The freq procedure will produce descriptive statistics, i.e., n-way frequency and crosstabulation tables, as well as perform inferential statistical analyses of the relationships among variables. Version 6.06 statements and options needed to produce descriptive statistics for printed output or output to a SAS® data set are emphasized. No instruction in the use of proc freq for inferential statistical analyses is provided.

This paper is geared for the beginning to intermediate level SAS user. This will be "streamlined" instruction; not everything will be covered, but certainly enough to get started. The tutorial begins with a basic explanation of the proc freq and tables statements. An explanation is given of the descriptive statistics output by the procedure. The handling of missing values, missing frequencies, and missing combinations with the missing, misprint, rare, and list options is presented. The use of other procedures (like sort and format) in conjunction with freq will be discussed. Examples of outputting to a data set using freq and manipulation of the variable values and counts within a data step for customized tables (data_null) are presented. Comparisons are made of the use of the by statement to the use of the * and () operators in the tables statement in the generation of multiple crosstabulation tables.

Some recommendations will be made for avoiding errors that could be made with proc freq. All instruction will be accompanied by examples of SAS code and output. For those who are interested in "learning by doing," this paper contains a homework assignment and an approach to solving the assignment.

Introduction
What can the freq procedure do? It can:

* produce descriptive statistics in the form of frequency counts and percentages;
* produce a data set of the counts and overall percentages. This data set may be merged with other data or reformatted into a customized table;
* perform inferential statistical analyses such as a Fisher's Exact Test or a Cochran-Mantel-Haenszel Test.

Some of the applications of the freq procedure are:

* to determine all the values that a variable can take on within a given data set, i.e., as an exploratory tool;
* for quick "n" dirty frequency table generation;
* to generate a data set to be used as input for customized table generation;
* to double-check descriptive statistics obtained by other methods;
* for particular inferential statistics available only through this procedure.

The freq procedure is a rather easy procedure to learn to use; that's an advantage it has over some of the more powerful procedures like tabulate, summary, and means. But there are certainly times when it is much more appropriate to use a proc tabulate rather than a proc freq, as will be discussed later on.

Data Set Used in Examples
There are many examples of SAS code and output contained in this paper, all of which were run on a hypothetical demographics data set from a clinical trial. The name of the data set is sug17dem, and here's the output from a proc contents run on this data set:

```
First Data Page:
---Alphabetic List of Variables and Attributes-----

# Variable Type Len Pos Label
9 ASH Num 6 58 PATIENT NUMBER
6 PATIENT Num 6 5 PATIENT NUMBER
6 PRT Chair 10 36 NUMBER OF PACKS SMOKED
6 RACE Chair 9 29 RACE
3 SEX Num 2 12 SEX
6 SMOKY Pos 6 28 PATIENT CURRENTLY SMOKES N/Y
6 TREATM Chair 10 54 TREATMENT SEQUENCE
7 TRS Chair 10 44 NUMBER OF PACKS SMOKED
```

Here's a proc print listing of the data set sug17dem:

```
Obs Patient Treatment Group Age Sex Verification Reason Patient Number
1 102 B 69 0 1 0
2 102 A 53 0 1 0
3 102 C 66 0 1 0
4 106 B 73 0 4 1
5 106 C 66 0 4 1
6 107 A 64 1 1 0
7 108 A 69 0 1 0
8 110 A 65 0 1 1
9 111 A 60 0 1 1
10 112 C 66 0 1 1
11 113 C 74 0 1 1
12 203 C 66 1 1 1
13 204 D 71 1 1 1
14 205 A 61 0 1 1
15 205 B 61 1 1 1
16 205 C 61 1 1 1
17 206 A 61 1 1 1
18 206 B 61 1 1 1
19 206 C 61 1 1 1
20 207 B 61 1 1 1
21 207 C 61 1 1 1
22 208 A 61 1 1 1
23 208 B 61 1 1 1
24 208 C 61 1 1 1
25 210 C 61 1 1 1
26 211 D 61 1 1 1
27 214 A 61 1 1 1
28 215 D 61 1 1 1
29 216 A 61 1 1 1
30 216 B 61 1 1 1
31 216 C 61 1 1 1
32 217 A 61 1 1 1
33 217 B 61 1 1 1
34 217 C 61 1 1 1
35 218 A 61 1 1 1
36 218 B 61 1 1 1
37 218 C 61 1 1 1
38 219 A 61 1 1 1
39 219 B 61 1 1 1
40 219 C 61 1 1 1
```

