Tables for Two: An Introduction to the TABULATE Procedure

Kathy McLeod, Computer Science Corporation

ABSTRACT

The SAS® System includes many powerful tools for summarizing and displaying data. PROC TABULATE is one of these tools. This tutorial provides an introduction to PROC TABULATE for programmers having little or no experience using the procedure to generate tables. Each statement used with PROC TABULATE will be discussed: the PROC TABULATE statement with its options; BY, CLASS, FORMAT, FREQ, LABEL, VAR and WEIGHT statements; the TABLE statement; and the KEYLABEL statement.

Emphasis will be put on the TABLE statement in creating a table that satisfies the application's requirements. The presentation will focus on how to cross (both across dimensions and within a dimension), concatenate, and group two variables to produce distinctly different tables. Once a basic table has been created, the tutorial will suggest methods of tailoring the program to display tables that satisfy more complex requirements. Since this paper is designed to be an introductory tutorial, the use of simple summary statistics will be mentioned, but advanced topics such as percentages will not be addressed.

INTRODUCTION

This paper will provide a discussion of definitions and concepts that apply to PROC TABULATE. It will continue with a description of the TABULATE statements and their options. Finally, it will conclude by presenting examples which build from a simple table application.

The data used in this paper will be limited to two dimensions and three variables. By keeping the examples simple, the effect of variable placement within the TABLE statement and use of other TABULATE statements will be easier to identify and explain. The tables presented as part of the examples have been modified to fit within the space allocated for each column.

Plan Your Input

As with any programming language, the key to success is how the input, process and output is controlled. Study the output specifications and work back to the input. By working through the specifications in this manner, only the necessary input variables will carried through the process (data step or procedure) which, in turn will provide for a more efficient program.

The SAS System offers users a wide range of easy-to-use tools for all phases of the program, but especially for use on input. Before coding any type of input statement, determine the most efficient way to read the data. This is especially important when reading external files with many records, or many potential SAS variables, or both. When processing many variables from non-SAS data sets, code potential BY variables or variables that will be used frequently early in the INPUT statement. For SAS data sets the RETAIN statement can be used to reorder the variables. Placing key variables near the beginning of the Program Data Vector (PDV) results in more efficient processing in later steps of the program. When reading many records, "filter" the raw data (external file or SAS data set) and keep only those observations and variables needed for the application.

Also, to minimize errors use clear and consistent coding techniques. For example, coding each variable in the INPUT statement on a separate line. This makes your code easier to read and maintain, and allows space for instream comments or definitions for each variable.

Preprocess data

In your review of the input versus the output specifications, plan in detail how to convert the input data into a SAS data set ready for PROC TABULATE. In many cases it is more efficient to use a procedure other than PROC TABULATE to preprocess the data. Many tables require a combination of statistics based on a select variable (i.e., number of observations, actual value, mean, percent of the total, etc.) or the same data displayed in many categories. PROC MEANS, PROC SUMMARY and PROC FREQ are examples of other procedures that can be used to "massage" the data. One or more SORTs or MERGEs are also often necessary.

Design on paper

Part of the planning (analysis) phase of developing any program is to design the output. If the program specifications do no include specifications for how the final product should appear, take the time to put the design on paper. This will become the model for the table. It is much easier to write a program when the outputs as well as the inputs are well defined. Such a design should include a drawing of the output table, with the relative position of each variable clearly shown. This will be your "blue print".

BASIC DEFINITIONS TO GET YOU GOING

Dimension

Dimensions in the TABULATE procedure refer to how the data is presented. Data can be displayed by the column; row and column; or page (or series of pages depending on the size of the table), row and column. Page, row and column define the dimensions of a table. A table that contains just columns is a one-dimensional, a table of rows and columns is two-dimensional and a table that includes pages, rows and columns is three-dimensional. An easy way to tell how many dimensions will be represented in a table is to count the number of commas. If no commas are present the table will be one-dimensional, one comma two-dimensional, and two commas three-dimensional.

Each dimension of a table is defined by an expression. An expression contains four parts:

- classification variable -
  defined by the CLASS statement.
- analysis variable -
  defined by the VAR statement.
- requested statistic -
  defined by keywords with a default of N (number of observations with non-missing values) for class variables and SUM (sum of the non-missing variable values) for analysis variables.
- format specifications -
  defined by format modifiers with a default of BEST12.2.

Each cell in a table has a unique definition based on the value of the four parts of the dimension expression.

Data Presentation

PROC TABULATE provides three ways to display two dimensions: crossing, concatenating and grouping.

Crossing (operator = '*')

Crossing variables for a table creates a matrix of relationships within the data. The first variable in a crossing is the most general category. With each crossing the categories get more specific (creating a hierarchy).

Concatenating (operator = ' ' (blank space))

Concatenating variables for a table is much the same as concatenating data sets. The values for the first variable are displayed than directly following are the values for the second variable.

