# How to use Proc Tabulate

Tasha Chapman, Oregon Department of Consumer and Business Services

Proc Tabulate can be a very powerful tool for creating professional looking tables. And unlike Proc Report, Proc Tabulate is compatible with multilabel formats, making it possible to create hierarchical tables with groups and sub-groups. However, the syntax for Proc Tabulate is not particularly intuitive, so many programmers tend to shy away from it.

The purpose of this paper is to acquaint the reader with the basics of Proc Tabulate syntax. By the end of this paper, you should know how to create a basic table, use formats in your table, change headers and data labels, add row and column totals, and implement basic statistics.

The following examples were created using SAS® version 9.1.3. The output was created using ODS PDF in the SASweb style.

## *Getting started – the basics of Proc Tabulate*

Before delving into the complexities[1] of Proc Tabulate, let's break this procedure down to the bare bones. The simplest Proc Tabulate includes a class statement and a table statement.

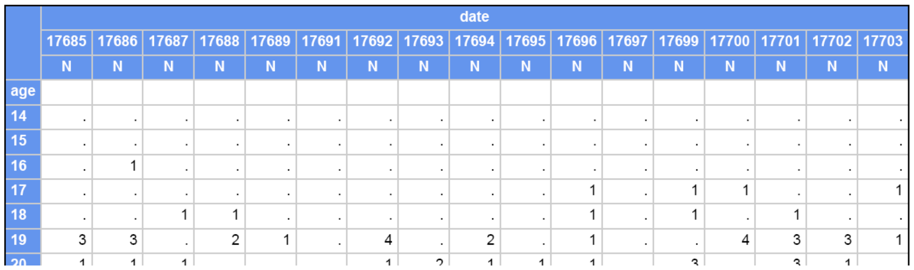
The **Class statement** tells SAS which variables you are going to be using in the table to categorize your observations. Any variable that will be used for categorization in your table must be in the class statement.

The **Table statement** is where you build your table, telling SAS which variables are row expressions, which are column expressions, and (if applicable) which are page expressions. Rows, columns, and pages are separated using commas, like so:

page-expression, row-expression, column-expression

Both page and row expressions are optional. Any table with a single dimension will simply show columns. In the first example, we'll show a two-dimensional table with rows and columns. "Age" will be the rows, and "Date" will be the columns:

```
proc tabulate;
class age date;
table age, date;
run;
```



| | date | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 17685 | 17686 | 17687 | 17688 | 17689 | 17691 | 17692 | 17693 | 17694 | 17695 | 17696 | 17697 | 17699 | 17700 | 17701 | 17702 | 17703 |
| | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N | N |
| age | | | | | | | | | | | | | | | | | |
| 14 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| 15 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| 16 | . | 1 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| 17 | . | . | . | . | . | . | . | . | . | . | 1 | . | 1 | 1 | . | . | 1 |
| 18 | . | . | 1 | 1 | . | . | . | . | . | . | 1 | . | 1 | . | 1 | . | . |
| 19 | 3 | 3 | . | 2 | 1 | . | 4 | . | 2 | . | 1 | . | . | 4 | 3 | 3 | 1 |
| 20 | 1 | 1 | 1 | | | | 1 | 2 | 1 | 1 | 1 | | 3 | | 3 | 1 | |

In this table both age and date are class variables, so they both appear in the class statement. This means that SAS is categorizing the observations according to these variables. In addition, because we didn't specify what statistics we wanted to display, Proc Tabulate defaulted to using the N statistic (the **n**umber of observations). In essence, SAS counted how many observations were in each category and displayed this count in the table.

® SAS is a registered trademark of SAS Institute in the USA and other countries. ® indicates USA registration.

[1] Don't panic… Complexities is a code word for really cool features!

## *Using formats with Proc Tabulate*

You've just created your first table using Proc Tabulate. Congratulations… but wow, that's ugly… Let's figure out how to fix it.

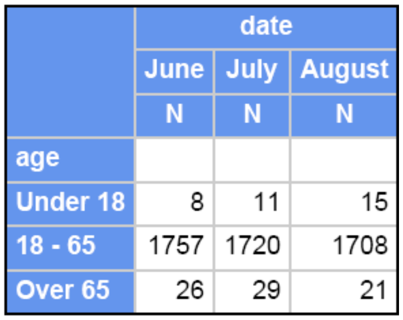
First of all, you'll notice that the variable "date" looks nothing like a date. That's because SAS stores dates as numbers (the number of days since January 1, 1960) and SAS is displaying this internal date value. Second, there are too many ages and too many dates in our dataset to make the table understandable. The table would be easier to read if we used groupings. Both of these issues can be fixed with formats.

