The Building Blocks of PROC TABULATE
Ron Coleman
Links Analytical, Inc.

There are a number of basic concepts that need to be understood before you can expect to work effectively with TABULATE. Without a thorough knowledge of the keywords and operators TABULATE can be very difficult. A little time and a building block approach will make this complex SAS® procedure easier to use and understand.

First, our example data set.

Our data is from a fictional company, Educational Demos, for 1995 and 1996. The company has clients in 6 states broken into North and South regions. The North region includes the states Connecticut, Maine, and Massachusetts. The South region includes Alabama, Florida, and Georgia.

Here is a partial PROC CONTENTS listing of the TABDATA data set.

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>REGION</td>
<td>Char</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>STATE</td>
<td>Char</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>YEAR</td>
<td>Num</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>MON</td>
<td>Num</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>DATE</td>
<td>Num</td>
<td>8</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>MONNAME</td>
<td>Char</td>
<td>3</td>
<td>31</td>
</tr>
<tr>
<td>7</td>
<td>QUARTER</td>
<td>Char</td>
<td>1</td>
<td>34</td>
</tr>
<tr>
<td>8</td>
<td>SALES</td>
<td>Num</td>
<td>8</td>
<td>35</td>
</tr>
<tr>
<td>9</td>
<td>NEWCUSTS</td>
<td>Num</td>
<td>8</td>
<td>43</td>
</tr>
</tbody>
</table>

The PROC TABULATE statement begins the TABULATE step.

The general form of the TABULATE statement is:

PROC TABULATE DATA=SAS-data-set <option(s)>;

The CLASS statement is used to list all classification variables to be used in a step. The general form of a CLASS statement is:

CLASS variable(s);

The VAR statement is used to list all analysis variables to be used within PROC TABULATE. Only one VAR statement may be used in a step. The general form of a VAR statement is:

VAR variable(s);

The TABLE statement is the workhorse of PROC TABULATE. This is where you define how you want your table to appear; which variables are in the table, row, and column headings, which analysis variables appear where and what statistics are displayed, what labels are used for the variables and/or statistics or is there to be no label printed.

The general form of the TABLE statement is:

TABLE <table-expression,> row-expression,> column-expression / <option(s)>;

Table expressions consist of variables, statistics, and operators. The variables come from the input data set. The available statistics for PROC TABULATE are listed in the table below.

<table>
<thead>
<tr>
<th>N</th>
<th>NMISS</th>
<th>MEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>STD</td>
<td>MIN</td>
<td>MAX</td>
</tr>
<tr>
<td>RANGE</td>
<td>SUM</td>
<td>USS</td>
</tr>
<tr>
<td>CSS</td>
<td>STDERR</td>
<td>CV</td>
</tr>
<tr>
<td>T</td>
<td>PRT</td>
<td>VAR</td>
</tr>
<tr>
<td>SUMWGT</td>
<td>PCTN</td>
<td>PCTSUM</td>
</tr>
</tbody>
</table>

The operators of PROC TABULATE are the most difficult topic to grasp. The operators used are:, (comma), * (asterisk or star), (space or blank), () (parentheses), " " (quotations), = (equal sign), <> (denominator), and (formatting).

The Comma

The first operator, mentioned on the previous page, is the comma. The Comma separates the table dimension-expressions for the table, row, and column dimensions. You must use a comma for any cross-tab type tables. All of the remaining operators are used within the dimension-expressions and can be used in all dimensions.

Enough words, let’s get to some tables!
The Building Blocks of PROC TABULATE

Exploring Dimensions
Begin with one CLASS variable. The default statistic when only CLASS variables are used in the N or number of observations in each category.

```
proc tabulate data=tabdata;
  class region;
  table region;
run;
```

The above example also illustrates the `all` operator. A `all` allows you to concatenate, or have appear next to on another, CLASS variables.

```
proc tabulate data=tabdata;
  class region year;
  table region year;
run;
```

Add another CLASS variable and another dimension to the table.

```
proc tabulate data=tabdata;
  class region year;
  table region, year;
run;
```

Add a third CLASS variable and a third dimension to the table.

```
proc tabulate data=tabdata;
  class region state year;
  table region, state, year;
run;
```

The next operator to learn in the `*` or asterisk. The asterisk causes a 'crossing' of CLASS variables. In the example below I specify `year*quarter`, that means for each value of `year`, give me the statistics for each value of `quarter`.

```
proc tabulate data=tabdata format=3.;
  class region year state;
  table region year state;
run;
```

```
proc tabulate data=tabdata format=3.;
  class region year quarter;
  table region year quarter;
run;
```

I also added the `format` option on the PROC TABULATE statement. This option sets the default format for all cells in the table. The standard default format is 12.2. You can also set specific formats for each column, We'll see how to do this later.

