PROC TABULATE – Building Tables With Style

Michael Eberhart, MPH, Department of Public Health, Philadelphia, PA

ABSTRACT

This paper describes how to build and customize tables using PROC TABULATE. Beginning with the code for a basic table, step by step instructions will be provided to expand and enhance any table with PROC TABULATE options and style elements. Topics discussed include the basic concepts of PROC TABULATE, classes and class levels, class data sets, row and column expressions, page expressions, formatting and style elements. Additional topics will include TABULATE statements to control keywords, labels, and analysis variables. Statistics that are available in PROC TABULATE, such as SUM, COLPCTN and ROWPCTN, and where they can be applied will also be discussed. Examples of PROC TEMPLATE code will be provided to modify existing SAS® styles, and picture formats will be used to enhance table output. Steps to save enhanced tables in various formats using the Output Delivery System (ODS) will also be described.

GETTING STARTED

There are several ways to produce tabular output using SAS® including PROC PRINT, PROC REPORT and PROC FREQ to name just a few. However, one of the best ways to make professional tables that include multiple data elements is PROC TABULATE. TABULATE has many features and options to help you make the table appear any way you like, and style elements can be controlled to customize the table in almost any way imaginable. The basic code for PROC TABULATE looks like this:

```
proc tabulate data=how2010.tabdata;
class ~variable list~;
var ~variable list~;
tables ~dimension expression~;
run;
```

The TABULATE statement identifies the procedure and the data set. The CLASS statement identifies all of the class variables that are defined for the procedure. A class variable can be either character or numeric, and is used by the procedure to classify the data into categories. Analysis variables that can be used to compute statistics are specified in the VAR statement. Analysis variables must be numeric. The procedure must include a CLASS statement or a VAR statement, and may include both. The TABLE statement defines the table structure produced by the procedure. A TABLE statement can have up to three dimensions; the column dimension (required), the row dimension (optional), and the page dimension (optional). The dimensions of the table are defined using expressions separated by commas.

With only the column expression, the output table will have one column for each unique value of the class variable ‘sex’.

```
/*Table with one dimension*/
proc tabulate data=how2010.tabdata;
class sex erace status;
tables sex; /*Column expression*/
run;
```

With the row expression AND the column expression, the output table will have one row for each unique value of the class variable ‘sex’ and one column for each unique value of the class variable ‘erace’.

```
/*Table with two dimensions*/
proc tabulate data=how2010.tabdata;
class sex erace status;
tables sex, /*Row expression*/
            erace; /*Column expression*/
run;
```

With the page expression AND the row expression AND the column expression, the output table will have one page for each unique value of the class variable 'sex', and each page will have one row for each unique value of the class variable 'erace' and one column for each unique value of the class variable 'status'.

```
/*Table with three dimensions*/
proc tabulate data=how2010.tabdata;
class sex erace status;
tables sex, /*Page expression*/
            erace, /*Row expression*/
                  status; /*Column expression*/
run;
```

There are several options that can be included in the TABULATE statement that will affect the output. A few that will be discussed in this paper include:

DATA=	identifies the input dataset
FORMAT=	specifies a format for table cells
MISSING	includes missing values in the table
ORDER=	determines the order for values of class variables
CLASSSDATA=	specifies combinations of class variable values
STYLE=	change or add style elements to the table

NOTE: Some of the options described in the SAS documentation are specific to certain output destinations. For example, some table options only apply to the default SAS monospace listing output, and will have no effect on tables generated using ODS. Therefore, you should consider coding your table design and development in the final output format to ensure that the table appears the way you expect. The example presented here are all produced using the ODS PDF destination.

MISSING VALUES

By default, observations with missing values in ANY of the class variables are excluded from the table, regardless of whether they are referenced in the table statement. To include observations with missing values for any of the class variables, add the MISSING option to the PROC TABULATE statement. To include observations with missing values for selected class variables, add the MISSING option to a CLASS statement. It should be noted here that you may include multiple CLASS statements in the procedure code – you would do this when you want specific options to affect specific class variables. You can run the procedure with the MISSING option in the TABULATE statement to identify any missing values that may be excluded.

TABLE DESIGN

The most common table design, and the design that this paper will focus on, has at least one column and several rows of data – also called a two-dimensional table because it utilizes two dimension expressions. The row expression may have several variables, such as race, sex and age, and the column expression will generally have one variable, a summary keyword, or both. The default statistic for class variables is N, or the frequency of observations. If your data are already summarized, you must include a FREQ statement to tell the procedure the name of the variable with the summary data.

