Paper 003-2008

SmryEachVar: A Data-Review Routine for All Data Sets in a Libref

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

ABSTRACT

The SAS® %INCLUDE statement is simple, yet powerful. This paper reviews the FreqAll data review program. (See Ronald Fehd, "Using PROC SQL List Processing with Dictionary.Columns to Eliminate Macro DO Loops" in *Proceedings of the SAS Global Forum, 2007.* This program produces a shortened data review report of frequencies — and appropriate statistics for numerics — for each variable in a data set. It provides routines called using call execute and %include to produce the same report for every data set in a libref.

This is another in the Journeymen's Tools series.

Audience data managers, intermediate to advanced users and macro programmers

Keywords attribute, call execute, data review, data structure dynamic programming, formats, includes, list processing, nrstr, routines, testing source2, subroutines

Information Provides utility programs WriteAttrib and WriteValue to repair datasets with missing metadata: formats or labels.

In this paper This paper contains the following topics.

INTRODUCTION

List of Topics

These are the topics discussed in the introduction.

Overview

This paper examines the list processing issues of calling one routine many times, using data set variables as a list of parameters. FreqAll uses SQL to generate macro calls to its subroutine FreqOf. SmryEachVar uses call execute to generate calls to its routines which are parameterized include files.

COMPARE ALGORITHMS

FreqAll

Fehd's FreqAll consisted of two parts: macro FreqOf and the FreqAll list processing routine. The calls to the subroutine macro FreqOf were generated by SQL writing text into a macro variable. The limitation of this algorithm is the program may run out of memory for the macro variable symbol table.
Example macro calls, which were all in one macro variable:

```
1  %FreqOf(Libname = sashelp
2         ,Memname = Class
3         ,Name    = Height
4         ,Type    = N
5         )
```

SmryEachVar

The macro FreqAll.FreqOf has been replaced with a parameterized include file, ProcFreq. The repetition of calls is handled by call execute.
Example parameterized include file calls:

```
1  %Let Libname = sashelp;
2  %Let Memname = Class  ;
3  %Let Name    = Height ;
4  %Let Type    = N      ;
5  %Include Project(ProcFreq);
```

SmryEachVar uses Proc Contents to make the lists of variables and list of data sets. These lists are used by each of the routines, CallXProc, and CallXRpt, to call execute the %include of the subroutines: ProcFreq, ProcSmry, ProcUniv, RptMemName.

Map of Calls

These tables show the calls of routines and subroutines by each module.

FreqAll
Map of Routine and Subroutine Calls

main	routine	subroutines
FreqAll	SQL	MakLists FreqOf RptMemN

SmryEachVar
Map of Routine and Subroutine Calls

main	routines	routines	subroutines
A0Smry	CallXProc	ProcFreq	MakLists DatStruc ProcMode
		ProcSmry	DatStruc
		ProcUniv	
	CallXRpt		RptMemN

EXAMPLE OUTPUT

Overview

Reports The SmryEachVar Report, per Memname, is a listing which contains:

attributes data structure list similar to Proc Contents

summary abbreviated freqency showing the high and low values and the number of levels; for numeric variables: mode, min, mean, max, n and nmiss

Utilities In addition two utilities are provided to repair data sets whose metadata is missing formats or labels:

WriteAttribute provides a text file containing a skeleton attribute statement

WriteValue provides a text file with skeleton proc format value statements

These files can be saved as programs, modified, and used to update data set metadata.

Attributes

The data structure is the primary item for consideration in data review.
There are two other considerations for each variable:

1. how many levels does each variable have?
2. is the variable unique, i.e. does the number of levels equal the number of observations?

The SmryEachVar listing contains the data set name in the title with the number of observations. As a data manager I am concerned to discover whether the data set has both labels and formats. If they are missing, I have to provide them; for this I use the text files from the utilites WriteAttribute and WriteValue.
Example report of data set attributes, see the demonstration file zqCITIDAY-report.txt:

```
SmryEachVar
Report Memname: SASHELP.CITIDAY nobs=1069 nvars=11
Report Memname: MemLabel=Citibase daily indicators: JAN88 FEB92
Report Memname: attributes

Var
Num  Name  Type  Length  Label                        Format  Level  Unique

  1  DATE   N        7    Date of Observation        DATE9.   1069      1
 10  DCD1M  N        8    INT.RATE:1MO CERTIFICATE             388      0
  9  DCP07  N        8    7 DAY COMMERCIAL PAPER               324      0
...
```

Summary

The FreqAll routine provided only the output from Proc Freq. SmryEachVar provides additional information:

1. when a format is present, the formatted value
2. number of levels, and a note if the variable is unique
3. Proc Summary or Univariate information; other information can be added for numerics

