

Journeyman's Tools: Data Review Macro FreqAll: Using Proc SQL List Processing with Dictionary.Columns to Eliminate Macro Do Loops

Ronald J. Fehd, Centers for Disease Control and Prevention, Atlanta, GA, USA

ABSTRACT

The SAS® macro language is simple, yet powerful. List Processing with Proc SQL is also simple, yet powerful. This paper provides a data review macro FreqAll which illustrates using Proc SQL reading Dictionary.Columns to replace macro %do loops.

Expected audience is advanced users, and macro programmers.

Keywords: data review, dynamic programming, list processing, do loop, macro, SQL.

INTRODUCTION

Whenever I receive a data set, the first thing I want to do is examine the data, looking first at the data structure, (Proc Contents) then at a frequency listing of all the variables (Proc Freq). In data sets containing continuous variables the frequency listing gets long quickly. What I really want is similar to Proc Univariate: a list of the 10 high and low values. How can I make Proc Freq do that?

In this paper I develop a macro with a macro %do loop for each variable using Dictionary.Columns and then replace that loop with Proc SQL list processing.

The finished product is a listing which contains:

1. data structure list similar to Proc Contents
2. abbreviated frequency showing the high and low values

Contents

Proc Contents	2
Proc Freq	2
Proc Univariate	3
Macro %Do Loop	3
Proc SQL select into	4
Macro Freq-Loop	4
FreqAll: SQL replaces macro do loop	5
Conclusion	8

PROC CONTENTS

There are several ways to access the data structure of a data set:

1. Proc Contents:

```
1 Proc Contents data = SAShelp.Class;
```

2. Proc Datasets:

```
1 Proc Datasets library = SAShelp
2 details nolist
3 memtype = data;
4 contents
5 data = Class;
6 quit;
```

3. Proc Print:

```
1 PROC Print data = SAShelp.Vcolumn
2 (where = ( Libname eq 'SASHELP'
3 and MemName eq 'CLASS'
4 ) );
```

4. Proc SQL, describe:

```
1 PROC SQL; describe table SAShelp.Class
2 ; quit;
```

5. Proc SQL, select:

```
1 PROC SQL; select Name, Type, Length, Label
2 from Dictionary.Columns
3 where Libname eq 'SASHELP'
4 and MemName eq 'CLASS'
5 ; quit;
```

I work with the SQL-select example, which provides both the data structure list and, as I show later, the loop of macro calls.

PROC FREQ

Proc Frequency provides a listing of all values of a variable. For large data sets with continuous variables, the listing gets long quickly.

```
1 PROC Freq data = SAShelp.Prdsal2;
2 tables _all_;
```

1800 lines ≈ 36 pages!

F-Freq.lst				
Month/Year				
MONYR	Frequency	Percent	Cumulative Frequency	Cumulative Percent
JAN95	24	2.40	24	2.40
FEB95	24	2.40	48	4.80

My goal is an output of few pages: one for data structure listing and others for the frequency listing of high and low values.

PROC UNIVARIATE

Proc Univariate provides a listing of the extreme values of a variable, but only of the numerics.

```
1  _____ F-Univariate.sas _____
2  PROC Univariate data = SAShelp.Class;
   var    _numeric_;
```

```
6  _____ F-Univariate.lst snip 1 _____
7  The UNIVARIATE Procedure
   Variable:  Age
```

```
55  ...
56  _____ F-Univariate.lst snip 2 _____
57  Extreme Observations
58  -----Lowest-----      -----Highest-----
59  Value      Obs      Value      Obs
60
61      11      18      15      8
62      11      11      15      14
63      12      16      15      17
64      12      13      15      19
65      12      10      16      15
```

The above examples illustrate my wish list: a list of variable attributes, and a limited frequency listing, showing only the extreme values. In the next section I show a macro %do loop, illustrate how to use it for a procedure, and examine its programming issues.

