Introduction to PROC TABULATE

Wendi L. Wright, Educational Testing Service, Princeton, NJ

ABSTRACT

This introduction to PROC TABULATE first looks at the basic syntax of PROC TABULATE and then, using a series of examples, shows how to produce one, two and three dimensional tables. The paper also illustrates how to use the TABLE statement and the difference between the CLASS and VAR statements. Also discussed are adding statistics for the table (including percents), labeling variables and statistics, adding totals and subtotals, and how PROC TABULATE handles missing data. Finally several examples are shown for how to clean up the tables using both standard PROC TABULATE options as well as several style options within ODS.

INTRODUCTION

PROC TABULATE is a procedure used to display descriptive statistics in tabular format. It computes many statistics that are computed by other procedures, such as MEANS, FREQ, and REPORT. PROC TABULATE then displays the results of these statistics in a table format. TABULATE will produce tables in up to three dimensions and allows, within each dimension, multiple variables to be reported one after another hierarchically. PROC TABULATE has some very nice mechanisms that can be used to label and format the variables and the statistics produced.

BASIC SYNTAX

```
PROC TABULATE <options>;
    CLASS variables < / options>;
    VAR variables < / options>;
    TABLE  <page> ,
           <row> ,
           column
           < / options> ;
    … other statements … ;
RUN;
```

Let's take a look at the basic syntax of the PROC TABULATE Procedure. We will start with three of the statements that you can use in PROC TABULATE, CLASS, VAR, and TABLE. As you can see each of these statements, as well as the PROC TABULATE statement itself allows options to be added. For each of the statements, the options need to be preceded with a '/'.

Note: two differences in the syntax from any other Procedure in SAS®; one) the variables in all three statements cannot be separated by commas; and two) the commas in the table statement are treated in a special way and mean a change in dimension.

OPTIONS FOR PROC TABULATE STATEMENT

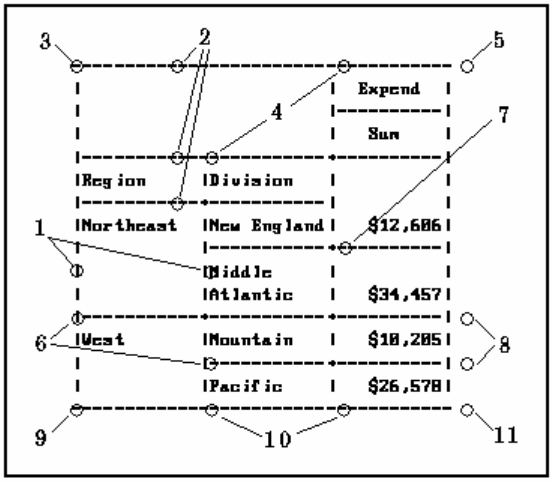
Let's take a look at a few of the options you can specify on the PROC TABULATE statement.

Data=	Specifies what input data to use.
Out=	Specifies the name of the output dataset to store calculated values
Format=	Option specifies a format to use for each cell in the table. Best12.2 is the default.
Formchar='...'	This specifies what line characters to use when drawing the

	table (more on this in a minute).
NoSeps	This option eliminates horizontal separators in the table (only affects traditional SAS monospace output destination).
Order=	Unformatted/Data/Formatted/Freq – orders how the CLASS values appear in the table
Missing	Tells SAS to treat missing values as valid
Style= Contents=	Used with ODS specifications
Exclusive Classdata=dset	To specify exact combinations of data to include

USING FORMCHAR= OPTION

Here is an example of the formchars for specifying the formchar statement. The example shows the default as well as where each character (by spacing) is used. If you want to change any of these, just change the formchar string. For example if you want to change the upper left corner to a '+', go to the third position in the formchar string and change the – to a +. The default value is formchar='|----|+|---'. The figure below shows how each position in the formchar string matches up to the table parts.



VAR STATEMENT

The VAR statement is used to list the variables you intend to use to create summary statistics. As such, they must be numeric. There are only two options that can be used with the VAR statement and, if present, these options appear after a '/'.

Style=	ODS style element definitions. Example might be to change the justification or the font.
Weight=	specify another variable that will weight the values of the variable with the following exceptions: (0 or <0 = counts observation in total number of observations, blank= exclude observation entirely)

TABLE STATEMENT

The Table statement consists of up to three dimensions expressions and the table options. To identify different dimensions, just use a comma. If there are no commas, SAS assumes you are only defining the column dimension (which is required), if there is one comma, then the row dimension is first, then the column, or, if you have three commas, then the order of expressions is page, then row, then column. Options appear at the end after a '/'.