251
It is a very good idea to keep a copy of the proc contents and a listing of the data (or a listing of just a subset of the data) handy for reference while programming is performed. Note that there are some missing values for age and sex in this data set, and that sex, termcode, and smkny are coded variables that will need to be decoded using a proc format. The data set has 40 observations.

freq Statement
The question "How many males and how many females are there in the study?" is posed. The SAS code for a simple proc freq to answer this question might look like this:

```
proc freq;
  data = wendy.sug17dem;
  tables sex;
```

And the output from this procedure is:

```
SEX Frequency Percent
0 31 81.58
1 9 21.42
Frequency Missing = 2
```

```
SEX (SEX) TREAT (TREATMENT SEQUENCE)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>21.42</td>
<td>0.00</td>
<td>5.26</td>
<td>10.18</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>88.58</td>
<td>100.00</td>
<td>94.74</td>
<td>89.82</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td>81.58</td>
<td>0.00</td>
<td>94.74</td>
<td>89.82</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td>21.42</td>
<td>0.00</td>
<td>5.26</td>
<td>10.18</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>88.58</td>
<td>100.00</td>
<td>94.74</td>
<td>89.82</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td>81.58</td>
<td>0.00</td>
<td>94.74</td>
<td>89.82</td>
<td>100.00</td>
</tr>
</tbody>
</table>
```

'.' Operator
To answer the question, "How many males and how many females are there in each treatment group?" another variable, treat, needs to be added to the tables statement:

```
proc freq;
  data = wendy.sug17dem;
  tables sex * treat;
```

The '.' operator separates the two variables that are to be crosstabulated. This procedure will generate a crosstabulation table between sex and treat:

```
MISSING Option
In the last sample output, how can it be determined whether the 2 missing observations are 2 observations of sex=., 2 observations of treat=, or one missing observation for each variable? This information can be displayed by using either the missing option or the missprint option. As many options as are desired may follow the '.' at the end of the tables statement.

proc freq;
  data = wendy.sug17dem;
  tables sex * treat / missing;
  title 'MISSING Option';
Note that the overall total number of observations is now 40 and the "Frequency Missing" line is no longer printed. There are 2 observations where sex=., one with treat=A, and the other with treat=C. These 2 observations are included in all the statistics calculated for the table.

To generate a table that displays where the missing observations occur but does not include the missing observations in the statistical calculations, the missprint option may be used.

### proc freq

```sas
proc freq data = wendy.sug17dem;
tables sex * treat /missprint;
label sex = 'Sex';
label treat = 'Treatment';
title; title7 'MISSPRINT Option';
title8 'TABLE OF SEX BY TREAT';
title9; title10 '(Default Format Directly from PROC FREQ)';
```

### proc format

```sas
proc format;
value sexfmt
.= 'Unknown'
0 = 'Male'
1 = 'Female';
```

### proc freq

```sas
proc freq data = wendy.sug17dem;
tables sex * treat /list;
title; title7 'LIST Option';
title8 'TABLE OF SEX BY TREAT';
title9; title10 '(Default Format Directly from PROC FREQ)';
```

It might save programming time for the programmer to have a written specification for the table format prior to beginning to program. It is not important whether someone else provides the specification, or whether the programmer sketches it out before he/she begins to write the code.

### list and sparse Options

The box format of output for a crosstabulation table from the freq procedure is not the only way to format the output. The data may be listed in a tabular format if the list option is used.

### proc freq

```sas
proc freq data = wendy.sug17dem;
tables sex * treat /list;
title; title7 'LIST Option';
title8 'TABLE OF SEX BY TREAT';
title9; title10 '(Default Format Directly from PROC FREQ)';
```

When the list option is used, the output has the same format as it would from a tables statement containing a single variable and omitting the list option. The row and column totals and percentages are not displayed when the list option is used. Note that there was not a record printed for sex=I and treat=B; that is because this
combination of values did not occur in the data set. The sparse option in combination with the list option may be used to get the procedure to list one record for each possible combination of variable values that are being crosstabulated.

