

USING PROC TABULATE - A REVIEW OF THE BASICS AND A LOOK AT USING THIS PROCEDURE WITH MULTIPLE RESPONSE DATA

Catherine M. Cerone, Cerone Consulting

Abstract

PROC TABULATE allows the user a good deal of flexibility in generating tables of percentages or frequencies. This flexibility allows to correctly calculate percentages on multiple response data. A multiple response question is one for which a panelist may respond with more than one response. Therefore, care must be taken when percentages are calculated on these questions.

This paper will begin with some basic information about using PROC TABULATE, take a look at some commonly used options and then discuss how to define percentages in PROC TABULATE to make an analysis of multiple response data more meaningful.

Overview

PROC TABULATE is one of the descriptive procedures available in the SAS system. It allows you to summarize data in tables with one, two, or three dimensions. A one dimensional table would contain a column of data representing the values of a specified variable. A two dimensional table would contain a column representing the values of one variable and rows representing values of a different variable. A three dimensional table would be made up of several two dimensional tables, one for each level of a third specified variable.

PROC TABULATE also allows you some control over the appearance of the table. For example, the report columns may be the values of one variable, the values of several

variables side by side, or the values of one variable grouped by the values of another variable. PROC TABULATE will also allow you to modify some of the text that appears in the table as well as modifying some of the lines that are drawn around the table.

Perhaps one of the most important areas in which PROC TABULATE gives you some flexibility is in defining how percentages are calculated. This flexibility is very beneficial in dealing with multiple response data. Multiple response questions frequently occur in survey data (i.e. brands used, comments, etc.). In analyzing these questions, percentages are frequently calculated. Unfortunately, the default method for calculating these percentages doesn't give us a usable number. Luckily, PROC TABULATE offers the user the ability to redefine how percentages are calculated in order to calculate a more useful percentage.

Using data collected from a survey developed to guide the formation of the Greater Cincinnati SAS Users Group, this paper will discuss each of the topics mentioned above.

The Data Set

Following the SUGI-16 meeting in New Orleans, several people from the Cincinnati area decided to investigate forming a local users group. It was decided that the best way to judge the level of interest in such a group was to send out a questionnaire to people in the Cincinnati area who were currently using the SAS system. This questionnaire included questions to

help us understand the users in the Cincinnati area as well as what they would like in a local users group. The questionnaire included some single response variables, such as "Type of organization", "Length of time you have been using SAS software", and "How often should the group meet?", and some multiple response variables such as "In what capacity do you use SAS software?", "Which SAS products have you used in the past?", "Which SAS products are you interested in learning more about?", and "Which areas would you be interested in helping with?". This is the data set that will be used for all examples shown in this paper.

PROC TABULATE syntax:

In order to generate a table with PROC TABULATE, at least three statements are needed. There must be:

1. The Proc Tabulate statement
2. Either a Class or Var statement or both. (A class variable would be used to identify class variables; a var statement would be used to identify analysis variables.)
3. A Table statement.

The Proc Tabulate statement may be followed by any of several options, including data, depth, format, formchar, missing, noseps, order, and vardef. The class and var statements would be followed by a list of variables. The table statement would include the variables that will be contained in the tables along with the operators necessary to specify the type of table being generated. It may also be followed by any of several options. Some of the options available in the Proc Tabulate and Table statements will be discussed

later in this paper.

Below is a table of the operators used in the table statement.

Operator	Purpose
comma ,	separates dimensions
asterisk *	crosses elements
blank space	concatenates elements
angle brackets < >	specify denominator

In addition to the statements mentioned above, the following statements can also be used in PROC TABULATE:

By
Format
Freq
Keylabel
Label
Weight

Some of these statements will also be discussed later in this paper.

Generating Tables:

As mentioned above, PROC TABULATE allows us to generate a variety of types of tables. The simplest table would contain data on just one variable. For example, suppose we wanted to look at the responses to the survey question "For what type of organization do you work?" We could use the following code:

```
PROC TABULATE DATA=DIR.ALL;
  CLASS TYPORG;
  TABLE TYPORG;
```

to generate the table shown on the next page.

