%ABC_FREQ: A SAS® Macro Utility to Generate Enhanced Frequency Reports
Sharon Matsumoto Hirabayashi, Westat, Rockville, MD

ABSTRACT
One-way frequency tables are routinely run for data checking purposes. When all variables in a file need to be examined, the volume of output can prove to be daunting. This macro utility is designed to assist the end user by supplementing the standard SAS PROC FREQ output. For faster lookup, output tables are ordered alphabetically. Variables can be dropped or kept from the full set using any valid SAS list syntax (without embedded blanks). Tables exceeding a user-specified number of levels can be suppressed and replaced with missing/non-missing counts. Formatted values or ranges (when applicable) are displayed with unformatted values, and counts for all formatted values are presented (when applicable), whether zero or not.

INTRODUCTION
In a statistical survey research company such as ours, one-way frequencies are routinely run for quality control. With large survey instruments, the number of variables to check and the resulting output can become rather sizeable. This macro was developed to provide both a simple interface for the programmer to generate the frequencies and a more functional report for the end user. Since the macro was written under SAS v8.0, the programmer should be aware of the 32K length limit for macro variables and any memory limitations affecting the number of variables that can be run at one time.

A simple example here illustrates the difference between output resulting from a standard PROC FREQ and that produced by the macro.

example data set (unformatted):

<table>
<thead>
<tr>
<th>teacher</th>
<th>class</th>
<th>birdtype</th>
<th>rackets</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gary</td>
<td>301</td>
<td>F</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Daphne</td>
<td>201</td>
<td>F</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Joyce</td>
<td>102</td>
<td>?</td>
<td>2</td>
<td>.</td>
</tr>
<tr>
<td>Bob</td>
<td>202</td>
<td>F</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Jim</td>
<td>101</td>
<td>P</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

example data set (formatted):

<table>
<thead>
<tr>
<th>teacher</th>
<th>class</th>
<th>birdtype</th>
<th>rackets</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gary</td>
<td>301</td>
<td>Feather</td>
<td>many</td>
<td>Excellent</td>
</tr>
<tr>
<td>Daphne</td>
<td>Interm</td>
<td>Feather</td>
<td>some</td>
<td>Good</td>
</tr>
<tr>
<td>Joyce</td>
<td>Adv Beg</td>
<td>?</td>
<td>some</td>
<td>.</td>
</tr>
<tr>
<td>Bob</td>
<td>Advanced</td>
<td>Feather</td>
<td>many</td>
<td>Good</td>
</tr>
<tr>
<td>Jim</td>
<td>Beginner</td>
<td>Plastic</td>
<td>one</td>
<td>Fair</td>
</tr>
</tbody>
</table>

standard PROC FREQ output:

Cumulative Cumulative
<table>
<thead>
<tr>
<th>teacher</th>
<th>Frequency</th>
<th>Percent</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>Daphne</td>
<td>1</td>
<td>20.00</td>
<td>2</td>
<td>40.00</td>
</tr>
<tr>
<td>Gary</td>
<td>1</td>
<td>20.00</td>
<td>3</td>
<td>60.00</td>
</tr>
<tr>
<td>Jim</td>
<td>1</td>
<td>20.00</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td>Joyce</td>
<td>1</td>
<td>20.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

class number

Cumulative Cumulative
<table>
<thead>
<tr>
<th>class</th>
<th>Frequency</th>
<th>Percent</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beginner</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>Adv Beg</td>
<td>1</td>
<td>20.00</td>
<td>2</td>
<td>40.00</td>
</tr>
<tr>
<td>Intermed</td>
<td>1</td>
<td>20.00</td>
<td>3</td>
<td>60.00</td>
</tr>
<tr>
<td>Advanced</td>
<td>1</td>
<td>20.00</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td>301</td>
<td>1</td>
<td>20.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