The first example is a two-dimensional table using three variables in the row dimension – sex, race and marital status – and the keyword variable ALL in the column dimension. This creates a table of summary data with rows for each unique value of the class variables in the row dimension – one row for male, one row for female, etc. The column created by the keyword ALL contains the count, or frequency, of all observations in the row.

The keyword ALL is a universal class variable that is used to summarize all of the class variables in a particular dimension. ALL can be used in any of the three dimensions, but is perhaps most useful in creating a column dimension for tabular data.

```
title 'Basic table with ALL keyword in column expression';
proc tabulate data=how2010.tabdata;
class sex erace status;
tables sex erace status,
            all;
run;
```

Basic table with ALL keyword in column expression

	All
	N
sex	
F	4717
M	3998
erace	
1	1017
2	12
3	54
4	6595
5	6
6	1031
status	
1	3464
2	5235
9	16

The universal class variable ALL can also be used in the row expression of the table. This is often useful as the first, or last, variable in the row expression to provide a total number of observations for the table.

```
title 'Basic table with ALL keyword in row and column expressions';
proc tabulate data=how2010.tabdata;
class sex erace status;
tables all sex erace status,
            all;
run;
```

Basic table with ALL keyword in row and column expressions

	All
	N
All	8715
sex	
F	4717
M	3998
erace	
1	1017
2	12
3	54
4	6595
5	6
6	1031
status	
1	3464
2	5235
9	16

FORMATTING AND LABELS

Formatting and labeling data in PROC TABULATE can be accomplished either through FORMAT statements in the procedure, format options in the TABULATE or CLASS statements, or through labels assigned in the table dimensions. If a label is assigned to a class variable in the input data set, the procedure will use the label as the row or column header. If no label is assigned, the procedure will use the class variable name. Row and column headers can also be assigned in the TABLE statement by adding ='Label Name' to the variable name in the dimension expression. This label method is appropriate for both variables and the statistics generated by the procedure. The next example uses formatted values from PROC FORMAT, and creates labels for the row headings. The statistic label 'N' is masked, or removed, using *(N=' ') after the universal class variable ALL. The summary data in the table cells are formatted using the FORMAT= option in the TABULATE statement.

```
proc format;
value $gender      'M'='Male'
                   'F'='Female';
value $erace       '1'='Hispanic'
                   '2'='Native American'
                   '3'='Asian'
                   '4'='Black'
                   '5'='Multi-Race'
                   '6'='White'
                   '7'='Other';
value $status      '1'='Divorced'
                   '2'='Married'
                   '8'='Separated'
                   '9'='Never Married';
value agecat       18-24='18 to 24 Years'
                   25-44='25 to 44 Years'
                   45-high='45+ Years';
run;
```

```
title 'Formatted table with ALL keyword in row and column expressions';
proc tabulate data=how2010.tabdata format=comma6.;
class sex erace status;
tables all='Total'
            sex='Gender'
             erace='Race/Ethnicity'
              status='Marital Status',
            all*(N=' ');
format sex $gender. erace $erace. status $status.;
run;
```

Formatted table with ALL keyword in row and column expressions

	All
Total	8,715
Gender	
Female	4,717
Male	3,998
Race/Ethnicity	
Hispanic	1,017
Native American	12
Asian	54
Black	6,595
Multi-Race	6
White	1,031
Marital Status	
Divorced	3,464
Married	5,235
Never Married	16

Notice that the amount of space allocated for the column header 'All' is decreased by masking, or removing the header for the summary statistic 'N'. Whenever row or column headings are suppressed, the procedure adjusts the output accordingly, creating a table that best fits the data provided. Adding larger row or column headings will also cause the table to be redrawn.

The next example adds the MISSING option to the TABULATE statement. This allows for missing values to be included in the output table.

```
title 'Formatted table with MISSING option';
proc tabulate data=how2010.tabdata format=comma6. missing;
class sex erace status;
tables all='Total'
            sex='Gender'
             erace='Race/Ethnicity'
              status='Marital Status',
            all*(N=' ');
format sex $gender. erace $erace. status $status.;
run;
```

Formatted table with MISSING option

	All
Total	8,775
Gender	
Female	4,754
Male	4,021
Race/Ethnicity	
	60
Hispanic	1,017
Native American	12
Asian	54
Black	6,595
Multi-Race	6
White	1,031
Marital Status	
Divorced	3,496
Married	5,263
Never Married	16

In the table above, the first category for the class variable 'erace' is blank to denote the number of observations with missing data. The formatted values for 'erace' also include a category for "Other" which is not included in the table above. There is also a formatted value of 'separated' for the 'status' variable that does not appear in the table. This is because none of the observations in the input data set included values of '7' for 'erace' or '8' for status. To include categories that are not represented in the data, create a class data set that contains all valid combinations of class variable values.