```plaintext
proc freq data = wendy.sug17dem;
tables sex * treat /sparse list;
title 'SPARSE and LIST Options';

```

The sparse option in combination with the Ust option may be used to get the procedure to list one record for each possible combination of variable values that are being crosstabulated.

```plaintext
proc rreq data = wendy.sug17dem;
tables sex * treat /sparse list;
title 'SPARSE and LIST Options';

```

Note that a record has been added for sex=1 and treat=B, and that the frequency of occurrence of this combination is 0. If the sparse option is used by itself, then the box format and frequencies that are generated are exactly the same as when the sparse option is omitted.

```plaintext
proc freq data = wendy.sug17dem;
tables sex * treat /sparse;
title 'SPARSE Option';

```

The noprint option may be used to suppress the printing of the default output from the freq procedure. The out= option may be used to specify a new data set to which the counts and percentages may be output. The out= option may be used by itself, or in combination with the noprint and other options. If there are multiple crosstabulations performed within a single tables statement, only the statistics from the last crosstabulation will be reported to the output data set. If the statistics from the other crosstabulations need to be reported to a data set also, then additional tables statements, each with an out= option, should be added to the procedure.

```plaintext
Customized Table Generation
The objective of the following series of data and proc steps will be to generate a customized crosstabulation table. This is the specification for the table that is to be produced:
```

```plaintext
data fixitop(keep = sex count1-count4);
set demog;
```

The demog data set contains 12 observations, but referring to the table specification, the data need to be combined into 3 observations. The next data step combines all 4 of the observations where sex=. into one observation, and then does the same for sex=0 and sex=1. The array variables count1-count4 correspond to treatments A, B, C, and D.
/* INITIALIZATION OF counts IN first.sex IS UNNECESSARY SINCE WE USED THE sparse OPTION IN THE freq. */

```plaintext
select(treat);
  when('A') counts(1) = count;
  when('B') counts(2) = count;
  when('C') counts(3) = count;
  when('D') counts(4) = count;
end;

if last.sex;
  proc print
data = fixltup;
title 'Listing of Transposed DEMOG Data Set';
Listing of Transposed DEMOG Data Set
OBS SEX COUNT1 COUNT2 COUNT3 COUNT4
  1  A    1  0  0  0
  2  A    1  1  0  0
  3  B    0  2  1  0
  4  B    0  2  1  0
  5  C    0  2  1  0
  6  C    0  2  1  0
  7  D    0  2  1  0
  8  D    0  2  1  0
  9  Unknown 0  2  1  0
 10  Unknown 0  2  1  0
 11  Unknown 0  2  1  0
 12  Unknown 0  2  1  0
```

But the records above are not in the order that the table specification requires; the order should be "Female, Male, Unknown" in the table. There is an option, order=, that may be used with the proc freq statement to specify the order that the records are to be output. But order= will not work in this case because missing values will still come first, and our specification requires that "Unknown" (sex=.) come last. So a new variable for sex, call it newsex, must be created whose values will be sorted in the desired order. Starting from the beginning again with the permanent data set:

```plaintext
data addfmt(keep = newsex treat);
  set wendy.sug17dem;
  attrib newsex length=$7;
  newsex = put(sex,sexfmt.);
proc freq
data = addfmt;
  tables newsex * treat/sparse
  noprint
  out = demog;
```

Note that the missing option is no longer necessary because the variable newsex does not have any missing values. The missing values were translated to the character string "Unknown" using the put function and sexfmt.