You can have multiple table statements in one PROC TABULATE. This will generate one table for each statement. All variables listed in the table statement must also be listed in either the VAR or CLASS statements.

In the table expressions, there are many statistics that can be specified. Among them are row and column percents, counts, means, and percentiles.

There are about a dozen options that can be specified. Here are a few of them.

Box =	Text and style for the empty box in the upper left corner.
Condense	Print multiple pages to the same physical page.
NoContinued	Suppress the continuation message
MissText	If a cell is blank, this text will print instead
PrintMiss	Print CLASS variable values, even if there is not data for them (this only works if somewhere there is at least one observation with that value.
Indent=	Number of spaces to indent nested row headings.
RTSpace =	Number of positions to allow for the row headings.
Style=[options]	Specify ODS style elements for various parts of the table.

CONSTRUCTING A TABLE STATEMENT – DIMENSION EXPRESSIONS

There are many elements you can use to construct a table expression. You start, of course, with the variables you want to include in the table, but you can also specify the universal CLASS variable ALL which allows you to calculate totals. You will also need to specify what statistics you want to put in the cells of the table. To make your table 'pretty', you can also specify formats, labels, and ODS style specifications in the expression.

So let's take a closer look at how to construct dimension expressions. Here is where PROC TABULATE differs from all the other Procedures in the SAS programming language. The syntax used here is very different.

- A comma specifies to add a new dimension.
- The asterisk is used to produce a cross tab of one variable with another (within the same dimension however, different from PROC freq).
- A blank is used to represent concatenation, or place this output element after the preceding one listed.
- Parenthesis will group elements and associate an operator with each element in the group
- Angle brackets specify a denominator definition for use in percentage calculations.

SIMPLE TABLE WITH ONE DIMENSION

The simplest table will have only one variable. You must specify this variable in either the CLASS or VAR statement, but the resulting table will be a little different depending on whether you specify the variable in the CLASS or VAR statement. If specified in the CLASS statement, you will get a count of observations in each category with the categories listed across the top of the page in columns. If the variable is specified in the VAR statement, then you will get a total sum across all observations.

Here is our example using the variable as a VAR variable. We should get the total income summed across all observations. The resulting table is shown to the right of the example.

```
PROC TABULATE data=one;
      VAR income;
      TABLE income;
RUN;
```

Income
Sum
1563354.00

ADDING STATISTICS

If you want something other than the default N or sum, you can do this by using the '*' and adding the name of the statistic you want instead. You can group multiple stats and variables with parentheses to get the results you want.

Descriptive Statistics	Quantile Statistics
COLPCTN PCTSUM COLPCTSUM MAX ROWPCTN MEAN ROWPCTSUM MIN STDDEV / STD N	MEDIAN | P50 P1 Q3 | P75 P90 P95 P5 P10 P99 Q1 | P25 QRANGE
STDERR NMISS SUM	Hypothesis Testing
PAGEPCTSUM PCTN VAR	ProbT T

```
PROC TABULATE data=one;
      VAR income;
      TABLE income * (N MEAN);
RUN;
```

Income	
N	Mean
30.00	52111.80

CLASS STATEMENT

Classification variables allow you to get stats by category. You will get one column or row for each value of the CLASS variable. You will need to be careful to use a categorical variable with only a limited number of categories or you may end up producing many, many pages of output.

The syntax for the CLASS statement is similar to the VAR statement. List the variables you want to use to group data followed by a '/' and any options you want. The variables here can be either numeric or character (unlike the VAR statement which required numeric). The statistics you can get for these variables are only counts and percents. The statistics will be produced for each LEVEL of the variable. This is almost like using a BY statement within the table.

The options you can use for the CLASS statement are different than for the VAR statement. Here are a few of them:

Ascending/Descending	Specify the order the CLASS variables values are displayed
Missing	Consider missing values valid with special missing values treated separately.
MLF	Enables use of multi-level formatting with overlapping ranges (ex – by state and by region at the same time)
Order=	Groups levels of CLASS variables in the order specified: • Internal (default) – use actual values in data • Data – same order the data is already sorted in • Formatted – use the formatted data values • Freq – highest counts first
Style=[options]	Give ODS style element definitions to these variables
PreLoadFMT	This will preload a format and will also (if other options are also specified), display all values in the table even if there are no observations present with some of the values.
exclusive	Will exclude from the table all combinations of CLASS variables not present in the data (normally used with the preloadfmt option.
groupinternal	Used to group values together by their internal values, not formatted.