preferred bird type

Cumulative Cumulative
<table>
<thead>
<tr>
<th>birdtype</th>
<th>Frequency</th>
<th>Percent</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>Feather</td>
<td>3</td>
<td>60.00</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td>Plastic</td>
<td>1</td>
<td>20.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

even rackets

Cumulative Cumulative
<table>
<thead>
<tr>
<th>rackets</th>
<th>Frequency</th>
<th>Percent</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>one</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>some</td>
<td>2</td>
<td>40.00</td>
<td>3</td>
<td>60.00</td>
</tr>
<tr>
<td>many</td>
<td>2</td>
<td>40.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

rating

Cumulative Cumulative
<table>
<thead>
<tr>
<th>rating</th>
<th>Frequency</th>
<th>Percent</th>
<th>Frequency</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>Excellent</td>
<td>1</td>
<td>20.00</td>
<td>2</td>
<td>40.00</td>
</tr>
<tr>
<td>Good</td>
<td>2</td>
<td>40.00</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td>Fair</td>
<td>1</td>
<td>20.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>
%abc_freq output (with default maxlevel):

--- var=BIRDTYPE [C]: preferred bird type -----

<table>
<thead>
<tr>
<th>fmtval</th>
<th>count</th>
<th>percent</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>F = Feather</td>
<td>3</td>
<td>60.00</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td>P = Plastic</td>
<td>1</td>
<td>20.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

-------- var=CLASS [N]: class number --------

<table>
<thead>
<tr>
<th>fmtval</th>
<th>count</th>
<th>percent</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>101 = Beginner</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>102 = Adv Beg</td>
<td>1</td>
<td>20.00</td>
<td>2</td>
<td>40.00</td>
</tr>
<tr>
<td>201 = Intermed</td>
<td>1</td>
<td>20.00</td>
<td>3</td>
<td>60.00</td>
</tr>
<tr>
<td>202 = Advanced</td>
<td>1</td>
<td>20.00</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td>301</td>
<td>1</td>
<td>20.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

-------- var=RACKETS [N]: extra rackets --------

<table>
<thead>
<tr>
<th>fmtval</th>
<th>count</th>
<th>percent</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>1 = one</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>2-5 = some</td>
<td>2</td>
<td>40.00</td>
<td>3</td>
<td>60.00</td>
</tr>
<tr>
<td>6-HIGH = many</td>
<td>2</td>
<td>40.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

-------- var=RATING [N] --------

<table>
<thead>
<tr>
<th>fmtval</th>
<th>count</th>
<th>percent</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>1 = Excellent</td>
<td>1</td>
<td>20.00</td>
<td>2</td>
<td>40.00</td>
</tr>
<tr>
<td>2 = Good</td>
<td>2</td>
<td>40.00</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td>3 = Fair</td>
<td>1</td>
<td>20.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

-------- var=TEACHER [C] --------

<table>
<thead>
<tr>
<th>fmtval</th>
<th>count</th>
<th>percent</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>Daphne</td>
<td>1</td>
<td>20.00</td>
<td>2</td>
<td>40.00</td>
</tr>
<tr>
<td>Gary</td>
<td>1</td>
<td>20.00</td>
<td>3</td>
<td>60.00</td>
</tr>
<tr>
<td>Jim</td>
<td>1</td>
<td>20.00</td>
<td>4</td>
<td>80.00</td>
</tr>
<tr>
<td>Joyce</td>
<td>1</td>
<td>20.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

%abc_freq output (with maxlevel = 2):

--------------- var=BIRDTYPE [C]*: preferred bird type ---------------

<table>
<thead>
<tr>
<th>fmtval</th>
<th>count</th>
<th>percent</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-missing</td>
<td>5</td>
<td>100.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

--------------- var=CLASS [N]*: class number ---------------

<table>
<thead>
<tr>
<th>fmtval</th>
<th>count</th>
<th>percent</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>101-301 = Beginner-301</td>
<td>5</td>
<td>100.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