For the date variable, we can use a SAS supplied date format. In this case, we'll use monname, which will display the name of the month. For the age variable, there is no applicable SAS format, so we'll need to create a value format[2] using Proc Format:

```
proc format;
value agefmt
low-17 = 'Under 18'
18-65 = '18 - 65'
66-high = 'Over 65';
run;
```

To apply the formats to the variables, we use a **Format statement**, similar to that seen in other procedures. To use the Format statement, simply list the variable (or variables), followed by their respective format, like so:

```
proc tabulate;
format age agefmt. date monname.;
class age date;
table age, date;
run;
```



| | date | | |
|---|---|---|---|
| | June | July | August |
| | N | N | N |
| age | | | |
| Under 18 | 8 | 11 | 15 |
| 18 - 65 | 1757 | 1720 | 1708 |
| Over 65 | 26 | 29 | 21 |

Congratulations, you can now create a basic table using Proc Tabulate!

## *Proc Tabulate operators*

In the next few examples, we'll discuss how to apply statistics to your table, add multiple variables to a single dimension, and apply formatting techniques to your table. But before continuing on, let's discuss operators in Proc Tabulate. Operators are the symbols (commas, spaces, asterisks, etc.) that are used to define the relationships among variables, statistics, and other elements.

The first operator that we've already used is the **comma**, which is used to separate the different dimensions of our table (page, row, and column).

The next operator we'll discuss is the **blank space**. A blank space concatenates – or links together – different elements within a single dimension. Elements linked in the Table statement by a blank space will appear side-by-side on the table.

[2] A value format replaces the specified field values or ranges with new data labels or category labels. By using a value format, all observations that fall within the specified values or ranges are grouped together under the new label.

Another common operator is the **asterisk**. The asterisk can be used for many purposes, such as associating statistical keywords and formats with a particular variable, and crossing elements within a dimension.

The **equal sign** is an operator most commonly used for formatting and table design. The equal sign can be used to override default cell formats or assign a label to an element.

**Parentheses** are another useful operator. Parentheses can be used to group elements and associate a particular operation with multiple elements.

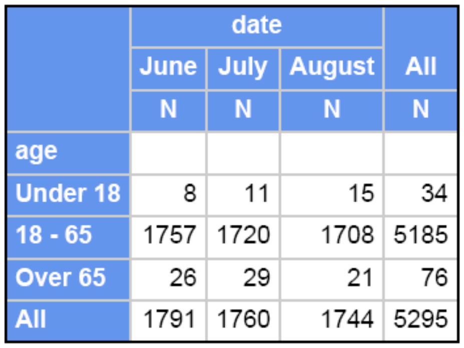
Appendix A provides a handy reference on the most commonly used operators in Proc Tabulate. You may want to refer to this as we continue through the next few sections.

## *Adding total rows and columns*

Let's spice our table up by adding a total row and a total column. Luckily for you, this is really easy to do. To add a total row or column, we add the keyword ALL to the table statement. (SAS refers to this as the "Universal Class Variable ALL.")

ALL[3] will create a sum total for whichever dimension(s) you choose. In this particular case, we want to have a total for both our row dimension and our column dimension, so we'll be adding the ALL keyword to both dimensions, like so:

```
proc tabulate;
format age agefmt. date monname.;
class age date;
table age all, date all;
run;
```



| | date | | | |
|---|---|---|---|---|
| | June | July | August | All |
| | N | N | N | N |
| age | | | | |
| Under 18 | 8 | 11 | 15 | 34 |
| 18 - 65 | 1757 | 1720 | 1708 | 5185 |
| Over 65 | 26 | 29 | 21 | 76 |
| All | 1791 | 1760 | 1744 | 5295 |

Placing the ALL keyword after the variable in question will create a total column in the last column and a total row in the last row. We can also place the ALL keyword before the variable to move the total column to the first column or the total row to the first row, like so:

```
table all age, all date;
```

## *Other class variable statistics*

As mentioned earlier, our simple Proc Tabulate defaulted to presenting the N statistic. However, there are a plethora of other statistics available for use in Proc Tabulate (see Appendix B for a complete list).