MACRO %DO LOOP

A %do loop in a macro is similar to a data step loop. On listing line 28, log line 8, the index, I, is incremented from the lower bound, 1, to the upper bound, the macro variable Dim-Item. The macro variable array, Item, contains 3 elements, the sequentially numbered macro variables: Item1, Item2, and Item3. The dimension of the macro array is Dim-Item. This naming convention is necessary in order for the loop to access each element in the loop with the reference: double ampersand, array-name, index — &&Item&I.— shown in log line 9.

```
21  _____ Macro-do-loop.log _____
22  1      %Macro Do_Loop;
23  2      %local Item1 Item2 Item3 Dim_Item I;
24  3      %Let Item1      = X1 ;
25  4      %Let Item2      = Y-2;
26  5      %Let Item3      = Z 3;
27  6      %Let Dim_Item = 3 ;
28  7
29  8      %Do I = 1 %to &Dim_Item.;
30  9          %Put Item&I<&&Item&I.>;
31 10          %end;
32 11      %mend;
33 12      %Do_Loop
34  Item1<X1>
35  Item2<Y-2>
   Item3<Z 3>
```

The problems associated with using macro arrays are:

	log line	statement
ensure the scope of macro array variables	2	%local Item1 ...
allocation of each element	3-5	%Let Item? = ...
allocation of the dimension (upper bound)	6	%Let Dim.Item = 3;

PROC SQL SELECT INTO

Fehd[Array-Contents] illustrates using a Proc Contents output data set to create a macro array. Fehd[Array-SQL] shows how to use Proc SQL to do the same, as shown here. Clay[Tight-Looping] provides comprehensive analysis of usage of macro arrays. Whitlock[Macro-Design] reviews macro design and list processing.

Note the system-generated automatic macro variable SQLObs, listing line 35, has the upper bound value, 5.

```
1  PROC SQL noprint;
2      select Name
3      into :Name1 - :Name999
4      from Dictionary.Columns
5      where Libname eq 'SASHELP'
6            and MemName eq 'CLASS'
7      ;quit;
8  %Put _User_;
```

```
35  GLOBAL SQLObs 5
36  GLOBAL SQLObs 38
37  GLOBAL NAME4 Height
38  GLOBAL NAME5 Weight
39  GLOBAL NAME2 Sex
40  GLOBAL NAME3 Age
41  GLOBAL NAME1 Name
```

MACRO FREQ-LOOP

In this section I show a demonstration macro which makes a macro array of the variables in a data set and then a macro %do loop.

As noted above, I use Dictionary.Columns to supply the listing of the data structure.

```
1  PROC SQL; describe table Dictionary.Columns;
2  quit;
```

This table contains these columns:

```
25  create table DICTIONARY.COLUMNS
26  (
27      libname char(8) label='Library Name',
28      memname char(32) label='Member Name',
29      memtype char(8) label='Member Type',
30      name char(32) label='Column Name',
31      type char(4) label='Column Type',
32      length num label='Column Length',
33      npos num label='Column Position',
34      varnum num label='Column Number in Table',
35      label char(256) label='Column Label',
36      format char(16) label='Column Format',
37      informat char(16) label='Column Informat',
```

The macro FreqAll in program FreqAll-Loop has parameters for: libref, data set name, how many extreme values to show, and testing (debugging).

```
2  %Macro FreqAll /* ----- */
3  (In_Lib = SASHELP
4  ,In_Data = Class
5  ,ShowHiLo = 5
6  ,Testing = 0
7  )/des = 'site: freq_all_ showing first N values'
```

Note that two macro arrays are created — Name and Type — in lines 28-29, and the scope of their variables is declared in lines 23-24.

```
23  /** Initialize macro array; %local SqlObs SqlObs;
24  %Do I = 1 %to 999; %local Var&I. Type&I.; %end;
25
26  PROC SQL noprint;
27      select Name, Type
28      into :Var1 - :Var999
29      ,:Type1 - :Type999
30      from Dictionary.Columns
31      where LibName eq "&In_Lib."
32            and MemName eq "&In_Data."
33      ;quit;
34
35  %If &Testing. %then %do; %Put SQLObs<&SQLObs.>; %end;
```

The loop begins on line 37 and ends on line 54.

