Paper 127-30

The Utter "Simplicity?" of the TABULATE Procedure

Dan Bruns, Chattanooga, TN

IN THE BEGINNING

Well, here we are again Tabulate fans. It never ceases to amaze me how many folks give up on using tabulate because they cannot quite figure out how to get it to produce the table they want. Once you learn the basics, it is quite simple to produce relatively simple reports/tables. But as you want to produce more complex tables, it does not seem to be quite as simple—especially percentages. During this workshop we will introduce you to some of the more advanced features TABULATE, i.e. percentages, multilevel formats, new statistics, new options, and maybe even a trick or two.

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:	27
Member Type:	DATA	Variables:	5
Engine:	V8xx	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

SOME BASICS

Here are few basic examples and their totally different looking outputs by simply changing where and how the variables are coded. If these are beyond your current proficiency with TABULATE, see my Beginning Tutorial and HandsOn Workshops papers entitled "The Utter Simplicity and Power of the TABULATE Procedure" in several of the previous SUGI proceedings and hang on to your hat because I'm starting from here and assuming you understand this much. (This will also give you a feel for the data.)

```
PROC TABULATE  DATA=CLASS ;
    CLASS ORG LOC DATE;
    VAR   SCORE;
    TABLE  ORG, LOC*SCORE*(N MEAN)*F=5.1;
RUN;
```

	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	89.1	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	89.1	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	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
Location				
A	10.0	84.1	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						All	
	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	4.0	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	89.1	2.0	70.7	3.0	79.8	9.0	81.9
All	10.0	84.1	5.0	83.9	10.0	83.3	25.0	83.7

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 built-in class variable that can be specified 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.

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 table entries; it accumulates each observations value and divides by the number of observations.

TITLES AND LABELS

You can see 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 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 the 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 30';
KEYLABEL
        N='Count'
        MEAN='Mean'
        ALL='Total' ;
```

SUGI 30	Location						Row Totals	
	A		B		C			
	Final Exam Score		Final Exam Score		Final Exam Score		Final Exam Score	
	Count	Mean	Count	Mean	Count	Mean	Count	Mean
Org								
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	89.1	2.0	70.7	3.0	79.8	9.0	81.9
Total	10.0	84.1	5.0	83.9	10.0	83.3	25.0	83.7

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

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

Final Exam Score	Location				Row Tota-ls
		A	B	C	
	Final Exam Score	Final Exam Score	Final Exam Score	Final Exam Score	Final Exam Score
Org					
	None	None	None	None	None
Energy	None	84.4	96.4	None	88.4
Mgt S	None	73.4	85.4	84.8	82.6
Power	None	89.1	70.7	79.8	81.9
--- Totals ---	None	84.1	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.

The one problem I had with this example was the MISSING applied to ***all*** variables and sometimes I was really only interested of a certain variable had missing values. Well, starting in Version 7 TABULATE now supports multiple CLASS statements with several options that let you control how each variable(s) is handled. Here are some of the new CLASS statement option:

- ASCENDING - specifies to sort the class variable levels in ascending order.
- DESCENDING - specifies to sort the class variable levels in descending order.
- EXCLUSIVE - excludes from the analysis all class variable values that are not found in the preloaded range of user-defined formats.
- GROUPINTERNAL - specifies not to apply formats to the class variables when TABULATE sorts the values to create combinations of class variables.
- MISSING - considers missing values as valid class variable levels.
- MLF - enables TABULATE to use the primary and secondary format labels for a given range or overlapping ranges to create the subgroup combinations when a multilabel format is assigned to a class variable.
- ORDER= - specifies the sort order for the levels of the class variables in the output.
- PRELOADFMT - specifies to preload all the formats for the class variables.
- MLF - allows you to make use of multiple labels when a multilabel format is assigned to a class variable in PROC FORMAT.