--------------- var=RACKETS [N]*: extra rackets ---------------

<table>
<thead>
<tr>
<th>fmtval</th>
<th>count</th>
<th>percent</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-8 = one-many</td>
<td>5</td>
<td>100.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

--------------- var=RATING [N]* ---------------

<table>
<thead>
<tr>
<th>fmtval</th>
<th>count</th>
<th>percent</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing</td>
<td>1</td>
<td>20.00</td>
<td>1</td>
<td>20.00</td>
</tr>
<tr>
<td>1-3 = Excellent-Fair</td>
<td>4</td>
<td>80.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

--------------- var=TEACHER [C]* ---------------

<table>
<thead>
<tr>
<th>fmtval</th>
<th>count</th>
<th>percent</th>
<th>count</th>
<th>percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-missing</td>
<td>5</td>
<td>100.00</td>
<td>5</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Notice the differences in the output between the standard PROC FREQ and the macro. The macro orders the variables alphabetically and the variable type is added in brackets next to the variable name. When a full frequency has been suppressed and replaced with missing/non-missing counts, an asterisk appears after the variable type. Also, for numeric variables that have been suppressed, the range of the non-missing values is given. The macro displays counts and percents for both formatted values or ranges (when applicable) and their associated unformatted values or ranges. Zero counts and percents are also displayed for formatted values (when applicable) that do not occur in the data.

THE MACRO CALL

The %ABC_FREQ macro has six keyword parameters: lib, mem, droplist, keeplist, maxlevel, and fmtlib. The programmer references the input SAS data set on which the frequencies will be run with the lib and mem parameters. The lib parameter should be specified with the libref associated with the input SAS data set. If none is specified, the default WORK library is referenced. The mem parameter should be specified with the input SAS data set name. If it is not specified, the macro prints an error message, and the macro is exited.

Only one parameter among keeplist and droplist can be specified, otherwise an error message is issued and the macro is exited. If neither is specified, one-way frequencies for all variables in the input data set are
generated in alphabetic order (based on uppercase names). The droplist and keeplist parameters allow the specification of variable lists to be dropped or kept, respectively, using any valid SAS list syntax (but without embedded blanks) including numbered ranges, named ranges, named prefixes, and the special named lists _CHARACTER_ and _NUMERIC_ (but with the exception of _ALL_). The items in the lists must be separated by blanks. Coding keeplist or droplist will result in one-way frequencies generated only for the variables kept (or not dropped) in alphabetic order (based on uppercase names).

The fmtlib parameter is optional and specifies a format catalog libref to search besides WORK and LIBRARY. If the fmtlib parameter is specified, there must be a format catalog named FORMATS in the fmtlib libref. If the fmtlib parameter is not specified, a warning message is issued and the value of the lib parameter is assigned to fmtlib. The formats referenced by the fmtlib parameter should not contain formats named _Smmnmfmt or mnmfmt, since these are generated by the macro.

If the number of rows (or observations in the output data set of counts) returned from PROC FREQ exceeds the maxlevel parameter, the full frequency output is suppressed and replaced with missing/non-missing counts. If maxlevel is not specified, a default of 50 levels is assigned.

Following the previous example, if the data set is stored in mydata.badminton; the formats stored in flib.formats; variables beginning with the letter “r” are to be dropped from the output; and the maximum number of levels before suppressing is 3, then the macro call is written as:

```
%abc_freq(lib= mydata, mem= badminton, fmtlib= flib, droplist= r:, maxlevel= 3)
```

**THE INNER WORKINGS**

The macro works by first, generating a list of variables (all in uppercase) in alphabetic order from the data set of interest. That list can contain all the variables in the data set or some subset of the variables. Once the list has been created, PROC FREQ is run with the counts for each variable in the list output to a separate data set. The macro then determines whether or not large output should be suppressed and replaced with missing/non-missing counts (the cutoff for what qualifies as sufficiently large being user-defined). The output is modified to display all formatted values (whether actually occurring in the data or not) with their associated unformatted values or ranges. Finally, the modified output for each variable is appended with the others before printing.