To apply a statistic other than N to the table, use an asterisk (*) to associate the desired statistic to the applicable variable. For example, in our previous example, let's say we wanted to show the column percent instead of the number of observations. To do this, we would simply associate the column percent keyword (COLPCTN) to our column variable (date), like so:

[3] Note: If your input dataset includes a variable named ALL, then enclose the Universal Class Variable ALL in quotation marks to distinguish it from your dataset variable.

```
proc tabulate;
format age agefmt. date monname.;
class age date;
table age, date*colpctn;
run;
```

| | date | | |
|---|---|---|---|
| | June | July | August |
| | ColPctN | ColPctN | ColPctN |
| age | | | |
| Under 18 | 0.45 | 0.63 | 0.86 |
| 18 - 65 | 98.10 | 97.73 | 97.94 |
| Over 65 | 1.45 | 1.65 | 1.20 |

Much like in other procedures, parentheses can be used to associate more than one statistic with a variable. For example, if we wanted to show both the column percent and the number of observations, then both keywords would need to be specified within the parentheses:

```
proc tabulate;
format age agefmt. date monname.;
class age date;
table age, date*(N colpctn);
run;
```

| | date | | | | | |
|---|---|---|---|---|---|---|
| | June | | July | | August | |
| | N | ColPctN | N | ColPctN | N | ColPctN |
| age | | | | | | |
| Under 18 | 8 | 0.45 | 11 | 0.63 | 15 | 0.86 |
| 18 - 65 | 1757 | 98.10 | 1720 | 97.73 | 1708 | 97.94 |
| Over 65 | 26 | 1.45 | 29 | 1.65 | 21 | 1.20 |

Statistical keywords can be applied to total rows and columns as well. You can apply the statistical keywords to both the variable and the Universal Class Variable ALL using parentheses:

```
table age, (date all)*(N colpctn);
```

Or, you can apply the statistical keyword(s) to ALL and not the variable:

```
table age, date all*(N colpctn);
```

Or, you can apply the statistical keyword(s) to variable and not ALL:

```
table age, date*(N colpctn) all;
```

Or, you can apply different statistical keywords to ALL and the variable:

```
table age, date*colpctn all*(N colpctn);
```

The possibilities are endless.

## *Placing more than one variable on a single dimension*

In the above examples we applied multiple statistics to a single variable. You can also apply multiple variables to a single dimension. There are two ways to do this. You can put the variables next to each other (known as "concatenation") or you can cross elements within a single dimension.

For example, in addition to date and age, we have another variable titled weekly wage (wkly_wage). Let's say that we've applied a value format to weekly wage that groups people into those who make less than $600 per week and those who make $600 per week or more. We'd like to see the observations categorized by date and by wage, in the same table, with the columns for wage next to the columns for date. To place these columns side-by-side, simply list the variables with a space between them:

```
proc tabulate;
format age agefmt. date monname.
       wkly_wage wagefmt.;
class age date wkly_wage;
table age, date wkly_wage;
run;
```

| | date | | | WKLY_WAGE | |
|---|---|---|---|---|---|
| | June | July | August | Less than $600 | $600 or more |
| | N | N | N | N | N |
| age | | | | | |
| Under 18 | 8 | 11 | 15 | 34 | . |
| 18 - 65 | 1757 | 1720 | 1708 | 3075 | 2110 |
| Over 65 | 26 | 29 | 21 | 56 | 20 |

## *Crossing elements within a dimension*

In addition to placing columns side-by-side, you can also place columns within each other. For example, in our table, we'd like to categorize our columns by both date and wage. This is known as crossing elements within a dimension. Instead of separating the variables using a space, crossing elements within a dimension is done using an asterisk:

```
proc tabulate;
format age agefmt. date monname. wkly_wage wagefmt.;
class age date wkly_wage;
table age, date*wkly_wage;
run;
```

| | date | | | | | |
|---|---|---|---|---|---|---|
| | June | | July | | August | |
| | WKLY_WAGE | | WKLY_WAGE | | WKLY_WAGE | |
| | Less than $600 | $600 or more | Less than $600 | $600 or more | Less than $600 | $600 or more |
| | N | N | N | N | N | N |
| age | | | | | | |
| Under 18 | 8 | . | 11 | . | 15 | . |
| 18 - 65 | 1039 | 718 | 1011 | 709 | 1025 | 683 |
| Over 65 | 19 | 7 | 21 | 8 | 16 | 5 |