Grouping (operator = ',')

Grouping variables for a table assigns an order to the presentation of the data. With the use of parenthesis, the order of how and when variables are concatenated or crossed is established.

PROC TABULATE STATEMENT AND ITS OPTIONS

The nine options listed below can be used as part of the PROC TABULATE Statement to help refine the table being generated. They can be presented in any order. Only those required for the table being generated need be present.

DATA =	Defines the SAS data set to be used. The default is to use the last created SAS data set.
DEPTH =	Defines the maximum number of crossings in the table. This includes statistics and format specifications. The default is 10.
FORMAT =	Defines a universal format to be used for all cells in the table. The default is BEST12.2.
FORMCHAR =	Defines the characters which will be used to display the borders for the rows and columns of the table (corners plus horizontal and vertical lines). The default is '|----|+|---'.
MISSING	Indicates that missing values are valid and should be included in the table. The default is to exclude missing values.
NOSEP	Indicates that the row separator lines not should be printed in the body of the table. The default is to include row separator lines.
ORDER =	Defines the order of the class variables values. The possible values are: DATA - The values are displayed in the order they occur in the SAS data set. FORMATTED - The values are displayed as if they had been sorted by their formatted value. FREQ - The values are displayed in descending order of number of occurrences. INTERNAL - The values are displayed in the order they would be in if the data had been sorted. The default is INTERNAL.
VARDEF =	Defines what the divisor will be when computing variance statistics. The possible values are: DF - Degrees of Freedom N - Number of Observations WDF - Sum of Weights - 1 WGT - Sum of Weights The default is DF (Degrees of Freedom).
NOCONTINUED	Indicates that the text used at the top on a continued page should not be printed. This option is new to Version 6.08 of SAS®. The default is for the text to be printed.

PROC TABULATE STATEMENTS

BY ***variable-list;***

The BY statement is used to generate separate tables for each group defined by the BY variables.

The BY statement is an alternative to the page dimension of the TABLE statement. The difference is that by defining a page dimension the incoming data set need not be sorted by the page dimension variable. For a BY statement to work properly the incoming data set must be sorted (ascending or descending) by the variables listed in the statement.

CLASS ***variable-list;***

The CLASS statement is used to identify selected variables in the data set defined by the DATA = option. These "class" variables usually have a limited number of values, which are discrete or categorical as opposed to continuous in nature. Class variables can be either numeric or character.

FORMAT ***variable1 format1 ...***
variablen formatn;

The FORMAT statement is used to assign another text string or value to a class variable. The FORMAT statement can be used to assign predefined (SAS or User) formats to one or many variable. The TABULATE procedure can contain more than one FORMAT statement. In duplication of variables exists, the last defined format will be used in the table.

FREQ ***variable;***

The FREQ statement is used to indicate a numeric variable containing the frequency of the observation. An example of this would be using a FREQ variable equal to 1000 and the analysis variable equal to 2 instead of 2000.

KEYLABEL ***statistics-keyword1 = 'label-text1' ...***
statistics-keywordn = 'label-textn';

The KEYLABEL statement is used to assign a label to the statistics keyword. (See ADDING THE VARIABLE STATISTICS below for a list of the statistics available in PROC TABULATE.) The TABULATE procedure can contain more than one KEYLABEL statement. If duplication of statistics variables exists, the last defined label text will be used in the table.

LABEL ***variable1 = 'label-text1' ...***
variablen = 'label-textn';

The LABEL statement is used to override the label assigned to variable in a prior data step. The label statement can be used to replace or add labels to some or all of the names of the variables used in the table. Like the LABEL statement in Base SAS, the label text can be up to 40 characters in length and must be enclosed in quotes (single or double). The TABULATE procedure can contain more than one LABEL statement. If duplication of variables exists, the last defined label text will be used in the table.

VAR ***variable-list;***

The VAR statement is used to identify the analysis variables in the data set defined by the DATA = option. These numeric variables can be either categorical or continuous.

WEIGHT = ***variable;***
(or WGT = ***variable;*****)**

The WEIGHT statement is used to identify a variable in the data set defined by the DATA = option that contains a weight to be applied to each analysis variable. If the WEIGHT = statement is used, the table will display weighted statistics.

TABLE ***[page-expression,]***
[row-expression,]
column-expression
[/ options];

The TABLE statement is required. It defines the dimensions of the table, all of the headings, the width of the columns and how the data is presented.

TABLE Statement Options

The following options for the TABLE statement must follow a single front slash (/). The options can be combined in any manner and there is no specific order required for presentation.

BOX = *'text-string'* or
BOX = *variable* or
BOX = _PAGE_

The BOX = option defines the text string, variable name or page dimension text (_PAGE_) that will be placed in the upper left corner of the table. In multi-dimensional tables this area is blank. The value of the text string will wrap from line to line within this area based on the value of the RTSPACE = option.