We can do crossings in any dimension of the table.

```
proc tabulate data=tabdata format=3.;
  class region state year quarter;
  table region*state, year*quarter;
run;
```
The Building Blocks of PROC TABULATE

Adding Analysis Variables

We can have tables with only analysis variables:

```plaintext
proc tabulate data=tabdata format=dollar11.2;
  var sales;
  table sales;
run;
```

The default statistic for analysis variables is the SUM. Analysis variables can only appear in one dimension of the table.

Most often we need to see the analysis variables in terms of categories so we need to mix CLASS and VAR variables to create more useful tables.

```plaintext
proc tabulate data=tabdata format=dollar11.2;
  class year;
  var sales;
  table year*sales;
run;
```

Again we use the * to do a crossing. Remember, the * means – for each value of the first variable, year, give me the values of the second. Since the second variable, sales, is an analysis variable we get a statistic.

Now let’s add our row dimension again.

```plaintext
proc tabulate data=tabdata format=dollar11.2;
  class year region;
  var sales;
  table year*sales;
run;
```

Now we will specify that we want a different statistic, the mean:

```plaintext
proc tabulate data=tabdata format=dollar11.2;
  class year region;
  var sales;
  table region, year*sales*(sum mean);
run;
```

Adding Statistics

Now let’s do two statistics. To do this we will need a new operator (), parentheses, to group items within the dimension-expression.

```plaintext
proc tabulate data=tabdata format=dollar11.2;
  class year region;
  var sales;
  table region, year*sales*(sum mean);
run;
```

Specify a smaller format for the mean:

```plaintext
proc tabulate data=tabdata format=dollar8.2;
  class year region;
  var sales;
  table region, year*sales*(sum mean=f=dollar8.);
run;
```

One more level, multiple analysis and multiple statistics.

```plaintext
proc tabulate data=tabdata format=dollar8.2;
  class year region;
  var sales newcosts;
  table region, year*sales*(sum mean=f=dollar8.) newcosts*sum=f=comm.8.);
run;
```
The Building Blocks of PROC TABULATE

Whoa! What did I just do?

Let's put the column dimension-expression into words:

For each value of the variable `year`, show the analysis variables `sales` and `newcusts`. For `sales` show the sum, formatted with the default format, and the mean, formatted using the `dollar8.` format. For `newcusts` show the sum, formatted using the `comma8.` format.

Building Tables

Now that we know how to present our data we need to dress it up! We can begin with a simple LABEL statement. NOTE: PROC TABULATE does not recognize the `split=` option of PROC PRINT.

```lisp
proc tabulate data=tabdata format=dollar11.;
  label year = 'Sales Year';
  label region = 'Sales Region';
  label sales = 'sales';
  label newcusts = 'New Buyers';
run;
```

When we specify the format for each column we say that the data will be displayed in `n` places within each cell. That defines the width of the cell and the labels are automatically wrapped or hyphenated to fit in the column.

You also have control over the space used for the row titles, called the RTS= option included the vertical lines within the row titles.

```lisp
proc tabulate data=tabdata format=dollar11.;
  label region, year = 'sales*{sum mean*f=dollar18.1
    newcusts*sum='Count'*f= comma8.}
  label region = 'Sales Region';
  label sales = 'Sales';
  label newcusts = 'New Buyers';
run;
```

LABEL and KEYLABEL statements are fine for setting the default labels but like with formats, TABULATE allows you to specify different labels within the TABLE statement.