Let's talk about how the CLASS variables are handled if they are missing. This applies to any Procedure where you can use a CLASS statement. If an observation has a missing value on even one of the CLASS variables, that observation is excluded from ALL calculations, even if they could have been included in some of the others. For example, a student has a gender value of 'F', and an education value of blank. He would not be included in the gender totals. To get him included wherever possible, use the 'missing' option.

FROM SAS Online Documentation:
"By default, if an observation contains a missing value for any CLASS variable, then PROC TABULATE excludes that observation from all tables that it creates. CLASS statements apply to all TABLE statements in the PROC TABULATE step. Therefore, if you define a variable as a CLASS variable, then PROC TABULATE omits observations that have missing values for that variable from every table even if the variable does not appear in the TABLE statement for one or more tables.

If you specify the MISSING option in the PROC TABULATE statement, then the Procedure considers missing values as valid levels for all CLASS variables. If you specify the MISSING option in a CLASS statement, then PROC TABULATE considers missing values as valid levels for the CLASS variable(s) that are specified in that CLASS statement."

In this example, we are adding more columns to the right of the two columns we already have from the previous example.

```
PROC TABULATE data=one;
      CLASS GENDER;
      VAR income;
      TABLE income * (N Mean)
            INCOME * MEAN * GENDER;
RUN;
```

Income		Income	
		Mean	
		Gender	
N	Mean	Female	Male
30.00	52111.80	52000.69	52238.79

MAKE A SINGLE DIMENSION TABLE VERTICAL

So far we have only looked at tables that are listed by column. Sometimes, especially if you have a single CLASS variable with many categories, it would be useful to have this display vertically. To do this, use the ROW=FLOAT option on the table statement.

```
PROC TABULATE data=one;
      CLASS ethnic;
      TABLE ethnic=' ' * N=' ' ,
            ALL='N'
            / ROW=FLOAT ;
RUN;
```

	N
Asian	4.00
Af.Amer.	6.00
Hisp.	7.00
Am. Ind.	4.00
White	11.00

TWO DIMENSIONAL TABLE

So let's take a look at creating two dimensional tables. All we need to do is add a comma BEFORE the column definition and then put in our row definition. Note the comma below after the statistic specifications (see arrows).

```
PROC TABULATE data=one;
      CLASS gender;
      VAR income;
      TABLE income * (N Mean) ,
            gender;
RUN;
```

		Gender	
		Female	Male
Income	N	16.00	14.00
	Mean	52000.69	52238.79

To swap rows and columns, you need only to switch what you put in front of the comma compared to what is after it.

```
PROC TABULATE data=one;
      CLASS gender;
      VAR income;
      TABLE gender ,
            income * (N Mean) ;
RUN;
```

	Income	
	N	Mean
Gender		
Female	16.00	52000.69
Male	14.00	52238.79

CHANGING WHERE STATISTICS ARE SPECIFIED

You can get very different table structures by changing where the statistic definitions are placed. The statistic definitions can be attached to either a VAR or the CLASS variable, but note that the numbers will ALWAYS be calculated using the VAR variable(s). Here we have the statistics attached to the VAR variable in the column dimension.

```
PROC TABULATE data=one;
      CLASS gender;
      VAR income;
      TABLE gender,
            income * (N Mean Max) ;
RUN;
```

	Income		
	N	Mean	Max
Gender			
Female	16.00	52000.69	93849.00
Male	14.00	52238.79	78695.00

If you move the statistic specification so that it is attached to the rows, the results look very different.

```
PROC TABULATE data=one;
      CLASS gender;
      VAR income;
      TABLE gender * (N Mean Max) ,
            income ;
RUN;
```

		Income
Gender		
Female	N	16.00
	Mean	52000.69
	Max	93849.00
Male	N	14.00
	Mean	52238.79
	Max	78695.00

TWO CLASSIFICATION VARIABLES

Here is an example with two classification variables. One is specified in each dimension.

```
PROC TABULATE data=one;
      CLASS gender fulltime;
      VAR income;
      TABLE gender ,
            Income * fulltime * ( n mean) ;
RUN;
```

	Income			
	Fulltime			
	Fulltime		Parttime	
	N	Mean	N	Mean
Gender				
Female	10.00	58680.40	6.00	40867.83
Male	9.00	52596.56	5.00	51594.80

You can also nest classification variables.

```
PROC TABULATE data=one;
      CLASS gender fulltime educ;
      VAR income;
      TABLE       fulltime * gender ,
            Income * educ * mean ;
RUN;
```

		Income			
		Educ			
		High School	Bachelors	Masters	Doctorate
		Mean	Mean	Mean	Mean
Fulltime	Gender				
Fulltime	Female		42771.20		74589.60
	Male	52506.67		50729.25	56466.00
Parttime	Female	44788.67	36947.00		
	Male	35600.00		62258.00	

ADDING TOTALS AND SUBTOTALS

In order to get totals in your table, you can use the 'ALL' keyword. You can use the keyword in multiple places. Depending on where you put the keyword, you can get different results. This is demonstrated in the following two examples. You can place the keyword on the row or the column dimensions or on both. Note in the examples below, we are not using a VAR statement and we are only requesting the count statistic (N).