---

Summary: output

Report Titles The data set metadata — number of observations, number of variables, and label — are in the titles.

```
SmryEachVar
Report Memname: SASHELP.CITIDAY nobs=1069 nvars=11
Report Memname: MemLabel=Citibase daily indicators: JAN88:FEB92
Report Memname: summary

      Type
Name  Len   Valu C                    formatted  Valu N    N    %    Level
```

Proc Freq Note: this example shows only the two lowest and highest values. The ProcFreq subroutine contains a parameter, Nobs2View — set in the autoexec — which controls how many levels to show.

```
DATE  N.7   .                         01JAN1988  10227.00  1    0.09    1
            .                         04JAN1988  10230.00  1    0.09    2
            .                         04FEB1992  11722.00  1    0.09 1068
            .                         05FEB1992  11723.00  1    0.09 1069
            levels=1069:is.primekey?  .               .    .     .   1069
```

Note that ValuC contains the number of levels. For sashelp.CitiDay, Date is the primary key: the row identifier, and its number of levels is equal to the number of rows.

ProcMode There is not a Proc Mode; see the ProcMode subroutine, which sorts the Proc Freq data set by descending count for this summary. Note that Proc Univariate produces a mode value.

```
            mode                      01JAN1988  10227.00  1    0.09    2
            mode                      01JAN1988  10230.00  1    0.09    3
            mode                      01JAN1988  10231.00  1    0.09    4
```

Proc Summary Note: other statistics may be added.

```
            min                   .          10227.00  .     .       .
            mean                  .          10975.40  .     .       .
            median                .          10975.00  .     .       .
            max                   .          11723.00  .     .       .
            n                     .           1069.00  .   100.00    .
            nmiss                 .              0.00  .     0.00    .
```

Note: Percent is calculated for the statistics N and Nmiss.

REVIEW OF CONCEPTS

Using Includes

The `%Include` statement reads all statements in a file. The option `source2` controls whether the statements are echoed to the log. The default value is `nosource2`. Note that the routines and subroutines check the value of option `source2` in order to self report while testing. The SmryEachVar suite uses the testing algorithms explained in Fehd [4, nesug2007.012].

Functions and Call Routines

These functions are used in the programs.

call execute submits statement for execution in next step; see also %nrstr

cat functions concatenation functions, replaces concatenation operator (!!)

- **cat:** no trim
- **cats:** remove leading and trailing blanks
- **catt:** remove trailing blanks
- **catx:** remove leading and trailing blanks, add separator specified in first argument

%eval evaluate numeric expression, return integer; used to test value of options during testing; see %sysfunc getoption and Fehd [4, nesug2007.012]

link goto named label; code is bracketed by `return;` statements

%nrstr: No Rescan String forces resolution of macro variable assignments and calls in next step; used with call execute

putlog write note to log; eliminates use of `file log;` statement

%sysfunc getoption returns current value of option in all caps

vname returns name of variable as text from array reference

REVIEW OF MAIN MODULE: A0Smry

Overview

This section examines the parts of the main program A0Smry.sas.

1. Parameters
2. Processing
3. Optional Reports

Parameters

The primary parameters are:

Libname: Libref of data sets to review

```
**   1 Prepare SmryEachVar report for:;
%Let Libname = Library;
```

Nobs to View: How many rows in the summary

```
**   2 Show lowest, highest values:;
%Let Nobs2View = 3;
```

Path2Txt: output file prefix, may include folder name

```
**   3 Write Smry report *.txt to folder:;
%Let Path2Txt  = zq;%*here: zq&MemName.*.txt;
```

If you want to write the reports into the same folder containing the data sets then enable these statements by removing the asterisk in column 2:

```
%*Let Path2Txt = ..\sas7b\;%*sibling folder;
 *Libname LibThis "&Path2Txt.";
%*Let Libname   = LibThis;
```

Processing

Input Program MakLists creates a data set which is used as a list of parameters for routines.

```
*input   : Make lists for CallX*;
           %Include Project(MakLists)       ;
```

Process Program CallXproc calls the subroutines ProcFreq, ProcMode, and ProcSmry

```
*process: Call procs freq, mode and summary;
           %Include Project(CallXProc)      ;
```

Output Program CallXrpt calls the reporting subroutine

```
*output : Print summary report, by MemName ;
           %Include Project(CallXRpt)       ;
```

Optional Reports

Additional programs are provided in the suite .zip for the following tasks:

RptNameA: by variable Name; compare that same named variables in different data sets have the same attributes: Type, Length and Label

WriteAttrib: write an attribute statement for the data set; if length needs changing or formats or labels are missing then this file can be used for modifications