## *Calculating means, medians, and other related statistics*

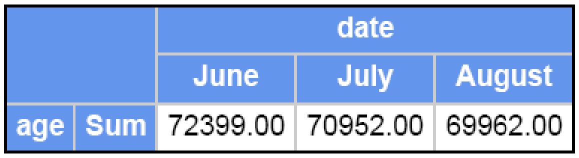
So far we've discussed how to use class variable statistics. These are statistics that rely on categorizing observations according to class variables, such as counts, percents, and the like. However, Proc Tabulate also allows the user to apply other statistics that rely on analyzing the value of numeric variables, such as mean, median, minimum, maximum, and other similar statistics.

For example, let's say we wanted to analyze the variable age. Age will no longer be a class variable, but will instead be considered an analysis variable. To declare age as an analysis variable, we must include a Var statement in our Proc Tabulate.

The **Var Statement** tells SAS which variables you are going to analyze in your table. Any analysis variable must be in the Var statement.

If we wanted to analyze age, while keeping date as a class variable, the simplest table might look like this:

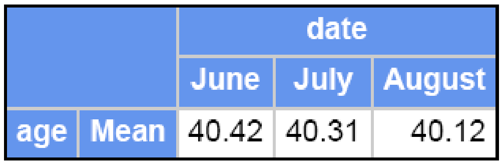
```
proc tabulate;
format date monname.;
class date;
var age;
table age, date;
run;
```



| | | date | | |
|---|---|---|---|---|
| | | June | July | August |
| age | Sum | 72399.00 | 70952.00 | 69962.00 |

Because no statistics were specified for our analysis variable, SAS defaulted to using the SUM statistic and added the age values of all our observations. A more meaningful statistic might be the MEAN, or average age. To calculate the mean, we must associate the statistical keyword with the analysis variable:

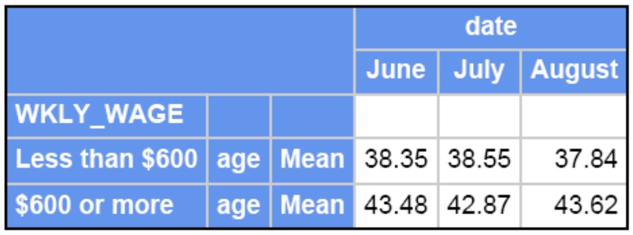
```
proc tabulate;
format date monname.;
class date;
var age;
table age*mean, date;
run;
```



| | | date | | |
|---|---|---|---|---|
| | | June | July | August |
| age | Mean | 40.42 | 40.31 | 40.12 |

The table can be further enhanced by adding a class variable to the row dimension, such as weekly wage:[4]

```
proc tabulate;
format date monname. wkly_wage wagefmt.;
class date wkly_wage;
var age;
table wkly_wage*(age*mean), date;
run;
```



| | | | date | | |
|---|---|---|---|---|---|
| | | | June | July | August |
| WKLY_WAGE | | | | | |
| Less than $600 | age | Mean | 38.35 | 38.55 | 37.84 |
| $600 or more | age | Mean | 43.48 | 42.87 | 43.62 |

## *Assigning formats to table cells*

Now that we know how to build a table, let's talk about how to format the table, applying headers, cell widths, and labels that will make the table easier to read.

The first thing we'll look at is formatting the values in the table cells. If no format is applied, SAS will default to using 12.2 in the table cells (although, as you can see from the examples above, the ODS style that you choose may adjust the default format). This default format can be overridden using the "format =" option. To assign a format to all the table cells, use the "format=" option in the Proc Tabulate statement:

```
proc tabulate format=comma6.;
```

However, Proc Tabulate gives us a little flexibility with table cell formats. It is possible to apply a particular format to only a specific statistic. Going back to our previous example showing N and COLPCTN, we could retain the comma6 format for the N statistics, but show the column percent to

[4] In this example, the parentheses are technically unnecessary, but can be visually helpful to distinguish between the class variable and the analysis variable.

1/10th of a percent (e.g., 14.6%), using a format with a width of 6.1 for this statistic.[5] To do this, you use the asterisk to associate the format modifier "Format=" to the COLPCTN statistic. "F=" can be used instead of "format=" to save keystrokes:

```
table age, date*(N colpctn*f=6.1);
```

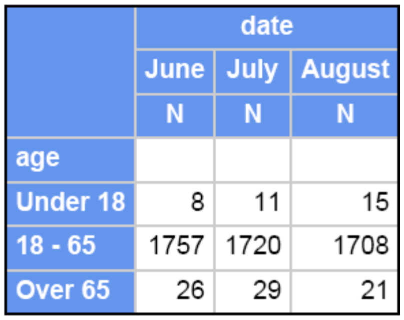
The format applied in the table statement will override the format in the Proc Tabulate statement for the specified instances.