The first example will place a total line at the bottom of the table. To place a total line at the top of the table, just list the ALL keyword first.

```
PROC TABULATE data=one;
      CLASS gender fulltime educ;
      TABLE fulltime * gender ALL,
            educ * N ;
RUN;
```

		Educ			
		High School	Bachelors	Masters	Doctorate
		N	N	N	N
Fulltime	Gender				
Fulltime	Female		5.00		5.00
	Male	3.00	1.00	4.00	2.00
Parttime	Female	3.00	4.00	2.00	
	Male	3.00	1.00	3.00	
All		9.00	11.00	9.00	7.00

In this example, we are adding a sub-total for total education by gender

```
PROC TABULATE data=one;
      CLASS gender fulltime educ;
      TABLE (fulltime ALL) * gender ALL,
            educ * N ;
RUN;
```

		Educ			
		High School	Bachelors	Masters	Doctorate
		N	N	N	N
Fulltime	Gender				
Fulltime	Female		5.00		5.00
	Male	3.00	1.00	4.00	2.00
Parttime	Female	3.00	4.00	2.00	
	Male	3.00	1.00	3.00	
All	Female	3.00	9.00	2.00	5.00
	Male	6.00	2.00	7.00	2.00
All		9.00	11.00	9.00	7.00

If you want to put a subtotal for gender within each education level, just change the placement of the ALL keyword. Here we are also adding a total column for the Educ group.

```
PROC TABULATE data=one;
      CLASS gender fulltime educ;
      TABLE fulltime * (gender ALL) ,
            (educ all)* N ;
RUN;
```

		Educ				
		High School	Bachelors	Masters	Doctorate	All
		N	N	N	N	N
Fulltime	Gender					
Fulltime	Female		5.00		5.00	10.00
	Male	3.00	1.00	4.00	2.00	10.00
	All	3.00	6.00	4.00	7.00	20.00
Parttime	Gender					
	Female	3.00	4.00	2.00		9.00
	Male	3.00	1.00	3.00		7.00
	All	6.00	5.00	5.00		16.00

ADDING LABELS

There are two ways to add labels for your variables. The first, and the simplest, is to just add =‘label’ to the dimension expression after the variable you want to label. This way works for labeling both the variables and the statistics.

The second way is to add a label statement to your code: LABEL var=‘label’. The label statement will not work for labeling the statistics. You need to use the KEYLABEL statement to label statistics: KEYLABEL stat=‘label’.

```
PROC TABULATE data=one;
      CLASS gender fulltime;
      VAR income;
      TABLE gender = 'Gender'    ALL = 'Total',
            Fulltime = 'Employment Status' * income * mean = 'Mean' ;
RUN;
```

	Employment Status	
	Fulltime	Parttime
	Income	Income
	Mean	Mean
Gender		
Female	58680.40	40867.83
Male	52596.56	51594.80
Total	55798.58	45743.73

Alternatively, you can also use this code to get the same table as shown above.

```
PROC TABULATE data=one;
      CLASS gender fulltime;
      VAR income;
      TABLE       gender ALL ,
                  Fulltime * income * mean ;
      LABEL gender='Gender' Fulltime='Employment Status';
      KEYLABEL mean='Mean' all='Total';
RUN;
```

HIDING LABELS

In order to hide variable or statistic labels, you can add =‘ ‘ as a label. Note the statistics MUST be attached to the row dimension and NOT the column dimension for this to work.

```
PROC TABULATE data=one;
      CLASS educ gender fulltime;
      VAR income;
      TABLE educ ,
            Fulltime='Employment Status' *
            gender = '  '  *
            income *
            mean = ' ' ;
RUN;
```

	Employment Status			
	Fulltime		Parttime	
	Female	Male	Female	Male
	Income	Income	Income	Income
Educ				
High School		52506.67	44788.67	35600.00
Bachelors	42771.20		36947.00	
Masters		50729.25		62258.00
Doctorate	74589.60	56466.00		