**GENERATING THE ORDERED LIST OF VARIABLES**

By accessing the DICTIONARY.COLUMNS table through PROC SQL and using the SELECT NAME INTO : clause, all of the variables in the data set can be stored in a macro variable and then ordered alphabetically (based on uppercase names) with an ORDER BY statement. This is possible because the DICTIONARY.COLUMNS table (a read-only object available during the SAS session) contains information about variables and variable attributes (like type, label, format, length, and position); while the SELECT INTO : clause provides the means to create a macro variable within PROC SQL. Since NAME in the DICTIONARY.COLUMNS table corresponds to variable, SELECT NAME INTO :_alphabetical_ creates the macro variable &_alphabetical_, which is a string of all the variable names in the data set, the order determined by the ORDER BY UPCASE(NAME) statement.

If droplist or keeplist is specified, then the variables in the macro variable list can be selectively dropped or kept in the PROC SQL step with a WHERE statement. Because droplist and keeplist can be specified using any valid SAS list syntax (but without embedded blanks) including numbered ranges, named ranges, named prefixes, and the special named lists _CHARACTER_ and _NUMERIC_ (but with the exception of _ALL_), each list item is parsed within a loop, and adds to a WHERE condition that is strung to the other conditions with ORs. The WHERE conditions selectively add or remove variables from the ordered list by checking some combination of the variable’s name (in uppercase), type, and position in the data set. To “turn off” the ORs, the last condition in the WHERE statement is “1 NE 1” (always false). The coding for droplist and keeplist is the same, except that when droplist is specified, the entire WHERE condition is preceded by a NOT.

To generate the WHERE conditions that select variables based on their order in the data set (where VARNUM between...), the %SYSFUNC macro is used. %SYSFUNC calls DATA step functions to generate text, and in this case is used to generate the numeric values that VARNUM must be between (VARNUM being another variable in the DICTIONARY.COLUMNS table containing the variable order).

**SUPPRESSING LARGE OUTPUT AND REPLACING WITH MISSING/NON-MISSING COUNTS**

Once PROC FREQ has been run and the one-way counts and percents have been output to a data set, the number of records in that data set is compared to the maxlevel parameter. If the number of records exceeds maxlevel, another PROC FREQ is run, but formatting the counts as missing and non-missing. For numeric variables, missing values are considered those ranging from _ to .z. To save processing time, instead of re-running the PROC FREQ on the entire input data set, the data set of counts output from the first PROC FREQ is read, each record weighted by its COUNT.
For numeric variables that switch to the missing/non-missing format, it is helpful for the end user to see the range of the non-missing values. But, when frequencies are run on variables having formats, the output data set from the PROC FREQ contains the counts attached to the smallest internal value for the formatted level, so that neither the minimum nor the maximum value for the range is necessarily stored. In order to determine the range then, macro variables for the minimum and maximum values are created in a PROC SQL step, with a SELECT MIN, MAX INTO : on the input data set.

DISPLAYING ALL FORMATTED VALUES OR RANGES WITH THEIR ASSOCIATED UNFORMATTED VALUES OR RANGES
It is often useful to see all possible formatted values displayed whether or not a particular value actually occurs in the data. Since one-way frequencies produced by PROC FREQ do not display formatted values with counts of zero, the output control data set of all formatted values (created through PROC FORMAT and accessed through PROC SQL) is initialized with count and percent values of 0, and then is updated with the data set output from PROC FREQ.

For variables without stored formats: if the number of levels resulting from PROC FREQ does not exceed maxlevel, all unformatted values are displayed; if maxlevel is exceeded, the full frequency is suppressed and replaced with missing/non-missing counts, with the range of the unformatted non-missing values displayed for NUMERIC variables.

For variables with stored formats: if the number of levels resulting from PROC FREQ does not exceed maxlevel, then all formatted values are displayed (whether 0 counts or not) along with the associated unformatted values or ranges. If maxlevel is exceeded, the full frequency is suppressed and replaced with missing/non-missing counts, with the range of formatted non-missing values displayed with their associated unformatted ranges for NUMERIC variables.

