24.0
Mgt S	07APR	10.00	82.56	99.40	40.0
	03MAY	10.00	82.56	99.40	40.0
	22JUN	10.00	82.56	99.40	40.0
	12OCT	10.00	82.56	99.40	40.0
Power	07APR	9.00	81.88	99.10	36.0
	03MAY	9.00	81.88	99.10	36.0
	22JUN	9.00	81.88	99.10	36.0
	12OCT	9.00	81.88	99.10	36.0

Notice here that by grouping with parenthesis you can get more than one statistic for a given variable. The way to look at this 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 specified as you'll see later.

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

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

Here again you can change the look of the table by simply moving the grouping to the row expression. Notice the statistics are not of ORG but simply 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
Org				
Energy	4.0	84.4	2.0	96.4
Mgt S	2.0	75.4	1.0	85.4
Power	4.0	89.0	2.0	70.7

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;

	Location	Final Exam Score			
		N	MEAN	MAX	PCTN
Org					
Energy	A	4.0	84.4	93.0	16.0
	B	2.0	96.4	100.0	8.0
Mgt S	A	2.0	75.4	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 MEAN MAX PCTN)*F=5.1;

		Final Exam Score			
		N	MEAN	MAX	PCTN
Org					
Energy		4.0	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
Location					
A		10.0	84.0	99.4	40.0
B		5.0	83.9	100.0	20.0
C		10.0	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
Org				
Energy	4.0	84.4	2.0	96.4
Mgt S	2.0	75.4	1.0	85.4
Power	4.0	89.0	2.0	70.7
ALL	10.0	84.0	5.0	83.9

What in the world happened in this last example? There's no variable named ALL in the dataset? That's right but ALL is kind of like a builtin 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 divide by the number of tables entries; it accumulates each observations value and divides by the number of observations.

Titles and Labels

You have seen that to have TABULATE put descriptive title 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 dataset 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 16';
 KEYLABEL
 N = 'Count'
 MEAN = 'Mean'
 ALL = 'Total';

SUBJ 16	Location							
	A		B		C		Row Totals	
	Final Exam Score	Final Exam Score	Final Exam Score	Final Exam Score	Final Exam Score	Final Exam Score	Final Exam Score	
	Count	Mean	Count	Mean	Count	Mean	Count	Mean
Orgs								
Energy	4.0	84.4	2.0	96.4			6.0	88.4
Mgt S	2.0	73.4	1.0	85.4	7.0	84.8	10.0	82.6
Power	4.0	89.0	2.0	70.7	3.0	79.8	9.0	81.9
Total	10.0	84.0	5.0	85.9	10.0	81.2	25.0	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 NOEPPS ;
  CLASS ORG LOC DATE;
  VAR SCORE;
/*  TITLES & LABELS */
  TABLE
    ORG ALL='— Totals —',
    (LOC ALL='Row Totals')
    *(SCORE*MEAN='*F=5.1)
  / BOX=SCORE ROW=FLOAT MISSTEXT='None';
```

Final Exam Score	Location			Row
	A	B	C	Totals
Final	Final	Final	Final	Final
Exam	Exam	Exam	Exam	Exam
Score	Score	Score	Score	Score
Org				
Energy	84.4	96.4	None	88.4
Mst S	73.4	85.4	84.8	82.4
Power	89.0	70.7	79.8	81.9
--- Totals ---	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;
```

TABLE

```

ORG ALL = '— Totals —',
(LOC ALL = 'Row Totals')
*(SCORE = '*' MEAN = '*' F = 6.1)
/ BOX = SCORE ROW = FLOAT MISSTEXT = 'None';

```

Final Exam Score	Location			Row Totals
	A	B	C	
Org				
Energy	84.4	96.4	None	88.4
Hgt S	73.4	85.4	84.8	82.6
Power	89.0	70.7	79.9	81.9
--- Totals ---	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.

Subtotalling

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

TABLE

```

ORG*(LOC ALL='Loc Subtotal') ALL='Org Total',
SCORE='Average Final Exam Score'*MEAN='*F=6.1
/RTS=25      BOX=SCORE      ROW=FLOAT
MISSTEXT='None';

```

Final Exam Score		Average Final Exam Score
Org	Location	
Energy	A	84.4
	B	96.4
	Loc	
	Subtotal	88.4
Mgt S	Location	
	A	73.4
	B	85.4
	C	84.8
	Loc	
	Subtotal	82.6
Power	Location	
	A	89.0
	B	70.7
	C	79.8
	Loc	
	Subtotal	81.9
Org Total		83.7

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

Percentages

Percentages is probably not a beginning tutorial topic either, but let's look at a few simple examples. In its simplest form the PCTN or PCTSUM is just another statistic like N or MEAN you can request.

```
PROC TABULATE DATA=CLASS FORMAT=6.1;
  CLASS ORG LOC DATE;
  VAR SCORE;
  TABLE ORG, (LOC ALL)*(N*F=3.0 PCTN);
```

	Location							
	A		B		C		ALL	
	N	PCTN	N	PCTN	N	PCTN	N	PCTN
Org								
Energy	61	22.21	21	7.41			81	29.61
Mgt S	21	7.41	11	3.71	71	25.91	101	37.01
Power	41	14.81	21	7.41	31	11.11	91	33.31

Unless specified the percentage is computed based on all the observations in the dataset. Notice that the PCTN under the ALL column does not add up to 100 due to rounding.

