SAS Advanced Programming with Efficiency in Mind: A Real Case Study

Quinn Liu has been using SAS since 2002 and is a SAS Certified Advanced programmer. He currently works as a application programmer/analyst lead for the University of Michigan Kidney Epidemiology and Cost Center, School of Public Health (UM-KECC) and works extensively on large database with base SAS programming, the Output Delivery System, SQL, the macro language, SAS/Graph, etc. He enjoys helping SAS users choose the most appropriate technique for their data situations and improve SAS programming efficiencies. He has authored/presented papers at Michigan SAS user group meetings and MWSUG conferences since 2006.

Name : Lingqun (Quinn) Liu
Organization: University of Michigan
Work Phone: 734-763-1603
E-mail: lqliu@umich.edu

SAS Advanced Programming with Efficiency in Mind: A Real Case Study

Lingqun Liu, University of Michigan, Ann Arbor

I. Background

- UM-KECC
- A Small SAS Application
- Performance

I. UM-KECC

- UM-KECC is a multidisciplinary research center within the UM School of Public Health (SPH). UM-KECC was formed in 1993 and its mission is "**to promote health, improve clinical practice and patient outcomes, optimize resource utilization, and inform public policy regarding organ failure and organ transplantation**." UM-KECC pursues this mission "**through high quality research, advances in biostatistics, and post-graduate education and training**." (www.kecc.sph.umich.edu).

I. 1 A Small SAS Application

- UM-KECC creates facility patient lists for quality measures each quarter : 5 jobs, one per measure, 21,870 /21,702 patient list files (201607/201604).
  - M1_DFC_Patient_Lists.sas
  - …
  - M5_DFC_Patient_Lists.sas

I.2 Process Time

Jobs	201604		201607	
	Real time	CPU time	Real time	CPU time
M1_DFC_Patient_Lists.sas	4:16:02.04	4:04:10.54	4:11:53.18	4:04:35.88
M2_DFC_Patient_Lists.sas	16:30.83	13:40.53	1:48:09.24	15:27.02
M3_DFC_Patient_Lists.sas	1:39.45	18.93	2:30.09	22.88
M4_DFC_Patient_Lists.sas	4:49:30.74	4:42:13.26	16:17:07.14	7:34:12.81
M5_DFC_Patient_Lists.sas	10:02:30.96	9:49:39.17	11:02:17.87	10:22:32.44
total	19:26:14.02	17:08:55.50	33:21:57.52	22:17:11.03

II. Code Analysis

- Code Logic
- Code Design/Structure
- SAS Features

II.1 Code Listing (part 1)

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_original_MWSUG.sas *

```
25
26 proc sort data=faclib.facinfo_&lookupdt. out= facinfo (keep= facid network provname provcity state)
27    where DFC_report=1;
28    by facid;
29 run;
30
31 %macro print_list(data,measure,name);
32
33 %IF &measure=M5  %then %do;
34 %let vars=firsts dialysis_90days age_ge_18 calcium_uncorrected in_facility modality elig_pm avg_3mo
35 %let varslabel= firsts='First^service^date' dialysis_90_days='Dialysis^ge^90^days' age_ge_18='Patie
36 in_facility="Meets^facility^requirement" modality="Meets^modality^requirement" elig_pm="Eligible^pa
37 hypercal_gt10_2="Hypercalemia-gt^10.2";
38 %end;
39
40 %put &vars;
41 %put &varslabel;
42
43 ******************************************************************************************;
44 *--------Sort measure files, keep only variables output to list---------------------------*;
45 ******************************************************************************************;
46
47 proc sort data=mlib.&data. out=temp(keep=patid facid &vars year month quarter);
48    by patid;
49 run;
50
51 /*****************************************************************************************
52 *--Merge individual measure files with patients to get patient identifiers---*;
53 /*****************************************************************************************/
54 data saflib.M5_plist_&dateit.;
55    merge temp (in=a)  saflib.patients (keep=patid surname first_name m_initial ssn);
56    by patid;
57    if a;
58    fname=trim(first_name)||' '||trim(m_initial);
59    Patient_id=_n_;
60    ssn1=ssn+0;
61     **** Note: In this step, a small percentage of pts have characters in their SSN.  This     ****;
62     *    causes warning messages in the log file because ssn1 cannot be calculated, and in the    *;
63     *    final patient list they will have a missing SSN.  Since the SSNs are not numeric, we     *;
64     *    assume they are not valid, so having missing SSN is not a problem.                      *;
65 run;
66
67 /*****************************************************************************************
68  *---Merge with facinfo to obtain provider name, city, state, etc---*
69 *****************************************************************************************/
70 proc sort data=saflib.M5_plist_&dateit.;
71    by facid;
72 run;
73
```