```plaintext
proc print
data = addfmt;
title 'Listing of Transposed DEMOG Data Set (in Desired Order for Printing)';
Listing of Transposed DEMOG Data Set (in Desired Order for Printing)
OBS newsex COUNT1 COUNT2 COUNT3 COUNT4
 1  Female 1  0  0  0
 2  Male 1  1  0  0
 3  Unknown 0  2  1  0
 4  Unknown 0  2  1  0
 5  Unknown 0  2  1  0
 6  Unknown 0  2  1  0
 7  Unknown 0  2  1  0
 8  Unknown 0  2  1  0
 9  Unknown 0  2  1  0
 10  Unknown 0  2  1  0
 11  Unknown 0  2  1  0
 12  Unknown 0  2  1  0
```

At last the data are ready to be printed. _null_ is used to specify that no data set needs to be created by this data step; only a file to be printed will be created.

```plaintext
data _null_; set fixltup;
file print;
if _n_ = 1 then do;
  uline = repeat('-',53);
  put uline @43 'Treatment' /
    @21 uline;
  put @25 'A' @40 'B' @55 'C' @70 'D';
  put @21 '--------'
    @21 '--------'
    @21 '--------'
    @21 '--------';
  put @7 'Sex' /;
end;
```

The records are now in the desired order which happens to be that of an alphabetical sort. Some other order might have required hardcoding.

```plaintext
data fixltup(keep = newsex count1-count4);
set demog;
  by newsex;
  retain count1-count4;
  array counts(4) count1-count4;
select(treat);
  when('A') counts(1) = count;
  when('B') counts(2) = count;
  when('C') counts(3) = count;
  when('D') counts(4) = count;
end;
if last.newsex;
  proc print
data = fixltup;
title 'Listing of Transposed DEMOG Data Set (In Desired Order for Printing)';
Listing of Transposed DEMOG Data Set (In Desired Order for Printing)
OBS newsex COUNT1 COUNT2 COUNT3 COUNT4
 1  Female 1  0  0  0
 2  Male 1  1  0  0
 3  Unknown 0  2  1  0
 4  Unknown 0  2  1  0
 5  Unknown 0  2  1  0
 6  Unknown 0  2  1  0
 7  Unknown 0  2  1  0
 8  Unknown 0  2  1  0
 9  Unknown 0  2  1  0
 10  Unknown 0  2  1  0
 11  Unknown 0  2  1  0
 12  Unknown 0  2  1  0
```

At last the data are ready to be printed. _null_ is used to specify that no data set needs to be created by this data step; only a file to be printed will be created.

