Paper 81-26

Automating the Process of Listing the Most Frequent Values of Thousands of Variables in Large Datasets

Haiping Luo, Dept. of Veterans Affairs, Washington, DC
Philip Friend, Dept. of Agriculture, Washington, DC

ABSTRACT

Listing variable values with high frequency for large datasets which have thousands of variables can be a time-consuming process. Using common procedures like 'proc freq' and 'proc tabulate' involves individually identifying possible value range, data discretion and/or class group before running the procedure. This **%getfreq** macro automates the process of listing the most frequent values of all or selected variables in a dataset. The value list can be tailored to reflect only those values with desirable frequencies and percentages, in the selected value ranges, and by certain groups. The resulting value list can serve as a useful reference to dataset users.

A CASE CALLING FOR THE NEED

In a process of recoding datasets from the Agriculture Resources Management Survey, we needed to know the range and the most frequent values of more than one thousand variables. There are some SAS procedures that can accomplish this task, but all of them require efforts to look into each target variable's features before running the procedure.

'Proc freq' can list the frequency count and percentage for each value of the target variable. But if the values are not very discrete, the 'freq' output table can be huge. Also this procedure can only output one variable's frequency count to a file. 'Proc tabulate' can list the value and its frequency count across variables. But it is suitable only to discrete variables with only a few values. 'Proc univariate', 'proc summary' both work only for numeric variables.

While these procedures list values, frequencies, and percentages variable by variable, extra coding efforts are needed to pull the output of each variable together into an integrated list for easy reference for a large dataset. When the datasets in hand are large with thousands of variables, the coding efforts can be tedious. This case triggered the development of **%getfreq** macro, aimed at automating the process of listing target variables' most frequent values for datasets of any size.

DESIGN

The core process of **%getfreq** has three steps:

1. Determine target variables and user-defined conditions;
2. Process the numeric variables - list their names, most frequent values, frequency counts, percentages, means, maximums, and minimums;
3. Process the character variables - list their names, most frequent values, frequency counts, and percentages.

One twist to this seemingly simple process is to allow the user to run batch by setting parameters in the calling code or to run interactively by inputting parameter values in an input window. Another twist is to allow the flexibility of selecting all or part of the variables, selecting value ranges, determining by groups, and determining frequency and percentage filtering criteria. A third twist is to append each variable's output value list to a comprehensive list which includes all target variables and their value statistics.

DETAIL OF THE CODE

1. COLLECT USER INPUT

Macro **%getfreq** needs these user inputs: input and output libnames and their physical paths; input and output datasets' names; and input variable filtering and sorting criteria. Two input methods are incorporated into the code: one is input by submitting parameter values in the %getfreq(…) string; the other is input by typing in parameter values in the pop up windows. The string method allows batch processing, while the pop-up-window method allows interactive processing.

To allow the user to choose the run method, a macro variable 'screen' is defined to represent the batch or interactive mode. An %if structure detects the value of &screen. When &screen is 1, a display macro will display pop-up windows to collect user's input, otherwise the parameter values in the **%getfreq()** call will be used to run the macro:

```
%macro getfreq(screen=1, …)
*** starts do window conditioning on &screen;
 %if &screen=1 %then %do;
  ** define window;
  %WINDOW first color=cyan …;
  ** define a display macro;
  %MACRO INTRFACE;
   %DISPLAY first.fileinfo;
   %DISPLAY first.varinfo blank;
  %MEND INTRFACE;
  ** run the display macro %intrface;
  %INTRFACE; …
 %end; *** ends do window;
```

After the input selection portion, the main code of the **%getfreq** is wrapped in a nested macro called %freqmain. The contents of %freqmain are described in the following sections 2 through 4.

