Summary Statistics in SAS II

Statistics 135

Autumn 2005

Copyright ©2005 by Mark E. Irwin

PROC TABULATE

The TABULATE procedure displays descriptive statistics in tabular format, using some or all of the variables in a data set. You can create a variety of tables ranging from simple to highly customized.

PROC TABULATE computes many of the same statistics that are computed by other descriptive statistical procedures such as MEANS, FREQ, and REPORT. PROC TABULATE provides

- simple but powerful methods to create tabular reports
- flexibility in classifying the values of variables and establishing hierarchical relationships between the variables
- mechanisms for labeling and formatting variables and procedure-generated statistics.

```
PROC TABULATE <option(s)>;
  BY <DESCENDING> variable-1 <...<DESCENDING> variable-n>
     <NOTSORTED>;
  CLASS variable(s) </ options>;
  CLASSLEV variable(s) / STYLE=<style-element-name | PARENT>
     <[style-attribute-specification(s)] >;
  FREQ variable;
  KEYLABEL keyword-1='description-1'
     <...keyword-n='description-n'>;
  KEYWORD keyword(s) / STYLE=<style-element-name | PARENT>
     <[style-attribute-specification(s)] >;
  TABLE <<page-expression,> row-expression,>
          column-expression</ table-option(s)>;
  VAR analysis-variable(s)</ options>;
  WEIGHT variable;
```

Many of the options mentioned earlier are valid here, such as `DATA=`, `OUT=`, `VARDEF=`, and `ALPHA=`. Another useful one is

- `FORMAT=format-name`: Specifies a default format for the value in each table cell. You can use any SAS or user-defined format. The default is `BEST12.2`, which uses a maximum of 12 spaces for each field with 2 digits after the decimal point.

The important statements as part of `PROC TABULATE` are

- `CLASS`: specifies one or more variables that the procedure uses to group the data. Variables in a CLASS statement are referred to as class variables. Class variables can be numeric or character. Class variables can have continuous values, but they typically have a few discrete values that define the classifications of the variable. You do not have to sort the data by class variables.

- `VAR analysis-variable(s) </ option(s)>`: The analysis variables.

- TABLE <<page-expression,> row-expression,> column-expression </ table-option(s)>: Defines how to cross-classify the variables and the statistics to calculate. The default statistic for analysis variables is SUM. Otherwise the default is N. All statistics that are valid with PROC MEANS are valid here, plus a few more, mainly dealing with row/column sums and percentages.

  Any variable mentioned in the TABLE statement must be mentioned in either the CLASS or VAR statement.

  When constructing the page-expression, row-expression, and column-expression, more than one variable can be used (as in the example). Variables within the same level are separated by *. * is also used the indicate which statistics are to be calculated for the analysis variable.

```
PROC FORMAT;
  VALUE $trainfmt 'TRUE' = 'Training'
                  'FALSE' = 'Test';
  VALUE potfmt 1 = 'Low'
               2 = 'Moderate'
               3 = 'High';

PROC TABULATE DATA = shingles2;
  CLASS training pot;
  VAR sales;
  TABLE training = 'Sample Type' * pot = 'Sales Potential' ,
        sales = 'Recent Sales' * (N MEAN STDDEV)
        / RTS=25 NOCONTINUED;
  FORMAT training $trainfmt. pot potfmt.;
  TITLE 'Sales Data by Region Potential and Training/Test Status';
```

Sales Data by Region Potential and Training/Test Status

		Recent Sales		
		N	Mean	StdDev
Sample Type	Sales Potential			
Testing	Low	4.00	154.98	77.22
	Moderate	15.00	178.25	74.06
	High	4.00	258.33	30.17
Training	Low	5.00	117.82	38.21
	Moderate	16.00	169.96	89.75
	High	5.00	223.38	77.77

PROC FREQ

Related to PROC TABULATE is PROC FREQ. It is used for creating and displaying and preforming some analysis on contingency tables (cross-classified categorical data). For display purposes, both can be useful, but if you need to look at standard tests on 2-way tables or association measures (on 2x2 tables), PROC FREQ is the way to go.

```
PROC FREQ < options > ;
  BY variables ;
  EXACT statistic-options < / computation-options > ;
  OUTPUT < OUT=SAS-data-set > options ;
  TABLES requests < / options > ;
  TEST options ;
  WEIGHT variable < / option > ;
```

WEIGHT here works like FREQ in other PROCs. The import option is TABLES which says which variables should be included in a table. The option EXACT requests exact tests and TEST requests asymptotic versions of tests.

For an example, lets look at 1973 Berkeley Admissions' data which looks at the number of admissions by gender for the 6 largest graduate programs at Berkeley that year.

Unfortunately for confidentiality reasons, the names of the majors cannot be revealed.