CONDENSE

The CONDENSE option tells the TABULATE procedure to combine logical pages to form a more condensed version of a requested table that has many columns and only a few rows. The data is not condensed, logical pages are combined. Pages will be combined only if one or more logical pages will fit on a single page.

FUZZ = *value*

The FUZZ = option defines a value to be used to eliminate outliers from analysis variables when computing descriptive statistics other than frequency counts. The default is "low values".

MISTEXT = *'text-string'*

The MISTEXT = option defines the text string to be used in table cells where the resulting value is missing. The value of MISTEXT can be up to twenty (20) characters and must be enclosed to single quotes.

PRINTMISS

The PRINTMISS option is used to include a row and/or column for all values of the class variables. Rows and/or columns are added in a crossing of class variables when the value of the one of the class variables exists, but not the other. PRINTMISS is also used to indicate that headings for all logical pages are identical.

ROW = FLOAT
ROW = CONSTANT (or ROW = CONST)

The Row = option is used to indicate if the headings in a row crossing will be allocated space when it is blank. If the ROW = option is set to CONSTANT (or CONST) than the space will be allocated. If it is set to FLOAT the space for the row headings will be divided equally among the remaining non-blank headings. The default is CONSTANT (or CONST).

RTSPACE = *value* (or RTS = *value*)

The RTSPACE option is used to define the amount of space to be used for row headings. It is the only way to define row heading space requirements.

HOW TO DEVELOP AND TEST PROC TABULATE CODE

The best way to develop and test PROC TABULATE code is to create a test SAS data set that contains the minimum number of observations required to have at least one, but preferably two observations that fit in each cell of the table. Once the data set has been developed, a mock-up of the table should also be created with the expected values in each shell. By doing this any "statistics" developed by PROC TABULATE will be verified. Many errors can be caught by taking the time to verify the table at this step of the process rather then on an entire SAS data set.

WRITING PROC TABULATE CODE

The easiest way to understand how to write the TABLE statement of PROC TABULATE is to examine the results of concatenating, crossing and grouping the same variables. Examples for each type of data presentation using the default values for requested statistics and format specification will be given. The examples will deal with data presentation for one- and two-dimensional tables. These basic examples will be enhanced during the remainder of this paper. Compare the statements required to move from one example to the next.

The following code, INPUT statement and data were used for all of the examples. (The names used in the FORMAT statement were borrowed from the Sports Car Club of America® (SCCA). The corresponding numerical data in no way represents factual information.):

```
  PROC FORMAT;
    VALUE $DIVFMT
       '1' = 'NORTHEAST'
       '2' = 'SOUTHEAST'
       '3' = 'MID-ATLANTIC';
    VALUE $REGFMT
      '01' = 'STEEL CITIES'
      '02' = 'NEW ENGLAND'
      '03' = 'FINGER LAKES'
      '04' = 'GLEN'
      '05' = 'MOHAWK-HUDSON'
      '06' = 'BUCCANEER'
      '07' = 'SOUTH FLORIDA'
      '08' = 'CAROLINA'
      '09' = 'ATLANTA'
      '10' = 'WASHINGTON D C'
      '11' = 'TRI-REGION'
      '12' = 'BLUE RIDGE';
RUN;
DATA MEMBERS;
  INPUT
    @  1 DIVISION $CHAR1.
    @  5 REGION   $CHAR2.
    @ 10 MEMCNT   7.;
DATALINES;
1   01   956
1   02   132
1   03   1096
1   04   50
1   05   25
2   06   200
2   07   309
2   08   12
2   09   95
3   10   654
3   11   20
3   12   987
RUN;
```

Concatenating variables in one dimension

```
PROC TABULATE DATA = MEMBERS;
  CLASS DIVISION REGION;
  TABLE DIVISION REGION;
RUN;
```

```
----------------------------------------------------------
|        DIVISION          |           REGION            |
|--------------------------+-----------------------------|
|   1    |   2    |   3    |   01    |   02    |   03    |
|--------+--------+--------+---------+---------+---------|
|   N    |   N    |   N    |   N     |   N     |   N     |
|--------+--------+--------+---------+---------+---------|
|   5.00|    4.00|    3.00|     1.00|     1.00|     1.00|
----------------------------------------------------------

(CONTINUED)
```

Explanation: The number of observations (N) for the variable DIVISION is displayed followed by the N for the variable REGION. (The table was truncated for space considerations. The word CONTINUED following the first part of the table indicates that the table is not complete.)