This is also a means of eliminating a label from the table while all of the data is still displayed.

```lisp
proc tabulate data=tabdata format=dollar11.;
  label region, year = 'sales*{sum mean*f=dollar18.1
    newcusts*sum='Count'\f= comma8.}
  label region = 'Sales Region';
  label sales = 'Sales';
  label newcusts = 'New Buyers';
run;
```

Note how PROC TABULATE wraps the titles automatically as with 'New Buyers'. You can also apply labels for the statistics using the KEYLABEL statement.

```lisp
proc tabulate data=tabdata format=dollar11.;
  label region, year = 'sales*{sum mean*f=dollar18.1
    newcusts*sum='Count'\f= comma8.}
  label region = 'Sales Region';
  label sales = 'Sales';
  label newcusts = 'New Buyers';
  keylabel sum = 'Total';
  keylabel mean = 'Average';
run;
```
If you have more than one variable in the row-expression then you need to calculate the RTS= amount carefully:

```
proc tabulate data=tabdata format=dollar11.;
var sales newcusts;
class year region state;
proc tabulate data=tabdata format=dollar11.;
var sales newcusts;
class year region state;
```

SAS Versions 6.10 and beyond, sorry not is 6.09E, has a useful table option INDENT=. This allows the use of many crossings without increasing the required row title space significantly.

```
proc tabulate data=tabdata format=dollar11.;
var sales newcusts;
class year region state;
```

Tabulate automatically shares the row title space equally with each crossing variable.

```
proc tabulate data=tabdata format=dollar11.;
var sales newcusts;
class year region state;
```

Percentages in PROC TABULATE

Each observation on our input data represents a salesman so we can use the N statistic to see how many salesmen were associated with the results. The statistic PCTN will also show us how the salesmen are distributed throughout the table. By default, the PCTN statistic gives percentages of the entire table.

```
proc tabulate data=tabdata format=dollar11.;
var sales newcusts;
proc tabulate data=tabdata format=dollar11.;
var sales newcusts;
```

We can specify the denominator for the percentages by using the angle bracket operators <>. We specify the dimension-expression piece we want to represent 100%.
The Building Blocks of PROC TABULATE

The other kind of percentage you can use is the PCTSUM statistic. In the example below we use PCTSUM with year as the denominator to find the relationship of the Total Sales in years 1995 and 1996.

```sas
proc tabulate data=tabdata format=dollar11.;
class year region state;
var sales newcusts;
table region*(state all), year='year';
sales*sum mean='sales'
newcusts*sum count='newcusts'
s=n=#/p='f=3. pctn<state all>pctn<state all>
/ rts=8 indent=3;
label year='Sales Year'
region='Sales Region'
state='State'
sales='Sales'
newcusts='New Buyers';
keylabel sum='Total'
mean='Average';
run;
```

The output of this code is:

```
      NB     MA     CT     ME     AL     NC     FL     GA     ALL
Total       $46,870,009  $34,038,985  $37,634,314  $37,634,314  $36,129,659  $35,216,114  $34,038,985  $34,038,985  $34,038,985

North       $11,617,872  $15,293,604  $16,251,251  $16,214,298  $13,526,146  $15,293,604  $16,251,251  $16,214,298  $16,214,298
South       $10,985,641  $13,494,644  $16,251,251  $16,214,298  $13,526,146  $15,293,604  $16,251,251  $16,214,298  $16,214,298

      NB     MA     CT     ME     AL     NC     FL     GA     ALL
Total       1,703,216  1,618,124  1,750,218  1,714,216  1,467,004  1,544,004  1,703,216  1,618,124  1,606,218

```

Next Steps

Here are some more powerful options that I don’t have time to cover in detail but you will find a need for some time in the future.

**TABULATE statement options:**

- **ORDER=** Defines how your categories will be displayed, in formatted order, in internal (actual value) order, or first encountered – second encountered – third, etc.
- **FORMCHAR=** Defines the 11 characters that are used to draw the lines for the box.
- **NOSEPS** Turns off all horizontal lines within the data portion of the table.
- **TABLE statement options:**
  - **BOX=** Allows you to specify text to be put into the big empty in the upper left corner of the table.
  - **MISSTEXT=** Allows you to specify the character or text to print in empty cells instead of the '.' that SAS uses for missing numeric values.

SAS and the Quality Partner program logo are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

© indicates USA registration.

---

**Author contact information:**

**Ron Coleman**

**3545-1 St Johns Bluff Rd., Suite 300**

**Jacksonville FL 32224**

**(904) 641-4766**

**rcoleman@linkanalytical.com**