	Men		Women	
Major	Rejected	Admitted	Rejected	Admitted
A	314	511	19	89
B	207	353	8	17
C	205	120	391	202
D	279	138	244	131
E	137	54	299	94
F	351	22	317	24

Using PROC FREQ we can create tables looking at different combinations of the variables. For example, to look at the three marginal distributions, we can use

```
PROC FREQ DATA = berkeley;
  WEIGHT count;
      /* count contains the number of people in
         each combination */
  TABLES major gender status / NOCUM;
  FORMAT gender genderfmt. status statusfmt.;
```

The FREQ Procedure

Major Applied to

major	Frequency	Percent
A	933	20.61
B	585	12.93
C	918	20.28
D	792	17.50
E	584	12.90
F	714	15.78

Gender

gender	Frequency	Percent
Men	2691	59.46
Women	1835	40.54

```
        Admittance Status

  status    Frequency     Percent
--------------------------------
Rejected        2771       61.22
Admitted        1755       38.78
```

Where the procedure is more interesting is looking at cross-classifications like

```
PROC FREQ DATA = berkeley;
  WEIGHT count;
  TABLES gender*status / CHISQ;
```

The CHISQ option will print out the standard tests for 2x2 tables. The resulting output is

Table of gender by status

gender(Gender) status(Admittance Status)

Frequency Percent Row Pct Col Pct	Rejected	Admitted	Total
Men	1493 32.99 55.48 53.88	1198 26.47 44.52 68.26	2691 59.46
Women	1278 28.24 69.65 46.12	557 12.31 30.35 31.74	1835 40.54
Total	2771 61.22	1755 38.78	4526 100.00

Statistics for Table of gender by status

Statistic	DF	Value	Prob
Chi-Square	1	92.2053	<.0001
Likelihood Ratio Chi-Square	1	93.4494	<.0001
Continuity Adj. Chi-Square	1	91.6096	<.0001
Mantel-Haenszel Chi-Square	1	92.1849	<.0001
Phi Coefficient		-0.1427	
Contingency Coefficient		0.1413	
Cramer's V		-0.1427	

Fisher's Exact Test

Cell (1,1) Frequency (F)	1493
Left-sided Pr <= F	2.854E-22
Right-sided Pr >= F	1.0000
Table Probability (P)	1.314E-22
Two-sided Pr <= P	4.836E-22

Sample Size = 4526

It appears that men were more likely than men were more likely to get into these programs in 1973 than women. Let look at what happens within each program by

```
PROC FREQ DATA = berkeley;
  WEIGHT count;
  TABLES major * gender * status / NOCOL NOPERCENT CHISQ;
```

This creates a two-way table (`gender` by `status`) for each program. The options `NOCOL`, `NOROW`, and `NOPERECENT` say don't print the column, row, and overall percentages. Here we want the row percentages so the option was omitted.

Controlling for major=A

gender(Gender) status(Admittance Status)

Frequency Row Pct	Rejected	Admitted	Total
Men	314 38.06	511 61.94	825
Women	19 17.59	89 82.41	108
Total	333 35.69	600 64.31	933 100.00

```
Controlling for major=B

gender(Gender)     status(Admittance Status)

Frequency|
Row Pct  |Rejected|Admitted|  Total
---------------------------------
Men      |    207 |    353 |    560
         |  36.96 |  63.04 |
---------------------------------
Women    |      8 |     17 |     25
         |  32.00 |  68.00 |
---------------------------------
Total          215      370      585
            36.75    63.25   100.00
```

The other tables look similar. When asking for CHISQ statistics, they are done for each 2x2 table when 3 or more variables are examined. For the 6 tables, they are

Major	X^2	P-value
A	17.4307	0.000
B	0.2537	0.614
C	0.7535	0.385
D	0.2980	0.585
E	1.2877	0.256
F	0.3841	0.535

The only case that there is a significant Pearson X^2 statistic is major A, and in that case women seem to be doing better.

Lets look at the other two way tables

```
PROC FREQ DATA = berkeley;
  WEIGHT count;
  TABLES major*gender / NOROW NOPERCENT;
  TABLES major*status / NOCOL NOPERCENT;
```

Table of major by gender

major(Major Applied to)
gender(Gender)

Frequency| Col Pct	Men	Women	Total
A	825 30.66	108 5.89	933
B	560 20.81	25 1.36	585
C	325 12.08	593 32.32	918
D	417 15.50	375 20.44	792
E	191 7.10	393 21.42	584
F	373 13.86	341 18.58	714
Total	2691	1835	4526

Table of major by status

major(Major Applied to)
status(Admittance Status)

Frequency Row Pct	Rejected	Admitted	Total
A	333 35.69	600 64.31	933
B	215 36.75	370 63.25	585
C	596 64.92	322 35.08	918
D	523 66.04	269 33.96	792
E	436 74.66	148 25.34	584
F	668 93.56	46 6.44	714
Total	2771	1755	4526

As can be seen, the men's preferred majors were A and B, which were the easiest to get into. However for the women, they avoided these two programs, preferring C through F, which were the most difficult to get into.