Type of Organization				
Banking/Investments	College/University	Consulting	Consumer Goods	Government Agency
N	N	N	N	N
2.00	37.00	17.00	23.00	26.00

Type of Organization				
Hospital/Healthcare	Insurance	Manufacturing	Utility	Other
N	N	N	N	N
5.00	17.00	51.00	8.00	24.00

When we are using PROC TABULATE, there is a universal class variable called ALL that can be very useful. ALL summarizes all the class variables in one dimension.

We can use this variable to help make a one dimensional table easier to read. We will create a table with the same information contained in the first table, but this time the organization types will form the rows of the table. This will allow all the organization types to be seen in one table instead of being split into two. Although we are really only dealing with one variable, we will write a table statement for a two dimensional table. The first variable (the row variable) will be TYPORG and the second variable (the column variable) will be ALL. The code shown below will produce the following table.

```
PROC TABULATE DATA=DIR.ALL;
CLASS TYPORG;
TABLE TYPORG,ALL;
RUN;
```

	ALL
	N
Type of Organization	
Banking/Investments	2.00
College/University	37.00
Consulting	17.00
Consumer Goods	23.00
Government Agency	26.00
Hospital/Healthcare	5.00
Insurance	17.00
Manufacturing	51.00
Utility	8.00
Other	24.00

Using this fairly simple table, lets take a look at some of the options we can use to modify the appearance of the table.

The first is an option called

NOSEP

that is used in the PROC TABULATE statement. It will remove the horizontal separating lines from the table. If we used the same code as was used in the previously example with the following change to the proc step

```
PROC TABULATE DATA=DIR.ALL NOSEP;
```

we would generate the following table.

	ALL
	N
Type of Organization	
Banking/Investment	2.00
College/University	37.00
Consulting	17.00
Consumer Goods	23.00
Government Agency	26.00
Hospital/Healthcare	5.00
Insurance	17.00
Manufacturing	51.00
Utility	8.00
Other	24.00

If, instead of removing the separators, we wanted to use different characters to form the box, we could use the FORMCHAR option. FORMCHAR is followed by a string of 11 characters which define parts of the table outline. These are summarized below.

Position	Outline Section
1	Vertical bar
2	Horizontal bar
3	Upper left corner
4	Upper middle corners
5	Upper right corner
6	Middle left corners
7	Middle middle corners
8	Middle right corners
9	Lower left corner
10	Lower middle corners
11	Lower right corners

So if the PROC TABULATE statement contained the following FORMCHAR option, it would produce the table shown below.

```
FORMCHAR='*%123456789'
1XXXXXXXXXXXXXXXXXXXXX2XXXXXXXXXXXXX3
*                * ALL *
*                4XXXXXXXXXXXXX6
*                * N *
4XXXXXXXXXXXXXXXXXXXXX5XXXXXXXXXXXXX6
*Type of * *
*Organization * *
4XXXXXXXXXXXXX6 *
*Banking/Investment *
*nts * 2.00*
4XXXXXXXXXXXXX5XXXXXXXXXXXXX6
*College/University *
*ty * 37.00*
4XXXXXXXXXXXXX5XXXXXXXXXXXXX6
*Consulting * 17.00*
4XXXXXXXXXXXXX5XXXXXXXXXXXXX6
*Consumer Goods * 23.00*
4XXXXXXXXXXXXX5XXXXXXXXXXXXX6
*Government Agency* 26.00*
4XXXXXXXXXXXXX5XXXXXXXXXXXXX6
*Hospital/Healthcare *
*are * 5.00*
4XXXXXXXXXXXXX5XXXXXXXXXXXXX6
*Insurance * 17.00*
4XXXXXXXXXXXXX5XXXXXXXXXXXXX6
*Manufacturing * 51.00*
4XXXXXXXXXXXXX5XXXXXXXXXXXXX6
*Utility * 8.00*
4XXXXXXXXXXXXX5XXXXXXXXXXXXX6
*Other * 24.00*
7XXXXXXXXXXXXX8XXXXXXXXXXXXX9
```

FORMCHAR can also be used with an index list if you only want to change some of the default settings. For example, to change the characters that appear on the corners to asterisks, you could use the following statement

```
FORMCHAR(3 5 9 11)='****'
```

in the PROC TABULATE statement and produce a table with asterisks at the corners and the default characters elsewhere.