2. DETERMINE VARIABLE NUMBERS BY TYPE

To determine how many numeric and character variables are in the user defined dataset, several data steps and procedures are used, together with %if structures to insert code conditionally:

```
*** check variable type and collect variable
numbers;
data _inall;
** read in dataset with its conditions;
 set %IF &PATHIN NE %THEN
%STR("&PATHIN.&indat"); %ELSE
%STR(&LIBIN..&INDAT);
  (%if &varlist ne  %then %do;
      keep=&varlist &byvar
   %end;
   %if (&filtlgic) ne  %then %do;
      where=(&filtlgic)
   %end;) end=lastobs;
  if lastobs then do;
     n=_n_;
    ** collect total observations;
     call symput('dataobs',n);
  end;
  drop n;
** sort if by group is defined;
%if &byvar ne %then %do;
   %LET byproc = %str(bygroup="&byvar");
   %LET bylab = label bygroup="Frequency by
This Group (bygroup)";
   proc sort data=_inall; by &byvar;
```

```
   run;
%end;
proc format;
 value tp 1 = "Number" 2 = "Char";
proc format;
 value tpvar
     1 = "No. of Numeric Variables"
     2 = "No. of Character Variables";
run;

** keep only the type for each variable;
proc contents noprint data=_inall
out=_intype(keep=type);
** list the count of variables by type;
proc freq data=_intype;
 tables type / list out=_intpsum;
 format type tpvar.;
run;
data _null_; set _intpsum;
 ** check if there are numeric and/or
charactor variables;
 if type=1 then do;
   ** collect numeric var number;
   call symput('nvnumber',count);
 end;
 else if type=2 then do;
   ** collect charactor var number;
   call symput('cvnumber',count);
 end;
run;
```

3. PROCESS NUMERIC VARIABLES

If there are numeric variables in the defined dataset, a %do loop is used to process each numeric variable to get its value list and each value's frequency and percentage. The variable's maximum, minimum and mean values by the defined 'by group' are also listed:

```
%if %eval(&nvnumber)>0 %then %do;
*** get numeric variables;
title "Numeric Variables";
data _in2; set _inall;
   keep _numeric_ &byvar;

** get the name, label of the numeric
variabls;
data _in4; set _in2(keep=_numeric_);
proc contents noprint data=_in4
out=_in3(keep=name label type varnum nobs);

** print a list of all numeric variables to
run frequency for;
title2 "Dataset &indat - variable
information: number of selected numeric
variables=&nvnumber";
proc print data=_in3;
     format type tp.;
run;
title2 "First 3 observations of the selected
numeric variables in dataset &indat";
proc print data=_in4(obs=3) label;
footnote;
run;

** go through each variable to get frequency;
%do i=1 %to &nvnumber;       * starts do i;

  ** determine the variable name;
  data _null_; set _in3;
     if varnum = &i then do;
        call symput('nname',name);
     end;
  data _2n; set _in2(keep=&nname &byvar);
     length value 8 varname $ 32;
      ** assign the value of the variable to a
common variable;
     value = &nname;
** assign the variable name as text to a
common variable;
     varname = "&nname";
label
value = "Value of Frequecy Variable (value)"
varname = "Frequency Variable (varname)";
  run;
```

The critical steps in the above code are to put the value of the processed variable into a common variable 'value', and the name of the processed variable into another common variable 'varname'. These enable the frequency calculation and the output appendance to focus on these two common variables instead of individual input variables.

```
  ** get frequency for each value of the
frequency variable;
  ** keep only those with frequency count >
&dropfrq in the &byvar group;
  proc freq data=_2n order=freq noprint;
     tables varname*value / list
out=_2out(where=(count gt &dropfrq));
     %if &byvar ne   %then %do;
        by &byvar;
     %end;
  run;

  ** get group mean, max, min, total obs
number for the frequency variable;
  proc means data=_2n (keep=&nname
            &byvar) noprint;
     var &nname;
     %if &byvar ne   %then %do;
        by &byvar;
     %end;
     output out=_2mean mean=mean max=max
       min=min n=totalobs nmiss=missobs;
     run;
  data _2mean;
     set _2mean(drop=_type_ _freq_);
     length varname $ 32;
     varname = "&nname";
  label
  mean = "Group Mean of This Variable (mean)"
  max  = "Group Maximum (max)"
  min  = "Group Minimum (min)"
  totalobs = "Non-missing Obs in This By
Group (totalobs)"
  missobs = "Missing Obs in This By Group
(missobs)"
  varname = "Frequency Variable (varname)";
  ** merge group means, etc. back to freq
list;
  data _2out;
     merge _2out(in=a) _2mean;
     by varname
     %if &byvar ne   %then %do;
        &byvar
     %end;
     ;
     if a;
     %if &byvar ne   %then %do;
```

```
            &byproc;
            &bylab;
        %end;
  run;
 ** create or append frequency list;
  %if &i=1 %then %do;
     data OUD.&outdat.n;
            set _2out;
     run;
  %end;
  %else %do;
  proc datasets ;
     append base=OUD.&outdat.n data=_2out

(where=(percent gt &droppct or percent = .));
     delete _2n _2out;
  run;
  quit;
  %end;
%end;                          *end do i;
** print frequency list;
title1 "&nvnumber Numeric variables (Dataset
observations = &dataobs)";
title2
  "Highest (>&droppct%) frequent (count >
&dropfrq) values (by &byvar)";
title3
  "of numeric variables in dataset &&indat
(this output was saved as
&pathout.&outdat.n)";

proc sort data=OUD.&outdat.n;
  by varname
  %if &byvar ne   %then %str(&byvar;);
  %else %str(;);
proc print data=OUD.&outdat.n label;
  by varname;
  var
  %if &byvar ne   %then %do;
     bygroup
     &byvar
  %end;
     value count percent mean max min
     totalobs missobs;
    sum count;
    sumby varname;
  footnote;
  run;
title;
%end; *** ends numeric variable process;
```