Note the four macro array element references (&Var&I.) in lines:

```
39
41
42 (&Type&I.)
46
```

For this demonstration I make a report with only the highest values. See the complete high and low processing in program FreqOf below.

```

                                FreqAll-Loop.sas snip 3
37 %Do I = 1 %to &SQLobs.;%* ----- *;
38 Proc Freq data = &In_Lib.&In_Data.;
39     tables    &Var&I.
40             / list missing noprint
41             out = Freq(rename = (&Var&I. =
42                             %If &Type&I. eq char %then ValuC;
43                             %Else                               ValuN;
44                             )                               );
45 DATA   Freq;
46 retain Var "&Var&I.";
47 if      0 then          %** copy structure;
48 set     FreqList;
49 set     Freq;
50
51 PROC Append base = FreqList
52             data = Freq(obs = &ShowHiLo.)
53             force;
54 %end;%*do I ..... *;
55
56 PROC Print data = FreqList;
57         by      Var notsorted;
58         id       Var;
59 run;%* ..... *;%Mend;
60 %*FreqAll;%*test;
61 %FreqAll(In_Data = PRDSAL2

```

FREQALL: SQL REPLACES MACRO DO LOOP

Where FreqAll was the name of the *macro* which contained the macro array and %do loop, here I name the *program* FreqAll and have placed the statements inside the %do loop into the macro subroutine FreqOf.

Note that the parameter names are aligned with the variables from the Dictionary.Columns data set: Name, Type, Length, Format and Label.

If the user desires not the highest and lowest *values* but the highest and lowest *frequencies*, then I have provided a parameter, Order = freq (lines 50: default, 52: must be enabled by opening comment on line 49 and closing comment on line 51), which shows the mode: the values occurring most and least often.

Note that the parameters InLib, InData, and Nobs2View refer to global macro variables set before the macro is called.

```

                                FreqAll.sas macro FreqOf pg. 1
43 %Macro Freqof
44 (Name      = /* var                                */
45 ,Type      = /* in (char,num)                      */
46 ,Length    = /* integer                            */
47 ,Format    = /* $char                              */
48 ,Label     = /* $char40                            */
49           /* for hi and low, use either: ***** */
50 ,Order     = internal /* default: hi and low values */
51           /* for mode use: ** *** ** * **** ** */
52 ,Order     = freq    /* hi and low descending count */
53 ,InLib     = &In_Lib. /* FreqAll scope: global */
54 ,InData    = &In_Data. /* FreqAll scope: global */
55 ,Nobs2View = &HiLowToView /* FreqAll scope: global */
56 ,Testing   = 0        /* show stuff? ,Testing=1 */
57 )/des      = 'site: FreqOf: subroutine of prog FreqAll'
58 ;/*change notes

```

Compare to program Freq-All-Loop.

The Proc Freq is the same except for the addition of the `order =` parameter. The macro array references (`&&Name&I.`, `&&Type&I.`) have been changed to parameter name (macro variable) references: `&Name`, `&Type`.

Data Freq, `attrib` standardizes the data set structure.