```plaintext
data _null_; set fixltup;
file print;
if _n_ = 1 then do;
  uline = repeat('-',53);
  put uline @43 'Treatment' /
    @21 uline;
  put @25 'A' @40 'B' @55 'C' @70 'D';
  put @21 '--------'
    @21 '--------'
    @21 '--------'
    @21 '--------';
  put @7 'Sex' /;
end;
```

255
Note that the variable `newsex` does not need a format because its values are the actual values to be printed. The final customized table looks like this:

### Patient Demographics by Treatment

**Number of Patients**

(Customized format - Data Set Output (OUT) from PROC FREQ and Reformatted)

- **Sex**
  - Female: 1
  - Male: 0
  - Unknown: 1

- **Treatment**
  - A: 3
  - B: 0
  - C: 2

- **Smokes**
  - No: 7
  - Yes: 7

---

More on the `tables` statement

The same descriptive statistics in the 2 tables above may be obtained using a different method. Multiple variables may be cross-tabulated within the `tables` statement using the `*` operator. In fact, the only limitation on the number of variables is the memory limitation of the computer being used. If many variables are going to be cross-tabulated all at once in the tables statement, it is strongly recommended that the `tabulate` procedure be used instead of the `freq` procedure. It is possible to reduce hundreds of pages of output from the `freq` procedure to a single table from proc `tabulate`.

### Using the BY Statement

(Proc `tabulate`)

<table>
<thead>
<tr>
<th>Sex</th>
<th>Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>Male</td>
<td>13.64</td>
</tr>
<tr>
<td></td>
<td>26.67</td>
</tr>
<tr>
<td>Female</td>
<td>4.55</td>
</tr>
<tr>
<td></td>
<td>25.00</td>
</tr>
<tr>
<td>Total</td>
<td>18.18</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Sex</th>
<th>Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>Male</td>
<td>31.25</td>
</tr>
<tr>
<td>Female</td>
<td>18.75</td>
</tr>
</tbody>
</table>

---

256
U6inq Additional Asterisks in the TABLE

TABLE 1 OF SEX BY TREAT
CONTROLLING FOR SMOKY-No

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Percent</th>
<th>Raw Pct</th>
<th>Col Pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>31.25</td>
<td>31.25</td>
<td>31.25</td>
</tr>
<tr>
<td></td>
<td>13.18</td>
<td>13.18</td>
<td>13.18</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Total</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
| Frequency Missing = 2

TABLE 2 OF SEX BY TREAT
CONTROLLING FOR SMOKY-Yes

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Percent</th>
<th>Raw Pct</th>
<th>Col Pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>25.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>16.67</td>
<td>28.51</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Total</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
| Frequency Missing = 2

The tables created using the by statement are nicer looking because the by variables can be labeled when the by statement is used. whereas the additional variables cannot be labeled when added to the tables statement as in the output above.

Parentheses may be used as a method of shorthand in the tables statement: tables smkny * sex * treat; is equivalent to the one in the example below:

proc freq
   data = wendy.sug17dem;
   tables smkny * (sex treat);
   format sex sexfmt. smkny smkfmt.;
   label sex = 'Sex';
   label treat = 'Treatment';
   label smkny = 'Does Patient Smoke?';
   title 'Using Parentheses in the TABLE Statement';
   title2 'Input Data Set Need Not Be SORTed';
   run;

Using Parentheses in the TABLE Statement (Input Data Set Need Not Be SORTed)

TABLE 1 OF SEX BY TREAT
CONTROLLING FOR SMOKY-No

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Percent</th>
<th>Raw Pct</th>
<th>Col Pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>25.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>16.67</td>
<td>28.51</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Total</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
| Frequency Missing = 2

But notice that the 2 tables that are generated above contain completely different statistics than when the following tables statement was used in a previous example:

Homework Assignment
True knowledge and ability to use the freq procedure will come with practice. Here's a homework assignment that may used for some practice with the freq procedure,

Produce a table that comes as close as possible to the following table specification using data, proc sort, proc format, and proc freq steps. (Hint: prior to using proc freq, a new variable will need to be created for age.) Try not to peek at the solution code until at least one draft table has been generated.

The following solution is just one approach to reaching the objective; there are many other ways these tables could have been generated.

data adagecat;
  set wendy.sug17dem;
  if age = 5 then agecat = 5;
  else if age < 65 then agecat = 1;
  else if age >= 65 then agecat = 2;
  else put 'PROB LEM - what is age? ' patient= age=;
The example above illustrates proc freq's weaknesses as much as it does its strengths. The final tables aren't very pretty, but the program was quick and easy to write and desired frequencies are presented. In the table for "Age=< 65 years", note that there is not a column for "Lost to Followup." This is because there were no observations in the data set for patients less than 65 years of age who were lost to followup in any of the 4 treatment groups. Even the sparse option can't help in this case; the freq procedure can't count things that don't exist.

To generate a nicer looking set of tables that more closely match the table specification, a data _null_ or another procedure would need to be used. It is recommended that sufficient knowledge about these other methods be gained so that the most appropriate method can be chosen for a particular task.

**Summary**

The primary advantages of the freq procedure over other similar procedures are that it is easy to learn and quick to use. An output data set can be created by the freq procedure for input to customized table generation. Particular inferential statistics are available only through the freq procedure. For exploratory tasks or some quickly generated frequency tables, the freq procedure is the way to go.

**Acknowledgements**

Thanks to the Analysis Programming Department at PRA for their suggestions for improvements to the paper and the presentation. Thanks also to Terry Herring and Amy Brown for their assistance with the manuscript and the slides.

**References**


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

**Author**

Wendy London
Supervisor of Analysis Programming
Pharmaceutical Research Associates, Inc.
1290 Seminole Trail
Charlottesville, VA 22901
phone: 804-978-7730
fax: 804-978-7084