FILLING THE BIG WHITE BOX

To fill in the big white box in the upper left, use the BOX= option.

```
PROC TABULATE data=one;
      CLASS gender fulltime;
      VAR income;
      TABLE fulltime = 'Employment Status',
            Gender * income * mean
            / BOX='Mean Income' ;
RUN;
```

Mean Income	Gender	
	Female	Male
	Income	Income
	Mean	Mean
Employment Status		
Fulltime	58680.40	52596.56
Parttime	40867.83	51594.80

THREE DIMENSIONAL TABLES

Three dimensional tables are easy to produce, just add another section BEFORE the row and column expressions in the table statement. PROC TABULATE will now interpret the dimension statements in this order, first page, then the row, then the columns.

Three dimensional tables allow you to utilize a neat trick to fill in the box, instead of the label of the page dimension appearing *above* the table, you can use the BOX=_page_ option to place that label *inside* the big white box. Only a part of the output is included below the sample code.

```
PROC TABULATE data=one;
      CLASS gender fulltime educ;
      VAR income;
      TABLE educ='Education',
            fulltime = 'Employment Status',
            Gender * income * mean
            / BOX=_PAGE_ ;
RUN;
```

Education High School	Gender	
	Female	Male
	Income	Income
	Mean	Mean
Employment Status		
Fulltime		52506.67
Parttime	44788.67	35600.00

Education Doctorate	Gender	
	Female	Male
	Income	Income
	Mean	Mean
Employment Status		
Fulltime	74589.60	56466.00

PRODUCING CELL PERCENTS

Here is how to get percentages in the cells of the table. PROC TABULATE allows you to put, in the same table, both percents and summary stats. To get percentages, you do not need to use numeric variables since the percents are just based on counts. So no VAR statement is needed unless you want to add other summary stats, you can use just the CLASS statement. There are several percents you can get and note that you can also construct your own percentages by specifying what denominator you wish to use. The use of a complex denominator will not be covered here. We will only cover the three percents that are most commonly used:

PCTN - percent of total.
ROWPCTN – percent across row (use row total as denominator).
COLPCTN – percent across column (use column total as denominator).

The first example shows the use of PCTN.

```
PROC TABULATE data=one;
      CLASS ethnic educ;
      TABLE ethnic * PCTN,
            Educ ;
RUN;
```

		Educ			
		High School	Bachelors	Masters	Doctorate
Ethnic					
Asian	PctN	3.13	9.38		
Af.Amer.	PctN	6.25	3.13	6.25	3.13
Hisp.	PctN	6.25	3.13	6.25	6.25
Am. Ind.	PctN		6.25	3.13	3.13
White	PctN	9.38	9.38	9.38	6.25

The next example shows the use of ROWPCTN. Note that the percents will add up to 100% (taking into account for rounding errors) across each row. COLPCTN works similarly only the columns will add up to 100%.

```
PROC TABULATE data=one;
      CLASS ethnic educ;
      TABLE ethnic * ROWPCTN,
            Educ ;
RUN;
```

		Educ			
		High School	Bachelors	Masters	Doctorate
Ethnic					
Asian	RowPctN	25.00	75.00		
Af.Amer.	RowPctN	33.33	16.67	33.33	16.67
Hisp.	RowPctN	28.57	14.29	28.57	28.57
Am. Ind.	RowPctN		50.00	25.00	25.00
White	RowPctN	27.27	27.27	27.27	18.18

HANDLING MISSING DATA ON A VAR STATEMENT VARIABLE

PROC TABULATE handles missing data differently depending on whether the variable is a VAR variable or a CLASS variable. By default, VAR variable observations are not dropped from the table if they are missing. To override this behavior (in other words, exclude missing observations), you could use the WHERE statement.

Here is an example demonstrating that TABULATE includes observations that have missing values on a variable listed in the VAR statement. If all the observations have a missing value for that variable for a particular cell in the table, the cell will be blank in most cases with the exception of when the statistic is either the count or a percent, when it shows a count of zero.

```
PROC TABULATE data=one;
      CLASS gender fulltime;
      VAR income;
      TABLE fulltime ALL,
            Income * (gender ALL) * mean;
RUN;
```

	Income		
	Gender		
	Female	Male	All
	Mean	Mean	Mean
Fulltime			
Fulltime	58680.40	52596.56	55798.58
Parttime	40867.83		40867.83
All	52000.69	52596.56	52215.20