This section either appends a small listing

or divides the list into high and low sets of values and adds a note indicating that values were removed.

```

65 _____ FreqAll.sas macro FreqOf pg. 2 _____
66 %local Nobs; %let Nobs = 0; %*initialize for symput;
67
68 PROC Freq data = &InLib..&InData.
69     order = &Order.;
70     tables &Name
71     / list missing noprint
72     out = Freq(rename = (&Name. =
73         %If &Type = char %then ValuC;
74         %else ValuN; ));
75
76 DATA Freq;
77     attrib Attributes length = %eval(32 +1 +4 +1 +4 +1 + 40)
78     ValuC length = $20
79     ValuN length = 8
80     Count length = 4 format = comma.
81     Percent length = 8 format = 6.2
82     Level length = 4;
83
84 retain Attributes "&Name. &Type.:&Length. &Label"
85     ValuC "." ValuN Level .;
86
87 do until(EndoFile);
88 set Freq end = EndoFile
89     nobs = Nobs;
90 Level ++1; %*increment retained counter;
91 output; end; %*do until EndoFile;
92 call symput( 'Nobs', compress(put(Nobs,32.)));
93 stop; run;
94 %If &Testing. %then %Put Note:&SysMacroName.: nobs<&Nobs.>;
95
96 %If &Nobs. le %eval(2 * &Nobs2View. + 2) %then %do;
97 PROC Append base = ListFreq
98     data = Freq; %end;
99
100 %Else %do;%* ----- *;
101 DATA Snipped;
102 set Freq(obs = 1);ValuC = '<snipped>';ValuN = .;
103 Count = .; Percent = .;
104 Level = .; output; stop;
105
106 PROC Append base = ListFreq %*high values;
107     data = Freq(obs = &Nobs2View.);
108
109 PROC Append base = ListFreq
110     data = Snipped;
111
112 PROC Append base = ListFreq %*low values;
113     data = Freq(firstobs = %eval( &Nobs.
114         - &Nobs2View. +1));
115 %end;%*Else do ..... *;
116 run;%* ..... *; %Mend Freqof;

```

In order for the global macro variables
In_Lib and In_Data

to be used in the select ... from
... where phrase, lines 128-129 and
133-134, each must be in ALL CAPS.

Proc SQL creates three objects:

1. line 124: table **ListAttributes** containing the variable attributes; this is the first page of the summary report.
2. line 137: macro variable **List** containing calls of macro FreqOf for each variable; these are executed on line 145.
To view the FreqOf statements, disable line 121:
%*Let SQLprint = noprint;
This select statement, lines 131-138, replaces the macro array and %do loop in the FreqAll-Loop program. Note: a macro variable for the upper bound is not needed.
3. line 139: macro variable **NobsData**: the number of observations of the input data set; this is used in the title2 statement, lines 148-149.

The report is printed in two parts: attributes, and frequencies.

Housecleaning: delete the program's global macro variables.

Changing line 121 to:

%*Let SQLprint = noprint;
produces this output, which shows the statements in the macro variable List.
Note: spaces have been added to align columns and improve readability.

This is the FreqAll report for SASHELP.PrdSal2; compare to program F-Freq.sas.
The first page of the FreqAll report contains Proc Contents information.

```

117 ----- FreqAll.sas program FreqOf snip -----
118 %Let In_Lib      = Library;
119 %Let In_Data     = MyData ;
120 %Let HiLowToView = 5      ;
121

```

```

122 ----- FreqAll.sas List Processing -----
123 %Let In_Lib = %upcase(&In_Lib.);
124 %Let In_Data = %upcase(&In_Data.);
125
126 %Let SQLprint = print;%*testing: view FreqOf statements;
127 %Let SQLprint = noprint;
128
129 PROC SQL &SQLprint.;
130     create table ListAttributes as
131     select Varnum, Name, Type, Length
132           , Format, Informat, Label, Npos
133           from Dictionary.Columns
134           where Libname eq "&In_Lib."
135                 and MemName eq "&In_Data."
136                 and MemType eq 'DATA'
137     ; select '%FreqOf(name = ' !! trim(Name)
138           !! ',type = ' !! trim(Type)
139           !! ',length = '
140           !! compress(put (Length,32.))
141           !! ',label = ' !! trim(Label)
142           !! ')'
143     into :List separated by ' '
144     from ListAttributes
145     ; select Nobs into :NobsData
146     from Dictionary.Tables
147     where Libname eq "&In_Lib."
148           and MemName eq "&In_Data."
149           and MemType eq 'DATA'
150
151     ; quit;
152 &List. ; %*execute macro calls;
153
154 %Let NobsData = &NobsData;%* remove leading blanks;
155 Title2
156     "&SysProcessName.: &In_Lib..&In_Data. nobs:&NobsData.";
157
158 PROC Print data = ListAttributes noobs;
159     Title3 "variable attributes";
160
161 PROC Print data = ListFreq;
162     var Valu: Count Percent Level;
163     by Attributes notsorted;
164     id Attributes ;
165     Title3 "list of variable frequencies";
166 run;
167 %symdel In_Data In_Lib HiLowToView List NobsData SQLprint;