II.1 Code Listing (part 2)

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_original_MWSUG.sas *

```
78  data  &measure._ptlist;
79     merge saflib.M5_plist_&dateit.(in=a)  facinfo (in=infacinfo);
80     by facid;
81     if a and infacinfo;
82     facility=trim(provname)||', '||trim(provcity)||', '||state;
83     format ssn1 ssn11.;
84     report_period=strip(year)||"    "||strip(month)||"    "||strip(quarter);
85  run;
86
87
88  proc sort data= &measure._ptlist;
89     by network facid facility surname first_name ;
90  run;
91
92
93  proc sql;
94    select count(distinct facid) into: numprovs
95    from  &measure._ptlist;
96  quit;
97
98  %put &numprovs;
99
100 data _null_;
101    length numprovsc $9.;
102    numprovsc=strip(&numprovs);
103    call symput('numprovsc', numprovsc);
104 run;
105
106
107 proc sql;
108   select distinct facid into :prov1  -:prov&numprovsc notrim
109   from  &measure._ptlist;
110 quit;
111
112
113 %do i=1 %to &numprovs;
114       %put '******************' &&prov&i;
115        data prvlevel ;
116           set  &measure._ptlist;
117           where facid="&&prov&i";
118           call symput ("facility", compress(facility,"'"));
119
120        run;
121        %put '******************' &facility;
122
123       ods listing file="&outfile";
124             title "CONFIDENTIAL: Patients included in the &name. measure reported in the";
125                    title2        "Quarterly Dialysis Compare-Preview for &month., &year. report.";
126                    title3 "MMM Certification Number=&&prov&i Facility=&facility";
127       options  ls=max ps=85;
128             proc print data=prvlevel noobs  split='^' uniform;
```

II.2 Code Logic

Two simple requirements

- **Create a data set**: to create a patient-measure level data set.
- **Create listing files:** to print patient-measure information by facility in plain text file with extension .txt.

II.2 Code Design & Structure

- ***Subtask 1***

  4 PROC SORTs, 2 DATA MERGEs.

- ***Subtask 2***

  2 PROC SQLs, 1 DATA _NULL_, a %MACRO %DO loop of 1 DATA step and ODS/PROC PRINT.

  - The second subtask is implemented with a %MACRO %do loop that creates and prints out one data set for each facility.

II.3 SAS Features

Many SAS features, including some pretty advanced ones, have been utilized in this SAS application.

- DATA STEP MERGE, PROC SQL, PROC SORT;
- %MACRO, &&VAR&N, CALL SYMPUT, INTO:, %DO loop; DATA _NULL_;
- Data type conversion (+0), function COMPRESS(), STRIP(), TRIM();
- ODS, Dynamic titles, PROC PRINT options, etc.
- System options: LS, NODATE, NONUMBER, NOCENTER, ERRORS, SOURCE2, MPRINT…

II.4 Critical Thinking

- Does it need to be so complicated?
- Does it need to use so many steps and features?
- Is the %macro really needed?
- Which features/steps did take most of the process time?

III. Log Analysis

- Review
- Estimation
- Utility
- Statistics

III.1 Review

Figure 2.1 Log Snapshot One

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG (2).log

```
NOTE: There were 2819069 observations read from the data set SAFKECC.PATIENTS.
NOTE: The data set SAFLIB.M5_PLIST_201604 has 6423888 observations and 21 variables.
NOTE: Compressing data set SAFLIB.M5_PLIST_201604 decreased size by 42.01 percent.
      Compressed is 64233 pages; un-compressed would require 110757 pages.
NOTE: DATA statement used (Total process time):
      real time           1:33.78
      cpu time            37.01 seconds

SYMBOLGEN:  Macro variable DATEIT resolves t
MPRINT(PRINT_LIST):   proc sort data=SAFLIB.
MPRINT(PRINT_LIST):   by provfs;
MPRINT(PRINT_LIST):   run;

NOTE: There were 6423888 observations read from the data set SAFLIB.M5_PLIST_201604.
NOTE: The data set SAFLIB.M5_PLIST_201604 has 6423888 observations and 21 variables.
NOTE: Compressing data set SAFLIB.M5_PLIST_201604 decreased size by 42.00 percent.
      Compressed is 64234 pages; un-compressed would require 110757 pages.
NOTE: PROCEDURE SORT used (Total process time):
      real time           1:36.95
      cpu time            36.65 seconds
```

Large DATA step MERGERING and PROC SORTING were fast.