Lastly, if you want no lines drawn around the table, you could use this statement

```
FORMCHAR='
```

in the PROC TABULATE statement and produce a table with no separating lines.

We can also modify the text and appearance of the numbers in the table. For example, since we are dealing with only whole numbers, we might want them to appear without the decimal points and trailing zeros. Also, we might want to replace the terms ALL and N in the table with words that are more meaningful. The following code will do that and produce the table shown below.

```
PROC TABULATE FORMAT=3.;
VAR TYPORG;
TABLE TYPORG,ALL='Total';
KEYLABEL N='Freq.';
```

	To-
	tal

	Fr-
	eq.

Type of	
Organization	

Banking/Investme-	
nts	2

College/Universi-	
ty	37

Consulting	17

Consumer Goods	23

Government Agency	26

Hospital/Healthc-	
are	5

Insurance	17

As you can see, this table has replaced ALL and N with Total and Freq. and has displayed the numbers in the 3. format. However, the column containing the numbers is too narrow to display either Total or Freq. We can fix this by changing the format option in the proc step to a wider format. Also, the values for organization type don't fit neatly in the left column. We can fix this by using the RTS option in the TABLE statement. The following code is a slightly modified version of the code shown above and will produce the table shown below.

```
PROC TABULATE FORMAT=6.;
VAR TYPORG;
TABLE TYPORG,ALL='Total'
/RTS=25;
KEYLABEL N='Freq.';
```

	Total

	Freq.

Type of Organization	

Banking/Investments	2

College/University	37

Consulting	17

Consumer Goods	23

Government Agency	26

Hospital/Healthcare	5

Insurance	17

Manufacturing	51

Utility	8

Other	24

We can take this table one more step and now have both the frequencies and percentage displayed. To do this, we will use the crossing operator, the asterisk. We will

cross frequency (n) and percent (pctn) with typorg to produce a table showing the frequency and percent for each organization type. Again, the following code is a modification of the code shown above and produces the table shown below.

```
PROC TABULATE FORMAT=6.;
  VAR TYPORG;
  TABLE TYPORG*(n,pctn),
    ALL='Total'
    /RTS=25;
  KEYLABEL N='Freq.'
    pctn='Percentage';
```

	Total
Type of	
Organizati-	
on	
Banking/In-	Freq. 2
vestments	
	Percentage 1
College/Un-	Freq. 37
iversity	
	Percentage 18
Consulting	Freq. 17
	Percentage 8
Consumer	Freq. 23
Goods	
	Percentage 11

Notice that in this table there is again not enough room to display the organization type. This is because 'Freq.' and 'Percentage' are taking up some of the 25 spaces set aside by the RTS option. If we change the RTS to 45, we would have sufficient space to display both the organization types and the words "Freq." and "Percentage".

Two and three dimensional tables

The table we have been working with so far gives us information on one

variable. We might want information on more than one variable. Suppose we wanted to see the preferred meeting frequency broken out by willingness to host a meeting. The following code would produce the table shown below.

```
PROC TABULATE DATA=DIR.ALL;
  CLASS OFTEN HOST;
  TABLE OFTEN,HOST;
  RUN;
```

	Willing to host		

		Don't	
		know	
		could	
	Yes	No	pursue

	Freq.	Freq.	Freq.
Meeting Frequency			
Monthly	2	3	27
Every other month	18	8	42
Quarterly	12	18	53
Semi-Annually	1	3	15
Other	.	.	1

Notice that in the table statement the two variables OFTEN and HOST are separated by a comma (.). In the table statement, the comma is used to separate the variables forming different dimensions of the table. In this case, we have a two dimensional table. The variable appearing before the comma (in this case OFTEN) is the row variable. Its values form the row headings in the table. The variable appearing after the comma (in this case HOST) is the column variable. Its values form the column headings in the table. If we were going to generate a three dimensional table, the variable defining the third dimension would be called the page

variable. In that case, the variables in the table statement would appear in the following order: PAGE,ROW,COLUMN.