```

```

1 ----- FreqAll-List-FreqOf-SASHELP-PrdSal2.txt -----
2 Program FreqAll: SASHELP.PRDSAL2 nobs:23040
3 %FreqOf(name = COUNTRY ,type = char,length = 10,label = Country)
4 %FreqOf(name = STATE ,type = char,length = 22,label = State/Province)
5 %FreqOf(name = COUNTY ,type = char,length = 20,label = County)
6 %FreqOf(name = ACTUAL ,type = num ,length = 8,label = Actual Sales)
7 %FreqOf(name = PREDICT ,type = num ,length = 8,label = Predicted Sales)

```

FreqAll.lst PrdSal2 pg. 1							
Program FreqAll: SASHELP.PRDSAL2 nobs:23040							
variable attributes							
varnum	name	type	length	format	informat	label	npos
1	COUNTRY	char	10	\$CHAR10.		Country	48
2	STATE	char	22	\$CHAR22.		State/Province	58
3	COUNTY	char	20	\$CHAR20.		County	80
4	ACTUAL	num	8	DOLLAR12.2		Actual Sales	0
5	PREDICT	num	8	DOLLAR12.2		Predicted Sales	8

The second page contains the abbreviated frequencies of each variable.

Note: the listing is truncated to save space.

The complete listing from program FreqAll of SAShelp.Prdsal2 is approximately 100 lines; three pages, instead of 20 from

Proc Contents: 2 pages
Proc Freq: 18 pages

FreqAll.lst PrdSal2 pg. 2

```

25 Program FreqAll: SASHELP.PRDSAL2 nobs:23040
26 list of variable frequencies
27
28 Attributes          ValuC          ValuN Count Percent Level
29
30 COUNTRY char:10 Country      U.S.A.      .    1,000 100.00    1
31
32 STATE char:22 State/Province California .    1,000 100.00    1
33
34 COUNTY char:20 County      .    1,000 100.00    1
35
36 ACTUAL num:8 Actual Sales .      8.16    1    0.10    1
37      .      10.20    1    0.10    2
38      .      13.60    1    0.10    3
39      .      17.00    1    0.10    4
40      .      27.20    1    0.10    5
41      <snipped>
42      .      3410.20    1    0.10   851
43      .      3413.60    1    0.10   852
44      .      3415.30    1    0.10   853
45      .      3440.80    1    0.10   854
46      .      3515.60    1    0.10   855
47
48 PREDICT num:8 Predicted Sales .      4.08    2    0.20    1

```

Fehd in [Prog-Doc] discusses necessary items in a program header.

```

1  /*      Name: FreqAll.sas
2
3  Requirements : description : Proc Freq of all vars in data set
4                purpose    : provide shorter listing than tables _all_
5
6  -----
7  Contexts    : program group: data review
8                program type: routine
9                SAS      type: program with parameters
10               uses routines: in-program macro %FreqOf
11
12  -----
13  Specifications: input  : libref
14                        data
15                        max number of high and low obs to view
16                process: SQL writes macro calls
17                        Proc Freq of each var
18                        save to data
19                        append to report
20                        subset if Nobs greater than Max-N-to-view
21                output : print report
22
23  -----
24  Information  : author: Ronald J. Fehd
25
26  -----
27  Usage Examples:
28  %Let In_Lib    = Library;
29  %Let In_Data   = MyData ;
30  %Let HiLowToView = 5      ;
31
32  %Inc 'FreqAll.sas';

```

To receive the latest edition of this program send an e-mail to the author
mailto:RJF2@cdc.gov
with the subject:
request FreqAll

Suggested Readings

- Abolafia[Data-Check] provides a macro to reproduce Proc DataChk.
- Carpenter[Guide-2e, ch. 9] discusses dynamic programming (list processing).
- Wobus and Gober[Data-Analysis] show data review with procs Summary and Univariate.