```
PROC FORMAT ;
    VALUE $LOCFMT
       'A'='Knoxville'
       'B'='Chattanooga'
       'C'='Nashville'
       'D'='Memphis';
       RUN;
PROC TABULATE  DATA=CLASS NOSEPS ;
    CLASS  LOC  / groupinternal;
    CLASS  DATE / MISSING ;
    CLASS  ORG  / DESCENDING MISSING;
    VAR   SCORE;
    Format loc $locfmt. ;
/*   TITLES & LABELS */
TABLE
    ORG ALL='--- Totals ---',
    (LOC ALL='Row Totals')
         *(SCORE*MEAN=' '*F=5.1)
    / BOX=SCORE  ROW=FLOAT
            MISSTEXT='None';
RUN;
```

Final Exam Score	Location			
	Knox-ville	Chat-tano-oga	Nash-ville	Row Tota-ls
	Final Exam Score	Final Exam Score	Final Exam Score	Final Exam Score
Org				
Power	89.1	70.7	79.8	81.9
Mgt S	73.4	85.4	84.8	82.6
Energy	84.4	96.4	None	88.4
	None	None	None	None
--- Totals ---	84.1	83.9	83.3	83.7

Will cover some of the other options later.

```
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';
RUN;
```

Final Exam Score	Location				
		A	B	C	Row Totals
Org					
	None	None	None	None	None
Energy	None	84.4	96.4	None	88.4
Mgt S	None	73.4	85.4	84.8	82.6
Power	None	89.1	70.7	79.8	81.9
--- Totals ---	None	84.1	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.

SUBTOTALING

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

```
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		Avera-ge 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	89.1
	B	70.7
	C	79.8
	Loc Subtotal	81.9
Org Total		83.7

Above we see the nesting of (LOC ALL) in ORG. That tells TABULATE to concatenate an ALL row after all the LOC rows for each ORG value.

PERCENTAGES

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);
RUN;
```

	Location						All	
	A		B		C			
	N	PctN	N	PctN	N	PctN	N	PctN
Org								
Energy	4	16.0	2	8.0	.	.	6	24.0
Mgt S	2	8.0	1	4.0	7	28.0	10	40.0
Power	4	16.0	2	8.0	3	12.0	9	36.0

Unless specified, the percentage is computed based on ***all*** the observations in the dataset.

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 N or SUM value, or what you are dividing by.***

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

	Location						All	
	A		B		C			
	N	PctN	N	PctN	N	PctN	N	PctN
Org								
Energy	4	40.0	2	40.0	.	.	6	24.0
Mgt S	2	20.0	1	20.0	7	70.0	10	40.0
Power	4	40.0	2	40.0	3	30.0	9	36.0

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 (25 observations) to all the observations for ORG within that columns (LOC) 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 number 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 10, 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. And for the ALL column we see each cell is divided by the total of all the ORG observations in all the locations.

Here is a handy rule-of-thumb:

> **To get percentages by column, use the row expression;**
> **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	4	66.7	2	33.3	.	.	6	100.0
Mgt S	2	20.0	1	10.0	7	70.0	10	100.0
Power	4	44.4	2	22.2	3	33.3	9	100.0

Notice in the above example that the entire column expression is coded as the denominator specification. If you don't, strange results or even errors can occur. As before, you are simply telling TABULATE which number of observations to total. So, in the above example, we see that organization 'Energy' cells are divided by 6, the total of all the LOC observations in that organization. For organization 'Mgt S' we see each cell is divided by 10, the total of all the LOC observations in that organization. For organization 'Power' we see each cell is divided by 9, the total of all the LOC observations in that organization. And for the ALL column we see each cell is divided by the total of all the LOC observations in that organization, thus the 100 percent.

