The Utter “Simplicity?” of the TABULATE Procedure - The Final Chapter?
Dan Bruns, Tennessee Valley Authority, Chattanooga, TN

IN THE BEGINNING

Well, here we are again TABULATE fans. I believe I have exhausted this topic (to DEATH some folks say), so I thought I would put it to rest in this FINAL CHAPTER with a paper on the truly advanced features of the TABULATE procedure. The problem is these advanced features are anything but simple. In this tutorial we look at some simpler advanced features, like FORMCHAR, column and row titling, and formatting, and then the one that is really a bear to understand – percentages (PCTN and PCTSUM). Some of the new Version 8 features will also be covered.

The output from a CONTENTS procedure below is just so you know a little about the dataset we will be working with.

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Len</th>
<th>Pos</th>
<th>Format</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td></td>
<td></td>
<td></td>
<td>Class</td>
</tr>
<tr>
<td>ORG</td>
<td></td>
<td></td>
<td></td>
<td>Location</td>
</tr>
<tr>
<td>LOC</td>
<td></td>
<td></td>
<td></td>
<td>Score</td>
</tr>
<tr>
<td>Power</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mgt S</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Energy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Org</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There’s no variable named ALL in the dataset?

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 paper in the SUGI 16 and 20 proceedings and my Advanced Tutorial paper in the SUGI 21 proceedings and hang on to your hat because I’m starting from here and assuming you understand this much.

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

--- Table Output ---
<table>
<thead>
<tr>
<th>Location</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4.0</td>
<td>84.4</td>
<td>93.0</td>
</tr>
<tr>
<td>B</td>
<td>1.0</td>
<td>85.4</td>
<td>85.4</td>
</tr>
<tr>
<td>C</td>
<td>7.0</td>
<td>84.8</td>
<td>98.3</td>
</tr>
</tbody>
</table>

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

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;

--- Table Output ---
<table>
<thead>
<tr>
<th>Location</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4.0</td>
<td>85.0</td>
<td>99.1</td>
</tr>
<tr>
<td>B</td>
<td>2.0</td>
<td>73.4</td>
<td>90.0</td>
</tr>
<tr>
<td>C</td>
<td>7.0</td>
<td>84.8</td>
<td>98.3</td>
</tr>
</tbody>
</table>

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;

--- Table Output ---
<table>
<thead>
<tr>
<th>Location</th>
<th>Score</th>
<th>Score</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>84.0</td>
<td>99.1</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>83.9</td>
<td>90.0</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>83.3</td>
<td>90.5</td>
</tr>
</tbody>
</table>

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 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.

```
PROC TABULATE DATA=CLASS
   MISSNOSPEPS;
   CLASS ORG LOC DATE;
   VAR SCORE;
   /* TITLES & LABELS */
   TITLE
   ORG ALL='--- Totals ---',
   (LOC ALL='Row Totals')
   *(SCORE*MEAN='*F=5.1)
   / BOX=SCORE ROW=FLOAT
   MISSTEXT='None';
```

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
   MISSNOSPEPS;
   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';
```

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. This example is the specification needed on an OS/390 mainframe (MVS).

```
PROC TABULATE DATA=CLASS
   MISSNOSPEPS;
   FORMCHAR='FABFACCCBCEB8FECABCB
   BB4E7E4F60AFE04C6E40'X ;
   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';
```
**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';
```

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<ORG>);
```

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 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 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. 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.

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

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 8, 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. 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.
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 NOEPS;
CLASS ORG LOC DATE;
VAR SCORE;
TABLE ORG*DATE,
(LOC ALL)* (N*F=3.0 PCTN<ORG*DATE>);

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

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

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.
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 expansion expression. 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.

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

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 table expression expansion. Typically ALL is used to do time at first until I discovered it is really only needed to satisfy the denominator for computing PCTN for that date in every ORG, thus the 50 percent for each one with a count of 1.

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

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.
```