To exclude this observation, you can use the following code. However, this will change the total number of observations included in the table and may change any total columns. Although in this example, the table did not change, in some cases, this can cause differences, so be careful.

```
PROC TABULATE data=one;
      WHERE income ne . ;
      CLASS gender fulltime;
      VAR income;
      TABLE       fulltime ALL,
            Income * (gender ALL) * mean;
RUN;
```

Rather than dropping the observations with missing income, a better solution is to label the empty cell. To do this, you can use the MISSTEXT option on the table statement. Any internal cell that is empty will have this text placed in it instead.

```
PROC TABULATE data=one;
      CLASS gender fulltime;
      VAR income;
      TABLE fulltime ALL,
            Income * (gender ALL) * mean
            / MISSTEXT = 'no data' ;
RUN;
```

	Income		
	Gender		
	Female	Male	All
	Mean	Mean	Mean
Fulltime			
Fulltime	58680.40	52596.56	55798.58
Parttime	40867.83	no data	40867.83
All	52000.69	52596.56	52215.20

HANDLING MISSING DATA ON A CLASS STATEMENT VARIABLE

Unlike with VAR statement variables, for CLASS variables, TABULATE *drops* observations from the entire table construction if they have a missing value on even one of the CLASS variables. A note of caution about this feature should be kept in mind: if you have multiple CLASS variables and one of them is only populated for a small portion of the records, PROC TABULATE would remove almost all the observations from the entire table. To override this behavior, you need to specify the MISSING option on the PROC TABULATE statement OR on the CLASS statement (after the '/'). This tells SAS to include an observation that may be missing on one of the CLASS variables, but not on the others.

Here is the table using the defaults for PROC TABULATE:

```
PROC TABULATE data=one;
      CLASS gender ethnic;
      VAR income;
      TABLE ethnic ALL,
            Income * (gender ALL) * n;
RUN;
```

	Income		
	Gender		
	Female	Male	All
	N	N	N
Ethnic			
Asian	1.00	0.00	1.00
Af.Amer.	3.00	2.00	5.00
Hisp.	4.00	1.00	5.00
Am. Ind.	3.00	1.00	4.00
White	3.00	8.00	11.00
All	14.00	12.00	26.00

To add these hidden 'missing' observations back in, you can specify the missing keyword on either the PROC TABULATE statement, or after a '/' on the CLASS statement. In the following example, I chose to add it to the CLASS statement. This option can also be used in other procedures (such as PROC MEANS) where you can also specify the CLASS statement.

```
PROC TABULATE data=one;
      CLASS gender ethnic / MISSING;
      VAR income;
      TABLE ethnic ALL,
            Income * (gender ALL) * n;
RUN;
```

	Income		
	Gender		
	Female	Male	All
	N	N	N
Ethnic			
	2.00	2.00	4.00
Asian	1.00	0.00	1.00
Af.Amer.	3.00	2.00	5.00
Hisp.	4.00	1.00	5.00
Am. Ind.	3.00	1.00	4.00
White	3.00	8.00	11.00
All	16.00	14.00	30.00

To label missing data in a row or column header (i.e. CLASS variable), you can create a format using PROC format and then assign the format to the variable using the format statement.

```
proc format;
      value $ethnic
            'W'='White'
            'H'='Hisp.'
            'I'='Am. Ind.'
            'A'='Asian'
            'B'='Af.Amer.'
            ' '='Missing'
            ;
RUN;

PROC TABULATE data=one;
      CLASS gender ethnic / missing;
      VAR income;
      TABLE       ethnic ALL,
            Income * (gender ALL) * n;
      FORMAT ethnic $ethnic. ;
RUN;
```

	Income		
	Gender		
	Female	Male	All
	N	N	N
Ethnic			
Missing	2.00	2.00	4.00
Asian	1.00	0.00	1.00
Af.Amer.	3.00	2.00	5.00
Hisp.	4.00	1.00	5.00
Am. Ind.	3.00	1.00	4.00
White	3.00	8.00	11.00
All	16.00	14.00	30.00

ADDING FORMATS TO THE STATISTICS AND REMOVING HORIZONTAL SEPARATORS

There are several options within PROC TABULATE to format the table to make it look cleaner and more professional. In this example we take a look at two of them. The first option allows you to specify formats for the numbers in the cells of the table using the *F=fmt. expression. The second option is the NOSEPS option which removes the horizontal dividers between the row header values from your table.

```
PROC TABULATE data=one NOSEPS;
      CLASS educ gender fulltime / missing;
      VAR income;
      TABLE educ=' ' * (fulltime=' ' ALL),
            gender=' ' * Income=' ' * ( N*F=6.  MEAN*F=Dollar8. );
RUN;
```

