PharmaSUG 2016 - Paper TT14

Setting the Percentage in PROC TABULATE

David Franklin, Quintiles Real Late Phase Research, Cambridge, MA

ABSTRACT

PROC TABULATE is a very powerful procedure which can do statistics and frequency counts very efficiently, but it also it has the capability of calculating percentages on many levels for a category. This paper looks at the automatic percentage calculations that are provided, and then delves into how a user can specify the denominator for your custom percentage.

INTRODUCTION

PROC TABULATE tends to be one of the procedures that many programmers forget about, too often overlooked by a series of PROC FREQs, PROC MEANs and finally PROC REPORT. The one feature of PROC TABULATE is the ability to do both frequency and statistical calculations on the data, and report in a tabular form, a bit clunky at times, but is still an option that every SAS® programmer should have under their belt.

This paper takes a quick look at the basics of the PROC TABULATE then quickly introduces percentages, taking a look at the “automatics”, then setting custom percentage calculations, then finish off with a quick look at a trick where a “Missing” category may be present, but the percentage should only be on non-missing categories.

It must be noted that the examples in this paper use the ODS LISTING output destination, but could be easily converted into nice looking output using ODS PDF, ODS RTF or ODS HTML output destinations that allow for substantial formatting and use of proportional fonts.

THE BASICS

One of the advantages of using PROC TABULATE is that it will do frequency and statistical analysis, as well a produce a readable output. The minimum syntax needed to produce a table is

```
PROC TABULATE DATA=dataset;
  CLASS categorical_variable(s);
  VAR numeric_variable(s);
  TABLES <<<page dimension,>row dimension,>column dimension>;
RUN;
;
```

When looking at using PROC TABULATE for a table, take a look at first the also known as the classification variables -- these are the ones that are going to be placed in the CLASS statement. Second, decide on any analysis variables from which statistics, not frequency counts, will be produced -- these are stated in the VAR statement. The last step is defining the layout in the TABLES statement which defines where the CLASS and VAR variables go, what statistics are to be calculated on VAR variables, and finally formatting and labels.

The best way to see show the CLASS, VAR and TABLES statements come together is see an example.

```
PROC TABULATE DATA=sashelp.cars FORMCHAR="|----|+|---" F=6.;
CLASS make type;
TABLE make,type;
WHERE origin='USA' AND type IN('SUV','Sedan');
RUN;
;
```

In the example at line one is the PROC TABULATE statement with a FORMCHAR option that sets the characters for the borders of the output, and a F= option that sets the default output column format for the result columns.

The second line lists the variables for the categorical analysis, in this case MAKE and TYPE.

The third line is the TABLE statement which sets up the dimensions to the table as well as calls for statistics, formatting and labels -- in the example we are just setting up a simple cross tabulation count with the variable MAKE as the ROW dimension, and the variable MAKE as the column dimension.

The dataset contains numerous other information but is being restricted by the WHERE statement selecting on two types of cars coming from the USA.

The output is given below:

	Type	
	SUV	Sedan
	N	N
Make		
Buick	2	7
Cadillac	2	4
Chevrolet	4	15
Chrysler	.	13
Dodge	1	8
Ford	4	11
GMC	3	1
Hummer	1	.
Jeep	3	.
Lincoln	2	7
Mercury	1	7
Oldsmobile	.	3
Pontiac	1	8
Saturn	1	6

As stated previously, this is just a simple frequency count; we are going to step this up a notch by looking at some automatically generated percentages.

THE ‘AUTOMATICS’

SAS by default will calculate a number of percentages automatically. These are:

- REPPCTN and REPPCTSUM -- percentage of the value in a single table cell in relation to the total of the values in the report.
- COLPCTN and COLPCTSUM -- percentage of the value in a single table cell in relation to the total of the values in the column.
- ROWPCTN and ROWPCTSUM -- percentage of the value in a single table cell in relation to the total of the values in the row.
- PAGEPCTN and PAGEPCTSUM -- percentage of the value in a single table cell in relation to the total of the values in the page.

In each case the ‘PCTN’ variables are in relation to the proportion of total counts, while the ‘PCTSUM’ variables relate to the proportion of the total sum.