The following table summarizes the order that variables appear in the table statement for each type of table.

Dimensions	Variable Order
1	Column
2	Row,Column
3	Page,Row,Column

Within a dimension, operators for crossing (the asterisk) and concatenation (the blank space) may be used.

Concatenation:

If we wanted to look at a table similar to the last one shown, but wanted to include "Willing to present" as well as "Willing to host" as column variables, we would concatenate those two variables in the column dimension. To concatenate two variables, we list them together separated by a blank space. The following code shows PRESENT and HOST as concatenated variables (separated by a blank space) in the column dimension and OFTEN as the row variable. The following table is generated by the code shown below.

```
PROC TABULATE DATA=DIR.ALL;
  CLASS OFTEN HOST PRESENT;
  TABLE OFTEN,HOST PRESENT;
```

						Willing to present

						Might
						or
			Willing to host	Would	might	Would
			-----	be	not be	not be
			Don't	willi-	willi-	willi-
			know -	ng to	ng to	ng to
			could	prese-	prese-	prese-
Yes	No	pursue	nt	nt	nt	nt
			-----	-----	-----	-----
			Freq.	Freq.	Freq.	Freq.
			-----	-----	-----	-----
Meeting Frequency						

Monthly	2	3	27	8	19	5

Every other month	18	8	42	13	42	13

Quarterly	12	18	52	12	63	7

Semi-Annually	1	3	15	1	14	4

Other	.	.	1	.	.	1

Crossing Variables:

In addition to concatenation, variables in the same dimension can be crossed. This means that a row (or column) will be created for every combination of the values of each of the two variables being crossed. The following code shows PRESENT and HOST as crossed variables (separated by an asterisk) in the row dimension and OFTEN as the column variable. The table generated by this code is shown on the next page.

```
PROC TABULATE DATA=DIR.ALL;
  CLASS OFTEN HOST PRESENT;
  TABLE PRESENT*HOST,OFTEN;
```

		Meeting Frequency				
		Every		Semi-		
		Month-	other	Quart-	Annua-	
		ly	month	erly	lly	Other
		Freq.	Freq.	Freq.	Freq.	Freq.
Willing to present	Willing to host					
Would be willing to present	Yes	1	7	6	.	.
	No	.	2	3	.	.
	Don't know - could pursue	7	4	3	1	.
Might or might not be willing to present	Yes	1	9	6	1	.
	No	2	4	12	2	.
	Don't know - could pursue	16	28	45	11	.
Would not be willing to present	Yes	.	2	.	.	.
	No	1	2	3	1	.
	Don't know - could pursue	4	9	4	3	1

Three Dimensions:

Finally, we can generate a separate two dimensional table for each level of a third variable. Since each two dimensional table is printed on separate page, the variable defining the third dimension is called a page variable. Suppose we wanted to generate separate tables of Meeting Frequency by Willing to Present for each level of Willing to host. The following code would do that. Notice that the variables appear in the order HOST,PRESENT,OFTEN. Therefore, HOST will be the page

variable, PRESENT will be the row variable, and OFTEN will be the column variable.

The code shown below would produce three tables, one for HOST=YES, one for HOST=NO, and one for HOST=DON'T KNOW. They would all have the same form as the table for HOST=YES, which follows the code.

```
PROC TABULATE DATA=DIR.ALL;
  CLASS OFTEN HOST PRESENT;
  TABLE HOST,PRESENT,OFTEN;
```

Willing to host Yes

		Meeting Frequency				
		Every		Semi-		
		Month-	other	Quart-	Annua-	
		ly	month	erly	lly	Other
		Freq.	Freq.	Freq.	Freq.	Freq.
Willing to present						
Would be willing to present		1	7	6	.	.
Might or might not be willing to present						
		1	9	6	1	.
Would not be willing to present		.	2	.	.	.