WriteValue: write a Proc Format value statement for each variable; this file can be used to prepare formats

---

Report: WriteAttrib

This is an example of the output from WriteAttrib.

```
1  *attrib statement for LIBRARY.CITIDAY;
2  attrib
3     Date      length =       7 format = date9.
4               label  = 'Date of Observation'
5     Dcd1m     length =       8 %*format =  Dcd1m.?;
6               label  = 'INT.RATE:1MO CERTIFICATES OF DEPOSIT, SH'
```

Note that the format is missing for variable Dcd1m and that the `format=` clause is disabled by a macro comment: `%*format = Dcd1m.?;`. If you desire to add formats to the data set then:

1. create a format, using the text file from WriteValue
2. enable the format= clause: remove the macro comment `%*` and closing semicolon `;`
3. apply the attribute statement to update the data structure

---

Report: WriteValue

This is an example of the output from WriteValue. For each variable it contains a skeleton of the value statement — note that the statement is disabled by a macro comment: `%*value DCD1M` — and that it contains the summary information.

```
1  PROC Format library = Library fmtlib;
2  Title2 'LIBRARY.CITIDAY';
3  %Let ValuBlank   = blank;
4  %Let ValuMissing = missing;
5  %*value  DCD1M
6             .  = ""
7             3.9499999993  = ""
8             10.069999997  = ""
9           %*min         3.9499999993 ;
10          %*max         10.069999997 ;
11          %*other = "&ValuMissing.";
12            ;
```

This report can be saved as a program, and modified for each format wanted.

HOW TO GET AND USE THESE PROGRAMS

Overview

In order to run these programs for a project, follow these steps:

1. Create Project Folders
2. Download the Suite Zip File
3. Set Up for the Demonstration
4. Run the Demonstration Program
5. Set Up for Use on a Library
6. Modifications and Testing

Create Project Folders

Create the following folders for the SmryEachVar project:

contains	recommended name
root	SmryEachVar
sas programs	sas
sas data sets	sas7b
temporary sas data sets	sas7bWork
text files	txt

When you are finished the directory structure might look like this:

```
C:\SAS\projects
C:\SAS\projects\SmryEachVar
C:\SAS\projects\SmryEachVar\sas
C:\SAS\projects\SmryEachVar\sas7b
C:\SAS\projects\SmryEachVar\sas7bWork
C:\SAS\projects\SmryEachVar\txt
```

Download the Suite Zip File

To get the code examples in this paper search www.sascommunity.org for Summarize Memnames in Libname.

1. download the .zip file
2. extract files to the project folder for sas programs

Set Up for the Demonstration

Open the sas programs folder and perform the following steps:

autoexec.sas review the title, filename and libname statements — these are for Windows — and ensure they conform to the operating system directory specifications

```
*options  source2;*enables self reporting when testing;
*  name: autoexec.sas;
Title    'SmryEachVar: Data Review for All Data Sets in Libref';
Filename Project '.'       ;
Libname  Library '..\sas7b';
Libname  LibWork (Work)    ;
```

The use of `options source2;` for testing is explained in Fehd [4, nesug2007.012].

CopySashelpToLibrary.sas submit this program to copy a few SAShelp data sets to the library

```
*name   : Copy-sashelp-to-library.sas;
*purpose: provide data sets in Library;
*         for demonstration and testing;
PROC Copy in      = sashelp
          out     = Library
          memtype = data;
          select    CitiDay CitiYr Class;
```

Run the Demonstration Program

Submitting the A0Smry program will create a set of text files for each member in the libref Library.

filename	usage	notes
zqCITIDAY-attrib.txt	example attribute statement	
zqCITIDAY-ProcFormatValues.txt	example value statement	
zqCITIDAY-report.txt	SmryEachVar report	see example above

Set Up for Use on Another Library

Make the following changes

autoexec.sas change the directory specification of `libname Library` to the directory of the data sets that you wish to report on

A0Smry.sas change the value of macro variable `Path2Txt` to the directory where you want the text reports written; this may be a full directory specification or a (Windows) sibling folder, as shown here:

Example: `%Let Path2Txt = ..\txt\;`

Modifications and Testing

A full set of test files for each routine and subroutine is provided in the suite .zip. These programs were developed using the testing algorithm explained in Fehd [4, nesug2007.012].

ROUTINES

List of Programs

This is the list of routines in this section.

CallXProc: CALL EXECUTE OF ROUTINES

Overview

This is the header record of this program.

