

The Mighty Macro versus the Lowly Link

Barbara M. Ison and Dania P. Ferguson
National Agricultural Statistics Service
U. S. Department of Agriculture

ABSTRACT

With the advent of macros and the resulting publicity and interest in their use, the power and efficiency of the LINK may easily be overlooked for many appropriate applications. Efficiency needs to be a concern to us all, particularly those of us using mainframe time-sharing environments. The CPU/cost savings potential of techniques such as those described below are substantial, especially when developing SAS[®] systems to be used repetitively in a production mode. The efficiencies we found under MVS hold true for other operating systems running SAS.

This paper will describe when LINKs can, and probably should, be used instead of macros. We will present our real-life experiences as examples to demonstrate the power, similarities and differences in these two approaches as well as the relative efficiencies.

INTRODUCTION

Our continuous challenge is to develop flexible SAS systems, tailored to individual applications by screen-oriented expert user interfaces (we use SAS/AF[®] and SAS/FSP[®]). These systems must produce consistently accurate results within tight time frames and budget constraints. In order to meet these requirements, we rely heavily on the power of the SAS macro language, but have learned it pays to spend time analyzing and testing relative efficiencies of different coding techniques.

A CASE STUDY

A new procedure needed to be developed for analyzing selected survey data for outlier detection. Outliers are data values outside logical or normal limits (e.g. Hired Farm Worker Age < 12 or > 70). Records with especially high and/or low expanded values were to be printed with additional items for reference; and charts were to be produced showing the expanded and/or unexpanded variable value for all observations. Expanded data are data weighted by the sampling rate (number of units a sample represents). Our data covers the U.S. with

differing sets of variables needing to be analyzed for each state. These variables often change from survey to survey.

Typically, we have 75 to 100 different variables to analyze for a single survey. A separate table was needed for each analysis variable. Each table had to have appropriate title information, supporting variables, tailored print formats and variable labels. The survey statisticians also wanted to designate for which states each different analysis table would be generated.

This seemed to be a natural for our generalized application approach, driven by parameters from which we generate tailored code (refer to our SUGI 16 paper)¹ Creating macros immediately came to mind and we generated code accordingly.

The examples presented here are simplified versions of the real thing but demonstrate the concepts and techniques used.

The macro approach:

First we designed FSEDIT procedure screens for the survey statisticians to use in defining analysis variables and all the accompanying information desired, including selecting the states for each table. Figure 1 shows a pair of FSEDIT screens used to define a table. The information entered on these two screens becomes one observation in the resulting parameter dataset. These observations are then read by a SAS program which creates the actual SAS statements shown in Figures 2, 3, 5 and 6. The two sets of SAS code were generated from one pass through the parameter SAS dataset.

Figure 2 shows the macro code generated to prepare the actual data for input into the outlier print and plot routines. Figure 3 shows the generated code to selectively invoke the macros. Figure 4 is the "base" SAS code designed to bring in and use the generated code.

What happened next

When we executed tests using the code in figures 2 through 4, the cost appeared to be extremely high -- 80 SRUs in this example to run through one state's data. SRUs are System Resource Units, the measure of cost by which we are charged for processing.

Analysis of the generated code

In looking at the actual SAS code generated for execution, we very quickly realized that, because the macro code was physically duplicated each time it was referenced, the executable code became VERY LONG! After exploring possible alternatives, we decided to try generating LINK routines the same way as the macros had been generated, and see if it helped.

The LINK approach

The code generating SAS program was modified to substitute LINKs and RETURNs in place of %MACROs and %MENDs -- See figures 5 and 6. The "base" code was re-ordered to place the LINKs at the end of the data step -- See Figure 7. A comparison of the code in figures 2 and 6 shows that very little has changed.

Results of the LINK approach (Are we having fun yet??)

When the revised code was executed for the same test data, the SRUs dropped from 80 to 30 -- a 62.5 percent saving! We tested both approaches with several sets of parameters and data, and found roughly the same savings. When you consider that this process will be used for all our major surveys (more than one per month) and for all states, the magnitude of savings is impressive. All this without changing the flexibility of the system or the accuracy of the results!

Subsequent elapsed time comparisons on the PC showed even more drastic differences, plus the macro code for this application caused "out-of-memory" problems when we ran more than 5 states!

The finished product

Figure 8 shows two of the output tables created from our case study code. The tables remained identical, only the method and cost changed!