This is an example of Simpson's paradox, where aggregating over third factor can switch the direction of association between two factors.

As mentioned earlier, in some cases tables created by `PROC TABULATE` are preferable to those from `PROC FREQ`. This usually occurs with 3 or higher way tables. To generate a the three way table this way, you can use the code

```
PROC TABULATE DATA = berkeley;
  FREQ count;
  CLASS major gender status;
  TABLE major , gender * status;
```

which gives

	Gender			
	Men		Women	
	Admittance Status		Admittance Status	
	Rejected	Admitted	Rejected	Admitted
	N	N	N	N
Major Applied to				
A	314.00	511.00	19.00	89.00
B	207.00	353.00	8.00	17.00
C	205.00	120.00	391.00	202.00
D	279.00	138.00	244.00	131.00
E	137.00	54.00	299.00	94.00
F	351.00	22.00	317.00	24.00

FORMAT

In many of the previous examples, the output was modified by a FORMAT statement. FORMAT can be used to modify how variables are displayed, read in, created, or stored. This can be particularly useful with data and time data, numbers representing money, etc. PROC FORMAT is used to create your own formats, which can be used in addition to the built in formats of **SAS**.

For example, earlier we had

```
PROC FORMAT;  /* create formats for table */
  VALUE $trainfmt 'TRUE' = 'Training'
                  'FALSE' = 'Test';
  VALUE potfmt 1 = 'Low'
               2 = 'Moderate'
               3 = 'High';
```

```
PROC TABULATE DATA = shingles2;
  CLASS training pot;
  VAR sales;
  TABLE training = 'Sample Type' * pot = 'Sales Potential' ,
        sales = 'Recent Sales' * (N MEAN STDDEV)
        / RTS=25 NOCONTINUED;
  FORMAT training $trainfmt. pot potfmt.;
```

anytime the value `TRUE` occurs in the variable `training` it will be displayed as `Training` and the value 1 occurs in the variable `pot` it will be displayed as `Low`. Note that this is how the values are displayed not stored. So in the case you could do something like `Low + Moderate` and get `High` as what you are really doing is `1 + 2 = 3`.

Another place where formats are useful are in `DATA` steps. Consider the example I've used before

```
IF potential > 15 then potentcat = 'High';
  ELSE IF potential < 6 then potentcat = 'Low';
  ELSE potentcat = 'Moderate';

IF _N_ < 27 THEN training = "TRUE";
  ELSE training = "FALSE";
```

If you printed out the variables potencat and training you would see something like

```
potentcat   training

  High        TRUE
  Mode        TRUE
  Mode        FALS
  High        FALS
  Low         FALS
  High        FALS
  Mode        FALS
```

As a default (at least with the setup in the labs), creating a text variable as above limits the number of characters to 4. By modifying the above code we can make the code work the way you think it should

```
FORMAT potentcat $8. training $5.;

IF potential > 15 then potentcat = 'High';
  ELSE IF potential < 6 then potentcat = 'Low';
  ELSE potentcat = 'Moderate';

IF _N_ < 27 THEN training = "TRUE";
  ELSE training = "FALSE";
```

This says that `potentcat` and `training` should be a character variables of length 8 and 5 respectively.

This type of code can be used to affect numeric variables as well.

Note that when referring to formats in a `FORMAT` statement, the name must end with a `'.'`.

In addition, FORMATs can be used to create a new variable. For example

```
PROC FORMAT;

  VALUE competefmt LOW-<10 = 'Low'
                   10-HIGH = 'High';

DATA shingles2;
  set shingles;

  FORMAT potentcat $8. training $5. compete $4.;

  compete = put(brands,competefmt.);
```

When FORMAT is used in a DATA step, it follows the variable through the analysis. However it doesn't have to be done this way. Instead it can be used only in PROCs of interest, as was done in the TABULAR example.

The structure of the PROC is

```
PROC FORMAT <option(s)>;
  EXCLUDE entry(s);
  INVALUE <$>name <(informat-option(s))> value-range-set(s);
  PICTURE name <(format-option(s))> value-range-set-1
              <(picture-1-option(s) )>
              <...value-range-set-n <(picture-n-option(s))>>;
  SELECT entry(s);
  VALUE <$>name <(format-option(s))> value-range-set(s);
```

The most important option is VALUE which describes the coding of the FORMAT. If the FORMAT is for a text variable, the name must start with $. The code OTHER is used to refer to values not included elsewhere in the definition.

```
PROC FORMAT;  /* create formats for table */
  VALUE $trainfmt 'TRUE' = 'Training'
                  'FALSE' = 'Test';

  VALUE potfmt 1 = 'Low'
               2 = 'Moderate'
               3 = 'High'
               . = 'Missing'
               OTHER = 'Miscoded';

  VALUE  competefmt LOW-<10 = 'Low'
                    10-HIGH = 'High';
```