CallXProc.sas

```
1  * name        : CallXProc.sas;
2  * description: Call Execute: Procs Freq, Mode, Summary, Univariate;
3  * purpose     : list processing of subroutine;
4
5  * parameters ;
6  * input       : ListNames;
7  * process     : 1.  for each Variable: call procs;
8    *             2.  add var Unique to ListNames;
9  * output      : 1.  from subroutines: ListSmry;
10   *             2.1 sort: out = ListNamesByName;
11   *             2.2 sort: out = ListSmryByName;
```

This program contains the following steps:

1. Data Structure
2. Make Statement
3. Call Subroutines
4. Link ExecStmnt
5. Add Information

Data Structure

Output from this data step is done by call execute so no output data set name is needed.

```
13 DATA   _Null_;
14 attrib Stmnt length = $132
15        Vname length = $ 32;
16 array  Mvar(*) $32 Libname MemName Name Type Format;
17
18 retain Testing %eval(0
19             or %sysfunc(getoption(Source2))
20                                 eq SOURCE2  );
```

Note `options source2;` can be set in autoexec; the use of the variable Testing is explained in Fehd [4, nesug2007.012].

Make Statement

For each character variable in the array make a global macro variable assignment statement.

```
21 do until(EndoFile);
22    set   LibWork.ListNames end = EndoFile;
23    %* make macro variable assignment statement:;
24    %* Stmnt = "%let Mvar = value";
25    do I = 1 to dim(Mvar);
26       call   vname(Mvar(I)    ,Vname);
27       Stmnt = catx(' ','%let ',Vname,'='
28                   ,Mvar(I)          ,';');
29       link ExecStmnt;
30       end;
```

Call Subroutines

This section calls the various summarization procedures: ProcFreq and, for numerics, ProcMode plus ProcSmry or ProcUniv. Note that ProcMode provides mode(s) from the freq data set, while ProcUniv calculates the mode.

```
31    Stmnt = cat('%Include Project(ProcFreq);');
32    link ExecStmnt;
33    if Type eq 'N' then do;
34       /** Choice 1: Mode from Freq and Summary **/
35       Stmnt = cat('%Include Project(ProcMode);');
36       link ExecStmnt;
37       Stmnt = cat('%Include Project(ProcSmry);');
38       /** Choice 2: Univariate with mode ********
39       Stmnt = cat('%Include Project(ProcUniv);');
40       /*****************************************/
41       link ExecStmnt;
42       end; %*if Type eq N;
```

Note: to swap the ProcMode plus ProcSmry with ProcUniv remove the slash ending the comment in line 34, and add a slash to the end of line 38.

Link ExecStmnt

This labeled section enables the program to self report when option source2 is true. `Options source2;` can be set in the autoexec. It writes the value of the variable Stmnt to the log. See Data Structure above for the allocation of the variable Testing.

```
45 return;
46 ExecStmnt: if   Testing then putlog    Stmnt=;
47            call execute(cats('%nrstr(',Stmnt,')'));
48 return;
49 run;       %*calls executed in this step;
```

Add Information

This section adds variable Unique to the report data set ListNames.

```
51 Data  LibWork.ListNames(drop = Count);
52 do    until(Endofile);
53 merge LibWork.ListSmry
54       (keep  = Libname MemName Name Count Level
55        where = (Count = . and Level))
56       LibWork.ListNames end = EndoFile;
57 by    Libname MemName Name;
58 Unique = (NobsData eq Level);
59 if    first.Name then output;
60 end;
61 stop;
```

CallXRpt: CALL EXECUTE REPORTING SUBROUTINE

Overview

This is the header record of this program.

```
1 * name        : CallXRpt.sas;
2 * description: Call Execute: Report MemName;
3 * purpose     : list processing of subroutine;
4
5 * parameters : ;
6 * input       : ListMemNames;
7 * process     : for each MemName: call RptMemN;
8 * output      : by subroutine;
```

This program uses the same algorithm as CallXProc to call the subroutine RptMemN.

ProcFreq: SAVE DATA SET

Overview

The routine ProcFreq is called by CallXProc. It is a parameterized include file modified from the FreqAll macro FreqOf. Its parameters are the global macro variables: LibName, MemName, Name, Type and Format. It calls the subroutine DatStruc.
This program contains the following steps:

1. Primary Process
2. Standardize Data Structure
3. Read and Output
4. Make Information
5. Output
6. Append

Primary Process

Save the proc freq output data set and rename the variable to the standardized names: ValuC or ValuN.

```
55 PROC Freq data   = &LibName..&MemName.
56           order  = &Order.;
57           format   &Name.;%*remove formatting;
58           tables   &Name.
59                  / list missing noprint
60              out = Freq(rename =
61                         (&Name. = Valu&Type.));
```

Note: the output data set Freq is used by the subroutine ProcMode.