		Female		Male	
		N	Mean	N	Mean
High School	Fulltime			3	$52,507
	Parttime	3	$44,789	2	$35,600
	All	3	$44,789	5	$45,744
Bachelors	Fulltime	5	$42,771	0	
	Parttime	3	$36,947	0	
	All	8	$40,587	0	
Masters	Fulltime			4	$50,729
	Parttime	0		3	$62,258
	All	0		7	$55,670
Doctorate	Fulltime	5	$74,590	2	$56,466
	All	5	$74,590	2	$56,466

CLEANING UP THE ROW LABELS EVEN MORE

Two more options to improve the look of your table are INDENT= which subsets row sub-headers and RTS=## which specifies how wide you want the row header field to be. Note, the RTS=## count includes the '|' bar dividers at the beginning and the end of the row header fields, so include these in your count.

```
PROC TABULATE data=one NOSEPS;
      CLASS educ gender fulltime / missing;
      VAR income;
      TABLE educ=' ' * (fulltime=' ' ALL),
            gender=' ' * Income=' ' * ( N*F=6.  MEAN*F=Dollar8. )
            / BOX='Income' INDENT=3  RTS=12;
RUN;
```

Income	Female		Male	
	N	Mean	N	Mean
High School				
Fulltime			3	$52,507
Parttime	3	$44,789	2	$35,600
All	3	$44,789	5	$45,744
Bachelors				
Fulltime	5	$42,771	0	
Parttime	3	$36,947	0	
All	8	$40,587	0	
Masters				
Fulltime			4	$50,729
Parttime	0		3	$62,258
All	0		7	$55,670
Doctorate				
Fulltime	5	$74,590	2	$56,466
All	5	$74,590	2	$56,466

MAKING THE TABLE PRETTY USING STYLE OPTIONS WITH ODS

You can also use ODS Style elements to clean up the table. Only some of the possible statements are listed below. I am not going to go into an ODS discussion here, but I will show you several examples that use some of these elements to make a table look more professional. These elements all work in HTML, PDF, RTF, PS and PCL destinations.

Foreground/Background=	modify color
BorderWidth=	specify thickness of borders
Just/Vjust=	specify horizontal and vertical justification
Font_Face/Font_Weight/Font_Size=	change font characteristics
Rules=	specify horizontal and vertical rule dividers
CellWidth/CellHeight=	change size of table cells
Cellpadding/CellSpacing=	specify white space and thickness of spacing around cell
OutputWidth=	specify width of table

You can get very different results depending upon where you place the style options. If you place the style options on the PROC TABULATE statement, for example, you will affect all the table cells in the tables. (See the table below for a list of places where you can put style options and what portion of the table they will affect.)

Note: for the CLASS, CLASSLEV, VAR, and KEYWORD statements, the style options can also be specified in the dimension expression in the Table statement.

Style Place In	**Part of Table Affected**
PROC TABULATE S=[…]	data cells
CLASS varname / S=[…]	heading for variable varname
CLASSLEV varname / S=[…]	class values for variable varname
VAR varname / S=[…]	heading for variable varname
KEYWORD stat / S=[…]	heading for named stat
TABLE page,row,col / S=[…]	table borders, rules, cell spacing
BOX={label=' ' S=[…] }	table Box

ADDING COLORS, CELL WIDTH AND JUSTIFICATION

is an example showing some of the style options mentioned above. Notice that we are applying a black foreground (type face) to all cells in the table, assigning a cell width of 200, and center justifying all internal cells. For the gender variable we are assigning a background of yellow to the values. On the table statement, for the ALL keyword, we are assigning a label of 'Tot' and right justifying it. For the mean statistic label we are using a foreground of white and a background of purple. For the box label, we are justifying it at the top left of the box.

You can use ODS Style options to add many things to the table. Here is an example that adds color and justification. We are setting the foreground (print type) to be black and all the cells should be centered. Note the use of the new CLASSLEV statement that we have not seen before.

```
ODS RTF file='c:\myfile.rtf';
PROC TABULATE data=one f=10.2 S=[FOREGROUND=BLACK CELLWIDTH=200 JUST=C];
       CLASS gender;
       CLASSLEV gender / S=[BACKGROUND=YELLOW];
       VAR income;
       TABLE gender=' ' all={label='Tot' s=[JUST=R]},
                     mean={s=[FOREGROUND=WHITE BACKGROUND=PURPLE]} * income
             /       box={label='Income' s=[VJUST=B JUST=R]};
Run;
ODS RTF close;
```

