Paper A05

FreqLibname: A Data Review Routine For All Memnames in a Libname

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

ABSTRACT

The SAS® include statement is simple, yet powerful. This paper reviews Fehd [3, sgf2007.028] FreqAll data review program which produces a shortened data review report of frequencies of 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 call execute, data review, dynamic programming, includes, list processing, nrstr, routines, source2, subroutines

---

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. FreqLibname 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:

```
%FreqOf(Libname = sashelp
       ,Memname = Class
       ,Name    = Height
       ,Type    = N
       )
```

FreqLibname

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:

```
%Let Libname = sashelp;
%Let Memname = Class  ;
%Let Name    = Height ;
%Let Type    = N      ;
%Include Project(ProcFreq);
```

FreqLibname 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, 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		MakLists
	SQL	FreqOf
		RptMemN

FreqLibname
Map of Routine and Subroutine Calls

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

EXAMPLE OUTPUT

Overview

The FreqLibname 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

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 FreqLibname 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.
Example report of data set attributes, see the demonstration file zqCITIDAYreport.txt:

```
SmryLibname
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. FreqLibname 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 information; note: other Proc Summary information can be added

Report Titles

```
SmryLibname
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, 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
```

ProcMode There is not a Proc Mode; see the ProcMode subroutine.

```
            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 opens and 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.

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

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 reports

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

LibWork: Libref of list processing data sets

```
**   2 Store Smry* data sets in libref:;
%Let LibWork  = Work;
```

Path2Txt: output file prefix, may include folder name

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

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

---

HOW TO GET AND USE THESE PROGRAMS

Overview

In order to run this use this program for your project, you need to do the following 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 Your Library
6. Modifications and Testing

Create Project Folders

Create the following folders for your FreqLibname project:

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

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

```
C:\SASprojects
C:\SASprojects\FreqLibname
C:\SASprojects\FreqLibname\sas
C:\SASprojects\FreqLibname\sas7b
C:\SASprojects\FreqLibname\sas7bWork
C:\SASprojects\FreqLibname\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 your project folder for sas programs

Set Up for the Demonstration

Open the sas programs folder and perform the following steps:

SASv9.cfg customize the configuration file for your project folder

rename file `SASv9copy.cfg` to `SASv9.cfg`

edit `SASv9.cfg`
change the value of `SASinitialFolder` from
`SASinitialFolder ’C:\SASprojects\SmryLibname\sas’`
to the name of your project folder:
e.g.: `SASinitialFolder ’C:\MyProjects\SmryLibname\sas’`

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

```
*  name: autoexec.sas;
Title   ’SmryLibname: Summary of each Memname in Libname’;
Filename Project ’.’           ;
Libname  Library ’..\sas7b’    ;
Libname  LibWork ’..\sas7bWork’;
```

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.

- zqCITIDAYreport.txt
- zqCITIYRreport.txt
- zqCLASSreport.txt

Set Up for Use on Your Library

Make the following changes

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

A0Smry.sas change the value of macro variable `Path2Txt` to the directory where you want your text reports written; this may be a full directory specification or a (Windows) sibling folder
Example: `%Let Path2Txt  = ..\txt\;`

Modifications and Testing

A full set of test files for each routine and subroutine is provided in the suite .zip.

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`

```
* name       : CallXProc.sas;
* description: Call Execute: Procs Freq, Mode, Smry;
* purpose    : list processing of subroutine;

* parameters ;
* input      : ListNames;
* process    : 1.  for each Variable: call procs;
  *            2.  add var Unique to ListNames;
* output     : 1.  from subroutines: ListSmry;
  *            2.1 sort: out = ListNamesByName;
  *            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.

```
DATA    _Null_;
attrib Stmnt length = $132
       Vname length = $ 32;
array  Mvar(*) $32 Libname MemName Name Type Format;

retain Testing %eval(0
            or %sysfunc(getoption(Source2))
                               eq SOURCE2  );
```

Make Statement

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

```
do until(EndoFile);
   set   &LibWork..ListNames end = EndoFile;
   %* make macro variable assignment statement:;
   %* Stmnt = "%let Mvar = value";
   do I = 1 to dim(Mvar);
      call   vname(Mvar(I)    ,Vname);
      Stmnt = catx(’ ’,’%let ’,Vname,’=’
                  ,Mvar(I)          ,’;’);
      link ExecStmnt;
      end;
```

Call Subroutines

This section calls the various summarization procedures: ProcFreq and, for numerics, ProcMode and ProcSmry.

```
   Stmnt = cat(’%Include Project(ProcFreq);’);
   link ExecStmnt;
   if Type eq ’N’ then do;
      Stmnt = cat(’%Include Project(ProcMode);’);
      link ExecStmnt;
      Stmnt = cat(’%Include Project(ProcSmry);’);
      link ExecStmnt;
      end; %*if Type eq N;
   end;    %*do until(EndoFile);
stop;
```

Link ExecStmnt

This labeled section enables the program to self report when option source2 is true. It writes the value of the variable Stmnt to the log. See above for the allocation of the variable Testing in the data structure.

```
return;
ExecStmnt: if   Testing then putlog    Stmnt=;
           call execute(cats(’%nrstr(’,Stmnt,’)’));
return;
run;       %*calls executed in this step;
```

Add Information

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

```
Data  &LibWork..ListNames(drop = Count);
do     until(Endofile);
merge &LibWork..ListSmry
      (keep  = Libname MemName Name Count Level
       where = (Count = . and Level))
      &LibWork..ListNames end = EndoFile;
by     Libname MemName Name;
Unique = (NobsData eq Level);
if     first.Name then output;
end;
stop;
```

CallXRpt: CALL EXECUTE REPORTING SUBROUTINE

Overview

This is the header record of this program.

```
* name       : CallXRpt.sas;
* description: Call Execute: Report MemName;
* purpose    : list processing of subroutine;

* parameters : ;
* input      : ListMemNames;
* process    : 1. for each MemName: call RptMemN;
* 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. Internal Parameters
2. Primary Process
3. Standardize Data Structure
4. Read and Output
5. Make Information
6. Output
7. Append

Internal Parameters

ProcFreq contains two macro variables which determine its output:

1. Nobs2View: the number of rows of the highest and lowest values
2. Order: which controls the proc freq output data set order; note: see also ProcMode.