```
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	
	Sum	PctSum	Sum	PctSum	Sum	PctSum	Sum	PctSum
Org								
Energy	337.6	63.7	192.8	36.3	.	.	530.4	100.0
Mgt S	146.7	17.8	85.4	10.3	593.5	71.9	825.6	100.0
Power	356.2	48.3	141.3	19.2	239.4	32.5	736.9	100.0

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>);
RUN;
```

		Location						All	
		A		B		C			
		N	PctN	N	PctN	N	PctN	N	PctN
Org	Class								
Energy	Date								
	03MAY	1	10.0	1	20.0	.	.	2	8.0
	22JUN	1	10.0	.	.	.	.	1	4.0
	12OCT	2	20.0	1	20.0	.	.	3	12.0
Mgt S	07APR	1	10.0	.	.	.	.	1	4.0
	03MAY	.	.	1	20.0	.	.	1	4.0
	22JUN	.	.	.	.	4	40.0	4	16.0
	12OCT	1	10.0	.	.	3	30.0	4	16.0
Power	07APR	1	10.0	2	40.0	.	.	3	12.0
	03MAY	1	10.0	.	.	.	.	1	4.0
	22JUN	1	10.0	.	.	3	30.0	4	16.0
	12OCT	1	10.0	.	.	.	.	1	4.0

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

		Location						All	
		A		B		C			
		N	PctN	N	PctN	N	PctN	N	PctN
Org	Class								
Energy	Date								
	03MAY	1	50.0	1	50.0	.	.	2	100.0
	22JUN	1	100.0	.	.	.	.	1	100.0
	12OCT	2	66.7	1	33.3	.	.	3	100.0
Mgt S	07APR	1	100.0	.	.	.	.	1	100.0
	03MAY	.	.	1	100.0	.	.	1	100.0
	22JUN	.	.	.	.	4	100.0	4	100.0
	12OCT	1	25.0	.	.	3	75.0	4	100.0
Power	07APR	1	33.3	2	66.7	.	.	3	100.0
	03MAY	1	100.0	.	.	.	.	1	100.0
	22JUN	1	25.0	.	.	3	75.0	4	100.0
	12OCT	1	100.0	.	.	.	.	1	100.0

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

		Location						All	
		A		B		C			
		N	PctN	N	PctN	N	PctN	N	PctN
Org	Class								
Energy	Date								
	03MAY	1	10.0	1	20.0	.	.	2	8.0
	22JUN	1	10.0	.	.	.	.	1	4.0
	12OCT	2	20.0	1	20.0	.	.	3	12.0
Mgt S	07APR	1	10.0	.	.	.	.	1	4.0
	03MAY	.	.	1	20.0	.	.	1	4.0
	22JUN	.	.	.	.	4	40.0	4	16.0
	12OCT	1	10.0	.	.	3	30.0	4	16.0
Power	07APR	1	10.0	2	40.0	.	.	3	12.0
	03MAY	1	10.0	.	.	.	.	1	4.0
	22JUN	1	10.0	.	.	3	30.0	4	16.0
	12OCT	1	10.0	.	.	.	.	1	4.0
All		10	100.0	5	100.0	10	100.0	25	100.0

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

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

		Location						All	
		A		B		C			
		N	PctN	N	PctN	N	PctN	N	PctN
Org	Class								
Energy	Date								
	03MAY	1	25.0	1	50.0	.	.	2	33.3
	22JUN	1	25.0	.	.	.	.	1	16.7
	12OCT	2	50.0	1	50.0	.	.	3	50.0
Mgt S	07APR	1	50.0	.	.	.	.	1	10.0
	03MAY	.	.	1	100.0	.	.	1	10.0
	22JUN	.	.	.	.	4	57.1	4	40.0
	12OCT	1	50.0	.	.	3	42.9	4	40.0
Power	07APR	1	25.0	2	100.0	.	.	3	33.3
	03MAY	1	25.0	.	.	.	.	1	11.1
	22JUN	1	25.0	.	.	3	100.0	4	44.4
	12OCT	1	25.0	.	.	.	.	1	11.1

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. You can also see the total number of observations for location B in ORG 'Mgt S' is 1, which becomes the denominator for computing PCTN for all those dates, thus the 100 percent on 03MAY. The total number of observations for the ALL column in ORG 'Power' is 9, which becomes the denominator for computing PCTN for all those dates.