Multiple Response Data

As mentioned previously, a multiple response question is one for which a respondent may give several answers. For example, the following question appeared in the GCSUG survey:

Which of the following time periods would you prefer to meet? (Check all that apply)

- ☐ Morning
- ☐ Afternoon
- ☐ Weekday Evenings
- ☐ Weekends

A person completing the survey could check one, two, three or all four responses. Therefore, although the number of people responding to the questionnaire was 211, there were a total of 311 responses to this question.

The data collected on a multiple response question is input differently than it is for a single response question. In order to accommodate the fact that there may be more than one response for an observation, each response in a multiple response question is treated as a separate variable. Therefore, the question discussed above generated four variables: MORNING, AFTNOON, EVENING, and WKEND. Then, for each observation, the variable value is either missing if the person did not check that response, or non-missing if they did check that response.

If we were to use PROC TABULATE to calculate percentages the way we did previously, we would wind up with 100% for each response. This happens due to the fact that each response is a separate variable with only one possible non-missing value. Therefore, for each of the variables, 100% of the non-missing observations have the same value.

We need to tell PROC TABULATE to use a different denominator in calculating the percentages. Luckily, PROC TABULATE allows us to do this by following the PCTN or PCTSUM by a denominator definition enclosed in angle brackets.

The first idea for a denominator might be the total number of response given to that question. However, this is not usually what we really want.

To demonstrate, consider this example. Suppose we surveyed 100 people and asked them the question

above. Suppose also that each of the 100 people checked all four boxes. This would mean that there were 400 responses to this question. If the denominator used in calculating the percentages was 400, we would have the following:

Preferred Time	Percentage
Morning	25%
Afternoon	25%
Weekday Evenings	25%
Weekends	25%

This might lead us to the conclusion that there is not a time when a majority of people could attend. It might even suggest that we could not get enough people together to have a good meeting.

However, if we define the denominator for the percents to be the number of people in the survey (100) instead of the number of responses on this question (400), we would get the following results:

Preferred Time	Percentage
Morning	100%
Afternoon	100%
Weekday Evenings	100%
Weekends	100%

This table shows that all the people responding to the survey would be able to attend meetings no matter when they were held.

Coding

In calculating the percents, we will define the denominator to be the number of people in the survey. In order to do this, we need an analysis variable that has non-missing values for every observation in the data set. In this case, we are using REGNO, which is an identification number. We can then use it in the denominator definition and get the expected results.

In order to establish REGNO as an analysis variable, we need to include it in the VAR statement. Then, in the TABLE statement we follow the word PCTN with REGNO enclosed in angle brackets. Using this statement, PROC TABULATE will use the total number of people in the survey as the denominator when calculating percents.

The following code includes the statements described above. In addition it incorporates some of the options discussed earlier in the paper including:

1. Defining a default format in the PROC step. (Note: A format of 5. is defined separately for N)

2. Substituting text for ALL and PCTN in the TABLE statement and substituting text for N in the KEYLABEL statement.

3. Using the RTS option to enlarge the first column.

```
PROC TABULATE FORMAT=9.2;
  TABLE ALL='Total Number of
Response Forms'*N=' '*F=5.
(MORNING AFTNOON EVENING
WKEND)*(N*F=5.
PCTN<REGNO>='% of
Respondents'),ALL='Total
```

```
Responses' /RTS=38 BOX='Time
Period Preferred for Meeting';
KEYLABEL N='Count';
```

This generates the following table.

Summary:

PROC TABULATE is a descriptive procedure used for creating tables of data. One, two, or three dimensional tables can be generated. In generating these tables, we can use options in PROC TABULATE that allow us to modify the appearance of the table, the text appearing in the table, and even how percents are calculated.

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

Other brand and product names are registered trademarks or trademarks of their respective companies.

Catherine M. Cerone
Cerone Consulting
6322 Churchview Lane
West Chester, OH 45069
(513) 777-2279