Paper 028-2007

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

Contents

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 freqency 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:

C-Contents.sas

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

2. Proc Datasets:

C-Datasets.sas

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

3. Proc Print:

C-Print.sas

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

4. Proc SQL, describe:

C-SQL-describe.sas

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

5. Proc SQL, select:

C-SQL-select.sas

```
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!

F-Freq.sas

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

16,600 lines ≅ 330 pages!

F-Freq.lst

```
16588                        Month/Year
16589
16590                                                Cumulative    Cumulative
16591  MONYR     Frequency     Percent     Frequency      Percent
16592  ---------------------------------------------------------------------
16593  JAN95           480        2.08           480         2.08
16594  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.

PROC UNIVARIATE

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

F-Univariate.sas

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

F-Univariate.lst snip 1

```
6  The UNIVARIATE Procedure
7  Variable:  Age
```

...

F-Univariate.lst snip 2

```
55          Extreme Observations
56
57 ----Lowest----          ----Highest---
58
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.

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.

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>
36 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;

SQL SELECT INTO :LIST

Fehd [4, sugi22.080] illustrates using a Proc Contents output data set to create a macro array. Fehd [5, sugi29.070] shows how to use Proc SQL to do the same, as shown here. Clay [3, sugi31.040] provides comprehensive analysis of usage of macro arrays. Whitlock [10, sugi29.244] reviews macro design and list processing.

SQL-select-into.sas

```
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_;
```

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

SQL-select-into.log

```
36 GLOBAL SQLOBS 5
37 GLOBAL SQLOOPS 27
38 GLOBAL NAME4 Height
39 GLOBAL NAME5 Weight
40 GLOBAL NAME2 Sex
41 GLOBAL NAME3 Age
42 GLOBAL NAME1 Name
```

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.

Describe-Dict-Columns.sas

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

This table contains these columns:

Describe-Dict-Columns.log

```
26 create table DICTIONARY.COLUMNS
27   (
28    libname char(8) label='Library Name',
29    memname char(32) label='Member Name',
30    memtype char(8) label='Member Type',
31    name char(32) label='Column Name',
32    type char(4) label='Column Type',
33    length num label='Column Length',
34    npos num label='Column Position',
35    varnum num label='Column Number in Table',
36    label char(256) label='Column Label',
37    format char(49) label='Column Format',
```

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

FreqAll-Loop.sas snip 1

```
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.

FreqAll-Loop.sas snip 2

```
23 %** Initialize macro array; %local SqlObs SqLoops;
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
```

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 occuring 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, 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.

FreqAll.sas macro FreqOf pg. 2

```
65  %local Nobs; %Let Nobs = 0;%*initialize for symput;
66
67  PROC Freq data   = &InLib..&InData.
68            order  = &Order.;
69            tables   &Name
70                   / list missing noprint
71               out = Freq(rename = (&Name. =
72                     %If &Type = char %then  ValuC;
73                     %else                   ValuN; ));
74
75  DATA    Freq;               %* name32 type4  $200 label40;
76  attrib  Attributes length = $%eval(32 +1 +4 +1 +4 +1 + 40)
77          ValuC      length = $20
78          ValuN      length =   8
79          Count      length =   4  format = comma.
80          Percent    length =   8  format = 6.2
81          Level      length =   4;
82
83  retain  Attributes "&Name. &Type.:&Length. &Label"
84          ValuC      "."     ValuN   Level .;
85
86  do      until(EndoFile);
87  set     Freq end  = EndoFile
88               nobs = Nobs;
89  Level   ++1;               %*increment retained counter;
90  output; end;               %*do until EndoFile;
91  call    symput(    'Nobs', compress(put(Nobs,32.)));
92  stop;   run;
93  %If &Testing. %then %Put Note:&SysMacroName.: nobs<&Nobs.>;
94
95  %If &Nobs. le %eval(2 * &Nobs2View. + 2) %then %do;
96  PROC Append base = ListFreq
97              data = Freq;                        %end;
98
99  %Else %do;%* -------------------------- *;
100 DATA Snipped;
101 set  Freq(obs = 1);ValuC = '<snipped>';ValuN   = .;
102                    Count = .;          Percent = .;
103                    Level = .;          output;      stop;
104
105 PROC Append base = ListFreq   %*high values;
106             data = Freq(obs = &Nobs2View.);
107
108 PROC Append base = ListFreq
109             data = Snipped;
110
111 PROC Append base = ListFreq   %*low values;
112             data = Freq(firstobs = %eval( &Nobs.
113                                         - &Nobs2View. +1));
114 %end;%*Else do ..................... *;
115 run;%* ............................. *; %Mend Freqof;
```

In order for the global macro variables `In_Lib` and `In_Data` ...

FreqAll.sas program, example parameters

```
28 %Let In_Lib      = Library;
29 %Let In_Data     = MyData ;
30 %Let HiLowToView = 5      ;
```

to be used in the `select ... from ... where` 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:
   `%*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.