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

		Location						All	
		A		B		C			
		N	PctN	N	PctN	N	PctN	N	PctN
Org	Class Date								
Energy	03MAY	1	50.0	1	50.0	.	.	2	50.0
	22JUN	1	50.0	.	.	.	.	1	11.1
	12OCT	2	50.0	1	100.0	.	.	3	37.5
Mgt S	07APR	1	50.0	.	.	.	.	1	25.0
	03MAY	.	.	1	50.0	.	.	1	25.0
	22JUN	.	.	.	.	4	57.1	4	44.4
	12OCT	1	25.0	.	.	3	100.0	4	50.0
Power	07APR	1	50.0	2	100.0	.	.	3	75.0
	03MAY	1	50.0	.	.	.	.	1	25.0
	22JUN	1	50.0	.	.	3	42.9	4	44.4
	12OCT	1	25.0	.	.	.	.	1	12.5

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 in that location, thus the 33.3 percent for each one with a count of 1. 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 that location, thus the 20 percent for each one with a count of 1. The total number of observations for location B with a date of 03MAY is 2, which becomes the denominator for computing PCTN for that date in every ORG in that location, thus the 50 percent for each one with a count of 1. The total number of observations for the ALL column with a date of 03MAY is 4, which becomes the denominator for computing PCTN for that date in every ORG, thus the 25 percent for each one with a count of 1.

The ALL in the denominator specification gave me a real hard time at first until I discovered it is really only needed to satisfy the table expression expansion. Typically ALL is used to do some sort of totaling and is thus concatenated not nested. So, all (ha! ha!) you have to do is include it in your denominator as shown below.

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

		Location						All	
		A		B		C			
		N	PctN	N	PctN	N	PctN	N	PctN
Org	Class								
Energy	Date								
	03MAY	1	50.0	1	50.0	.	.	2	50.0
	22JUN	1	50.0	.	.	.	.	1	11.1
	12OCT	2	50.0	1	100.0	.	.	3	37.5
Mgt S	07APR	1	50.0	.	.	.	.	1	25.0
	03MAY	.	.	1	50.0	.	.	1	25.0
	22JUN	.	.	.	.	4	57.1	4	44.4
	12OCT	1	25.0	.	.	3	100.0	4	50.0
Power	07APR	1	50.0	2	100.0	.	.	3	75.0
	03MAY	1	50.0	.	.	.	.	1	25.0
	22JUN	1	50.0	.	.	3	42.9	4	44.4
	12OCT	1	25.0	.	.	.	.	1	12.5
All		10	100.0	5	100.0	10	100.0	25	100.0

If you leave it out of the denominator specification, you will get the messages:

ERROR: PCTN base is not in table.
ERROR: A PCTN crossing has no denominator.

Where the ALL gets real complicated is when you nest the ALLs in groupings, then you will need to expand the "crossings" as the SAS manuals indicate to be sure you get the proper denominator.

To get a better feel for the use of percentages, let's use the subtotaling example from earlier and add a subtotal percentage.