## *Changing row and column headers*

Using the "format=" modifier is only the tip of the iceberg. Proc Tabulate offers a wide variety of options for modifying and customizing your table.

Let's use one of our earlier tables as an example:

```
proc tabulate;
format age agefmt. date monname.;
class age date;
table age, date;
run;
```



| | date | | |
|---|---|---|---|
| | June | July | August |
| | N | N | N |
| age | | | |
| Under 18 | 8 | 11 | 15 |
| 18 - 65 | 1757 | 1720 | 1708 |
| Over 65 | 26 | 29 | 21 |

The most informative elements of a table are row and column headers, but the default headers (the variable names "age" and "date") are not very descriptive. Let's put in meaningful titles instead.

Applying data labels in Proc Tabulate is fairly simple. For example, you can change the data label for a particular variable within the table statement, like so:

```
table age='Age of claimant', date='Date of acceptance';
```

Instead of showing "age", Proc Tabulate will display "Age of claimant," and instead of "date," the new column header will be "Date of acceptance."

You can also change the label for the Universal Class Variable ALL in the same way. Without a data label, Proc Tabulate will label the total row or column with the word "all." To use a different header instead, simply assign a data label as we did with the other variables:

```
table age='Age of claimant' all='All claimants',
      date='Date of acceptance' all='Three month total';
```

## *Changing data labels using Keylabel*

You can change the data label for a statistical keyword (such as N) in either the table statement, or in the optional Keylabel statement. Let's talk about the Keylabel statement first.

[5] In format widths, the number before the period is the total character width, while the number after the period is the number of digits after the decimal point. So a format with a width of 6.1 would be six total characters long, with one digit after the decimal point. (Be aware that the decimal point also counts as a character when calculating the width.) Format widths can be used by themselves, or with a format type such as "best," "comma," or "dollar."

The **Keylabel statement** will change the label of a particular keyword for the duration of the Tabulate procedure. Since we want the new label for N to be applied to each and every instance in which "N" currently appears, the Keylabel statement is the best option. In the example below, Proc Tabulate will print "Number of observations" instead of "N" in the resulting table.

```
proc tabulate;
format age agefmt. date monname.;
class age date;
keylabel N='Number of observations';
table age='Age of claimant', date='Date of acceptance';
run;
```

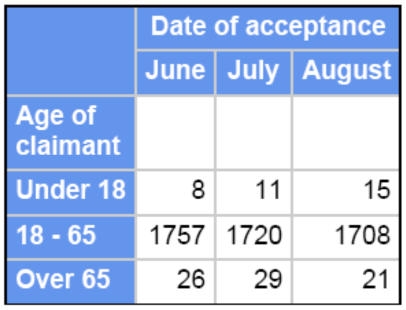
However, N is the only statistic that we're using, so this label is a bit unnecessary. Let's get rid of it. If you don't want any label to be applied to a keyword, you can replace this label with a blank space. For example:

```
keylabel N=' ';
```

Instead of printing the label "N," Proc Tabulate will print no label. The entire row full of N's in the current table will be deleted.

Let's put these two pieces together and see what it looks like:

```
proc tabulate;
format age agefmt. date monname.;
class age date;
keylabel N=' ';
table age='Age of claimant',
      date='Date of acceptance';
run;
```



| | Date of acceptance | | |
|---|---|---|---|
| | June | July | August |
| Age of claimant | | | |
| Under 18 | 8 | 11 | 15 |
| 18 - 65 | 1757 | 1720 | 1708 |
| Over 65 | 26 | 29 | 21 |