To specify how the percentage is computed you simply attach a denominator specification to PCTN or PCTSUM using the inequality signs less-than (<) and greater-than(>). The real trick to

understanding how the denominator specification works is to remember you are telling TABULATE what the denominator is to divide into the cell N or SUM value.

TABLE ORG, (LOC ALL)*(N*F=3.0 PCTN < ORG >);

	Location							
	A		B		C		ALL	
	N	PCTN	N	PCTN	N	PCTN	N	PCTN
Org								
Energy	61	50.01	21	40.01			81	29.61
Mgt S	21	16.71	11	20.01	71	70.01	101	37.01
Power	41	33.31	21	40.01	31	30.01	91	33.31

The above example shows the row expression in the denominator specification. Notice that none of the counts(N) have changed but the PCTN values have because the denominator has changed from the entire dataset (27 observations) to all the observations for ORG within that locations value. Observe that since PCTN is nested in LOC that the denominator specification is saying to divide each cell under that location by the total of observations that are in that location. So why do you specify the row expression? Because that is simply telling TABULATE which number of observations to total. So, in the above example, we see that location A cells are divided by 12, the total of all the ORG observations in that location. For location B we see each cell is divided by 5, the total of all the ORG observations in that location. And for location C we see each cell is divided by 10, the total of all the ORG observations in that location.

Here is a handy rule-of-thumb:

To get percentages by column, use the row expression; and to get percentages by row, use the column expression.

TABLE ORG, (LOC ALL)*(N*F=3.0 PCTN < LOC ALL >);

	Location						ALL	
	A		B		C			
	N	PCTN	N	PCTN	N	PCTN	N	PCTN
Org								
Energy	61	75.01	21	25.01			81	100.01
Hst S								
	21	20.01	11	10.01	71	70.01	101	100.01
Power	41	44.41	21	22.21	31	33.31	91	100.01

Notice in the above example that the entire column expression is coded as the denominator specification. If you don't, strange results of even errors can occur.

TABLE ORG,
(LOC ALL)*SCORE*
(SUM*F=5.1 PCTSUM < LOC ALL >);

	Location				ALL
	A	B	C		
	Final Exam Score	Final Exam Score	Final Exam Score	Final Exam Score	
Org					
Energy	337.61	63.71192.81	26.31		530.61
Hst S	146.71	17.8185.41	10.31593.51	71.91825.61	100.1
Power	1356.21	48.31161.31	19.21239.41	32.51736.91	100.1

This example is just to show that the PCTSUM works in the same way. (The summing of exams scores doesn't seem to make much sense, but it is the only numeric variable in the dataset.)

The following examples show that the same rules apply for nesting variables and using ALL.

PROC TABULATE DATA=CLASS FORMAT=6.1 NOSEPS;
CLASS ORG LOC DATE;
VAR SCORE;

TABLE ORG*DATE,
(LOC ALL)*(N*F=3.0 PCTN < ORG*DATE >);

		Location							
		A		B		C		ALL	
		N	PCTN	N	PCTN	N	PCTN	N	PCTN
Org	Class								
Energy	Date								
	07APR	11	8.31	11		11		11	3.71
	03MAY	11	8.31	11	20.01	11		21	7.41
	22JUN	11	8.31					11	3.71
	12OCT	31	25.01	11	20.01	11		41	14.81
Hst S	07APR	11	8.31	11		11		11	3.71
	03MAY			11	20.01	11		11	3.71
	22JUN	11				41	40.01	41	14.81
	12OCT	11	8.31			31	30.01	41	14.81
Power	07APR	11	8.31	21	40.01	11		31	11.11
	03MAY	11	8.31			11		11	3.71
	22JUN	11	8.31			31	30.01	41	14.81
	12OCT	11	8.31			11		11	3.71

TABLE ORG*DATE,
(LOC ALL)*(N*F=3.0 PCTN < LOC ALL >);

		Location							
		A		S		C		ALL	
		N	PCTN	N	PCTN	N	PCTN	N	PCTN
Org	Class								
Energy	Date								
	07APR	11	100.01	11	50.01	11	50.01	11	100.01
	03MAY	11	50.01	11	50.01	11	50.01	21	100.01
	22JUN	11	100.01	11	25.01	11	25.01	11	100.01
	12OCT	31	75.01	11	25.01	11	25.01	41	100.01
Hst S	07APR	11	100.01	11	50.01	11	50.01	11	100.01
	03MAY	11	50.01	11	100.01	11	100.01	11	100.01
	22JUN	11	100.01	11	25.01	11	25.01	41	100.01
	12OCT	11	25.01	11	25.01	11	75.01	41	100.01
Power	07APR	11	33.31	21	66.71	11	66.71	31	100.01
	03MAY	11	100.01	11	25.01	11	25.01	11	100.01
	22JUN	11	25.01	11	25.01	11	75.01	41	100.01
	12OCT	11	100.01	11	25.01	11	25.01	11	100.01