```
TABLE
    ORG*(LOC ALL='Loc Subtotal')
       ALL='Org Total',
    SCORE*(SUM*F=6.1 PCTSUM<LOC ALL>)
    / RTS=25 BOX=SCORE
       ROW=FLOAT MISSTEXT='None';
```

Final Exam Score		Final Exam Score	
		Sum	PctSum
Org	Location		
Energy	A	337.6	63.7
	B	192.8	36.3
	Loc		
	Subtotal	530.4	100.0
Mgt S	Location		
	A	146.7	17.8
	B	85.4	10.3
	C	593.5	71.9
	Loc		
	Subtotal	825.6	100.0
Power	Location		
	A	356.2	48.3
	B	141.3	19.2
	C	239.4	32.5
	Loc		
	Subtotal	736.9	100.0
Org Total		2092.9	100.0

It is amazing that after all these years of SAS programmers trying….**and trying….*and trying*** to figure out how to make the @!#$%^**& TABULATE denominator specification work right, ***they make a statistic to do it for you!***

```
PROC TABULATE  DATA=CLASS
       FORMAT=7.1 NOSEPS;
    CLASS  LOC  / DESCENDING ;
    CLASS  DATE / MISSING ;
    CLASS  ORG  / MISSING;
    VAR   SCORE;
    TABLE  ORG*DATE ALL,
        (LOC ALL)*(N*F=3.0 COLPCTN);
RUN;
```

		Location						All	
		C		B		A			
		N	ColPctN	N	ColPctN	N	ColPctN	N	ColPctN
Org	Class								
	Date								
	.	.	.	.	.	1	9.1	1	3.8
Energy	03MAY	.	.	1	20.0	1	9.1	2	7.7
	22JUN	.	.	.	.	1	9.1	1	3.8
	12OCT	.	.	1	20.0	2	18.2	3	11.5
Mgt S	07APR	.	.	.	.	1	9.1	1	3.8
	03MAY	.	.	1	20.0	.	.	1	3.8
	22JUN	4	40.0	.	.	.	.	4	15.4
	12OCT	3	30.0	.	.	1	9.1	4	15.4
Power	07APR	.	.	2	40.0	1	9.1	3	11.5
	03MAY	.	.	.	.	1	9.1	1	3.8
	22JUN	3	30.0	.	.	1	9.1	4	15.4
	12OCT	.	.	.	.	1	9.1	1	3.8
All		10	100.0	5	100.0	11	100.0	26	100.0

PROC TABULATE supports all the new statistics that PROC MEANS and SUMMARY also now supports:

- COLPCTN
- COLPCTSUM
- MEDIAN
- P1
- P5
- P10
- P90
- P95
- P99
- PAGEPCTN
- PAGEPCTSUM
- Q1
- Q3
- QRANGE
- REPPCTN
- REPPCTSUM
- ROWPCTN
- ROWPCTSUM

```
PROC TABULATE  DATA=CLASS
       OUT=CLASSOUT
       FORMAT=5.1 NOSEPS;
    CLASS ORG LOC DATE;
    VAR   SCORE;
    TABLE  ORG*DATE all,
    SCORE*(LOC ALL)*
       (N*F=2. MEDIAN*F=5.1
          COLPCTN*F=3. REPPCTN*F=3. )
   /RTS=15 CONDENSE NOCONTINUED;
RUN;
```

		Final Exam Score															
		Location												All			
		A				B				C							
		N	Median	ColPctN	RepPctN	N	Median	ColPctN	RepPctN	N	Median	ColPctN	RepPctN	N	Median	ColPctN	RepPctN
Org Energy	Class Date																
	03MAY	1	88.9	10	4	1	92.8	20	4	.	.	.	.	2	90.9	8	8
	22JUN	1	69.9	10	4	.	.	.	.	.	.	.	.	1	69.9	4	4
	12OCT	2	89.4	20	8	1	100.0	20	4	.	.	.	.	3	93.0	12	12
Mgt S	07APR	1	99.4	10	4	.	.	.	.	.	.	.	.	1	99.4	4	4
	03MAY	.	.	.	.	1	85.4	20	4	.	.	.	.	1	85.4	4	4
	22JUN	.	.	.	.	.	.	.	.	4	87.7	40	16	4	87.7	16	16
	12OCT	1	47.3	10	4	.	.	.	.	3	86.8	30	12	4	76.7	16	16
Power	07APR	1	99.1	10	4	2	70.7	40	8	.	.	.	.	3	90.0	12	12
	03MAY	1	93.5	10	4	.	.	.	.	.	.	.	.	1	93.5	4	4
	22JUN	1	70.1	10	4	.	.	.	.	3	81.2	30	12	4	75.7	16	16
	12OCT	1	93.5	10	4	.	.	.	.	.	.	.	.	1	93.5	4	4
All		10	91.0	100	40	5	90.0	100	20	10	84.0	100	40	25	88.9	100	100

ADDITIONAL FEATURES

There are several additional features that we just do not time to cover. And until I can fully understand (and try!) all the new features of TABULATE, I will just list them for you here. Many of them are in response to the SASware Ballot.

The PROC TABULATE statement supports these new options:

- CLASSDATA= - specifies a data set that contains the combinations of class variable values to include in analysis.
- CONTENTS= - allows you to name the link in the HTML table of contents that points to the ODS output of the first table produced.
- EXCLNPWGT - excludes observations with nonpositive weights from the analysis.
- EXCLUSIVE - excludes from the analysis all class variable combinations that are not in the CLASSDATA= data set.
- NOTRAP - disables trapping mathematical errors due to overflow.
- OUT= - names the output data set.
- QMARKERS= - specifies the default number of markers to use for the P2 (fixed space) quantile estimation method.
- QMETHOD - specifies the method to process the input data to compute quantiles.
- QNTLDEF= - specifies the mathematical definition used to compute quantiles.
- TRAP - enables trapping mathematical errors due to overflow.

In the TABLE statement, the following options have been enhanced:

- CONDENSE - prints multiple logical pages on a physical page.
- CONTENTS= - allows you to name the link in the HTML table of contents that points to the ODS output of the table produced using the TABLE statement.
- NOCONTINUED - suppresses the printing of the "(Continued)" continuation message for tables that span physical pages.

PRELOADED FORMATS

What an interesting feature. I was not sure I could find a use for this, but then it dawned on me. Our data only has three locations, but there are actually four……our data file just did not happen to have location D in it (Memphis). By using the PRELOADFMT option on the Class statement, I can now force the table show Memphis even thought there are not observations containing it.