OTHER OUTPUT MODIFICATIONS
As in the standard PROC FREQ output, variable labels are attached in the macro output when applicable. But unlike standard PROC FREQ output, the variable’s type is supplied in brackets next to the variable name. Again, the labels and types are determined using the %SYSFUNC macro. Also, variables that have had their full counts suppressed and replaced with missing/non-missing counts are flagged with asterisks.

Since the output data set from PROC FREQ only includes COUNT and PERCENT and not the cumulative counts and percents, those cumulatives need to be constructed before the output from all of the variables is appended. The appended output is then printed in PROC PRINT (by variable) to reduce the length of the output.

CONCLUSION
The %ABC_FREQ macro offers a simple interface for the programmer to generate an enhanced report when one-way frequencies need to be run. When those frequencies are requested for all variables in a data set, the programmer need only supply the libname and memname of the data set and the location of any stored formats. If not all variables need to be run, the programmer can use any valid SAS list syntax (without embedded blanks) to drop or keep variables.

The enhancements for the user include: ordering the variables alphabetically for a faster table lookup; including the variable type; suppressing and flagging large output and replacing it with missing/non-missing counts; displaying the range of non-missing values for suppressed numeric variables; displaying formatted values or ranges (whether actually occurring in the data or not) with unformatted values or ranges; and reducing the volume of output by appending each variable’s counts before printing.

REFERENCES

ACKNOWLEDGMENTS
The author wishes to thank Judith Kuhn, Ian Whitlock, and Shigeru Ochi for their comments and encouragement. And a special thank you to the contributors to the example data set.

TRADEMARKS
SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

CONTACT INFORMATION
Questions and comments are welcomed by the author:
Sharon Hirabayashi
Westat
1650 Research Boulevard
Rockville, MD 20850
(301)517-8093
SharonHirabayashi@WESTAT.COM
%macro abc_freq (lib= work, mem= , droplist= , keeplist= , maxlevel= 50, fmtlib= );

%let lib = %upcase(&lib);
%let mem = %upcase(&mem);
/* use upcase because droplist/keeplist can contain special characters */
%let droplist = %upcase(&droplist);
%let keeplist = %upcase(&keeplist);
%let fmtlib = %upcase(&fmtlib);

/* if &mem is blank, print an error message */
%if &mem = %then
  %do;
  %put ERROR:  YOU MUST SPECIFY MEM!!!;
  %goto exit;
  %end;

/* if &fmtlib is blank, have it point to &lib, and print a warning message */
%if &fmtlib = %then
  %do;
  %put WARNING:  YOU DID NOT SPECIFY FMTLIB, SO IT WILL POINT TO &LIB;
  %let fmtlib = &lib;
  %end;

/* if both &droplist and &keeplist are specified, print an error message */
%if &droplist ne & &keeplist ne %then
  %do;
  %put ERROR:  YOU MAY NOT SPECIFY BOTH DROPLIST AND KEEPLIST!!!;
  %goto exit;
  %end;
%if &droplist ne %then %let list = &droplist;
%else if &keeplist ne %then %let list = &keeplist;
%else if &droplist = & &keeplist = %then %let list = ;
%let dsid = %sysfunc(open(&lib..&mem));
%let numstr = 0123456789;
%let _alphabetic_ = ;

/************
 /* formats */
/************

/* missing/non-missing formats to replace *big* format output */
proc format cntlout= addedfmt;
  value $cmnmfmt
    ' ' = 'Missing    '
    other = 'Non-missing'
  ;
  value mnmfmt
    _-.z  = 'Missing    '
    other = 'Non-missing'
  ;
run;

/* bring in user's stored formats */
%if &fmtlib ne %then
  %do;
  options fmtsearch = (&fmtlib);
  proc format library= &fmtlib cntlout= infmt;
  run;
  %end;