## *Changing data labels using in the Table statement*

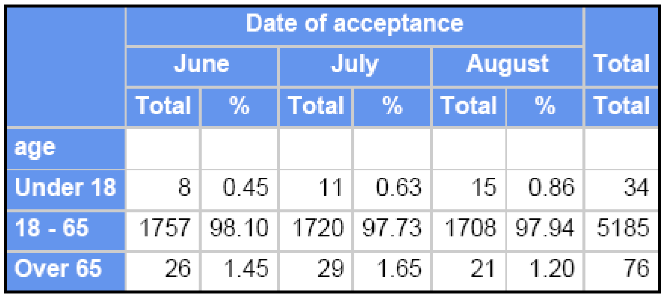
As we've already seen, by default SAS will use the name of the statistical keyword as the label. As in the last example, this can be overridden using the Keylabel statement. This method works fine if you want to replace every instance with the same label. However, this might not always be the case. The assigned labels in the Keylabel statement can also be overridden for specific instances using the Table statement. You can change the label for a statistical keyword in the Table statement, in the same way that labels for variables are changed:

```
table age, date='Date of acceptance'*colpctn='Percent';
```

This is particularly useful if you are applying different statistics to a class variable and a total row or column. For example, let's say we had a table that showed both N and COLPCTN. This table also has a total column (created using ALL), which shows only N.[6] We've also used the Keylabel statement to change "N" and "COLPCTN" to "Total" and "%" instead:

[6] N does not need to be explicitly requested for the total column like it is shown here. Because we are dealing with only class variables, if no statistics are specified Proc Tabulate will default to displaying N. However, it is explicitly stated here to prepare for the next example.

```
proc tabulate;
format age agefmt. date monname.;
class age date;
keylabel N='Total' colpctn='%';
table age, date='Date of acceptance'*(N colpctn) all='Total'*N;
run;
```



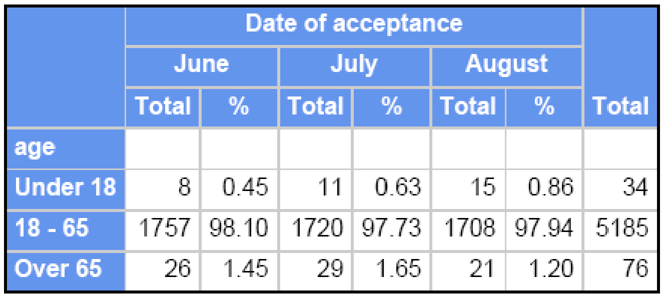
| | Date of acceptance | | | | | | Total |
|---|---|---|---|---|---|---|---|
| | June | | July | | August | | Total |
| | Total | % | Total | % | Total | % | Total |
| age | | | | | | | |
| Under 18 | 8 | 0.45 | 11 | 0.63 | 15 | 0.86 | 34 |
| 18 - 65 | 1757 | 98.10 | 1720 | 97.73 | 1708 | 97.94 | 5185 |
| Over 65 | 26 | 1.45 | 29 | 1.65 | 21 | 1.20 | 76 |

The table looks a little silly because the label "Total" appears twice in the total column. We can override the label for the N statistic in the total column, by defining a new label using the Table statement. For example, if we wanted the label to be "Number," the code would be:

```
table age, date='Date of acceptance'*(N colpctn) all='Total'*N='Number';
```

The other option is to get rid of the label entirely for the N statistic of the total column. We do this by replacing the label with a blank space:

```
proc tabulate;
format age agefmt. date monname.;
class age date;
keylabel N='Total' colpctn='%';
table age, date='Date of acceptance'*(N colpctn) all='Total'*N=' ';
run;
```



| | Date of acceptance | | | | | | Total |
|---|---|---|---|---|---|---|---|
| | June | | July | | August | | |
| | Total | % | Total | % | Total | % | |
| age | | | | | | | |
| Under 18 | 8 | 0.45 | 11 | 0.63 | 15 | 0.86 | 34 |
| 18 - 65 | 1757 | 98.10 | 1720 | 97.73 | 1708 | 97.94 | 5185 |
| Over 65 | 26 | 1.45 | 29 | 1.65 | 21 | 1.20 | 76 |

## *Controlling the box in the upper left hand corner*

So far we've spent a lot of time talking about column headers. Let's discuss the row header. The current placement for the label for "age" is creating a blank row, which is a bit unsightly. However, above the label "age" there is a blank box in the upper left hand corner. Let's get rid of that blank row, and move the label into the empty box.

The first thing we need to do is get rid of the current label for "age." We'll do this by replacing the label in the Table statement with a blank space, similar to what we did in earlier examples:

```
table age=' ', date='Date of acceptance';
```

This will get rid of the label for "age," as well as the blank row that the label is creating.