Standardize Data Structure

Call subroutine DatStruc.

```
63 %Include Project(DatStruc);
64
```

Read and Output

Read the data set and output only the lowest and highest rows.

```
65  do RowNmbr = 1 to    NobsFreq;
66     set  Freq nobs  = NobsFreq
67               point = RowNmbr;
68               %* case 1: output all rows;
69     if   NobsFreq le %eval(2 * &Nobs2View. + 2)
70          then link Assigns;
71     else do;  %* case 2: lo and hi &Nobs2View. rows;
72          if      RowNmbr le            &Nobs2View.
73               or RowNmbr ge NobsFreq - &Nobs2View.
74               then link Assigns;
75          else if RowNmbr gt            &Nobs2View.
76               then do;
77               RowNmbr  = NobsFreq - &Nobs2View.;
78               Level    = RowNmbr;
79               end;  %*else if RowNmbr gt &Nobs2View.;
80          end;       %*else do: case 2;
81     end;            %*do RowNmbr;
```

Note: Compare this single pass algorithm to FreqAll.FreqOf macro.

Make Information

After the output of the lowest and highest rows make the information row, which contains the number of rows (Levels) of the proc freq data set and, if the variable is unique, adds a note saying that the variable is unique: `'is.primekey?'`.

```
83  %* make information row;
84  ValuC    = cats('levels=',NobsFreq);
85  if NobsData eq NobsFreq then
86     ValuC = cats(ValuC,':is.primekey?');
87  Format   = ' ';   ValuF   = '.';   ValuN = . ;
88  Count    =  . ;   Percent =  . ;   Level = . ;
89  output;
```

Output

Add the formatted value.

```
91  return;
92  Assigns: Level+ +1;
93           if Format  ne ' ' then do;
94              if Type eq 'C' then ValuF = putC(ValuC,Format);
95              else                ValuF = putN(ValuN,Format);
96              end;
97           output;
98  return;
99  stop;
100 run;     %*execute calls here;
```

Append

The freq output is appended to the report data set.

```
102 PROC Append base = LibWork.ListSmry
103             data = CommonDataStructure;
```

ProcSmry: SAVE DATA SET

Overview

This subroutine is called by CallXProc.Its parameters are the same as ProcFreq: LibName, MemName, Name.
This program contains the following steps:

1. Proc Summary
2. Basic Statistics
3. Extra Statistics
4. Transpose
5. Standardize Data Structure
6. Read and Output
7. Append

Proc Summary

```
3  PROC Summary data    = &LibName..&MemName.;
4               var       &Name.;
5               output
6                  out = Summary
7             (  drop =  _Type_  _Freq_)
```

Basic Statistics

These are the basic statistics useful in understanding the distribution of a numeric variable.

```
12    min      (&Name.) = min      %*;
13    mean     (&Name.) = mean     %*;
14    median   (&Name.) = median   %*p50;
15    max      (&Name.) = max      %*;
16    n        (&Name.) = n        %*;
17 %*;nmiss    (&Name.) = nmiss    %*;
```

Extra Statistics

Other statistics may be enabled by adding a semicolon in column 3 which closes the macro comment and enables the statement. Refer to line 17 for Nmiss, above, for an example.

```
18 %* p1       (&Name.) = p01      %*;
19 %* p5       (&Name.) = p05      %*;
20 %* p10      (&Name.) = p10      %*;
21 %* p25      (&Name.) = p25      %*q1;
22 %* p50      (&Name.) = p50      %*median;
```

Transpose

The Proc Summary output data set is one row; the Proc Transpose changes the data structure to one row per statistic.

```
39 PROC Transpose data   = Summary
40                out    = SummaryT
41               (keep   = Col1         ValuC
42                rename =(Col1 = ValuN      ))
43                name   =              ValuC  ;
```

Standardize Data Structure

Call subroutine DatStruc.

```
45 %Include Project(DatStruc);
46
```

Read

Note calculations of percent for N and Nmiss.

```
47 do until(             EndoFile);
48    set  SummaryT end = EndoFile;
49    if   ValuC in ('n','nmiss') then
50         Percent = 100*(ValuN/NobsData);
51    else Percent = .;
52    output;
53    end;    %*do until EndoFile;
54 stop;
55 run;
```

Append

```
57 PROC Append base = LibWork.ListSmry
58              data = CommonDataStructure;
```

Data set CommonDataStructure is named by subroutine DatStruc.