ANALYSIS OF RESULTS

Because the link acts very much like a 2nd or 3rd generation subroutine, and is not physically reproduced in the executable code, the executable SAS code was much shorter. That and the elimination of the macro overhead for those calls resulted in much more efficient SAS code.

OTHER USES AND LIMITATIONS OF LINK

The main limiting factor of LINK is that it must be used within a data step but, as demonstrated in our case study, you can often design your application within this constraint. Keep in mind that LINK routines can be made flexible in many ways, such as setting certain variable values before calling LINK and having your LINK routine vary depending on the value passed to it. Also, macro variables can be used to vary the routine from run to run.

SUMMARY (The Moral Of The Story)

Macros are extremely powerful and versatile but they are not necessarily the best answer to every situation requiring flexibility. Faster, more efficient code can result from taking the time to analyze and test alternatives. The Lowly LINK has many useful places in your SAS code!

References

1. Ison, Barbara M. and Ferguson, Dania P. (1991), Using SAS Software as a Programming Language to Develop Generalized Systems, Proceedings of the Sixteenth Annual SAS Users Group International Conference, SAS Institute Inc., Cary, NC

* SAS, SAS/AF and SAS/FSP are the registered trademarks of SAS Institute Inc., Cary, NC

For further information contact:

Barbara Ison
USDA/SID, User Services Branch
Room 4808, South Building
Washington, D. C. 20250
(202) 720-3362

Dania Ferguson
USDA/SID, System Services Branch
Room 5862, South Building
Washington, D. C. 20250
(202) 720-9248

FIGURE 1. EXAMPLE OF FSEDIT SCREENS FOR PARAMETER INPUT:

```

EDIT SAS DATA SET: SASIN.QASINFO                                SCREEN
COMMAND ==>                                                    OBS    1

      SETUP FOR CHARTS AND TABLES FOR POTENTIAL OUTLIER PRINTS    TABLE SEQ.
(Use 'n' to move to screen 'n' or use PF11 to shift to screen 2)  NUMBER  22

VARIABLE TO ANALYZE: D2          FORMAT: 10.0          LABEL ** EXPANDED*CHANGE

TITLE1: MINIMUM EXPANDED DIFFERENCE IN ACRES FOR CORN
TITLE2: MARCH VS. DECEMBER
TITLE3: DIFFERENCES THAT EXPAND TO ZERO HAVE BEEN EXCLUDED

  Extra   Print   Labels for extra   ** Use an asterisk
  Variables Print Formats   Variables **   (*) to split each
1. _____ _____ _____   label into max.
2. _____ _____ _____   of 3 print lines.
3. CROPLAND 9.0     CURRENT *CROPLAND* ACRES   analysis variable
4. HCRNXXPL 9.0     DECEMBER* CORN * ACRES   may have only 2
5. CCRNXXPL 9.0     CURRENT* CORN * ACRES   print lines in
6. ED       9.1     ADJ * EXP * FACTOR     its label.)
7. _____ _____ _____

Expansion type: NONE (NONE, XT, XF, XN, XW, XTC, XFC, XNC, XWC)
This will control minimum expansions using SPS Data Listings expansion macro.

Additional sort variable for Hi-Lo print: _____
Plot Character Variable Name: FRAME
Hi-Lo Type: 1 (1=high 20; 2=10 high/10low; 3=10high/10 low positive)

```

```

EDIT SAS DATA SET: SASIN.QASINFO                                SCREEN
COMMAND ==>                                                    OBS    21

      (Use 'n' or PF10 for screen 1, PF11 for screen 3)

      Mark all types of analysis to perform for each state
      'C' for CHART/HILO    'O' for OUTLIER EDIT    'S' for PRE-SUMMARY

AL(01) C    AK(02) _____    AZ(04) _____    AR(05) C    CA(06) C
CO(08) C    CT(09) C    DE(10) _____    FL(12) _____    GA(13) C
HI(15) _____    ID(16) C    IL(17) C    IN(18) C    IA(19) C
KS(20) C    KY(21) C    LA(22) C    ME(23) C    MD(24) C
MA(25) _____    MI(26) C    MN(27) C    MS(28) C    MO(29) C
MT(30) C    NE(31) C    NV(32) C    NH(33) _____    NJ(34) _____
NM(35) C    NY(36) _____    NC(37) C    ND(38) C    OH(39) C
OK(40) C    OR(41) C    PA(42) C    RI(44) _____    SC(45) C
SD(46) C    TN(47) C    TX(48) C    UT(49) C    VT(50) _____
VA(51) C    WA(53) C    WV(54) _____    WI(55) C    WY(56) _____

Select criteria for commodity. Data values of -1 are always excluded.

Include records with 0 values (Y/N) M
Include records for which frame(s): B ('L'=List, 'A'=Area, 'B'=Both)

