Journeymen’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.

Audience: intermediate 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
WISH LIST

DATA STRUCTURE

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

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

2. Proc Datasets:
   ```
   Proc Datasets library = SAShelp;
   details nolist
   memtype = data;
   contents
   data = Class;
   quit;
   ```

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

4. Proc SQL, describe:
   ```
   PROC SQL; describe table SAShelp.Class
   ; quit;
   ```

5. Proc SQL, select:
   ```
   PROC SQL; select Name, Type, Length, Label
   from Dictionary.Columns
   where Libname eq 'SASHELP'
   and MemName eq 'CLASS'
   ; 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!

16,600 lines = 330 pages!

```
Month/Year  Frequency  Percent  Cumulative  Cumulative
----------  ---------  --------  -----------  -----------
JAN95      480       2.08     480         2.08
FEB95      480       2.08     960         4.17
```

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

**MACRO ARRAY AND %DO LOOP**

A %do loop in a macro is similar to a data step loop. On listing line 29, 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.

The problems associated with using macro arrays are:

<table>
<thead>
<tr>
<th>log line</th>
<th>statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>%local Item1 ...</td>
</tr>
<tr>
<td>3-5</td>
<td>%Let Item? = ...</td>
</tr>
<tr>
<td>6</td>
<td>%Let Dim_Item = 3;</td>
</tr>
</tbody>
</table>

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.
SQL SELECT INTO :LIST


Note the system-generated automatic macro variable SqlObs, listing line 36, has the upper bound value, 5.

BUILDING THE APPLICATION

MACRO ARRAY AND %DO 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.

This table contains these columns:

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

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

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

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

**SQL SELECT INTO :LIST**

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.
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, lines 95–97, either appends a small listing...

or, lines 99–114, divides the list into high and low sets of values and adds a note indicating that values were removed.
In order for the global macro variables `In_Lib` and `In_Data` ...

to be used in the `select ... from ...` phrase, lines 128–129 and 141–142, each must be in ALL CAPS, lines 117–118.

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:

```sas
%*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:

```sas
%*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.

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 120 lines; three pages, instead of over 300 from Proc Contents: 2 pages
Proc Freq: 330 pages

Fehd [6, sugi30.067] discusses necessary items in a program header.

To receive the latest edition of this program send an e-mail to the author mailto:RJF2@cdc.gov with the subject: request FreqAll
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 for his many contributions to SAS-L, the on-line SAS User Group; he raises the bar.

Suggested Readings

- Carpenter [2] saspress.59224 discusses dynamic programming (list processing) in ch. 9.
- Fehd [8] sasl.225107 posts code to replace macro array with call execute.

BIBLIOGRAPHY


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. R⃝ indicates USA registration.

Author: Ronald Fehd  
mailto:RJF2@cdc.gov
Centers for Disease Control
4770 Buford Hwy NE
Atlanta GA 30341-3724

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 since1997

Document Production: This paper was typeset in LaTeX. For further information about using LaTeX 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