ProcUniv: SAVE DATA SET

Overview

This subroutine is called by CallXProc.Its parameters are the same as ProcFreq: LibName, MemName, and Name.
This program contains the following steps:

1. Proc Univariate
2. Basic Statistics
3. Extra Statistics — Note the following steps are the same as ProcSmry
4. Transpose
5. Standardize Data Structure
6. Read and Output
7. Append

Proc Univariate

```
3 PROC Univariate data   = &LibName..&MemName.
4                          noprint;
5                 var      &Name.;
6                 output
7                    out = Univariate
```

Basic Statistics

These are the basic statistics useful in understanding the distribution of a numeric variable.

```
12    max              = max      %*;
13    mean             = mean     %*;
14    median           = median   %*p50;
15    min              = min      %*;
16    mode             = mode     %*;
17    n                = n        %*;
18    nmiss            = nmiss    %*;
19 %*;nobs             = nobs     %*;
```

Extra Statistics

Other statistics may be enabled by adding a semicolon in column 3 which closes the macro comment and enables the statement. Refer to the line for Nobs, above, for an example.

```
20 %* range            = range    %*;
21 %* skewness         = skewness %*;
22 %* std              = std      %*;
```

Other Steps

. . . are the same as ProcSmry.

SUBROUTINES

List of Programs

This is the list of subroutines in this section.

DatStruc: COMMON DATA STRUCTURE

Overview

The purpose of the DatStruc subroutine is to standardize the data structure of each of the procedure outputs. This program is called by ProcFreq, ProcSmry and ProcUniv.
This program contains the following steps:

1. Output Data Set Name
2. Read Identifiers
3. Set Length of ValuC
4. Allocate Data Structure
5. Initialize Values
6. Self Report When Testing

Output Data Set Name

Each of the calling routines gets the data set CommonDataStructure.

```
DATA   CommonDataStructure
      (label = 'attrib for Procs Freq Smry Univ'
       keep  = LibName MemName Name  TypeLen
               ValuC   ValuF   ValuN
               Count   Percent Level Label);
```

Read Identifiers

Read one row from the list processing data set which contains the identifiers and retain all variables.

```
set   LibWork.ListNames(where = (
           upcase(LibName) eq "%upcase(&LibName.)"
       and upcase(MemName) eq "%upcase(&MemName.)"
       and    upcase(Name) eq    "%upcase(&Name.)"));
retain _all_;
```

Set Length of ValuC

Set maximum length of the variable ValuC.

```
%Let   LenValuC =
       %length(levels=123,456,789:is.primekey?);
```

Allocate Data Structure

Use the attribute statement to declare the common data structure.

```
attrib TypeLen length = $ %length(C.32767)
                label =   'Type Len'
       ValuC   length = $ &LenValuC.
                label =   'Valu C'
       ValuF   length = $ &LenValuC.
                label =   'formatted'
       ValuN   length =   8        format = best.
                label =   'Valu N'
       Count   length =   4        format = comma.
                label =   'N'
       Percent length =   8        format = 6.2
                label =   '-%-'
       Level   length =   4
       Testing length =   4;
```

Initialize Values

Note: the Proc Freq output data set supplies either ValuC or ValuN. This retaining ensures the append works correctly.

```
retain Testing %eval(0
                or %sysfunc(getoption(Source2))
                                    eq SOURCE2  )
       ValuC   '.'    ValuF   '.'   ValuN .
       Count    .     Percent  .    Level .
       TypeLen '?.9'  ;
       TypeLen = cats(upcase(substr(Type,1,1))
                     ,'.',Length);
```

The use of the variable Testing is explained in Fehd [4, nesug2007.012].

Self Report When Testing

Conditionally write test messages to log.

```
51 if Testing then do;
52    put _all_;
53    call execute('%nrstr(%put _global_;)');
54    end;
```

MakLists: MAKE DATA SET FOR LIST PROCESSING

Overview

Subroutine MakLists is called by the main module A0Smry; it prepares the list processing data set used by both CallXProc and CallXRpt. This program has the following steps:

1. Save Proc Contents output
2. Split Proc Contents output
3. Standardize data structure
4. Read data set
5. Recode Contents.Type
6. Assemble Format
7. Output

Save Proc Contents Output

In FreqAll I used Proc Sql; Phil Mason noted in a private conversation that Proc Contents is faster.

```
12 PROC Contents data   = &Libname.._all_
13                        noprint
14              out     = LibWork.ListNames
15             (where   = (MemType eq 'DATA')
16              rename  = (Nobs = NobsData)   );
```

The variable Nobs is renamed to differentiate it from NobsFreq, the number of observations of the Proc Freq data set.