```

FIGURE 2

```

EXAMPLE OF GENERATED MACROS (CHMACS):

%MACRO CHL1; ** CHART/HILO %FOR CROPLAND ;
IF CROPLAND GT 0 THEN DO;
  IF FRAME = 'AREA' THEN PLOTCODE = 'AREA';
  ELSE PLOTCODE = 'LIST'; SORTVAL = 0;
  SEQ =1 ;VARNAME = 'CROPLAND'; EFVALUE = XW ;
  XVAL1 =.; XAVAL1 ='';XVAL2 =HCROPLAN ; XAVAL2 ='';
  XVAL3 =.; XAVAL3 ='';XVAL4 =.; XAVAL4 ='';
  XVAL5 =.; XAVAL5 ='';XVAL6 =.; XAVAL6 ='';
  XVAL7 =.; XAVAL7 ='';
  SYSAHILO = 1 ; UNEXPVNL = CROPLAND ;
  VALUE = CROPLAND * EFVALUE;
  OUTPUT SASOUT.CALC2;
END;
%MEND CHL1; ** END OF CHL1 ;

NOTE: Macros for tables 2 through 21 not shown

%MACRO CHL22; ** CHART/HILO %FOR D2 ;
IF D2 GT 0 THEN DO;
  IF FRAME = 'AREA' THEN PLOTCODE = 'AREA';
  ELSE PLOTCODE = 'LIST'; SORTVAL = 0;
  SEQ =22 ;VARNAME = 'D2'; EFVALUE = 1;
  XVAL1 =.; XAVAL1 ='';XVAL2 =.; XAVAL2 ='';
  XVAL3 =CROPLAND ; XAVAL3 ='';XVAL4 =HCRNXXPL ;
  XAVAL4 ='';XVAL5 =CCRNXXPL ;
  XAVAL5 ='';XVAL6 =ED ; XAVAL6 ='';
  XVAL7 =.; XAVAL7 ='';
  SYSAHILO = 1 ; UNEXPVNL = D2 ; VALUE = D2 ;
  OUTPUT SASOUT.CALC2;
END;
%MEND CHL22; ** END OF CHL22;

NOTE: Macros for tables 23 through 78 are not shown

```

FIGURE 3

```

EXAMPLE OF GENERATED CODE TO USE MACROS FROM
FIGURE 2 (CHSELS):

LENGTH XAVAL1-XAVAL7 $ 1;
SELECT (STATE) ;
WHEN (1) DO;
  %CHL1 ; ** CROPLAND ; %CHL2 ; ** CCRNXXPL ;
  %CHL3 ; ** CSOYXXPL ; %CHL4 ; ** CSRGXXPL ;
  %CHL22 ; ** D2 ; %CHL23 ; ** D3 ;
  %CHL24 ; ** D4 ; %CHL26 ; ** D6 ;
  %CHL74 ; ** L2 ; %CHL75 ; ** L3 ;
  %CHL76 ; ** L4 ; %CHL78 ; ** NHOGTOTL ;
END;

Note: Generated code for states 2 through 55 not shown

WHEN (56) DO;
  %CHL1 ; ** CROPLAND ;
  %CHL2 ; ** CCRNXXPL ; %CHL9 ; ** CDEBXXPL ;
  %CHL76 ; ** L4 ; %CHL78 ; ** NHOGTOTL ;
  END;
  OTHERWISE ;
END;
RETURN;

```

FIGURE 4

```

EXAMPLE OF SAS CODE INCORPORATING FIGURES 2 & 3

** BRING IN GENERATED MACROS;

%INCLUDE CHMACS; *** CODE FROM FIGURE 2 ;

RUN;

DATA SASOUT.CALC2(KEEP=&BREAKIDS &HILOEFS VARNAME
VALUE SORTVAL SYSAHILO EFVALUE
XAVAL1-XAVAL7 SEQ XVAL1-XVAL7
UNEXPVNL);
LENGTH VARNAME $ 8;
SET IN.SASMSTR;
IF REPTIN GT 0 AND CRITERR LT 1;

*** CALL IN MACRO TO CALCULATE EXPANSIONS ON
DATA VALUES;

%EXPAND;

*** BRING IN GENERATED CODE TO CALL MACROS;

%INCLUDE CHSELS; *** CODE FROM FIGURE 3 ;