TABLE ORG*DATE ALL,
(LOC ALL)*(N*F=3.0 PCTN < ORG*DATE ALL >);

		Location							
		A		B		C		ALL	
		N	PCTN	N	PCTN	N	PCTN	N	PCTN
Org	Class								
Energy	Date								
	07APR	11	8.31					11	3.71
	03MAY	11	8.31		20.01			21	7.41
	22JUN	11	8.31					11	3.71
	12OCT	31	25.01	11	20.01			41	14.81
Hst S	07APR	11	8.31					11	3.71
	03MAY			11	20.01			11	3.71
	22JUN					41	40.01	41	14.81
	12OCT	11	8.31			31	30.01	41	14.81
Power	07APR	11	8.31	21	40.01			31	11.11
	03MAY	11	8.31					11	3.71
	22JUN	11	8.31			31	30.01	41	14.81
	12OCT	11	8.31					11	3.71
ALL		121	100.01	51	100.01	101	100.01	271	100.01

These last two examples are without a doubt beyond the scope of Beginning Tutorials.

TABLE ORG*DATE,
(LOC ALL)*(N*F=3.0 PCTN < DATE >);

		Location							
		A		B		C		ALL	
		N	PCTN	N	PCTN	N	PCTN	N	PCTN
Org Energy	Class								
	Date								
	07APR	11	16.71					11	12.51
	03MAY		16.71	11	50.01			21	25.01
	22JUN		16.71					11	12.51
Hst S	12OCT	31	50.01	11	50.01			41	50.01
	07APR	11	50.01					11	10.01
	03MAY			11	100.01			11	10.01
	22JUN					41	57.11	41	60.01
	12OCT	11	50.01			31	42.91	41	60.01
Power	07APR	11	25.01	21	100.01			31	33.31
	03MAY		25.01					11	11.11
	22JUN	11	25.01			31	100.01	41	66.61
	12OCT	11	25.01					11	11.11

The same rules apply for denominator specifications that are NOT the entire expression. With the denominator specification of

DATE, TABULATE will use the total number of observations for all dates in that column as the denominator. But since DATE is nested within ORG, it will only use those observations that belong to that ORG. So, in the above example the total number of observations for location A in ORG Energy is 6, which becomes the denominator for computing PCTN for all those dates.

TABLE ORG*DATE.
(LOC ALL)*(N*F=3.0 PCTN<ORG>);

		Location							
		A		B		C		ALL	
		N	PCTN	N	PCTN	N	PCTN	N	PCTN
Org Energy	Class Date								
	07APR	1	33.3					1	20.0
	03MAY	1	50.0	1	50.0			2	50.0
	22JUN	1	50.0					1	11.1
Inst S	12OCT	3	60.0	1	100.0			4	44.4
	07APR	1	33.3					1	20.0
	03MAY			1	50.0			1	25.0
	22JUN					4	57.1	4	44.4
Power	12OCT	1	20.0			3	100.0	4	44.4
	07APR	1	33.3	2	100.0			3	60.0
	03MAY	1	50.0					1	25.0
	22JUN	1	50.0			3	42.9	4	44.4
	12OCT	1	20.0					1	11.1

With the denominator specification of ORG, TABULATE will use the total number of observations for all organizations in that column as the denominator. But since ORG is nested with DATE, it will only use those observations that belong to that DATE. So, in the above example the total number of observations for location A with a date of 07APR is 3, which becomes the denominator for computing PCTN for that date in every ORG. And the total number of observations for location A with a date of 12OCT is 5, which becomes the denominator for computing PCTN for that date in every ORG.

In Summary

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

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

The base SAS statements required are:

- o PROC TABULATE DATA=sasdataset options;

options: MISSING FORMAT= ORDER= FORMCHAR= NOSEPS

- o CLASS variables:
 - classification variables: character or numeric
- o VAR variables:
 - analysis variables: numeric only
- o KEYLABEL keyword='text';
- o TABLE expression, expression, expression / options;
- o Others used: LABEL, FORMAT, BY, TITLEN

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

- o Operators
 - * for nesting
 - space for concatenation
- o Operands
 - () for grouping
 - class variables(from CLASS stmt) or ALL
 - analysis variables(from VAR stmt)
 - statistics keywords
 - title specification(='text')
 - format specification(F=w.d)
- o Standard statistics names from MEANS or SUMMARY
 - N, NMISS, MEAN, STD, MIN, MAX, SUM, etc.
- o Plus two new ones: PCTN and PCTSUM

For a complete discussion of TABULATE and its uses see the "SAS Guide to TABULATE Processing, Second Edition".

Acknowledgements

SAS is a registered trademark of the SAS Institute, Inc., Cary, NC

Author

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

Dan Bruns
Tennessee Valley Authority
150 West 11th Street
MP 2B
Chattanooga, TN 37401
(615) 751-6430