Split Proc Contents Output

The Proc Contents data set contains more variables than I need so I split it into two data sets similar to proc SQL's Dictionary.Columns and Dictionary.Tables. Note that NobsData is saved in both data sets. ListNames is the list processing data set; ListMemnames is the first of the final report data sets.

```
18 DATA   LibWork.ListNames
19           (keep = LibName  MemName
20                   Name     Type     Length
21                   Label    Format
22                   Varnum   NobsData        )
23        LibWork.ListMemnames
24           (keep = LibName  MemName MemLabel
25                   NobsData Nvars           );
```

Standardize Data Structure

The attribute statement declares the order of the variables in the data structure.

```
attrib LibName                label = 'LibName'
       MemName                label = 'MemName'
       Name                   label = 'Name'
       Type     length = $ 1  label = 'Type'
       Length                 label = 'Length'
       Label                  label = 'Label'
       %*$49==sql.dictionary.columns.format length;
       Format   length = $49  label = 'Format'
       NobsData length =   4  label = 'Nobs Data'
       Nvars    length =   4  label = 'N vars'
       VarNum                 label = 'Var Num'    ;
retain Nvars 0;
```

Read Data Set

```
do until(EndoFile);
   set LibWork.ListNames
      (rename = (Type = TypeN))
       end    =  EndoFile;
   by  Libname MemName;
   if  first.MemName then Nvars = 0;
   Nvars+ +1;
   if  last.MemName  then
       output LibWork.ListMemNames;
```

Note data set ListMemNames is like SQL's Dictionary.Tables.

Recode Contents.Type

Recode the Contents.Type numeric variable into a character variable.

```
   select(TypeN);%*convert to SQL.Dict.Columns.Type;
       when(1) Type = 'N';
       when(2) Type = 'C';
       otherwise;
       end;
```

Assemble Format

Assemble the format from its parts: Format, FormatL and FormatD.

```
   if Format ne ' ' then do;
      if FormatL then Format = cats(Format,FormatL
                                    ,'.');
      else            Format = cats(Format,'.');
      if FormatD then Format = cats(Format,FormatD);
      end;
```

Output

This output statement is for the list of variable names, which is like SQL's Dictionary.Columns.

```
   output LibWork.ListNames;
   end;   %* do until(EndoFile);
```

ProcMode: SORT FREQ DATA SET

Overview

This subroutine is an optional procedure for numeric variables; it can be disabled in CallXProc.
Variable values are added to the Proc Freq output data set, which is then sorted by descending count. Only the most frequently occurring rows are appended to the summary report data set.

Add Variable Values

```
6  DATA   Freq;
7  set    CommonDataStructure(obs=1);
8  retain _all_; %*identifiers;
9  retain Level  0 ;
10
11 do     until(EndoFile);
12         set Freq  end = EndoFile;
13         Level+ +1;
14         ValuC  = 'mode';
15         output;
16         end;
17 stop;
```

Sort

```
19 PROC Sort data   = Freq;
20           by       descending Count;
```

Append

```
22 PROC Append base = LibWork.ListSmry
23             data = Freq
24            (obs  = &Nobs2View.);
25 run;
```

RptMemN: REPORT BY MEMNAME

Overview

RptMemN is called by CallXRpt. It writes one summary report for each data set to a text file. This program contains the following steps:

1. Overview
2. Description
3. Open Output Text File
4. Put Information in Titles
5. Print Attributes
6. Print Summary
7. Close Output

Description

```
* name        : RptMemN.sas;
* description: Report of MemName     ;
*              attributes and summary;
* purpose     : write summary report to text file;

* parameters : global: Libname, Memname;
*              local : ReportName;
* input       : ListMemNames ListNames ListSmry;
* process     : get Nobs, Nvars, MemLabel for titles;
*              print;
* output      : to text file;
```

Open Output Text File

Note: the macro variable Path2Txt is set in main module A0Smry.

```
Proc PrintTo   new
     print   = "&Path2Txt.&MemName.-report.txt";
```

Put Information in Titles

Read one row from the list processing data set ListMemNames, which contains the items for the title statements.

```
%Let NobsData   = 0;
%Let Nvars      = 0;
%Let ReportName = Report-Memname;
PROC SQL noprint;
         select       NobsData,  Nvars,  MemLabel
                into :NobsData, :Nvars, :MemLabel
         from   LibWork.ListMemnames
         where  Libname eq "%upcase(&Libname.)"
           and  Memname eq "%upcase(&Memname.)";
         quit;
%*note: reassignment == remove leading blanks;
%Let NobsData = &NobsData.;
%Let Nvars    = &Nvars.;

Title2 "&ReportName.: &Libname..&Memname."
                           " nobs=&NobsData."
                           " nvars=&Nvars."  ;
Title3 "&ReportName.: MemLabel=%unquote(&MemLabel.)";
```

Note: MemLabel is unquoted because it may contain either special characters, such as ampersands or percent signs, or unmatched quotes.