III.1 Review

Figure 2.2 Log Snapshot Two

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG (2).log *

```
515 NOTE: The data set WORK.M5_PTLIST has 6423888 observations and 27 variables.
516 NOTE: Compressing data set WORK.M5_PTLIST decreased size by 51.77 percent.
517       Compressed is 88522 pages; un-compressed would require 183540 pages.
518 NOTE: PROCEDURE SORT used (Total process time):
519       real time           2:47.55
520       cpu time            1:04.28
```

Large DATA step MERGERING and PROC SORTING were fast.

III.2 Estimation

Figure 2.3 Log Snapshot Three

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG (2).log

```
578  NOTE: There were 1404 observations read from the data set WORK.M[redacted]PTLIST.
579        WHERE p[redacted]';
580  NOTE: The data set WORK.PRVLEVEL has 1404 observations and 27 variables.
581  NOTE: Compressing data set WORK.PRVLEVEL decreased size by 53.66 percent.
582        Compressed is 19 pages; un-compressed would require 41 pages.
583  NOTE: DATA statement used (Total process time):
584        real time           5.28 seconds
585        cpu time            5.28 seconds
```

It used a few seconds or so per facility

Figure 2.4 Log Snapshot Four

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG (2).log

```
457571  MPRINT(PRINT_LIST):   ods listing close;
457572  MLOGIC(PRINT_LIST):  %DO loop index variable I is now 6376; loop will not iterate again.
457573  MLOGIC(PRINT_LIST):  Ending execution.
457574  294
457575  295
457576
457577  NOTE: SAS Institute Inc., SAS Campus Drive, Car
457578  NOTE: The SAS System used:
457579        real time           10:02:30.96
457580        cpu time            9:49:39.17
457581
```

The stop value of the %DO loop was 6,375 for this case. Therefore, the total run time was about 5.28*6375/(60*60) seconds = 9.35 hours

III.3 Log Analysis Utility

Programmer's File Editor - [log_analysis_MWSUG.sas]

File Edit Options Template Execute Macro Window Help

```
1  ***********************************************************************;
2  *Program name: log_analysis.sas
3  *By           : lqliu@umich.edu 2008, 2016
4  *Purpose      : to process SAS log file to analyze SAS application
5  *             :   --structure and performance
6  *             :
7  *Input        : SAS log file
8  *Output       : two txt file and two datasets, plus ...
9  *
10 *Note         : internal use only
11 *             : some lines in step 3 need revision to reflect individual needs
12 ************************************************************************;
13 option mprint;
14 ***************** STEP 1 ******************************************;
15 ** usage: ------------------------------------------------------------;
16 ** %log_IO_search(log= [your log file].log,
17                    doc=[results txt file].txt);
18 ************************************************************************;
19 %macro log_IO_search(log=,doc=);
20 %*if not %index(&log,'.') %then %let log=&log.*.log;
21 data _null_;
22 length logname f logline $200 ;
23 infile "&log" filename=f  end=done;
24 file "&doc";
25 logname=f;
26 if logname ne lag(logname) then do;
27     if line then put line "lines read";
28     put // '-----' logname '------';
29          line=0;
30 end;
31 input ;
32 line + 1;
33 output = index(_infile_,'NOTE: The data set')  and
34          not index(_infile_,'-- NOTE:')
35                  or
36          index(_infile_,'were written to the file')
37 ;
38 input = index(_infile_,'read from')  or   index(_infile_,'WHERE ')  ;
39 time  = index(_infile_,' time')  ;
40 logline = _infile_;
41 keep= ifc(input,'INPUT ','OUTPUT') ;
42 keep= ifc(input,keep,'TIME  ') ;
43 if input or output or time then put keep logline;
44 if done then put line "lines read";
45 run;
46 %mend;
47
48 %let log=I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.log;
49 %let doc=I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.txt;
50 %log_IO_search(log=&log,doc=&doc);
```