/************************************************************************************/
/* create list of variables in alphabetic order in the macro variable &_alphabetic_ */
/************************************************************************************/
proc sql noprint;
  select upcase(name) into :_alphabetic_ separated by " "
  from dictionary.columns
  where
  ;
  /***********/
  /* loop through all items in &list */
  /***********/
  %let i = 1;
  /* use bquote because list can contain - */
  %let item = %bquote(%scan(&list, %eval(&i), %str( )));
%do %until(&item= );
  %if &i = 1 %then
    %do;
      %if &droplist ne %then not;
    %end;
    /********************************************************************************/
    /* check for _character_ (all character) in list */
    /*******************************************************************************/
    %if %index(&item, _CHARACTER_) > 0 %then trim(upcase(type)) = 'CHAR' or;
    /*******************************************************************************/
    /* check for _numeric_ (all numeric) in list */
    /*******************************************************************************/
    %else %if %index(&item, _NUMERIC_) > 0 %then trim(upcase(type)) = 'NUM' or;
    /*******************************************************************************/
    /* check for dropping/keeping by variable number and determine start and end */
    /*******************************************************************************/
    %else %if %index(&item, -CHARACTER-) > 0 or %index(&item, -NUMERIC-) > 0 or
      %index(&item,--) > 0 %then
      %do;
      /* determine first variable in list */
      %let var1 = %substr(&item, 1, %eval(%index(&item, -)-1));
      /* determine last variable in list */
      %if %index(&item, -CHARACTER-) > 0 %then %let incr = 11;
      %else %if %index(&item, -NUMERIC-) > 0 %then %let incr = 9;
      %else %if %index(&item, --) > 0 %then %let incr = 2;
      %let var2 = %substr(&item, %eval(%index(&item, -)+%eval(&incr)));
      /* determine variable numbers of first and last variables in list */
      %let start = %sysfunc(varnum(&dsid, &var1));
      %let end = %sysfunc(varnum(&dsid, &var2));
    /*******************************************************************************/
    /* check for -- (pdv order) in list */
    /*******************************************************************************/
    %if %index(&item, --) > 0 %then varnum between &start and &end or;
    /*******************************************************************************/
    /* check for -character/-numeric (all character/numeric in pdv order) in list */
    /*******************************************************************************/
    %else %if %index(&item, -CHARACTER-) > 0 or %index(&item, -NUMERIC-) > 0 %then
      %do;
      (varnum between &start and &end
        %if %index(&item, -CHARACTER-) > 0 %then & trim(upcase(type)) = 'CHAR';
        %else & trim(upcase(type)) = 'NUM';
      ) or
      %end;
    %end;
    /********************************************************************************/
    /* check for - (variable list) in list */
    /********************************************************************************/
    %else %if %index(&item, -) > 0 %then %do;
      /* determine parts before and after - in variable list */
      %let before = %substr(&item, 1, %eval(%index(&item, -)-1));
      %let after = %substr(&item, %eval(%index(&item, -)+1));
      /* determine first and last number indices in variable list */
      %let lastch1 = %length(&before);
      %do %while (%verify(%substr(&before, %eval(&lastch1), 1), &numstr) = 0);
        %let lastch1 = %eval(&lastch1 - 1);
      %end;
      %let firstind = %substr(&before, %eval(&lastch1 + 1));
      %let lastch2 = %length(&after);
      %do %while (%verify(%substr(&after, %eval(&lastch2), 1), &numstr) = 0);
        %let lastch2 = %eval(&lastch2 - 1);
      %end;
      %let lastind = %substr(&after, %eval(&lastch2 + 1));
    %end;
  %end;
/* determine the character part of the variable list */
%let charpt1 = %substr(&before, 1, %eval(&lastch1));
%let charpt2 = %substr(&after, 1, %eval(&lastch2));
%if charpt1 ne charpt2 %then
  %do;
    %put ERROR: THE VARIABLE LIST WAS SPECIFIED INCORRECTLY!!!;
    %go to exit;
  %end;
%end;
%do k = %eval(&firstind) %to %eval(&lastind);
  trim(upcase(name)) = "&charpt1.&k" or
%end;