Print Attributes

```
PROC Print data  =  LibWork.ListNames
          (where = (Libname eq "%upcase(&Libname.)"
             and    Memname eq "%upcase(&Memname.)"
          )        )noobs;
           label    Level = 'Levels';
           var      VarNum Name Type Length Label
                    Format Level Unique;
           Title4  "&ReportName.: attributes";
```

Print Summary

```
43
44 PROC Print data  =  LibWork.ListSmry
45            (where = (Libname eq "%upcase(&Libname.)"
46               and    Memname eq "%upcase(&Memname.")
47            )        )label noobs;
48             var      ValuC ValuF ValuN Count Percent
49                      Level; %* Label;
50             format   ValuN;
51             by       Name notsorted TypeLen;
52             id       Name           TypeLen;
53             Title4  "&ReportName.: summary";
```

Close Output

```
55 Proc PrintTo;
56 run;
```

CONCLUSION

Summary

FreqAll	The data review utility program FreqAll provides a short data set summary using Proc Freq.
SmryEachVar	The data review utility suite SmryEachVar provides more information, especially for numerics.
Call Execute	This paper shows that call execute is a powerful method for list processing.
%Includes	Doing list processing with call execute of %Includes can eliminate the use of macros. This yields clearer code.

Suggested Reading

Proc DataCheck Abolafia [1, sugi22.229] provides a macro to replace the SUGI Supplemental Library Proc DataChk which summarizes numerics.

Fehd: FreqAll Fehd [3, sgf2007.028] wrote the original proc freq code upon which this paper is based.

Fehd: update After presenting FreqAll in fall 2007 Fehd wrote to the SAS-L listserv to provide an update: Fehd [2, sasl.225107] which used call execute with a list from sashelp.vcolumn.

Call Execute and %nrstr Fehd and Carpenter [5, sgf2007.113] demonstrate the timing of the error of using call execute of macros without the macro function %nrstr.

Testing Fehd [4, nesug2007.012]shows the use of options echoauto, mprint, source2, and verbose while testing modules, routines and subroutines.

Acknowledgements My section chair at NESUG, **Rob Russell**, suggested that I develop FreqAll for every MemName; **Toby Dunn** requested formatted values and reviewed early drafts; **Peter Flom** and **David Cassell** requested nvars, Proc Summary median, n and nmiss; **Phil Mason** noted that Proc Contents was faster than Proc SQL Dictionary.Columns; and the usual suspects on SAS-L provided much other encouragement. My colleagues at CDC **Susan Katz** and **Elizabeth Perez** reviewed output and provided suggestions for renaming output text files.

BIBLIOGRAPHY

[1] Jeffrey M. Abolafia. Proc DataChk revisited: The DataChk macro. In *Proceedings of the 22nd Annual SAS Users Group International Conference*, 1997. URL `http://www2.sas.com/proceedings/sugi22/POSTERS/PAPER229.PDF`. Posters, 6 pp.; macro to replace SUGI Supplemental Library Proc DataChk; provides summary of numeric variables.

[2] Ronald Fehd. Re: tip: macro FreqAllVars. In *Archives of the SAS-L listserve*, 4 Jan. 2007. URL `http://www.listserv.uga.edu/cgi-bin/wa?A2=ind0701A&L=sas-l&P=R14003`. Updated algorithm: replace macro array with call execute.

[3] Ronald J. Fehd. Journeymen's tools: Data review macro FreqAll — using Proc SQL list processing with Dictionary.Columns to eliminate macro do loops. In *Proceedings of the SAS Global Forum*, 2007. URL `http://www2.sas.com/proceedings/forum2007/028-2007.pdf`. Coder's Corner, 10 pp.; attributes, dictionary.columns, metadata, proc append, proc freq, proc sql, program header; bibliography.

[4] Ronald J. Fehd. Writing testing-aware programs that self-report when testing options are true. In *Proceedings of the NorthEast SAS User Group Conference*, 2007. URL `http://www.nesug.info/Proceedings/nesug07/cc/cc12.pdf`. Coders' Corner, 20 pp.; topics: using options echoauto, mprint, source2 and verbose to test modules, routines or subroutines.

[5] Ronald J. Fehd and Art Carpenter. List processing basics: Creating and using lists of macro variables. In *Proceedings of the SAS Global Forum*, 2007. URL `http://www2.sas.com/proceedings/forum2007/113-2007.pdf`. Hands On Workshop, 20 pp.; comparison of methods: making and iterating macro arrays, scanning macro variable, writing calls to macro variable, write to file then include, call execute; 11 examples, bibliography.

To get the code examples in this paper search www.sascommunity.org for Summarize Memnames in Libname.

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: 12+ SUG papers	
SAS-L:	author: 4,000+ messages to SAS-L since1997	
	Most Valuable SAS-L contributor:	2001, 2003

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
```