4. PROCESS CHARACTER VARIABLES

The procedure for getting character variables' frequent value list is similar to that for the numeric variables, except that there is no calculation of group means, maximums, and minimums. The following code presents in both the numeric and the character portions of the program, but it is more critical to the character portion. By default, SAS' numeric variables have a length of eight bytes, while a character variable's length is either defined explicitly by the user or implicitly by the length of the first value ever read in. To prevent the common output variable 'value' being truncated to the length of the first character variable in the input sequence, the following code obtains the maximal length of all character variables; the maximal length is then used to set the length of the common output variable 'value'.

```
** get the name, label, length, and number of
observations of the character variables;
data _ic4; set _ic2(keep=_character_);
proc contents noprint data=_ic4
out=_ic3(keep=name label type varnum nobs
length);

** get the longest charactor value to set the
length of the value variable;
proc means data=_ic3(keep=length) noprint;
     output out=_ic32 max=max;
run;
data _null_;
     set _ic32;
     call symput('maxlen',max);
run;
. . .
** go through each variable to get frequency;
%do i=1 %to &cvnumber;       * starts do i;

  ** determine the variable name;
  data _null_; set _ic3;
     if varnum = &i then do;
        call symput('cname',name);
     end;
  run;
  data _2c; set _ic2(keep=&cname &byvar);
    ** set the value to the maximal length;
     length value $ &maxlen
            varname $ 32;
     ** assign the value of this variable;
     value = &cname;
     ** assign the variable name as text to
varname;
     varname = "&cname";
     label
value = "Value of Frequecy Variable (value)"
varname = "Frequency Variable (varname)"
      ;
  run;
. . .
%end;                        *end do i;
```

After processing the character variables and performing some housekeeping chores, the definition of the %freqmain sub macro ends. The parent macro **%getfreq** then calls %freqmain, and itself is closed by the "%mend getfreq;" statement.

USAGE

%Getfreq has been tested under SAS for Windows versions 6.12, 7, 8, and for IBM MVS/TSO version 6.09. The following code can call **%getfreq** on the Windows platform:

```
/* put this option at the beginning of your
code*/
options sasautos="path-to-this-macro-
directory";
/*assume popup windows, works for V8*/
%getfreq
/*assume popup windows, works for V6.12, V7*/
%getfreq()
/*no windows, take all default values defined
in the macro code*/
%getfreq(screen=0)
/*no windows, run the test dataset work.a */
%getfreq
      (screen=0,
       pathin=, libin=work, indat=a,
       pathout=d:\temp\, outdat=getfreq,
       varlist=%str(x y datet ),
       dropfrq=1, droppct=0,
       filtlgic=%str(x>0 or y ne . ),
      byvar=%str(studname)  )
```

There is a detail explanation of the parameters in the code file. To control the frequency output, the most important parameters are &droppct and &dropfrq. Setting 'dropfrq=1' will drop all the values which only occur once. This will prevent the continuous value being included in the output file. For a dataset with millions of observations, only drop "frequency count equals to one" may not be enough. The &droppct parameter allows the values with low frequency percentage being dropped from the output. For example, 'droppct=5' will keep only those values which's frequencies are higher than 5 percent of the total observations in the calculation group.

%Getfreq generates one dataset for numeric variables and one for character variables. The dataset(s) contains variables' name, frequent values, the frequency count and the percentage of each value, and some group statistics if it is a numeric variable. The datasets are saved in the &pathout directory as "getfreqn.sas7bdat" for numeric variables and "getfreqc.sas7bdat" for character variables. The file extension assumes that the code was run under SAS version 7 or 8.

