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 will 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 will be a listing which contains:

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

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

1. Proc Contents:

   C-Contents.sas
   Proc Contents data = SAShelp.Class;

2. Proc Datasets:

   C-Datasets.sas
   Proc Datasets library = SAShelp
   details nolist
   memtype = data;
   contents
   data = Class;
   quit;

3. Proc Print:

   C-Print.sas
   PROC Print data = SAShelp.Vcolumn
   (where = ( Libname eq 'SASHELP'
   and MemName eq 'CLASS' ) );

4. Proc SQL, describe:

   C-SQL-describe.sas
   PROC SQL; describe table SAShelp.Class
   ; quit;

5. Proc SQL, select:

   C-SQL-select.sas
   PROC SQL; select Name, Type, Length, Label
   from Dictionary.Columns
   where Libname eq 'SASHELP'
   and MemName eq 'CLASS'
   ; quit;

I will be working with the SQL-select example, which will provide both the data structure list and 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.

PROC Freq data = SAShelp.Prusal2;
   tables _all_;

1800 lines ≈ 36 pages!

My goal is an output of few pages: one for data structure listing and others for frequency listing.
PROC UNIVARIATE

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

```
PROC Univariate data = SAShelp.Class;
  var _numeric_;
```

---

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 will show a macro do loop, illustrate how to use it for a procedure, and examine its programming problems.

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.

```
%Macro Do_Loop;
  %local Item1 Item2 Item3 Dim_Item I;
  %Let Item1 = X1 ;
  %Let Item2 = Y-2;
  %Let Item3 = Z 3;
  %Let Dim_Item = 3;
  %Do I = 1 %to &Dim_Item.;
    %Put Item&I<&&Item&I.>;
  %end;
%mend;
```

The problems associated with using macro arrays are:

- ensure the scope of macro array variables
- allocation of each element
- allocation of the dimension (upper bound)

PROC SQL SELECT INTO

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

MACRO FREQ-LOOP

In this section I will 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 will use Dictionary.Columns to supply the listing of the data structure.

This table contains these columns:

Macro FreqAll has three parameters: libref, data set name and how many extreme values to show.

Note that two macro arrays are created — Name and Type — in lines 29-30, and the scope of their variables is declared in lines 25-26.
The loop begins on line 39 and ends on line 56.

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

41
43
44 (&&Type&I.)
48

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

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, which will show the mode: the values occurring most and least often.

Note that the parameters InLib, InData, and Nobs2View refer to global macro variables.
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.
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:

```sas
%*Let SQLprint = noprint;
```

This select statement, lines 131-138, replaces the macro array and do loop in the FreqAll-Loop program.

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 (.lst) was saved to text (.txt) and the CR/LF removed for printing here; the listing is truncated to save space.

The complete listing from program FreqAll is 95 lines; three pages, instead of 20 from Proc Contents: 2 pages
Proc Freq: 18 pages

---

### FreqAll.sas documentation

/* Name: FreqAll.sas */

**Requirements:** description: Proc Freq of all vars in data set
purpose: provide shorter listing than tables _all_

**Contexts:** program group: data review
program type: routine
SAS type: program with parameters
uses routines: in-program macro %FreqOf

**Specifications:** input: libref
data
max number of high and low obs to view
process: SQL writes macro calls
Proc Freq of each var
save to data
append to report
subset if Nobs greater than Max-N-to-view
output: print report

**Information:** author: Ronald J. Fehd

**Usage Examples:**
%Let In_Lib = Library;
%Let In_Data = MyData;
%Let HiLowToView = 5 ;
%Inc 'FreqAll.sas';

---

To receive the latest edition of this program header send an e-mail to the author with the subject: request FreqAll

---

### SUGGESTED READINGS

Wobus and Gober [Data-Analysis] show data review with procs Summary and Univariate. Abolafia [Data-Check] provides a macro to reproduce Proc DataChk.
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.

REFERENCES


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

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

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. \textcircled{R} 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