```

FIGURE 5

```

EXAMPLE OF GENERATED LINK CODE (CHLINKS):

CHL1 ;; ** CHART/HILO LINK FOR CROPLAND ;
IF CROPLAND GT 0 THEN DO;
  IF FRAME = 'AREA' THEN PLOTCODE = 'AREA';
  ELSE PLOTCODE = 'LIST'; SORTVAL = 0;
  SEQ =1 ;VARNAME = 'CROPLAND'; EFVALUE = XW ;
  XVAL1 =.; XVAL1 ='';XVAL2 =HCROPLAN ; XVAL2 ='';
  XVAL3 =.; XVAL3 ='';XVAL4 =.; XVAL4 ='';
  XVAL5 =.; XVAL5 ='';XVAL6 =.; XVAL6 ='';
  XVAL7 =.; XVAL7 ='';
  SYSAHILO = 1 ; UNEXPNVL = CROPLAND ;
  VALUE = CROPLAND * EFVALUE;
  OUTPUT SASOUT.CALC2;
END;
RETURN; ** END OF CHL1 ;

Note: Link code for tables 2 through 21 not shown

CHL22 ;; ** CHART/HILO LINK FOR D2 ;
IF D2 GT 0 THEN DO;
  IF FRAME = 'AREA' THEN PLOTCODE = 'AREA';
  ELSE PLOTCODE = 'LIST'; SORTVAL = 0;
  SEQ =22 ;VARNAME = 'D2'; EFVALUE = 1;
  XVAL1 =.; XVAL1 ='';XVAL2 =.; XVAL2 ='';
  XVAL3 =CROPLAND ; XVAL3 ='';XVAL4 =HCRNXXPL ;
  XVAL4 =''; XVAL5 =CCRNXXPL ;
  XVAL5 ='';XVAL6 =ED ; XVAL6 ='';
  XVAL7 =.; XVAL7 ='';
  SYSAHILO = 1 ; UNEXPNVL = D2 ; VALUE = D2 ;
  OUTPUT SASOUT.CALC2;
END;
RETURN; ** END OF CHL22 ;

Note: Link code for tables 23 through 78 not shown

```

NOTE: As you can see, the code remains very similar to that in Figures 2, 3, and 4. The links must be positioned physically within the data step instead of preceding it as the macros do.

FIGURE 6

```

EXAMPLE OF GENERATED CODE TO USE LINKS FROM
FIGURE 5 (CHSELS):

LENGTH XVAL1-XVAL7 $ 1;
SELECT (STATE) ;
WHEN (1) DO;
  LINK CHL1 ; ** CROPLAND ; LINK CHL2 ; ** CCRNXXPL;
  LINK CHL3 ; ** CSOYXXPL ; LINK CHL4 ; ** CSRGXXPL;
  LINK CHL22 ; ** D2 ; LINK CHL23 ; ** D3 ;
  LINK CHL24 ; ** D4 ; LINK CHL26 ; ** D6 ;
  LINK CHL74 ; ** L2 ; LINK CHL75 ; ** L3 ;
  LINK CHL76 ; ** L4 ; LINK CHL78 ; ** NHOGTOTL ;
END;

Note: Generated code for states 2 through 55 not shown

WHEN (56) DO;
  LINK CHL1 ; ** CROPLAND ;
  LINK CHL2 ; ** CCRNXXPL ; LINK CHL9 ; ** CDEBXXPL;
  LINK CHL76 ; ** L4 ; LINK CHL78 ; ** NHOGTOTL ;
END;
OTHERWISE ;
END;
RETURN;

```

FIGURE 7

```

EXAMPLE OF SAS CODE INCORPORATING FIGURES 5 AND 6:

DATA SASOUT.CALC2(KEEP=&BREAKIDS &HILOEFS VARNAME
  VALUE SORTVAL SYSAHILO EFVALUE
  XVAL1-SVAL7 SEQ XVAL1-XVAL7
  UNEXPNVL);
  LENGTH VARNAME $ 8;
  SET IN.SASMSTR;
  IF REPTIN GT 0 AND CRITERR LT 1;

  *** CALL IN MACRO TO CALCULATE EXPANSIONS ON
  DATA VALUES;

  %EXPAND;

  *** BRING IN GENERATED CODE TO CALL MACROS;

  %INCLUDE CHSELS; *** CODE FROM FIGURE 7 ;

  RETURN;

  *** BRING IN GENERATED LINKS;

  %INCLUDE CHLINKS; *** CODE FROM FIGURE 6 ;