```
PROC FORMAT ;
    VALUE $LOCFMT
       'A'='Knoxville'
       'B'='Chattanooga'
       'C'='Nashville'
       'D'='Memphis';
       RUN;

PROC TABULATE  DATA=CLASS
       FORMAT=7.1 NOSEPS;
    CLASS  LOC
       / DESCENDING PRELOADFMT;
    CLASS  DATE / MISSING ;
    CLASS  ORG  / MISSING;
    Format loc  $locfmt. ;
    VAR   SCORE;
    TABLE  ORG*DATE ALL,
         (LOC)*(N*F=3.0 COLPCTN)
    / PRINTMISS ;
RUN;
```

		Location							
		Memphis		Nashville		Chattanooga		Knoxville	
		N	ColPctN	N	ColPctN	N	ColPctN	N	ColPctN
Org	Class								
	Date								
	.	.	.	.	0.0	.	0.0	1	9.1
	07APR	.	.	.	0.0	.	0.0	.	0.0
	03MAY	.	.	.	0.0	.	0.0	.	0.0
	22JUN	.	.	.	0.0	.	0.0	.	0.0
	12OCT	.	.	.	0.0	.	0.0	.	0.0
Energy	.	.	.	.	0.0	.	0.0	.	0.0
	07APR	.	.	.	0.0	.	0.0	.	0.0
	03MAY	.	.	.	0.0	1	20.0	1	9.1
	22JUN	.	.	.	0.0	.	0.0	1	9.1
	12OCT	.	.	.	0.0	1	20.0	2	18.2
Mgt S	.	.	.	.	0.0	.	0.0	.	0.0
	07APR	.	.	.	0.0	.	0.0	1	9.1
	03MAY	.	.	.	0.0	1	20.0	.	0.0
	22JUN	.	.	4	40.0	.	0.0	.	0.0
	12OCT	.	.	3	30.0	.	0.0	1	9.1
Power	.	.	.	.	0.0	.	0.0	.	0.0
	07APR	.	.	.	0.0	2	40.0	1	9.1
	03MAY	.	.	.	0.0	.	0.0	1	9.1
	22JUN	.	.	3	30.0	.	0.0	1	9.1
	12OCT	.	.	.	0.0	.	0.0	1	9.1
All		.	.	10	100.0	5	100.0	11	100.0