In its place, let's put a new row header in the empty box in the upper left hand corner of the table. This box is actually controlled by an optional command in the table statement. All options in the Table statement follow a slash:

```
table age=' ', date='Date of acceptance' / box='Age of claimant';
```

Putting the row header in the upper left hand box is only a matter of style. This box can be used for many things, such as the title of the table, or even a graphic. For a quick reference of the multiple formatting options for Proc Tabulate discussed in this paper, from using the "format =" option to control table cells to using the "box=" option, see Appendix C.

## *Other important features*

Now you can build a basic table using Proc Tabulate, and even format it to your liking. Before sending you off into the world, here are a few more important features that you should know about.

### Missing option

As with many SAS procedures, observations with missing data in a class variable, by default, are excluded from the analysis. In other words, if you had 100 observations, but 5 were missing values for "date," only the remaining 95 would be included in the analysis. To include observations with missing data in the analysis, the **missing option** must be utilized in the Proc Tabulate statement:

```
proc tabulate missing;
```

The missing option considers missing values as valid values when creating the combinations of class variables. The missing option can also be utilized in the Class statement to apply to only certain class variables. In the example below, observations with missing weekly wage (wkly_wage) will be excluded, but observations with missing age will be included:

```
proc tabulate;
format age agefmt. wkly_wage wagefmt.;
class wkly_wage;
class age /missing;
table age, wkly_wage;
run;
```

### Misstext option

By default, table cells with no observations will display a period (.) to denote the missing value. However, instead of a period, this can be changed to a dash or a zero or whatever the user  chooses (up to 256 characters) by utilizing the **misstext option** in the Table statement:

```
proc tabulate;
format age agefmt. date monname. wkly_wage wagefmt.;
class age date wkly_wage;
table age, date*wkly_wage / misstext='-';
run;
```

## *Final thoughts*

Hopefully you have found this paper useful. What we've covered here should help you through most of your Proc Tabulate endeavors, but additional tips and tricks are being discovered everyday.

For additional resources, check out SAS Help and Documentation and sasCommunity.org (www.sascommunity.org/wiki).

For a guide on how to use multilabel formats with Proc Tabulate effectively, check out Making the Most Out of Multilabel Formats, by Tasha Chapman, presented at the 2008 Pacific Northwest SAS Users Group conference and available at sasCommunity.org.

Additional questions can also be addressed to the author:
Tasha Chapman – Tasha.L.Chapman@state.or.us

Many thanks to all the people who first taught me how to use Proc Tabulate and helped me along the way.

## ***Appendix A: Proc Tabulate operators***

Operators are symbols that tell Proc Tabulate what actions to perform on the variables, statistics, and other elements. The following table lists the common operators in Proc Tabulate:

| Operator | Action |
|---|---|
| **, comma** | separates dimensions of the table (*page, row, column*) |
| *** asterisk** | crosses elements within a dimension;<br>associates statistical keywords and formats with variables |
| **Blank space** | concatenates elements within a dimension |
| **( ) parentheses** | groups elements;<br>associates an operator with each element in the group |
| **= equal** | overrides default cell format or assigns label to an element |

Information taken in large part from the SAS Help and Documentation

## *Appendix B: Statistics that are available in Proc Tabulate*

Use the following keywords to request statistics in the TABLE statement or to specify keywords in the KEYWORD or KEYLABEL statement. If a variable name (class or analysis) and a statistic name are the same, then enclose the statistic name in single quotation marks – for example, 'MAX'.

***Descriptive statistic keywords***

| | | |
|---|---|---|
| COLPCTN | N | SKEWNESS \| SKEW |
| COLPCTSUM | NMISS | STDDEV \| STD |
| CSS | PAGEPCTN | SUM |
| CV | PCTN | SUMWGT |
| KURTOSIS \| KURT | PCTSUM | UCLM |
| LCLM | RANGE | USS |
| MAX | REPPCTSUM | VAR |
| MEAN | ROWPCTN | |
| MIN | ROWPCTSUM | |

***Quantile statistic keywords***

| | | |
|---|---|---|
| MEDIAN \| P50 | Q1 \| P25 | P99 |
| P1 | Q3 \| P75 | QRANGE |
| P5 | P90 | |
| P10 | P95 | |

***Hypothesis testing keywords***

| | |
|---|---|
| PROBT | T |

To compute standard error of the mean (STDERR) or Student's t-test, you must use the default value of the VARDEF= option, which is DF. The VARDEF= option is specified in the PROC TABULATE statement.