```

FIGURE 8

EXAMPLE OF PRINT GENERATED USING LINK OR MACRO CODE

MARCH 1992 AG SURVEY -- POTENTIAL OUTLIER PRINTS
 MINIMUM EXPANDED ACRES FOR CROPLAND
 REPORTS THAT EXPAND TO ZERO HAVE BEEN EXCLUDED

ATLANTIS (01)

PLOT CODE	STRATA CODE	SEGMENT /LSF ID	OL/NOL		JUNE TRACT ACRES	JUNE FARM ACRES	CURR FARM ACRES	DECEMBER CROPLAND ACRES	EXPANDED CROPLAND ACRES	
			EXPN FACT	CODE /LAF						
***** 20 LARGEST *****										
AREA	3103	250	714.3	1.00	25	25	145	0	145	103578
AREA	1304	9004	649.4	1.00	35	35	120	32	115	74677
AREA	2001	9181	1179.8	1.00	98	98	99	30	60	70789
AREA	1312	24	651.1	1.00	138	203	220	140	150	66631
AREA	1303	1027	126.5	1.00	389	490	500	400	470	47249
LIST	70	770294080	19.9	1.00	.	.	3000	.	2000	39859
LIST	70	790532510	19.9	1.00	.	.	1720	.	1645	32784
AREA	3104	1258	714.3	1.00	45	160	160	105	153	31080
AREA	1303	1027	126.5	1.00	30	30	275	250	240	30259
LIST	70	857513440	19.9	1.00	.	.	1600	.	1500	29894
AREA	1301	13	126.5	1.00	201	202	240	180	180	22646
AREA	2012	204	229.8	1.00	139	239	235	125	168	22456
AREA	1302	14	649.4	1.00	50	138	138	75	95	22351
AREA	2008	9188	229.8	1.00	139	163	170	90	110	21559
AREA	2012	204	229.8	1.00	150	160	160	100	100	21547
LIST	70	867674130	19.9	1.00	.	.	1103	.	1072	21365
AREA	2011	9131	229.8	1.00	177	460	450	240	240	21225
LIST	70	770060220	19.9	1.00	.	.	1100	.	900	17937
AREA	3105	9245	714.3	1.00	33	33	34	30	25	17858
AREA	1311	23	649.4	1.00	86	550	280	80	170	17261

MARCH 1992 AG SURVEY -- POTENTIAL OUTLIER PRINTS
 MINIMUM EXPANDED DIFFERENCE IN ACRES FOR CORN
 MARCH VS. DECEMBER
 DIFFERENCES THAT EXPAND TO ZERO HAVE BEEN EXCLUDED

ATLANTIS (01)

PLOT CODE	STRATA CODE	SEGMENT /LSF ID	JUNE		CURR FARM ACRES	CURRENT CROPLAND ACRES	DECEMBER CORN ACRES	CURRENT CORN ACRES	ADJ EXP FACTOR	EXPANDED CHANGE
			TRACT ACRES	FARM ACRES						
***** 20 LARGEST *****										
AREA	1312	24	138	203	220	150	20	50	651.1	13326
LIST	70	790529320	.	.	927	800	350	500	49.8	7474
AREA	1308	9008	20	20	0	0	18	0	392.8	7071
LIST	70	770139290	.	.	653	270	0	130	49.8	6477
LIST	70	790392320	.	.	447	380	60	190	49.8	6477
LIST	70	790484200	.	.	0	0	110	0	49.8	5481
LIST	66	857509450	.	.	0	0	62	0	87.9	5447
LIST	70	770334330	.	.	0	0	105	0	49.8	5232
AREA	1303	1027	30	30	275	240	80	120	126.5	5043
LIST	70	770302510	.	.	0	0	100	0	49.8	4982
LIST	66	770990860	.	.	197	140	75	20	87.9	4832
LIST	70	770157340	.	.	423	363	160	250	49.8	4484
AREA	2003	1147	147	276	218	50	50	20	229.8	3665
LIST	71	771598810	.	.	0	0	120	0	27.0	3246
AREA	1312	9072	10	10	40	38	7	15	393.9	3151
LIST	66	790155890	.	.	70	30	35	0	87.9	3075
LIST	66	790712950	.	.	130	128	65	100	87.9	3075
AREA	2008	9128	7	115	133	125	60	20	1179.8	3064
AREA	1311	23	86	550	280	170	20	50	649.4	3046
AREA	1303	1027	389	490	500	470	80	110	126.5	3016