```
PROC CONTENTS  DATA=CLASSOUT;
       RUN;
```

The CONTENTS Procedure

Data Set Name:	SASUSER.CLASSOUT	Observations:	31
Member Type:	DATA	Variables:	10
Engine:	V8	Indexes:	0
Created:	8:53 Friday, October 13, 20xx	Observation Length:	72
Last Modified:	8:53 Friday, October 13, 20xx	Deleted Observations:	0
Protection:		Compressed:	NO
Data Set Type:		Sorted:	NO
Label:			

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

#	Variable	Type	Len	Pos	Format	Label
5	_PAGE_	Num	8	8		Page for Observation
6	_TABLE_	Num	8	16		Table for Observation
4	_TYPE_	Char	3	63		Type of Observation
3	DATE	Num	8	0	DATE.	Class Date
2	LOC	Char	1	62		Location
1	ORG	Char	6	56		Org
8	SCORE_Median	Num	8	32		
7	SCORE_N	Num	8	24		
10	SCORE_PctN_000	Num	8	48		
9	SCORE_PctN_010	Num	8	40		

```
PROC PRINT  DATA=CLASSOUT; RUN;
```

Obs	ORG	LOC	DATE	_TYPE_	_PAGE_	_TABLE_	SCORE_N	SCORE_Median	SCORE_Pct_N_0	SCORE_Pct_N_0
1	Energy	A	03MAY80	111	1	1	1	88.90	10	.
2	Energy	B	03MAY80	111	1	1	1	92.80	20	.
3	Energy	A	22JUN80	111	1	1	1	69.90	10	.
4	Energy	A	12OCT80	111	1	1	2	89.40	20	.
5	Energy	B	12OCT80	111	1	1	1	100.00	20	.
6	Mgt S	A	07APR80	111	1	1	1	99.40	10	.
7	Mgt S	B	03MAY80	111	1	1	1	85.40	20	.
8	Mgt S	C	22JUN80	111	1	1	4	87.65	40	.
9	Mgt S	A	12OCT80	111	1	1	1	47.30	10	.
10	Mgt S	C	12OCT80	111	1	1	3	86.80	30	.
11	Power	A	07APR80	111	1	1	1	99.10	10	.
12	Power	B	07APR80	111	1	1	2	70.65	40	.
13	Power	A	03MAY80	111	1	1	1	93.50	10	.
14	Power	A	22JUN80	111	1	1	1	70.10	10	.
15	Power	C	22JUN80	111	1	1	3	81.20	30	.
16	Power	A	12OCT80	111	1	1	1	93.50	10	.
17	Energy	A	03MAY80	111	1	1	.	.	.	4
18	Energy	B	03MAY80	111	1	1	.	.	.	4
19	Energy	A	22JUN80	111	1	1	.	.	.	4
20	Energy	A	12OCT80	111	1	1	.	.	.	8
21	Energy	B	12OCT80	111	1	1	.	.	.	4
22	Mgt S	A	07APR80	111	1	1	.	.	.	4
23	Mgt S	B	03MAY80	111	1	1	.	.	.	4
24	Mgt S	C	22JUN80	111	1	1	.	.	.	16
25	Mgt S	A	12OCT80	111	1	1	.	.	.	4
26	Mgt S	C	12OCT80	111	1	1	.	.	.	12
27	Power	A	07APR80	111	1	1	.	.	.	4
28	Power	B	07APR80	111	1	1	.	.	.	8
29	Power	A	03MAY80	111	1	1	.	.	.	4
30	Power	A	22JUN80	111	1	1	.	.	.	4
31	Power	C	22JUN80	111	1	1	.	.	.	12
32	Power	A	12OCT80	111	1	1	.	.	.	4
33	Energy		03MAY80	101	1	1	2	90.85	.	8
34	Energy		22JUN80	101	1	1	1	69.90	.	4
35	Energy		12OCT80	101	1	1	3	93.00	.	12
36	Mgt S		07APR80	101	1	1	1	99.40	.	4
37	Mgt S		03MAY80	101	1	1	1	85.40	.	4
38	Mgt S		22JUN80	101	1	1	4	87.65	.	16
39	Mgt S		12OCT80	101	1	1	4	76.65	.	16
40	Power		07APR80	101	1	1	3	90.00	.	12
41	Power		03MAY80	101	1	1	1	93.50	.	4
42	Power		22JUN80	101	1	1	4	75.65	.	16
43	Power		12OCT80	101	1	1	1	93.50	.	4
44	Energy		03MAY80	101	1	1	.	.	.	8
45	Energy		22JUN80	101	1	1	.	.	.	4
46	Energy		12OCT80	101	1	1	.	.	.	12
47	Mgt S		07APR80	101	1	1	.	.	.	4
48	Mgt S		03MAY80	101	1	1	.	.	.	4
49	Mgt S		22JUN80	101	1	1	.	.	.	16
50	Mgt S		12OCT80	101	1	1	.	.	.	16
51	Power		07APR80	101	1	1	.	.	.	12
52	Power		03MAY80	101	1	1	.	.	.	4
53	Power		22JUN80	101	1	1	.	.	.	16
54	Power		12OCT80	101	1	1	.	.	.	4
55		A		010	1	1	10	90.95	100	.
56		B		010	1	1	5	90.00	100	.
57		C		010	1	1	10	84.00	100	.
58		A		010	1	1	.	.	.	40
59		B		010	1	1	.	.	.	20
60		C		010	1	1	.	.	.	40
61				000	1	1	25	88.90	.	100
62				**000**	**1**	**1**	.	.	.	**100**

But what I think is one of the most useful new features of version 8 is ODS. If you get a chance, attend one of Ray Pass's workshops or papers on ODS; they are very well done and very informative. Actually any class or paper on ODS would be useful. Virtually any output SAS can generate can use ODS.