CLASSDATA

To precisely control which categories of class variables appear in the output table, create a data set that contains one observation for each possible combination of values for the variables referenced in the CLASS statement. This tells the procedure exactly which categories should appear in the table, regardless of the actual values in the data. For example, if you wanted the category 'Other' to appear in the race table, but none of the observations in the data set indicates a race of 'Other', you would need to include the category in a class data set.

```
/*Create classes data set so that all possible combinations
      are included in the table.  */
data classes;
do sex='M','F';
do erace='1', '2', '3', '4', '5', '6', '7';
do status= '1', '2', '8', '9';output;
end;end;end;run;
```

The class data set is referenced in the TABULATE statement using the CLASSDATA= option. The MISSTEXT='0' in the TABLE statement options tells the procedure to use the value '0' for cells that have no summary data – in this case for erace='other' and status='separated'.

```
title 'Formatted table with MISSING option and CLASSDATA';
proc tabulate data=how2010.tabdata missing format=comma6.
classdata=classes;
class sex erace status;
tables all='Total'
            sex='Gender'
             erace='Race/Ethnicity'
              status='Marital Status',
            all*(N=' ') / misstext='0';
format sex $gender. erace $erace. status $status.;
run;
```

Formatted table with MISSING option and CLASSDATA

	All
Total	8,775
Gender	
Female	4,754
Male	4,021
Race/Ethnicity	
	60
Hispanic	1,017
Native American	12
Asian	54
Black	6,595
Multi-Race	6
White	1,031
Other	0
Marital Status	
Divorced	3,496
Married	5,263
Separated	0
Never Married	16

ANALYSIS VARIABLES

To add an analysis variable to the table, such as age, include a VAR statement indicating the name of the numeric variable. An analysis variable can be included as a separate entry in the row dimension of the table, or as part of a cross-tab in the example below. A missing value for an analysis variable DOES NOT exclude the observation from the summary table, however the missing value is NOT included in the calculation of statistics. The next example calculates the median age of observations for males and females using the * operator between the class variable 'sex' and the analysis variable 'age'. The default statistic for analysis variables is SUM, but numerous alternate statistics can also be calculated. To choose a different statistic, use the * operator followed by the name of the statistic in parentheses. For a list of available statistics, see the SAS 9.1.3 Help and Documentation.

```
title 'Age as Analysis Variable';
proc tabulate data=how2010.tabdata missing format=comma6.
classdata=classes;
class sex erace status;
var age;
tables all='Total'*(N=' ')
            sex='Gender'*age='Age'*(median='Median' n='N' )
             erace='Race/Ethnicity'*(N=' ')
              status='Marital Status'*(N=' '),
                  all / misstext='0';
format sex $gender. erace $erace. status $status.;
run;
```

Age as Analysis Variable

			All
Total			8,775
Gender			
Female	Age	Median	35
		N	4,534
Male	Age	Median	40
		N	3,935
Race/Ethnicity			
			60
Hispanic			1,017
Native American			12
Asian			54
Black			6,595
Multi-Race			6
White			1,031
Other			0
Marital Status			
Divorced			3,496
Married			5,263
Separated			0
Never Married			16

The cross-tab of sex and age creates two columns of header information – one for sex values and one for age values – along with a third column for the statistic headings, in this case ‘median’ and ‘n’. The remaining parts of the table also include a column for the statistic header, however that value is suppressed by the N=’ ‘ in the TABLE statement.