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 specify 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.
- `QNTLDEF= -` specifies the mathematical definition used to compute quantiles.
- `QMARKERS= -` specifies the default number of markers to use for the P2 (fixed space) quantile estimation method.
- `QMETHODO -` specifies the method to process the input data to compute quantiles.
- `VERSION 8 FEATURES

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.
- `QNTLDEF= -` specifies the mathematical definition used to compute quantiles.
- `QMARKERS= -` specifies the default number of markers to use for the P2 (fixed space) quantile estimation method.
- `QMETHODO -` specifies the method to process the input data to compute quantiles.
- `VERSION 8 FEATURES

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 specify 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.
- `QNTLDEF= -` specifies the mathematical definition used to compute quantiles.
- `QMARKERS= -` specifies the default number of markers to use for the P2 (fixed space) quantile estimation method.
- `QMETHODO -` specifies the method to process the input data to compute quantiles.
- `VERSION 8 FEATURES

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 specify 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.
- `QNTLDEF= -` specifies the mathematical definition used to compute quantiles.
- `QMARKERS= -` specifies the default number of markers to use for the P2 (fixed space) quantile estimation method.
- `QMETHODO -` specifies the method to process the input data to compute quantiles.
class variables.

- **MLF** - allows you to make use of multiple labels when a multilabel format is assigned to a class variable in PROC FORMAT.

**TABULATE** also supports multiple VAR statements.

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.

**PROC TABULATE** supports these new statistics that **PROC MEANS** and **SUMMARY** also now supports:

- **COLPCTN**
- **COLPCTSUM**
- **MEDIAN**

**P1**

**P5**

**P10**

**P90**

**P95**

**P99**

**PAGEPCTN**

**PAGEPCTSUM**

**Q1**

**Q3**

**GRANGE**

**REPPCTN**

**REPPCTSUM**

**ROWPCTN**

**ROWPCTSUM**

The first thing I had to try was some of the new statistics.

**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. REPFTCTN*F=3. )

/ **RTS=15** **CONDENSE NOCONTINUED**

**RUN;**

It is amazing that after all these years of SAS programmers trying...and 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!

Let's see what the output dataset looks like.

**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: 8:13 Friday, October 13, 2000 Observation Length: 72

Last Modified: 8:13 Friday, October 13, 2000 Deleted Observations: 0

Protection: Compressed: NO

Data Set Type: Sorted: NO

Label:

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

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Pos</th>
<th>Format</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><em>PAGE</em></td>
<td>Num</td>
<td>8</td>
<td>B</td>
<td>8</td>
<td>Page for Observation</td>
</tr>
<tr>
<td>6</td>
<td><em>TABLE</em></td>
<td>Num</td>
<td>8</td>
<td>16</td>
<td></td>
<td>Table for Observation</td>
</tr>
<tr>
<td>4</td>
<td>TYPE_</td>
<td>Char</td>
<td>3</td>
<td>63</td>
<td></td>
<td>Type of Observation</td>
</tr>
<tr>
<td>3</td>
<td>DATE_</td>
<td>Num</td>
<td>8</td>
<td>0</td>
<td>DATE_</td>
<td>Class Date</td>
</tr>
<tr>
<td>2</td>
<td>LOC_</td>
<td>Char</td>
<td>2</td>
<td>62</td>
<td></td>
<td>Location</td>
</tr>
<tr>
<td>1</td>
<td>ORG_</td>
<td>Char</td>
<td>2</td>
<td>56</td>
<td></td>
<td>Org</td>
</tr>
<tr>
<td>8</td>
<td>SCORE_Median</td>
<td>Num</td>
<td>8</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SCORE_N</td>
<td>Num</td>
<td>8</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SCORE_PctN_000</td>
<td>Num</td>
<td>8</td>
<td>48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>SCORE_PctN_010</td>
<td>Num</td>
<td>8</td>
<td>40</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**PROC PRINT DATA=CLASSOUT;**

**RUN;**

Multiple **CLASS** statements give you the ability to specify different options to different variables.

**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;**

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=\"e:\sugi1\advtabl.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;**
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 7 and 8 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 straightforward.

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.

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