Use both LCLM and UCLM to compute a two-sided confidence limit for the mean. Use only LCLM or UCLM to compute a one-sided confidence limit. Use the ALPHA= option in the PROC TABULATE statement to specify a confidence level.

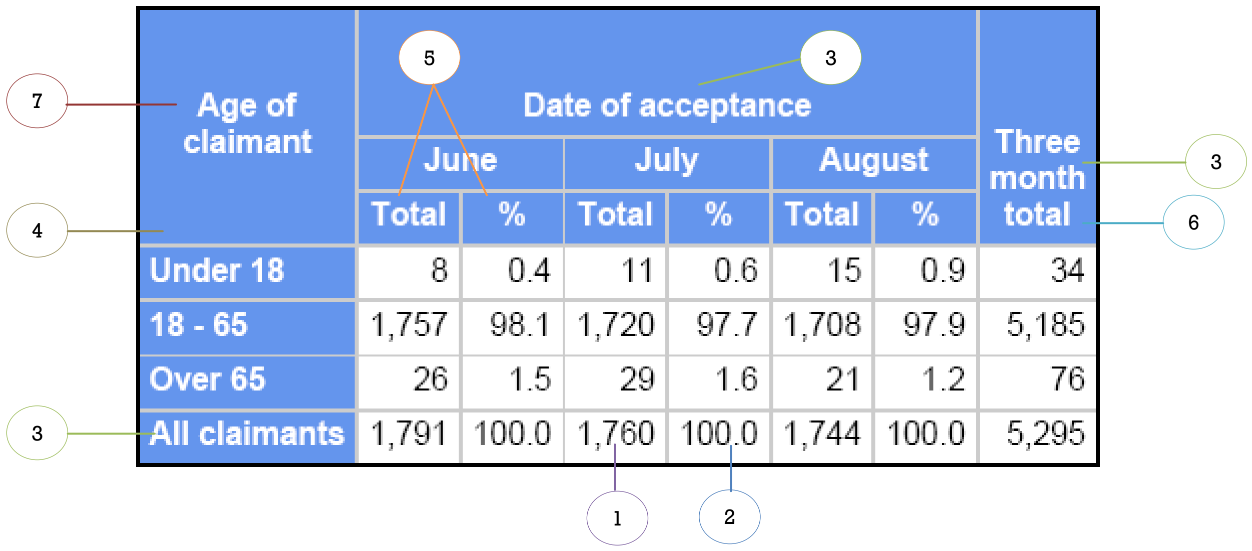
To compute weighted quantiles, you must use QMETHOD=OS in the PROC TABULATE statement.

Information taken in large part from the SAS Help and Documentation

## *Appendix C: Putting it all together - Formats*

Now that we've seen what each piece of formatting code can do, let's put it all together.

```
proc tabulate format=comma6.;
format age agefmt. date monname.;
class age date;
keylabel N='Total' colpctn='%';
table age=' ' all='All claimants',
      date='Date of acceptance'*(N colpctn*f=6.1)
      all='Three month total'*N=' '
      / box = 'Age of claimant';
run;
```



| Age of claimant | Date of acceptance | | | | | | Three month total |
|---|---|---|---|---|---|---|---|
| | June | | July | | August | | |
| | Total | % | Total | % | Total | % | |
| Under 18 | 8 | 0.4 | 11 | 0.6 | 15 | 0.9 | 34 |
| 18 - 65 | 1,757 | 98.1 | 1,720 | 97.7 | 1,708 | 97.9 | 5,185 |
| Over 65 | 26 | 1.5 | 29 | 1.6 | 21 | 1.2 | 76 |
| All claimants | 1,791 | 100.0 | 1,760 | 100.0 | 1,744 | 100.0 | 5,295 |

1. Using the **format** = option in the Proc Tabulate statement sets the default format for the table cells.
2. Using the **format** = option in the Table statement overrides the default format for the table cells in the specified instances.
3. Headers for rows, columns, and totals can be changed in the Table statement.
4. When a header is set to blank, no label will be displayed. In this case the row header is set to blank. No label for the row header is displayed.
5. Using the Keylabel statement will specify labels for statistical keywords.
6. The assigned labels in the Keylabel statement can also be overridden for specific instances using the Table statement. In this case the label is being set to blank, so no label will be printed.
7. The box in the upper left hand corner is filled with the text "Age of claimant" using the **box =** option in the Table statement.