```
ODS  html  body="c:\sugi\advtab1.html";
PROC TABULATE  DATA=CLASS
       FORMAT=7.1 NOSEPS;
    CLASS  LOC  / DESCENDING ;
    CLASS  DATE / MISSING ;
    CLASS  ORG  / MISSING;
    VAR   SCORE;
    TABLE  ORG*DATE ALL,
         (LOC ALL)*(N*F=3.0 COLPCTN);
 RUN;
 ODS  html close;
```

		Location						All	
		C		B		A			
		N	ColPctN	N	ColPctN	N	ColPctN	N	ColPctN
Org	Class Date								
	.	.	.	.	.	1	9.1	1	3.8
Energy	03MAY	.	.	1	20.0	1	9.1	2	7.7
	22JUN	.	.	.	.	1	9.1	1	3.8
	12OCT	.	.	1	20.0	2	18.2	3	11.5
Mgt S	07APR	.	.	.	.	1	9.1	1	3.8
	03MAY	.	.	1	20.0	.	.	1	3.8
	22JUN	4	40.0	.	.	.	.	4	15.4
	12OCT	3	30.0	.	.	1	9.1	4	15.4
Power	07APR	.	.	2	40.0	1	9.1	3	11.5
	03MAY	.	.	.	.	1	9.1	1	3.8

		Location						All	
		C		B		A			
		N	ColPc tN	N	ColPc tN	N	ColPc tN	N	ColPc tN
	22JUN	3	30.0	.	.	1	9.1	4	15.4
	12OCT	.	.	.	.	1	9.1	1	3.8

	Location						All	
	C		B		A			
	N	ColPc tN	N	ColPc tN	N	ColPc tN	N	ColPc tN
All	10	100.0	5	100.0	11	100.0	26	100.0

IN SUMMARY

I never thought when I first wrote the first version of this in 1992 that it would survive until the next version. Well, it has and versions 8 and 9 have introduced some new features…***So is this really the FINAL CHAPTER?!?!*** Only time will tell!

The one thing SAS has done over the years is virtually guarantee code written in prior versions will continue to work in later ones … upward compatibility. TABULATE is no different; even though some of the new features make old techniques obsolete; it is still good to know them.

This paper is not intended to be a cure for all your TABULATE problems. Every use of TABULATE is unique in some ways. All I have attempted to do is give you a good starting point or foundation to better understand how to get TABULATE to give you what you want. The more complicated your "crossings", as the SAS manuals refer to them, the tougher it is going to be to determine the denominator specification. Most everything else about TABULATE is very straight forward.

So good luck and happy tabulating!!!

ACKNOWLEDGEMENTS

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