Our next example will take the previous example and add a percentage based on each row total, and use an ALL option in the TABLE statement to add the type column frequency results together, not outputting the percentage in the ALL group (which will add up to 100% so is redundant in this case).

```
PROC TABULATE DATA=sashelp.cars FORMCHAR="|----|+|---" F=6.;
  CLASS make type;
  TABLE make, (type*(n ROWPCTN='%'*F=6.1)) ALL;
  WHERE origin='USA' AND type IN('SUV','Sedan');
RUN;
```

In the row dimension, things remain the same. However in the column dimension things that changed -- the TYPE variable is now broken into counts (n) and row percentages (ROWPCTN), and for the percentage we are setting the column label to read '%' and the default format is 6.1. The output will look similar to that given below:

	Type				
	SUV		Sedan		All
	N	%	N	%	N
Make					
Buick	2	22.2	7	77.8	9
Cadillac	2	33.3	4	66.7	6
Chevrolet	4	21.1	15	78.9	19
Chrysler	.	.	13	100.0	13
Dodge	1	11.1	8	88.9	9
Ford	4	26.7	11	73.3	15
GMC	3	75.0	1	25.0	4
Hummer	1	100.0	.	.	1
Jeep	3	100.0	.	.	3
Lincoln	2	22.2	7	77.8	9
Mercury	1	12.5	7	87.5	8
Oldsmobile	.	.	3	100.0	3
Pontiac	1	11.1	8	88.9	9
Saturn	1	14.3	6	85.7	7

The next example uses the COLPCTN variable by setting the percentage by the column total. Lets look at the code first, but this time the CLASS dataset from the SASHELP directory is being used:

```
DATA _class;
  SET sashelp.class;
  agegrp=age;
  LABEL agegrp='Age Group';
RUN;
PROC TABULATE DATA=_class FORMHAR="|----|+|---" F=6.;
  CLASS sex agegrp;
  VAR age;
  TABLES (age*(N MEAN*F=6.1 MIN MAX))
         agegrp*(N COLPCTN='%'*F=6.1)
        ,sex /RTS=30;
RUN;
;
```

The datastep just takes the value of AGE and copies it to a new variable AGEGRP, the reason being that in this example a basic statistical analysis is done on age along with a frequency analysis, by SEX, in the singe PROC TABULATE call -- this is often call 'stacking'.

In the CLASS statement the variables SEX and AGEGRP are being treated as frequency class variables, while AGE is being set as numerical type variable.

The first line of the TABLES statement sets up an analysis in the row dimension for an AGE analysis computing the N, MEAN, MIN and MAX values, setting a format for the MEAN of 6.1 (the default for the table is 6.0). The second line of the TABLE statement sets up the frequency analysis calculating the percentage based on the column total,

with a format of 6.1. The variable SEX is used as the column dimension, and the RTS option sets aside a width of 30 characters to the field that will contain the labels.

Output from the code is shown below:

		Sex	
		F	M
Age	N	9	10
	Mean	13.2	13.4
	Min	11	11
	Max	15	16
Age Group			
11	N	1	1
	%	11.1	10.0
12	N	2	3
	%	22.2	30.0
13	N	2	1
	%	22.2	10.0
14	N	2	2
	%	22.2	20.0
15	N	2	2
	%	22.2	20.0
16	N	.	1
	%	.	10.0

One problem with the example above is that the percentage is based on all categories, meaning that if a 'Missing' category exists in your output table then it is going to be used in the denominator calculation. Do not fret, there is a way to 'trick' PROC TABULATE into not including the Missing category, but we are going to have to create our own new dataset:

```
DATA _class;
  LENGTH subnum sex 8;
  INFILE CARDS;
  INPUT subnum trt sex age @@;
  sexk=N(sex);  *Set flag, 1=sex value present;
  LABEL trt='Treatment' sex='Gender' age='Age';
CARDS;
1 2 1 25 2 1 2 28 3 1 1 46 4 2 1 24 5 1 1 63
6 2 1 47 7 1 2 56 8 2 1 23 9 2 1 43 10 1 . 55
11 2 1 53 12 2 1 25 13 2 1 50 14 2 2 44 15 2 1 48
16 2 1 64 17 2 1 59 18 1 1 59 19 1 2 47 20 2 1 65
;
RUN;
```

Now we have the data, notice that at the end of the second line, the record has a TRT value, no SEX value, and an AGE value. There also the line

sexy=n(sex);

which sets a flag 1 if the value fox SEX is present, else it will set to 0. This is used in the calculation of the percentage which will be seen shortly.