NUMERIC DATA AS A CLASS

If you are not using the CLASSDATA= option in the TABULATE statement and you want to include a numeric variable such as ‘age’ using the formatted categories, you can simply add the variable ‘age’ to the CLASS statement, include the variable ‘age’ as an entry in one of the table dimensions, and include a FORMAT statement indicating the variable and format names. In this case, missing values would be handled as class variables rather than analysis variables. However, when using a class data set, all of the possible combinations of class variable values must be included. The simplest way to handle this situation is to create a new class variable called ‘agecat’. To add the variable age, which is a numeric variable, using the values associated with the $agecat format, you must first create a new variable using the PUT function and the format. Then you can recreate the classes data set, including the variable agecat and its four categories, and re-run the PROC TABULATE.

```
/*Create categorical variable agecat from numeric variable age*/
data how2010.tabdata; set how2010.tabdata;
agecat=put(age,agecat.);run;

/*Add agecat to classes data set*/
data classes;
do sex='M','F';
do erace='1', '2', '3', '4', '5', '6', '7';
do status= '1', '2', '8', '9';
do agecat='18 to 24 Years','25 to 44 Years','45+
Years','Unknown';output;end;end;end;end;run;

title 'Formatted table with MISSING option and CLASSDATA';
proc tabulate data=how2010.tabdata missing classdata=classes;
class sex erace status agecat;
tables all='Total'
            sex='Gender'
             erace='Race/Ethnicity'
              status='Marital Status',
                  agecat='Age Category'*(n=' ')
            all / misstext='0';
format sex $gender. erace $erace. status $status.;
run;
```

Formatted table with MISSING option and CLASSDATA

	Age Category				All
	18 to 24 Years	25 to 44 Years	45+ Years	Unknown	N
Total	1,026	5,111	2,332	306	8,775
Gender					
Female	695	2,841	998	220	4,754
Male	331	2,270	1,334	86	4,021
Race/Ethnicity					
	7	29	21	3	60
Hispanic	100	615	266	36	1,017
Native American	2	8	2	0	12
Asian	8	32	14	0	54
Black	823	3,831	1,714	227	6,595
Multi-Race	1	5	0	0	6
White	85	591	315	40	1,031
Other	0	0	0	0	0
Marital Status					
Divorced	540	2,042	814	100	3,496
Married	484	3,067	1,518	194	5,263
Separated	0	0	0	0	0
Never Married	2	2	0	12	16

In the above example the column widths of the class variable AGECAT are varied. This column width can be controlled using the CLASSLEV statement and the STYLE= option.

```
title 'CLASSLEV and STYLE to Control Column Width';
proc tabulate data=how2010.tabdata missing classdata=classes;
class sex erace status agecat;
classlev agecat / style=[outputwidth=.75in];
tables all='Total'
            sex='Gender'
             erace='Race/Ethnicity'
              status='Marital Status',
                  agecat='Age Category'*(n=' ')
            all / misstext='0';
format sex $gender. erace $erace. status $status.;
run;
```

CLASSLEV and STYLE to Control Column Width

	Age Category				All
	18 to 24 Years	25 to 44 Years	45+ Years	Unknown	N
Total	1,026	5,111	2,332	306	8,775
Gender					
Female	695	2,841	998	220	4,754
Male	331	2,270	1,334	86	4,021
Race/Ethnicity					
	7	29	21	3	60
Hispanic	100	615	266	36	1,017
Native American	2	8	2	0	12
Asian	8	32	14	0	54
Black	823	3,831	1,714	227	6,595
Multi-Race	1	5	0	0	6
White	85	591	315	40	1,031
Other	0	0	0	0	0
Marital Status					
Divorced	540	2,042	814	100	3,496
Married	484	3,067	1,518	194	5,263
Separated	0	0	0	0	0
Never Married	2	2	0	12	16

STYLES

As seen in the previous example, specific style attributes, such as the column width, can be controlled or changed using the STYLE= option. The STYLE= options is used to change specific style attributes for specific sections of the output table. Style attributes denoted in the TABULATE statement affect the entire table, style attributes in a CLASS or CLASSLEV statement affect only the variables in the class(es) referenced, and style attributes in the TABLE statement affect only the dimension in which they are referenced.