FreqAll.sas List Processing

```
117 %Let In_Lib  = %upcase(&In_Lib.);
118 %Let In_Data = %upcase(&In_Data.);
119
120 %Let SQLprint =   print;%*testing: view FreqOf statements;
121 %Let SQLprint = noprint;
122
123 PROC SQL &SQLprint.;
124          create table  ListAttributes as
125                 select Varnum, Name, Type, Length
126                      , Format, Informat, Label, Npos
127                        from    Dictionary.Columns
128                        where     Libname eq "&In_Lib."
129                              and MemName eq "&In_Data."
130                              and MemType eq 'DATA'
131        ; select '%FreqOf(name = ' !! trim(Name)
132                 !!     ',type = ' !! trim(Type)
133                 !!   ',length = '
134                 !!            compress(put(Length,32.))
135                 !!    ',label = ' !! trim(Label)
136                 !!    ')'
137                 into  :List  separated by ' '
138                 from   ListAttributes
139        ; select Nobs   into :NobsData
140                 from   Dictionary.Tables
141                 where      Libname eq "&In_Lib."
142                        and MemName eq "&In_Data."
143                        and MemType eq 'DATA'
144        ; quit;
145 &List. ; %*execute macro calls;
146
147 %Let NobsData = &NobsData;%* remove leading blanks;
148 Title2
149    "&SysProcessName.: &In_Lib..&In_Data. nobs:&NobsData.";
150
151 PROC Print data = ListAttributes noobs;
152            Title3 "variable attributes";
153
154 PROC Print data = ListFreq;
155            var    Valu: Count Percent Level;
156            by     Attributes notsorted;
157            id     Attributes          ;
158            Title3 "list of variable frequencies";
159 run;
160 %symdel In_Data In_Lib HiLowToView List NobsData SQLprint;
```

FreqAll-List-FreqOf-SAShelp-PrdSal2.txt

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

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.

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

FreqAll.lst PrdSal2 pg. 2

```
Program FreqAll: SASHELP.PRDSAL2 nobs:23040
list of variable frequencies

Attributes                  ValuC         ValuN  Count Percent Level

COUNTRY char:10 Country     Canada            .   4,608  20.00     1
                            Mexico            .   4,608  20.00     2
                            U.S.A.            .  13,824  60.00     3

STATE char:22 State/Province Baja Calif       .   1,152   5.00     1
                            British Co        .   1,152   5.00     2
                            California        .   1,152   5.00     3
                            Campeche          .   1,152   5.00     4
                            Colorado          .   1,152   5.00     5
                            <snipped>         .       .      .     .
                            Ontario           .   1,152   5.00    12
                            Quebec            .   1,152   5.00    13
                            Saskatchew        .   1,152   5.00    14
                            Texas             .   1,152   5.00    15
                            Washington        .   1,152   5.00    16
```

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`

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';
```

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

- Abolafia [1, sugi22.229] provides a macro to reproduce Proc DataChk.
- Carpenter [2, saspress.59224] discusses dynamic programming (list processing) in ch. 9.
- Fehd [7, pnwsug2006.012] shows list processing using proc sql.
- Fehd [8, sasl.225107] posts code to replace macro array with call execute.
- Fehd and Carpenter [9, sgf2007.113] discusses basics of list processing.
- Wobus and Gober [11, sugi22.042] show data review with procs Rank, Summary and Univariate.

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 user-written proc DataChk.

[2] Arthur L. Carpenter. *Carpenter's Complete Guide to the SAS Macro Language, Second Edition*. Cary, NC: SAS Institute Inc., 2004. URL `http://support.sas.com/publishing/bbu/companion_site/59224.html`. 13 chap., 475 pp., appendices: 5, glossary: 3 pp., bibliography: 19 pp., index: 13 pp.

[3] Ted Clay. Tight looping with macro arrays. In *Proceedings of the 31st SAS User Group International Conference*, 2006. URL `http://www2.sas.com/proceedings/sugi31/040-31.pdf`. Coders' Corner, 8 pp.; two macro functions: array and do_over.

[4] Ronald Fehd. Array: Construction and usage of arrays of macro variables. In *Proceedings of the 22nd SAS User Group International Conference*, 1997. URL `http://www2.sas.com/proceedings/sugi22/CODERS/PAPER80.PDF`. Coders' Corner, 4 pp.; using proc contents output and symput, bibliography.

[5] Ronald Fehd. Array: Construction and usage of arrays of macro variables. In *Proceedings of the 29th SAS User Group International Conference*, 2004. URL `http://www2.sas.com/proceedings/sugi29/070-29.pdf`. Coders' Corner, 6 pp.; using proc sql, bibliography.

[6] Ronald Fehd. Journeymen's tools: The writing for reading and reuse program header. In *Proceedings of the 30th SAS User Group International Conference*, 2005. URL `http://www2.sas.com/proceedings/sugi30/067-30.pdf`. Coders' Corner, 4 pp.; documentation, quality, theory of program development, time spent in design and testing; bibliography.

[7] Ronald Fehd. How to use proc sql select into for list processing. In *Proceedings of the Pacific NorthWest SAS User Group Conference*, 2006. URL `http://www.pnwsug.com/Conference_2006/Procedings/PNWSUGotherfiles/PN12FehdSQL.pdf`. Hands On Workshop, 15 pp.; sql select syntax, comparison of procedures vs. sql, writing constant text into macro variable, using dictionary tables; bibliography.

[8] 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.

[9] Ronald 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/sgf2007/113-2007.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.

[10] Ian Whitlock. A second look at SAS macro design issues. In *Proceedings of the 29th SAS User Group International Conference*, 2004. URL http://www2.sas.com/proceedings/sugi29/244-29.pdf. Tutorials, 18 pp.; 6 problems from sas-l.

[11] Diana Zhang Wobus and John C. Gober. A step-by-step illustration of building a data analysis tool with macros. In *Proceedings of the 22nd SAS User Group International Conference*, 1997. URL http://www2.sas.com/proceedings/sugi22/ADVTUTOR/PAPER42.PDF. Advanced Tutorials, 7 pp.; using procs rank, summary and univariate.

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	
	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
```