The next step is set our format, as shown below:

```
proc format;
  value trtf 1='Arm A' 2='Arm B';
  value sexf .='Missing' 1='Male' 2='Female';
run;
```

These set up decoded values for the coded variables -- note that for the format SEXF a missing values is set to MISSING. Next is the PROC TABULATE call, as shown below:

```
PROC TABULATE DATA=_class MISSING FORMCHAR="|----|+|---" F=6.;
  CLASS trt sex;
  VAR age sexk;
  TABLES age*(N MEAN*f=6.1 MIN MAX)
         sex*sexk='Group'*(N COLPCTSUM='%'*F=6.1)
         ,trt ALL/RTS=30;
  FORMAT sex sexf. trt trtf.;
RUN;
```

The SEXK variable, as we defined above, is used as a numeric, and with the COLPCTSUM in the TABLES statement, only does the percentage calculation based on the sum of the SEX frequencies where SEXK is equal to 1, i.e. Male or Female. See the output below for the result of the PROC TABULATE call:

			Treatment		
			Arm A	Arm B	All
Age		N	7	13	20
		Mean	50.6	43.8	46.2
		Min	28	23	23
		Max	63	65	65
Gender					
Missing	Group	N	1	.	1
		%	0.0	.	0.0
Male	Group	N	3	12	15
		%	50.0	92.3	78.9
Female	Group	N	3	1	4
		%	50.0	7.7	21.1

As a check, for Treatment Arm A, three males out of six males and females is 3*100/6=50%.

Setting Your Own Denominator

Above we were using the ‘automatics’, but it is possible to set your own denominator for the percentage. There are two formats for this approach:

- PCTN<*var*> -- Frequency counts, *var* being the variable to use as the denominator
- PCTSUM<*var*> -- Sums of data, *var* being the variable to use as the denominator

This approach to the denominator is very useful if you have stacked tables as often the value to be used is not the column total or row total. The following example shows the case, going back to the CARS dataset that was earlier, where the denominator is the MAKE by TYPE:

```
PROC TABULATE DATA=sashelp.cars FORMCHAR="|----|+|---" F=6.;
  CLASS make type;
  TABLE make, (type*(n PCTN<make>='%'*F=6.1)) ALL;
  WHERE origin='USA' AND type IN('SUV','Sedan');
RUN;
```

Note that the denominator in this case is the total cars in the MAKE category by type, thus the percentage total for the SUV type will be 100% as is the total for the Sedan type. In a stacked TABLES statement call, there may be different denominators for each row or column group. See below for the output from the PROC TABULATE call.

	Type				
	SUV		Sedan		All
	N	%	N	%	N
Make					
Buick	2	8.0	7	7.8	9
Cadillac	2	8.0	4	4.4	6
Chevrolet	4	16.0	15	16.7	19
Chrysler	.	.	13	14.4	13
Dodge	1	4.0	8	8.9	9
Ford	4	16.0	11	12.2	15
GMC	3	12.0	1	1.1	4
Hummer	1	4.0	.	.	1
Jeep	3	12.0	.	.	3
Lincoln	2	8.0	7	7.8	9
Mercury	1	4.0	7	7.8	8
Oldsmobile	.	.	3	3.3	3
Pontiac	1	4.0	8	8.9	9
Saturn	1	4.0	6	6.7	7

CONCLUSION

Adding a percentage to the PROC TABULATE call is not as daunting as some would make the SAS programmer below. The use of the so called ‘automatics’ assists in getting that percentage out to the report correctly, and even will allow for Missing categories to be counted without being used in the percentage. Also touched on was the ability to set your own denominator for the calculation of the percentage.

CONTACT INFORMATION

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

Name: David Franklin
Enterprise: Quintiles Real Late Phase Research
E-mail: David.Franklin@Quintiles.com