Crossing variables in one dimension

```
PROC TABULATE DATA = MEMBERS;
  CLASS DIVISION REGION;
  TABLE DIVISION * REGION;
RUN;
```

```
-------------------------------------------------------------
|                        DIVISION                           |
|-----------------------------------------------------------|
|                         1                        |   2    |
|--------------------------------------------------+--------|
|                       REGION                     | REGION |
|--------------------------------------------------+--------|
|   01   |   02    |   03    |   04    |   05    |   06     |
|--------+---------+---------+---------+---------+----------|
|   N    |   N     |   N     |   N     |   N     |   N      |
|--------+---------+---------+---------+---------+----------|
|    1.00|     1.00|     1.00|     1.00|     1.00|      1.00|
-------------------------------------------------------------
```

```
(CONTINUED)
```

Explanation: The number of observations (N) for each REGION within a DIVISION is displayed.

Crossing variables in two dimensions

```
PROC TABULATE DATA = MEMBERS;
  CLASS DIVISION REGION;
  VAR MEMCNT;
  TABLE DIVISION * REGION, MEMCNT;
RUN;
```

```
--------------------------------
|                  |  MEMCNT   |
|                  |-----------|
|                  |   SUM     |
|------------------+-----------|
|DIVISION|REGION   |           |
|--------+---------|           |
|1       |01       |     956.00|
|        |---------+-----------|

...

|        |---------+-----------|
|        |05       |      25.00|
|--------+---------+-----------|
|2       |06       |     200.00|
|        |---------+-----------|

...

|        |---------+-----------|
|        |09       |      95.00|
|--------+---------+-----------|
|3       |10       |     654.00|
|        |---------+-----------|
|        |11       |      20.00|
|        |---------+-----------|
|        |12       |     987.00|
--------------------------------
```

Explanation: The rows are REGION within DIVISION with the SUM statistic for MEMCNT being displayed for the column.

Grouping variables in one dimension

```
PROC TABULATE DATA = MEMBERS;
  CLASS DIVISION REGION;
  VAR MEMCNT;
  TABLE (DIVISION REGION), MEMCNT;
RUN;
```

```
--------------------------------
|                |   MEMCNT    |
|                |-------------|
|                |    SUM      |
|----------------+-------------|
|DIVISION        |             |
|----------------|             |
|1               |      2259.00|
|----------------+-------------|
|2               |       616.00|
|----------------+-------------|
|3               |      1661.00|
|----------------+-------------|
|REGION          |             |
|----------------|             |
|01              |       956.00|
|----------------+-------------|
|02              |       132.00|
|----------------+-------------|

...

|----------------+-------------|
|11              |        20.00|
|----------------+-------------|
|12              |       987.00|
--------------------------------
```

Explanation: The rows are DIVISION followed by REGION with the SUM statistic for MEMCNT being displayed for the column.

CUSTOMIZING THE PROC TABULATE OUTPUT

Title

There are two ways to add titles to your TABULATE output: use the TITLE statement, or specify the title in the page dimension of the table.

The TITLE statement, the more flexible of the two, can be used anywhere in a SAS program. The format of the TITLE statement is as follows:

TITLE[n] [*'text-string'*];

- where [n] is the line number where the title is to be placed. Up to ten titles can be defined at any point in time.
- where ['text-string'] is the text to be used as a title for the output. Any title line can be up to 132 characters long. Titles of less than 132 characters are centered within the 132 character print line. With a little planning and some leading and trailing spaces the title can be positioned anywhere within the print line. All titles must be enclosed in single quotes.

Hint: *To continue a title or footnote beyond 72 characters, enter characters up to the seventy-second position of one line and than continue the title starting in the first position of the next line. Remember the semi-colon indicates the end of the line (logical line).*

Once defined it will be used on every page of output until you cancel all or part of the TITLE statements. To cancel all of the TITLE statements, use the TITLE statement without a number suffix and title. To cancel part of the TITLE statements, use the TITLE statement with a number suffix and without a title. This cancels the title lines starting with the number suffix on through the remaining title lines. If a title line requires changing, all title lines following the line to be changes must be redefined.

```
TITLE1 '1994 MEMBERSHIP DATA';
TITLE3 'EASTERN SPORTS CAR CLUB';
PROC TABULATE DATA = MEMBERS;
  CLASS DIVISION REGION;
  VAR MEMCNT;
  TABLE (DIVISION REGION), MEMCNT;
RUN;
```

```
              1994 MEMBERSHIP DATA

             EASTERN SPORTS CAR CLUB

----------------------------------
|                 |   MEMCNT   |
|                 |------------|
|                 |    SUM     |
|-----------------+------------|
|DIVISION         |            |
|-----------------|            |
|1                |     2259.00|
|-----------------+------------|
|2                |      616.00|
|-----------------+------------|
|3                |     1661.00|
|-----------------+------------|
|REGION           |            |
|-----------------|            |
|01               |      956.00|
|-----------------+------------|
|02               |      132.00|
|-----------------+------------|

...

|-----------------+------------|
|11               |       20.00|
|-----------------+------------|
|12               |      987.00|
----------------------------------
```

Defining a title as part of the page dimension in the TABLE statement is the easier of the two options, but it does not allow the flexibility of the TITLE statement. The format of the page dimension the TABLE statement when defining a title is as follows:

TABLE ALL = '*text-string*' *
  ALL = '*text-string*', ...