SAS also provides several complete style definitions that can be applied to the output file that contains your table. Style definitions control aspects of color, font and size for the entire document. Each style definition is made up of a series of style elements. Style elements are made up of style attributes that refer to specific parts of the output file, such as the body, table, header and footer. A style attribute is used to specify one feature of the output such as foreground color or text justification. Several style definitions are included in Base SAS, such as the “statistical” style definition in the example below.

```
ods listing close;
ods pdf file="&fileloc.test6.pdf" style=statistical ;
title 'Style Templates - Statistical';
proc tabulate data=how2010.tabdata missing classdata=classes ;
class sex erace status agecat;
classlev agecat / style=[outputwidth=.75in];
tables all='Total'
            sex='Gender'
             erace='Race/Ethnicity'
              status='Marital Status',
                  agecat='Age Category'*(n=' ')
            all / misstext='0';
format sex $gender. erace $erace. status $status.;
run;
ods pdf close;
ods listing;
```

Style Templates - Statistical

	Age Category				All
	18 to 24 Years	25 to 44 Years	45+ Years	Unknown	N
Total	1,026	5,111	2,332	306	8,775
Gender					
Female	695	2,841	998	220	4,754
Male	331	2,270	1,334	86	4,021
Race/Ethnicity					
	7	29	21	3	60
Hispanic	100	615	266	36	1,017
Native American	2	8	2	0	12
Asian	8	32	14	0	54
Black	823	3,831	1,714	227	6,595
Multi-Race	1	5	0	0	6
White	85	591	315	40	1,031
Other	0	0	0	0	0
Marital Status					
Divorced	540	2,042	814	100	3,496
Married	484	3,067	1,518	194	5,263
Separated	0	0	0	0	0
Never Married	2	2	0	12	16

PROC TEMPLATE

You can create a customized style template using one of the existing definitions as a 'parent'. All of the style templates developed by SAS use the 'default' template as a parent, and make changes to different elements to create a new style. PROC TEMPLATE allows anyone to create their own style template while either making only a few adjustments, as in the example below, or changing as many of the default settings as desired. Unless you want to create a template from scratch, you will want to choose one of the SAS-provided templates as a parent, and make changes to that.

```
proc template;
   define style temp;
      parent = styles.statistical;
      replace colors /
            'tablebg' = cxFFFFFF
            'headerbg' = white;
      style Table from Output /
            frame =  BOX
                  rules=none;
   end;
run;
```

Most of the style elements from styles.statistical are retained, however the table background ('tablebg') and header background ('headerbg') are both set to the color 'white'. SAS recognizes colors by name or by hexadecimal code for RGB.

Style elements for the table include FRAME= and RULES=. FRAME refers to how the box is drawn around the entire table, and RULES refers to the lines drawn between rows and columns.

The options for FRAME= are:

ABOVE	a border at the top
BELOW	a border at the bottom
BOX	borders at the top, bottom, and both sides
HSIDES	borders at the top and bottom
LHS	a border at the left side
RHS	a border at the right side
VOID	no borders
VSIDES	borders at the left and right sides

The options for RULES= are:

ALL	between all rows and columns
COLS	between all columns
GROUPS	between the table header and the table and between the table and the table footer, if there is one
NONE	no rules anywhere
ROWS	between all rows

In the PROC TEMPLATE example code above, the FRAME= options is set to BOX, which will draw a box around the entire table, and the RULES= options is set to NONE indicating that no lines be drawn between the rows or columns in the table.

```
ods listing close;
ods pdf file="&fileloc.table12.pdf" style=temp;
title 'Custom Style Template';
proc tabulate data=how2010.tabdata missing format=comma6.
classdata=classes ;
class sex erace status agecat;
classlev agecat / style=[outputwidth=.75in];
tables all='Total'
            sex='Gender'
             erace='Race/Ethnicity'
              status='Marital Status',
                  agecat='Age Category'*(n=' ')
            all / misstext='0';
format sex $gender. erace $erace. status $status.;
run;
ods pdf close;
ods listing;
```

Custom Style Template

	Age Category				All
	18 to 24 Years	25 to 44 Years	45+ Years	Unknown	N
Total	1,026	5,111	2,332	306	8,775
Gender					
Female	695	2,841	998	220	4,754
Male	331	2,270	1,334	86	4,021
Race/Ethnicity					
	7	29	21	3	60
Hispanic	100	615	266	36	1,017
Native American	2	8	2	0	12
Asian	8	32	14	0	54
Black	823	3,831	1,714	227	6,595
Multi-Race	1	5	0	0	6
White	85	591	315	40	1,031
Other	0	0	0	0	0
Marital Status					
Divorced	540	2,042	814	100	3,496
Married	484	3,067	1,518	194	5,263
Separated	0	0	0	0	0
Never Married	2	2	0	12	16

MORE STYLE

Style attributes can be changed to highlight row header (centered text in a different color or font), as well as background of cells, either in the row/column headers or the summary data cells. Style options in the CLASS statement affect the headers of CLASS variables, and style options in the CLASSLEV statement affect the value headers of class variables. ORDER= in the CLASS statement changes the order of the values in the class variable(s). The default order is the order of the actual unformatted data. ORDER=FREQ changes the order to the values in descending frequency.