%Getfreq also generates the following lists at the output window for numeric and/or character variables:

- Number of variables;
- Variable names and labels;
- The first 3 observations;
- The most frequent values for each variable.

The following is a sample output generated to the output window:

A 3 Numeric variables (Dataset observations = 24)
Highest (>0%) frequent (count > 1) values (by studname)
of numeric variables in dataset a
(this output was saved as d:\temp\getfreqn.*)

Frequency Variable (varname)=X

B

Frequency by This Group (bygroup)	Student name (studname)	Value of Frequecy Variable (value)	Frequency Count	Percent of Total Frequency
studname	Alex Dickins	0	2	66.667
studname	John Graves	1	2	66.667
studname	Larry Lin	0	2	66.667
studname	Linda Brooks	0	2	66.667
studname	Lynne Miller	3	2	66.667
studname	Mitch Tramp	1	3	100.000
studname	Susan Williams	1	2	100.000

VARNAME			15	

(*print out continues)*

Group Mean of This Variable (mean)	Group Maximum (max)	Group Minimum (min)	Non-missing Obs in This By Group (totalobs)	Missing Obs in This By Group (missobs)
1.00000	3	0	3	0
0.66667	1	0	3	0
0.33333	1	0	3	0
1.00000	3	0	3	0
2.33333	3	1	3	0
1.00000	1	1	3	0
1.00000	1	1	2	0

VARNAME

C
Input and output PATHS: or work; d:\temp\
Input dataset: a(where=( ) keep=( ))
Output datasets: numeric freq - d:\temp\getfreqn,
char. freq - d:\temp\getfreqc,
sorted by studname, where=((count > 1) and (percent > 0))

Part **A** in the above list is a title. The title indicates that: the dataset name is a; there are 24 observations in it; there are 3 numeric variables in dataset a; this portion of the list is for variable x (frequency variable=x); this list displays the value of x which have frequency count more than 1 (count>1); the frequency is run by a variable "studname"; the output file is saved as "d:\temp\getfreqn.*".

Part **B** is the contents of the list. There are 10 columns in the table. We can read the table this way: under the group variable(studname)'s first value, "Alex Dickins", the frequency variable x has a most frequent value, 0, which occurred twice, and accounted for 66.67 percent of all values of x under this student name "Alex Dickins"; the mean value of x under "Alex Dickins" is 1, while the maximum is 3 and the minimum is 0. The last two columns show the total number of non-missing and missing observations for the group calculation. There is a summation by 'varname' at the end of the table under column 4 'Frequency Count". The value 15 indicates that 15 out of 24 total observations from the dataset are included in this most frequent value list for this variable x.

Part **C** is a footnote for the list. It reports the input and output file paths, libnames, dataset names, variable filtering criteria, etc.

The mainframe version of **%Getfreq** does not contain the input window and the output library 'libname OUD . . .' statement. Assuming that a JCL statement defines the output files like this:

```
//OUD DD DSN=USER1.SAS.DATA.GETFREQ,...
```

The calling statement for the test dataset 'a' is:

```
%getfreq
      (pathin=, libin=work, indat=a,
       pathout=oud, outdat=getfreq,
       varlist=,
       filtlgic=,
       dropfrq=1,droppct=0,
       byvar=studname
       );
```

The mainframe code also limited the line length to 80 characters, moved '/' from the first column to avoid confusion with JCL, and removed quotation marks from comment area to prevent error in reading. The output of the mainframe version of **%getfreq** are the same as the Windows version.

CONCLUSION

Obtaining most frequent values of many variables in large datasets can be a tedious task. This **%getfreq** macro automates the process and saves the frequent values to files. This macro allows a user to tailor the results to include all or part of variables, to report only the values with desirable frequencies and percentages, and to run and group results by certain variables. The resulting frequent value list can serve as a useful reference for dataset users. To obtain a complete copy of the most current version of **%getfreq**, please go to these links:

http://chinafoundation1.org/getfreq.sas
http://chinafoundation1.org/getfreq-mainframe.sas

Further development of **%getfreq** will focus on improving efficiency, allowing more options, testing for other platforms, and making the execution more robust.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please contact the authors at:

Author Names Haiping Luo (hpluo@yahoo.com)
Philip Friend (pfriend@ers.usda.gov)

Code Web URL http://chinafoundation1.org/getfreq.sas

SAS Discussion Board
http://clubs.yahoo.com/clubs/sas
http://go.to/sas-net

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