- This is a special use of the ALL variable. In this case the ALL is used to generically define the page.
- The page dimension variable can also be used in this manner. The title will be printed followed by the value of the page dimension variable. Note that the title is not centered.

```
PROC TABULATE DATA = MEMBERS;
  CLASS DIVISION REGION;
  VAR MEMCNT;
  TABLE ALL = '1994 MEMBERSHIP DATA' *
        ALL = 'EASTERN SPORTS CAR CLUB',
        (DIVISION REGION), MEMCNT;
RUN;
```

```
1994 MEMBERSHIP DATA
AND EASTERN SPORTS CAR CLUB
----------------------------------
|                 |   MEMCNT   |
|                 |------------|
|                 |    SUM     |
|-----------------+------------|
|DIVISION         |            |
|-----------------|            |
|1                |     2259.00|
|-----------------+------------|
|2                |      616.00|
|-----------------+------------|
|3                |     1661.00|
|-----------------+------------|
|REGION           |            |
|-----------------|            |
|01               |      956.00|
|-----------------+------------|
|02               |      132.00|
|-----------------+------------|

...

|-----------------+------------|
|11               |       20.00|
|-----------------+------------|
|12               |      987.00|
----------------------------------
```

Headings

Adding headings to columns or rows in the table is like using the page dimension to add a title to the table. The general format is the follow the dimension variable by an equal sign (=) and then the text string that defines the dimension. The text string must be enclosed in single quotes.

All of the multi-dimensional examples presented to this point in the paper have had a box in the upper left corner of the table that is blank. To use this box to further customize the TABULATE output use, the BOX = option of the TABLE statement.

```
PROC TABULATE DATA = MEMBERS;
  CLASS DIVISION REGION;
  VAR MEMCNT;
  TABLE ALL = '1994 MEMBERSHIP DATA',
        (DIVISION = 'NATIONAL AREA'
         REGION = 'LOCAL CLUB'),
         MEMCNT
      / RTS = 25 BOX = 'EASTERN SPORTS CAR CLUB';
  RUN;
```

```
1994 MEMBERSHIP DATA
--------------------------------------
|EASTERN SPORTS CAR CLUB|   MEMCNT   |
|                       |------------|
|                       |    SUM     |
|-----------------------+------------|
|NATIONAL AREA          |            |
|-----------------------|            |
|1                      |     2259.00|
|-----------------------+------------|
|2                      |      616.00|
|-----------------------+------------|
|3                      |     1661.00|
|-----------------------+------------|
|LOCAL CLUB             |            |
|-----------------------|            |
|01                     |      956.00|
|-----------------------+------------|
|02                     |      132.00|
|-----------------------+------------|

...

|-----------------------+------------|
|11                     |       20.00|
|-----------------------+------------|
|12                     |      987.00|
--------------------------------------
```

- The '/' indicates that the remaining portion of the TABLE statement contains selected options.

Column and Row Headings

Column and row headings can be customized by using user-defined formats.

```
PROC TABULATE DATA = MEMBERS;
  CLASS DIVISION REGION;
  VAR MEMCNT;
  FORMAT DIVISION $DIVFMT.;
  FORMAT REGION $REGFMT.;
  TABLE ALL = '1994 MEMBERSHIP DATA',
        (DIVISION = 'NATIONAL AREA'
         REGION = 'LOCAL CLUB'),
         MEMCNT
      / RTS = 25 BOX = 'EASTERN SPORTS CAR CLUB';
RUN;
```

```
1994 MEMBERSHIP DATA
--------------------------------------
|EASTERN SPORTS CAR CLUB|   MEMCNT   |
|                       |------------|
|                       |    SUM     |
|-----------------------+------------|
|NATIONAL AREA          |            |
|-----------------------|            |
|NORTHEAST              |     2259.00|
|-----------------------+------------|
|SOUTHEAST              |      616.00|
|-----------------------+------------|
|MID-ATLANTIC           |     1661.00|
|-----------------------+------------|
|LOCAL CLUB             |            |
|-----------------------|            |
|STEEL CITIES           |      956.00|
|-----------------------+------------|
|NEW ENGLAND            |      132.00|
|-----------------------+------------|

...

|-----------------------+------------|
|TRI-REGION             |       20.00|
|-----------------------+------------|
|BLUE RIDGE             |      987.00|
--------------------------------------
```

Internal and External "Borders" for the Data

There are two ways to modify the internal table borders, but only one way to modify the external borders. The value of the FORMCHAR = option defines what characters will be used to form the internal and external borders. Each position in the 11 position value specifies a type of line or corner. The positions of FORMCHAR are defined as follows:

● Position 1:	(¦) Horizontal Line
● Position 2:	(-) Vertical Line
● Position 3:	(-) Upper Left Corner
● Position 4:	(-) Upper Center Intersection
● Position 5:	(-) Upper Right Corner
● Position 6:	(¦) Center Left Intersection
● Position 7:	(+) Center Intersection
● Position 8:	(¦) Center Right Intersection
● Position 9:	(-) Lower Left Corner
● Position 10:	(-) Lower Center Intersection
● Position 11:	(-) Lower Right Intersection.