/**********************************
/* check for : (wildcard) in list */
/**********************************
/* change sas wildcard notation : to sql notation % */
%else %if %index(&item, :) > 0 %then
  trim(upcase(name)) like "%substr(&item, 1, %index(&item, :)-%index(&item, :)-1)%" or;
/**********************************
/* check for variable name in list */
/**********************************
%else %if &item ne %then trim(upcase(name)) = "&item" or;
/**********************************
/* check for no keeplist/droplist */
/**********************************
%else %if &item = %then 1=1 or;
/**********************************
/* go to next item in list */
/**********************************
%let i = %eval(&i + 1);
%let item = %bquote(%scan(&list, %eval(&i), %str( )));
%end;
/* finish off the "or"s */
%if &i > 1 %then 1 ne 1) &;
/* point to input sas dataset */
trim(libname) = &lib %trim(upcase(memname)) = &mem
/* in v8, names are case-sensitive (based on first occurrence); order names here by uppercase */
/* order by upcase(name) */
%
/*******************************/
/* create table of all formats */
/*******************************/
/* note that fmtname will not contain a leading $ for character formats, nor a trailing . */
create table allfmts as
  %if &fmtlib ne %then
    select fmtname, start, end, label from infmt outer union corr
    select fmtname, start, end, label from addedfmt
    order by fmtname, start
  ;
quit;
/* do not continue if _alphabetic is blank */
%if &_alphabetic_ = %then
  %do;
    %put WARNING: YOUR QUERY PRODUCED NO VARIABLES;
    %go to exit;
  %end;

/******************************/
/* run the freqs */
******************************/
/* loop through each variable */

%let a = 1;
%let freqvar = %scan(&_alphabetic_, %eval(&a), %str( ));
%do %until (&freqvar = );
  tables &freqvar / list missing out= &freqvar;
  %let a = %eval(&a + 1);
  %let freqvar = %scan(&_alphabetic_, %eval(&a), %str( ));
%end;
run;

/******************************************************/
/\ loop through each variable */
/******************************************************/

%let a = 1;
%let freqvar = %scan(&_alphabetic_, %eval(&a), %str( ));
%do %until (&freqvar = );

  %let varnum = %sysfunc(varnum(&dsid, &freqvar));
  %let vartype = %sysfunc(vartype(&dsid, &varnum));
  /* use bquote because varlabel can contain unmatched single quote */
  %let varlabel = %bquote(%sysfunc(varlabel(&dsid, &varnum)));
  /* note that varfmt will contain , as last character */
  %let varfmt = %sysfunc(varfmt(&dsid, &varnum));
  %let varlen = %sysfunc(varlen(&dsid, &varnum));

data _null_;
  if _n_ = 1 then
    do;
      call symput("count", put(nobs, 6.));
      set &freqvar nobs= nobs;
      stop;
    end;
  run;

/******************************************************/
/* if the freq output is too big (> &maxlevel) then replace with missing/nonmissing freq */
/******************************************************/

%if &count > &maxlevel %then
  %do;
    /* don't need to re-run full freq, just use output counts as weights for new freq */
    proc freq noprint data= &freqvar (rename= (count= wt));
      weight wt;
      tables &freqvar / list missing out= &freqvar;
      format &freqvar if &vartype = N %then
        %do;
          mnmfmt.
          %let holdfmt = &varfmt;
          %let varfmt = mnmfmt.;
        %end;
        %else if &vartype = C %then
          %do;
            $cmnmfmt.
            %let holdfmt = ;
            %let varfmt = $cmnmfmt.;
          %end;
          %cmnmfmt.
          %let holdfmt = ;
          %let varfmt = %cmnmfmt.;
    %end;
  run;