Income	Mean
	Income
Female	52000.69
Male	52238.79
Tot	52111.80

REMOVING LINES FROM THE TABLE

With the addition of the rules, cellspacing and cellpadding options we can remove lines from the table.

```
ODS RTF file='c:\myfile.rtf';
PROC TABULATE data=one f=10.2 S=[foreground=black cellwidth=200 just=c];
       CLASS gender;
       CLASSLEV gender / S=[background=yellow];
       VAR income;
       TABLE gender=' ' all={label='Tot' s=[just=R]},
                     mean={s=[foreground=white background=purple]} * income
             /       s=[RULES=NONE CELLSPACING=0 CELLPADDING=10]
                     box={label='Income' s=[VJust=B Just=R]};
Run;
ODS RTF close;
```

Income	Mean
	Income
Female	52000.69
Male	52238.79
Tot	52111.80

ADDING A FLYOVER TEXT BOX USING ODS AND PROC FORMAT

Using PROC FORMAT, there are a number of ways you can get ODS to do some very interesting tricks. One thing you can do is create flyover text box for your table using ODS HTML. To do this you need to use the FLYOVER=fmt. Option. The result can't be demonstrated here, but what happens is when you hold your curser

over the column header when you view the table in a Web Browser, a little text box will open underneath that shows the text you have in your format.

```
PROC FORMAT;
      VALUE $ethnic
            'W'='White - Non-Hispanic'
            'H'='Hispanic American'
            'I'='American Indian'
            'A'='Asian'
            'B'='African American'
            ' '='Missing';
ODS HTML file='c:\myfile.HTML';
PROC TABULATE data=one;
      CLASS ethnic gender;
      CLASSLEV ethnic / s=[flyover=$ethnic.];
      VAR income;
      TABLE gender,
            ethnic * income;
RUN;
ODS HTML CLOSE;
```

ADDING FIGURES TO YOUR TABLE

Another trick is to add a figure to the your row or column header cell. Note that in this example, the gif files needed to be in the same directory that the HTML file was in.

```
PROC FORMAT;
      VALUE $gendGif
            'M'='bluebutterfly.gif'
            'F'='pinkbutterfly.gif';
RUN;
ODS HTML file='c:\myfile.HTML';
PROC TABULATE data=one;
      CLASS GENDER;
      CLASSLEV gender / S=[VJust=T postimage=$gendGif.  Cellwidth=80];
      VAR income;
      TABLE  income * (N Mean)
            INCOME * MEAN * GENDER;
RUN;
ODS HTML CLOSE;
```

Income		Income	
		Mean	
		Gender	
N	Mean	Female	Male
30	52111.80	52000.69	52238.79

ADDING COLORS TO CELLS BASED ON VALUES IN THE CELLS

The last trick I am going to mention is how to use a format to highlight different cells with different colors based on the final value in the cell (generally called traffic highlighting).

```
title *Example 30 - Traffic Highlighting;
ODS HTML FILE='c:\myfile.html';
PROC FORMAT;
      VALUE watchit
            0 - 20000 = 'Green'
            20001 - 40000 = 'Orange'
            40001 - 50000 = 'Blue'
            50001 - 60000 = 'Purple'
            60001 - high = 'Red' ;
RUN;
PROC TABULATE data=one S=[foreground=watchit.] ;
      CLASS gender ethnic / MISSING;
      VAR income;
      TABLE gender ALL,
            Income * (ethnic ALL) * Mean;
RUN;
ODS HTML CLOSE;
```

	Income						
	Ethnic						All
	Missing	Asian	Af.Amer.	Hisp.	Am. Ind.	White	
	Mean	Mean	Mean	Mean	Mean	Mean	Mean
Gender							
Female	60398.50	26948.00	62083.67	49304.25	56945.00	43321.00	52000.69
Male	57147.50		54039.50	35600.00	78695.00	49334.25	52238.79
All	58773.00	26948.00	58866.00	46563.40	62382.50	47694.27	52111.80

CONCLUSION AND TRADEMARKS

The options described here are by no means exhaustive of what PROC TABULATE can do. Please refer to the documentation to learn what else is possible to improve the look of your tables.

There are several very good resources that you can use to learn more about PROC TABULATE. Among them are:

PROC TABULATE by Example, by Lauren Haworth
SAS Guide to Report Writing: Examples, by Michele M. Burlew
SAS Guide to TABULATE Processing

AUTHOR CONTACT

Your comments and questions are valued and welcome. Contact the author at:

Wendi L. Wright
71 Black Pine Ln.
Levittown, PA 19054-2108
Phone: (215) 547-3372
E-mail: wwright@ets.org