The value of FORMCHAR can be redefined completely or partially. To completely redefine FORMCHAR all 11 positions must be specified. Any character or hexadecimal representation can be used to redefining FORMCHAR.

```
PROC TABULATE DATA = MEMBERS
  FORMCHAR = '**+$+$+$+$+';
  CLASS DIVISION REGION;
  VAR MEMCNT;
  FORMAT DIVISION $DIVFMT.;
  FORMAT REGION $REGFMT.;
  TABLE ALL = '1994 MEMBERSHIP DATA',
        (DIVISION = 'NATIONAL AREA'
         REGION = 'LOCAL CLUB'),
         MEMCNT
      / RTS = 25 BOX = 'EASTERN SPORTS CAR CLUB';
RUN;
        1994 MEMBERSHIP DATA
        +***********************$************+
        *EASTERN SPORTS CAR CLUB*   MEMCNT   *
        *                       $************$
        *                       *    SUM     *
        $***********************+************$
        *NATIONAL AREA          *            *
        $***********************$            *
        *NORTHEAST              *     2259.00*
        $***********************+************$
        *SOUTHEAST              *      616.00*
        $***********************+************$
        *MID-ATLANTIC           *     1661.00*
        $***********************+************$
        *LOCAL CLUB             *            *
        $***********************$            *
        *STEEL CITIES           *      956.00*
        $***********************+************$
        *NEW ENGLAND            *      132.00*
        $***********************+************$

        ...
```

```
$************************+*************$
*TRI-REGION              *        20.00*
$************************+*************$
*BLUE RIDGE              *       987.00*
+************************$*************+
```

The horizontal lines within the body of the table can be removed by using the NOSEP option. The NOSEP option actually removes the horizontal lines from the table where redefining FORMCHAR to all spaces leaves a blank line.

HINT: *Setting FORMCHAR to 11 spaces will eliminate all lines, intersections, plus corners from a table, but space will be left in the table for the position where the lines, intersections or corners should have been. By adding the NOSEP option, the horizontal lines which are now blank lines will be removed from the table. This combination will greatly reduce the number of lines required by any table.*

```
PROC TABULATE DATA = MEMBERS
  FORMCHAR = '**+$+$+$+$+' NOSEP;
  CLASS DIVISION REGION;
  VAR MEMCNT;
  FORMAT DIVISION $DIVFMT.;
  FORMAT REGION $REGFMT.;
  TABLE ALL = '1994 MEMBERSHIP DATA',
        (DIVISION = 'NATIONAL AREA'
         REGION = 'LOCAL CLUB'),
         MEMCNT
       / RTS = 25 BOX = 'EASTERN SPORTS CAR CLUB';
RUN;
```

```
1994 MEMBERSHIP DATA
+************************$*************+
*EASTERN SPORTS CAR CLUB*    MEMCNT    *
*                        $*************$
*                        *     SUM     *
$************************+*************$
*NATIONAL AREA           *             *
*NORTHEAST               *      2259.00*
*SOUTHEAST               *       616.00*
*MID-ATLANTIC            *      1661.00*
*LOCAL CLUB              *             *
*STEEL CITIES            *       956.00*
*NEW ENGLAND             *       132.00*

...

*TRI-REGION              *        20.00*
*BLUE RIDGE              *       987.00*
+************************$*************+
```

Data

The data in the table can only be modified in appearance. Format modifiers, the fourth part of a dimension expression, are crossed with the analysis variable to change the appearance of the data. Standard SAS formats or user-defined formats can be used as format modifiers. The Format procedure can be used to provide formats (or new values) for character as well as numeric data.

Hint: *Each crossing is counted by the procedure and compared to the maximum value of the DEPTH = option.*

```
PROC TABULATE DATA = MEMBERS
  FORMCHAR = '**+$+$+$+$+' NOSEP;
  CLASS DIVISION REGION;
  VAR MEMCNT;
  FORMAT DIVISION $DIVFMT.;
  FORMAT REGION $REGFMT.;
  TABLE ALL = '1994 MEMBERSHIP DATA',
        (DIVISION = 'NATIONAL AREA'
         REGION = 'LOCAL CLUB'),
         MEMCNT = 'MEMBER COUNT' * F = COMMA20.2
       / RTS = 25 BOX = 'EASTERN SPORTS CAR CLUB';
RUN;
```

```
1994 MEMBERSHIP DATA
+************************$*********************+
*EASTERN SPORTS CAR CLUB*    MEMBER COUNT      *
*                        $*********************$
*                        *         SUM         *
$************************+*********************$
*NATIONAL AREA           *                     *
*NORTHEAST               *             2,259.00*
*SOUTHEAST               *               616.00*
*MID-ATLANTIC            *             1,661.00*
*LOCAL CLUB              *                     *
*STEEL CITIES            *               956.00*
*NEW ENGLAND             *               132.00*

...

*TRI-REGION              *                20.00*
*BLUE RIDGE              *               987.00*
+************************$*********************+
```