CONCLUSION

FreqAll The data review utility program FreqAll provides a shorter data set summary with more information.

Proc SQL List processing (select ...into :List) can eliminate the use of macro arrays and %do loops. This yields clearer code.

Acknowledgements

My colleagues at CDC, too many to mention here, provided the dirty data for which I originally developed this routine in the early 1990s. Toby Dunn provided commentary and critique. Dianne Rhodes whispered SQL encouragement to me. I am grateful to Ian Whitlock and for his many contributions to SAS-L, the on-line SAS User Group; he raised the bar.

REFERENCES

[Data-Check] Jeffrey M. Abolafia, *PROC DATACHK Revisited: The DATACHK Macro*, Proceedings of the 22nd Annual SAS® Users Group International Conference, 1997.
<http://www2.sas.com/proceedings/sugi22/POSTERS/PAPER229.PDF>

[Guide-2e] Art Carpenter, 2004. *Carpenter's Complete Guide to the SAS® Macro Language, Second Edition*, Cary, NC: SAS Institute Inc.
http://support.sas.com/publishing/bbu/companion_site/59224.html

[Tight-Looping] Ted Clay, *Tight Looping With Macro Arrays*, Proceedings of the 31st Annual SAS® Users Group International Conference, 2006.
<http://www2.sas.com/proceedings/sugi31/040-31.pdf>

[Array-Contents] Ronald Fehd, *%Array: construction and usage of arrays of macro variables*, Proceedings of the 22nd Annual SAS® Users Group International Conference, 1997.
<http://www2.sas.com/proceedings/sugi22/CODERS/PAPER80.PDF>

[Array-SQL] Ronald Fehd, *%Array: construction and usage of arrays of macro variables*, Proceedings of the 29th Annual SAS® Users Group International Conference, 2004.
<http://www2.sas.com/proceedings/sugi29/070-29.pdf>

[Prog-Doc] Ronald Fehd, *Journeyman's Tools: The Writing For Reading And Reuse Program Header*, Proceedings of the 30th Annual SAS® Users Group International Conference, 2005.
<http://www2.sas.com/proceedings/sugi30/067-30.pdf>

[Macro-Design] Ian Whitlock, *A Second Look at SAS® Macro Design Issues*, Proceedings of the 29th Annual Conference of the SAS® Users Group, Int'l, 2004.
<http://www2.sas.com/proceedings/sugi29/244-29.pdf>

[Data-Analysis] Diana Zhang Wobus and John C. Gober, *A Step-by-Step Illustration of Building a Data Analysis Tool with Macros*, Proceedings of the 22nd Annual SAS® Users Group International Conference, 1997.
<http://www2.sas.com/proceedings/sugi22/ADVTUTOR/PAPER42.PDF>

Author: Ronald Fehd <mailto:Ronald.Fehd@cdc.hhs.gov>
Centers for Disease Control MS-G23 e-mail: RJF2@cdc.gov
4770 Buford Hwy NE
Atlanta GA 30341-3724 bus: 770/488-8221

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.

about the author:		
education:	B.S. Computer Science, U/Hawaii,	1986
	SUGI attendee	since 1989
	SAS-L reader	since 1994
experience:	programmer: 20+ years	
	data manager at CDC, using SAS: 18+ years	
	author: 10+ SUG papers	
SAS-L:	author: 3,000+ messages to SAS-L since 1997	
	Most Valuable SAS-L contributor: 2001, 2003	

Document Production: This paper was typeset in L^AT_EX. For further information about using L^AT_EX to write your SUG paper, consult the SAS-L archives:

<http://www.listserv.uga.edu/cgi-bin/wa?S1=sas-l>
Search for :
The subject is or contains: LaTeX
The author's address : RJF2
Since : 01 June 2003