To finalize the output table size, we can drop the MISSING option from the TABULATE statement, and add a final column to the universal class variable ALL to indicate the column percent of the frequency. By adding the COLPCTN statistic to the ALL table entry, and applying the picture format 'pctfmt' to display the % after the calculated statistic, the final column in the table will include both the number and percent of observations for each category.

```
proc format;
   picture pctfmt low-high='009.0 %';
run;


ods listing close;
ods pdf file="&fileloc.table13.pdf" style=temp;
title 'Custom Style Template';
```

```
proc tabulate data=how2010.tabdata  format=comma6. classdata=classes ;
class sex  status agecat / style=[foreground=blue just=c];
class erace / order=freq style=[foreground=blue  just=c];
classlev agecat / style=[outputwidth=1in] ;
tables all='Total'
            sex='Gender'
             erace='Race/Ethnicity'
              status='Marital Status',
                  agecat='Age Category'*(n=' ')
            all='Total'*(n='N' COLPCTN='Col %'*f=pctfmt.) /
misstext='0';
format sex $gender. erace $erace. status $status.;
run;
ods pdf close;
ods listing;
```

Custom Style Template

	Age Category				Total	
	18 to 24 Years	25 to 44 Years	45+ Years	Unknown	N	Col %
Total	1,019	5,082	2,311	303	8,715	100.0 %
Gender						
Female	690	2,825	985	217	4,717	54.1 %
Male	329	2,257	1,326	86	3,998	45.8 %
Race/Ethnicity						
Black	823	3,831	1,714	227	6,595	75.6 %
White	85	591	315	40	1,031	11.8 %
Hispanic	100	615	266	36	1,017	11.6 %
Asian	8	32	14	0	54	0.6 %
Native American	2	8	2	0	12	0.1 %
Multi-Race	1	5	0	0	6	0.0 %
Other	0	0	0	0	0	0.0 %
Marital Status						
Divorced	536	2,026	804	98	3,464	39.7 %
Married	481	3,054	1,507	193	5,235	60.0 %
Separated	0	0	0	0	0	0.0 %
Never Married	2	2	0	12	16	0.1 %

A LITTLE COLOR

One sure way to make tables pop off the page or a presentation is to include a background color for the table cells. Again, style attributes in different sections of the code will affect different parts of the table. In the next example style attributes for background color in the TABLE statement dimensions will affect cells in the summary data, changing the cells associated with sex and status to light red, and the cells associated with 'erace' and ALL to light blue. The cells for the row headings will remain unchanged.

The style attribute for border color is also changed in the TABLE statement options, thus affecting the border around the entire table. In this case the border color is changed to blue.

```
ods listing close;
```

```
ods pdf file="&fileloc.table14.pdf" style=temp;
title 'Custom Style Template';
proc tabulate data=how2010.tabdata format=comma6. classdata=classes ;
class sex  status agecat / style=[foreground=blue just=c];
class erace / order=freq style=[foreground=blue  just=c];
classlev agecat / style=[outputwidth=1in] ;
tables all='Total'*[style=[background=lightblue]]
           sex='Gender'*[style=[background=lightred]]
            erace='Race/Ethnicity'*[style=[background=lightblue]]
       status='MaritalStatus'*[style=[background=lightred]],
                 agecat='Age Category'*(n=' ')
           all='Total'*(n='N' COLPCTN='Col %'*f=pctfmt.) /
misstext='0'  box=_page_ style=[bordercolor=blue];
format sex $gender. erace $erace. status $status.;
run;
ods pdf close;
ods listing;
```

Custom Style Template

	Age Category				Total	
	18 to 24 Years	25 to 44 Years	45+ Years	Unknown	N	Col %
Total	1,019	5,082	2,311	303	8,715	100.0 %
Gender						
Female	690	2,825	985	217	4,717	54.1 %
Male	329	2,257	1,326	86	3,998	45.8 %
Race/Ethnicity						
Black	823	3,831	1,714	227	6,595	75.6 %
White	85	591	315	40	1,031	11.8 %
Hispanic	100	615	266	36	1,017	11.6 %
Asian	8	32	14	0	54	0.6 %
Native American	2	8	2	0	12	0.1 %
Multi-Race	1	5	0	0	6	0.0 %
Other	0	0	0	0	0	0.0 %
Marital Status						
Divorced	536	2,026	804	98	3,464	39.7 %
Married	481	3,054	1,507	193	5,235	60.0 %
Separated	0	0	0	0	0	0.0 %
Never Married	2	2	0	12	16	0.1 %