Footnotes

The TABULATE procedure does not offer a facility to add footnotes to a table, but the SAS System has a FOOTNOTE statement that can be used anywhere. The FOOTNOTE statement operates in the same manner as the TITLE statement. There can be up to 10 footnote lines on any page, and once defined they will be used on every following page until canceled. The format of the FOOTNOTE statement is:

FOOTNOTE[n] [*'text-string'*];

- where [n] indicates the order of the footnote lines. Up to 10 footnotes can be defined.
- where ['text-string'] is the text to be used as the footnote for the output. Any footnote line can be up to 132 characters long. Footnotes of less than 132 characters are centered within the 132 character print line. With a little planning and some leading and trailing spaces the footnote can be positioned anywhere within the print line. All footnotes must be enclosed in single quotes.

```
FOOTNOTE1 'NEW MEMBERS SINCE 01/01/94';
FOOTNOTE3 'CUTOFF DATE 02/28/94';
PROC TABULATE DATA = MEMBERS
  FORMCHAR = '**+$+$+$+$+' NOSEP;
  CLASS DIVISION REGION;
  VAR MEMCNT;
  FORMAT DIVISION $DIVFMT.;
  FORMAT REGION $REGFMT.;
  TABLE ALL = '1994 MEMBERSHIP DATA',
        (DIVISION = 'NATIONAL AREA'
         REGION = 'LOCAL CLUB'),
         MEMCNT = 'MEMBER COUNT' * F = COMMA20.2
       / RTS = 25 BOX = 'EASTERN SPORTS CAR CLUB';
```

```
1994 MEMBERSHIP DATA
+***********************$*********************+
*EASTERN SPORTS CAR CLUB*    MEMBER COUNT     *
*                       $*********************$
*                       *         SUM         *
$***********************+*********************$
*NATIONAL AREA          *                     *
*NORTHEAST              *            2,259.00*
*SOUTHEAST              *              616.00*
*MID-ATLANTIC           *            1,661.00*
*LOCAL CLUB             *                     *
*STEEL CITIES           *              956.00*
*NEW ENGLAND            *              132.00*

...

*TRI-REGION             *               20.00*
*BLUE RIDGE             *              987.00*
+***********************$*********************+

              NEW MEMBERS SINCE 01/01/94

                 CUTOFF DATE 02/28/94
```

ADDING THE "ALL" VARIABLE

The ALL variable is a special universal class variable. It can be used in any dimension of the table or within a parenthetical group. The results of the ALL variable will be placed in an additional row or column according to the placement of the term in the TABLE statement. Parentheses many have to be added to dimension definitions to achieve desired results.

```
    FOOTNOTE1 'NEW MEMBERS SINCE 01/01/94';
    FOOTNOTE2 'CUTOFF DATE 02/28/94';
    PROC TABULATE DATA = MEMBERS
      FORMCHAR = '           ' NOSEP;
      CLASS DIVISION REGION;
      VAR MEMCNT;
      FORMAT DIVISION $DIVFMT.;
      FORMAT REGION $REGFMT.;
      TABLE ALL = '1994 MEMBERSHIP DATA',
            (DIVISION = 'NATIONAL AREA' *
             REGION = 'LOCAL CLUB')
             ALL = 'TOTAL NEW MEMBERS',
             MEMCNT = 'MEMBER COUNT' * F = COMMA20.2
           / RTS = 40 BOX = 'EASTERN SPORTS CAR CLUB';
    RUN;
```

```
1994 MEMBERSHIP DATA
EASTERN SPORTS CAR CLUB                      MEMBER COUNT
                                                      SUM

NATIONAL AREA     LOCAL CLUB
NORTHEAST         STEEL CITIES                     956.00
                  NEW ENGLAND                      132.00
                  FINGER LAKES                   1,096.00
                  GLEN                              50.00
                  MOHAWK-HUDSON                     25.00
SOUTHEAST         BUCCANEER                        200.00
                  SOUTH FLORIDA                    309.00
                  CAROLINA                          12.00
                  ATLANTA                           95.00
MID-ATLANTIC      WASHINGTON D C                   654.00
                  TRI-REGION                        20.00
                  BLUE RIDGE                       987.00
TOTAL NEW MEMBERS                                4,536.00

                 NEW MEMBERS SINCE 01/01/94
                    CUTOFF DATE 02/28/94
```