  %if &vartype = N %then
    %do;
      proc sql noprint;
        select min(&freqvar), max(&freqvar)
          if &holdfmt ne %then
            , min(&freqvar) format &holdfmt., max(&freqvar) format &holdfmt.;
        into :min separated by " ", :max separated by " ";
      from &lib..&mem
    ;
      quit;
    %end;
  %end;
%end;
%if &varfmt ne &varfmt ne mnmfmt. &varfmt ne mmmfmt. %then
  %do;
    data table;
      %if &vartype = C %then length &freqvar $&varlen;
    retain count percent 0;
    set allfmts;
    where %if &vartype = C %then "$"||upcase(left(trim(fmtname)))||'.' = %upcase(%cmpres(&varfmt))||'.';
    %if &vartype = N %then &freqvar = input(left(start), best12.);
    %else %if &vartype = C %then &freqvar = start;
    run;
  proc sort data= table;
    by label;
    run;
  data &freqvar;
    set &freqvar;
    label = put(&freqvar, &varfmt.);
  run;
  proc sort data= &freqvar;
    by label;
    run;
  data &freqvar;
    set &freqvar;
    label = put(&freqvar, &varfmt.);
  run;
  proc sort data= &freqvar;
    by label;
    run;
  data &freqvar;
    update table
      &freqvar;
    by label;
    run;
  proc sort data= &freqvar;
    by &freqvar;
    run;
  %end;
%else %if &varfmt = mnmfmt. %then
  %do;
    data &freqvar;
      length start end 92;
      set &freqvar;
      if &freqvar > . then do;
        %if &holdfmt ne %then
          %do;
            start = "&fmin";
            end = "&fmax";
          %end;
        %else
          %do;
            start = "&min";
            end = "&max";
          %end;
        end;
      else
        do;
          start = ' ';
          end = ' ';
        end;
      end;
  %end;
%else
  %do;
    data &freqvar;
      length start end $16;
      set &freqvar;
      retain start " ";
      set &freqvar;
      run;
  %end;
data &freqvar (keep= var fmtval count percent cumct cumpct);
length var fmtval $100;
set &freqvar (keep= var fmtval start end count percent);
var = &freqvar || ' [' || %if &vartype = N %then 'N' %else 'C' || ' ]';
%if &count > &maxlevel %then || '*' || ' '; %if &varlabel ne %then || ': ' || &varlabel';
%if &varfmt = %then
%do;
%if &vartype = N %then %let varfmt = best32.;
%else %if &vartype = C %then %let varfmt = $80.;
%end;
if count = 0 or trimn(start) = '' && trimn(end) = '' then
fmtdval = trim(left(put(&freqvar, &varfmt)));
%if &varfmt = mnmfmt. %then
%do;
%if &holdfmt = %then
&fmtval = &min-&max;
%do;
%else
&fmtval = &min-&max;
%end;
%else
%do;
else if start ne end then &fmtval = &min-&max = " || trim(left(start)) || '-' || trim(left(end))
else &fmtval = &min-&max = " || trim(left(start));
%end;
%end;
%else
%do;
else if start ne end then &fmtval = trim(left(start)) || '-' || trim(left(end))
else &fmtval = trim(left(start)) || '=' || trim(left(put(&freqvar, &varfmt)));
%end;
/* get cumulatives */
cumct + count;
cumpct + percent;
run;
/* append output from all variables */
proc append base=all data= &freqvar;
run;
/* end of looping through each variable */
%let a = %eval(&a + 1);
%let freqvar = %scan(&_alphabetic_, %eval(&a), %str( ));
%end;
/*************************/
/* print output */
/*************************/
proc print data= all label split='*';
label COUNT = 'count'
PERCENT = 'percent'
cumct = 'cumulative*count'
cumpct = 'cumulative*percent';
format percent cumpct 6.2;
by var notsorted;
id fmtdval;
var count percent cumct cumpct;
run;
/*************************/
/* null out macro variables */
/*************************/
%exit: %mend abc_freq;