```
%Let Nobs2View = 3;   %* show how many rows?;

%Let Order = internal;%*default: hi and low values;
%* for mode use:        replaced by ProcMode;
%*Let Order = freq;   %*descending count;
```

Primary Process

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

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

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

Standardize Data Structure

Call subroutine DatStruc.

```
%Include Project(DatStruc);

```

Read and Output

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

```
do RowNmbr = 1 to    NobsFreq;
   set  Freq nobs  = NobsFreq
             point = RowNmbr;
             %* case 1: output all rows;
   if   NobsFreq le %eval(2 * &Nobs2View. + 2)
        then link Assigns;
   else do;  %* case 2: lo and hi &Nobs2View. rows;
        if      RowNmbr le            &Nobs2View.
             or RowNmbr ge NobsFreq - &Nobs2View.
             then link Assigns;
        else if RowNmbr gt            &Nobs2View.
             then do;
             RowNmbr  = NobsFreq - &Nobs2View.;
             Level    = RowNmbr;
             end;  %*else if RowNmbr gt &Nobs2View.;
        end;      %*else do: case 2;
   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?’.

```
%* make information row;
ValuC    = cats(’levels=’,NobsFreq);
if NobsData eq NobsFreq then
   ValuC = cats(ValuC,’:is.primekey?’);
Format   = ’ ’;   ValuF   = ’.’;   ValuN =  . ;
Count    =  . ;   Percent =  . ;   Level = Level -1;
link     Assigns;

```

Output Add the formatted value.

```
return;
Assigns: Level+ +1;
         if Format  ne ’ ’ then do;
            if Type eq ’C’ then ValuF = putC(ValuC,Format);
            else                ValuF = putN(ValuN,Format);
            end;
         output;
return;
stop;
run;     %*execute calls here;
```

Append The freq output is appended to the report data set.

```
PROC Append base = &LibWork..ListSmry
            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

```
PROC Summary data   = &LibName..&MemName.;
             var      &Name.;
             output
                out = Summary
            (  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    output;
52    end;    %*do until EndoFile;
53 stop;
54 run;
```

Append

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

SUBROUTINES

List of Programs

This is the list of subroutines in this section.

DatStruc: COMMON DATA STRUCTURE

Overview

The purpose of DatStruc is to standardize the data structure of each of the procedure outputs. This subroutine is called by ProcFreq and ProcSmry.
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 ProcFreq and ProcSmry’
       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);
```

Self Report When Testing

Conditionally write test messages to log.

```
if Testing then do;
   put _all_;
   call execute(’%nrstr(%put _global_;)’);
   end;
```

MakLists: MAKE DATA SET FOR LIST PROCESSING

Overview

MakLists.sas is called by the main module A0Smry; it prepares the list processing data set used by both CallXProc and CallXRpt. In FreqAll I used Proc Sql; Phil Mason noted in a private conversation that Proc Contents is faster. 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

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

```
PROC Contents data   = &Libname.._all_
                       noprint
              out    = &LibWork..ListNames
             (where  = (MemType eq ’DATA’)
              rename = (Nobs = NobsData)  );
```

Split Proc Contents Output

The Proc Contents data set contains more variables than I need so I split it. 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.

```
DATA   &LibWork..ListNames
          (keep = LibName  MemName
                  Name     Type     Length
                  Label    Format
                  Varnum   NobsData        )
       &LibWork..ListMemnames
          (keep = LibName  MemName MemLabel
                  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;
```

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

ProcMode: SORT FREQ DATA SET

Overview

This 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

```
DATA   Freq;
set    CommonDataStructure(obs=1);
retain _all_; %*identifiers;
retain Level  0 ;

do     until(EndoFile);
       set Freq  end = EndoFile;
       Level+ +1;
       ValuC  = ’mode’;
       output;
       end;
stop;
```

Sort

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

Append

```
PROC Append base = &LibWork..ListSmry
            data = Freq
           (obs  = &Nobs2View.);
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; %*Npos;
           Title4  "&ReportName.: attributes";
```

Print Summary

```
PROC Print data  = &LibWork..ListSmry
          (where = (Libname eq "%upcase(&Libname.)"
             and    Memname eq "%upcase(&Memname.")
          )        )label noobs;
           var      ValuC ValuF ValuN Count Percent
                    Level; %* Label;
           format   ValuN;
           by       Name notsorted TypeLen;
           id       Name           TypeLen;
           Title4  "&ReportName.: summary";
```

Close Output

```
Proc PrintTo;
run;
```

CONCLUSION

Summary

FreqAll	The data review utility program FreqAll provides a short data set summary using Proc Freq.
FreqLibname	The data review utility suite FreqLibname 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 [4, sgf2007.113] demonstrate the timing of the error of using call execute of macros without the macro function %nrstr.

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 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
```