ADDING THE VARIABLE STATISTICS

To request any statistics other than the defaults (N for class variables and SUM for continuous variables) the statistic keyword must be crossed with the variable in question. Here again, remember each crossing is counted as a depth and compared with the value of DEPTH =. The following descriptive statistics are available in the TABULATE procedure. (SAS® Guide to TABULATE Processing, Page 22):

●	CSS	Sum of the Squares corrected for the mean
●	CV	Percent coefficient of variation
●	MAX	Maximum value
●	MEAN	Arithmetic mean
●	MIN	Minimum value
●	N	Number of observations with non-missing variable values
●	NMISS	Number of observation with missing variable values
●	PCTN	Percentage that one frequency represents of another
●	PCTSUM	Percentage that one sum represents of another
●	PRT	Two-tailed *p*-value for student's *t* with *n* - 1 degrees of freedom, the probability under the null hypothesis of obtaining an absolute value of *t* greater than the *t* value observed in this sample
●	RANGE	Maximum value - minimum value
●	STD	Standard deviation
●	STDERR	Standard error of the mean
●	SUM	Weighted sum
●	SUMWGT	Sum of weights
●	USS	Uncorrected sum of squares
●	T	Student's *t* for H_0: population mean = 0
●	VAR	Variance.

```
    FOOTNOTE1 'NEW MEMBERS SINCE 01/01/94';
    PROC TABULATE DATA = MEMBERS
      FORMCHAR = '           ' NOSEP;
      CLASS DIVISION REGION;
      VAR MEMCNT;
      FORMAT DIVISION $DIVFMT.;
      FORMAT REGION $REGFMT.;
      KEYLABEL
          SUM = ' '
        PCTSUM = ' ';
      TABLE ALL = '1994 MEMBERSHIP DATA',
            (DIVISION = 'NATIONAL AREA' *
             REGION = 'LOCAL CLUB')
             ALL = 'TOTAL NEW MEMBERS',
             MEMCNT = 'MEMBER COUNT' * F = COMMA20.2
             MEMCNT = 'PERCENT' * PCTSUM * F = 8.4
           / RTS = 40 BOX = 'EASTERN SPORTS CAR CLUB';
    RUN;
```

```
1994 MEMBERSHIP DATA

EASTERN SPORTS CAR CLUB                    MEMBER COUNT  PERCENT

NATIONAL AREA   LOCAL CLUB
NORTHEAST       STEEL CITIES                     956.00  21.0758
                NEW ENGLAND                      132.00   2.9101
                FINGER LAKES                   1,096.00  24.1623
                GLEN                              50.00   1.1023
                MOHAWK-HUDSON                     25.00   0.5511
SOUTHEAST       BUCCANEER                        200.00   4.4092
                SOUTH FLORIDA                    309.00   6.8122
                CAROLINA                          12.00   0.2646
                ATLANTA                           95.00   2.0944
MID-ATLANTIC    WASHINGTON D C                   654.00  14.4180
                TRI-REGION                        20.00   0.4409
                BLUE RIDGE                       987.00  21.7593
TOTAL NEW MEMBERS                              4,536.00 100.0000

                NEW MEMBERS SINCE 01/01/94
```

CONCLUSION

The TABULATE procedure is a very powerful, but somewhat complicated tool for presenting data in a table (spreadsheet) format. The best way to get the most out of this procedure is to experiment on a small known set of variables. Record the input, process (PROC TABULATE statements), and output for future reference. In actuality, this paper is a partial record of my experiments. To further acquaint yourself with the TABULATE procedure, review the latest documentation from the SAS Institute.

REFERENCES

SAS Institute Inc. (1985), SAS® User's Guide: Basics, Version 5 Edition, Cary, NC: SAS Institute, INC.

SAS Institute Inc. (1989), SAS® Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute, INC.

SAS Institute Inc. (March 1993), SAS® Technical Report P-242, SAS® Software: Changes and Enhancements, Second Edition, Cary, NC: SAS Institute, INC.

SAS Institute Inc. (1993), SAS® Guide to TABULATE Processing, Second Edition, Cary, NC: SAS Institute, INC.

Getting the Most Out of PROC TABULATE (1994), Rockville, MD: Computer Sciences Corporation, Professional Services Group

Jaffe, Jay (1989), Mastering the SAS® System, New York, NY: Van Nostrand Reinhold

ACKNOWLEDGEMENTS

I would like to acknowledge the SAS Institutes Manual SAS® Guide to TABULATE Processing as a comprehensive guide and tutorial for the TABULATE procedure. This paper is a record of my experiments with the TABULATE procedure and relied very heavily on the contents of the TABULATE manual.

I would like to thank Eli Henrickson, Lisa Shutta, Tom McLeod, Elsie Fletcher and Russ Johnson, who reviewed this paper and offered many constructive suggestions.

The author can be reached at:

Computer Sciences Corporation
Professional Services Group
1301 Piccard Drive, Second Floor
Rockville, MD 20850