FINISHING TOUCHES

The style setting for the cell background affects the rows of cells with summary data, as well as the blank cells to the right of the class header. This does not quite work to create a blank line between class variables in the table. In order to insert a blank line between the different classification variables in the row dimension, the universal class variable ALL was inserted into the table with a style element indicating that the foreground color, or the color of the text, should be white – therefore rendering it invisible on a white background. The variable labels were also manipulated so that the blank line of data would have a label referencing the next class variable in the table.

While style elements for the row and column headings of class variables can be altered in the CLASS and CLASSLEV statements, the style elements for row and column headings of universal class variables are controlled

using the KEYWORD statement. In the next example, the KEYWORD style elements for the foreground is set to blue and the justification is set to centered. The style for the forground text in the summary data cells is modified in the STYLE= option in the TABLE statement where the keyword is used. In addition, the output width is set to 1 ¾ inches for the row headings by changing the style attribute in the CLASSLEV statement for the row dimension class variables.

```
ods pdf file="&fileloc.table15.pdf" style=temp;
title 'Custom Style Template';
proc tabulate data=how2010.tabdata  format=comma6. classdata=classes;
class sex  status agecat /  style=[foreground=blue just=c];
class erace /  order=freq style=[foreground=blue  just=c];
classlev agecat / style=[outputwidth=1in] ;
classlev sex status erace / style=[outputwidth=1.75in];
keyword all / style=[foreground=blue just=c];
tables all='Total'*[style=[foreground=blue]]
            all='Gender'*[style=[foreground=white]]
             sex=' '*[style=[background=lightred]]
              all='Race/Ethnicity'*[style=[foreground=white]]
               erace=' '*[style=[background=lightblue]]
                  all='Marital Status'*[style=[foreground=white]]
                 status=' '*[style=[background=lightred]],
            agecat='Age Category'*(n=' ')
            all='Total'*(n='N' COLPCTN='Col %'*f=pctfmt.) /
misstext='0'  box=_page_ style=[bordercolor=blue];
format sex $gender. erace $erace. status $status.;
run;
ods pdf close;
```

Custom Style Template

	Age Category				Total	
	18 to 24 Years	25 to 44 Years	45+ Years	Unknown	N	Col %
Total	1,019	5,082	2,311	303	8,715	100.0 %
Gender						
Female	690	2,825	985	217	4,717	54.1 %
Male	329	2,257	1,326	86	3,998	45.8 %
Race/Ethnicity						
Black	823	3,831	1,714	227	6,595	75.6 %
White	85	591	315	40	1,031	11.8 %
Hispanic	100	615	266	36	1,017	11.6 %
Asian	8	32	14	0	54	0.6 %
Native American	2	8	2	0	12	0.1 %
Multi-Race	1	5	0	0	6	0.0 %
Other	0	0	0	0	0	0.0 %
Marital Status						
Divorced	536	2,026	804	98	3,464	39.7 %
Married	481	3,054	1,507	193	5,235	60.0 %
Separated	0	0	0	0	0	0.0 %
Never Married	2	2	0	12	16	0.1 %

CONCLUSION

With all of the tools and options available with the TABULATE procedure, along with the almost endless customization options using PROC TEMPLATE and the styles provided by SAS, the PROC TABULATE output can have as much style as you want. As with many SAS procedures, there are often several ways to get the same result. The SAS 9.1.3 Help and Documentation can usually point you in the right direction, but there is no substitute for trial and error.

REFERENCES

SAS Institute Inc., "Why Use PROC TABULATE?" http://support.sas.com/publishing/pubcat/chaps/56514.pdf
SAS Institute Inc., SAS 9.1.3 Help and Documentation, Cary, NC: SAS Institute Inc., 2000-2004.

CONTACT INFORMATION

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

Michael Eberhart, MPH
Philadelphia Department of Public Health
1101 Market Street
8th Floor
Philadelphia, PA 19107
Work Phone: 215-685-4772
Email: michael.eberhart@phila.gov