III.3 Log Analysis Utility

```
******** STEP 2*************************************;
** usage: ------------------------------------------------------------;
** %log_IO_data(log= [results txt file from step 1 above].txt,
                    doc=[results txt file].txt);
*********************************************************************;
%macro log_IO_data(log=,doc=);
data log_runtime_messy log_runtime(keep= dsn ntime ctime procdat  obs);
length logname f logline $200 dsn $32 PROCDAT $6;
retain dsn obs;
infile "&log" filename=f  end=done;
 file "&doc";  *optional;
logname=f;
if logname ne lag(logname) then do;
    if line then put line "lines read";
    put // '-----' logname '------';
        line=0;
end;
input @;
 if
 index(_infile_,'TIME NOTE: The data set')
 or index(_infile_,'TIME NOTE: DATA statement used (Total process time):')
 or index(_infile_,'TIME NOTE: PROCEDURE SORT used (Total process time):')
 or index(_infile_,'TIME NOTE: PROCEDURE SQL used (Total process time):')

```

III.3 Log Analysis Utility

\MWSUG16__BB18\log_analysis_MWSUG.sas

```
     input /;
     ctime = scan(_infile_,4,"");
     if index(ctime,':') then do;
        if countc(ctime,':')=1 then ctime='0:'||ctime ;
        ntime=input(strip(ctime), time11.2);
     end;
     else ntime=ctime+0;
         if PROCDAT='DATA: ' then do; DATA_TIME+ntime; DATA_steps+1; end;
         else if PROCDAT='SORT: ' then do;SORT_TIME+ntime; SORT_steps+1; end;
         else do;SQL_TIME+ntime; SQL_steps+1;end;
     output;
      put PROCDAT _infile_  @46 OBS comma10.0 ' ' @60 DSN  "---" ntime=mmss8.2;  *optional;
    end;
   else input;
 end;
 else input;
 if done then do;
     put DATA_steps " DATA steps -- total process time " DATA_TIME=time11.2  ;  *optional;
         put SORT_steps " SORT steps -- total process time " SORT_TIME=time11.2; *optional;
         put SQL_steps " SQL steps -- total process time " SQL_TIME=time11.2  ;   *optional
end;

run;
%mend;
option nprint;
 %let log=I:\MWSUG16__BB18\MS_DFC_Patient_Lists_MWSUG.txt;
 %let doc=I:\MWSUG16__BB18\MS_DFC_Patient_Lists_MWSUG2.txt;
 %log_IO_data(log=&log,doc=&doc);

** STEP 3 ***********************************************************;
** summarize the results;
*********************************************************************;
proc means data=log_runtime   mean max min sum;
class procdat dsn ;
var ntime obs;
types dsn procdat procdat*dsn ;
run;

** ENDSAS ***;
```

III.4 Log Analysis: Output 1

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.txt

```
-----I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.log ------
TIME real time           0.10 seconds
TIME cpu time            0.06 seconds
INPUT NOTE: There were 6553 observations read from the data set FACLIB.FACINFO_201601.
INPUT WHERE DFC_report=1;
TIME NOTE: The data set WORK.FACINFO has 6553 observations and 5 variables.
TIME NOTE: PROCEDURE SORT used (Total process time):
TIME real time           0.74 seconds
TIME cpu time            0.06 seconds
INPUT NOTE: There were 6554484 observations read from the data set MLIB.M5_PATIENT_LIST.
TIME NOTE: The data set WORK.TEMP has 6554484 observations and 14 variables.
TIME NOTE: PROCEDURE SORT used (Total process time):
TIME real time           19.53 seconds
TIME cpu time            18.54 seconds
```

III.4 Log Analysis: Output 2

I:\MWSUG16_BB18\M5_DFC_Patient_Lists_MWSUG2.txt

```
-----I:\MWSUG16__BB18\M5_DFC_Patient_Lists_MWSUG.txt ------
SORT: TIME real time          0.74 seconds      6,553    WORK.FACINFO ---ntime=0:00.74
SORT: TIME real time          19.53 seconds  6,554,484    WORK.TEMP ---ntime=0:19.53
DATA: TIME real time          27.00 seconds  6,554,484    SAFLIB.M5_PLIST_201607 ---ntime=0:27.00
SORT: TIME real time          18.59 seconds  6,554,484    SAFLIB.M5_PLIST_201607 ---ntime=0:18.59
DATA: TIME real time          26.34 seconds  6,554,484    WORK.M5_PTLIST ---ntime=0:26.34
SORT: TIME real time          29.15 seconds  6,554,484    WORK.M5_PTLIST ---ntime=0:29.15
SQL : TIME real time          7.20 seconds   6,554,484    WORK.M5_PTLIST ---ntime=0:07.20
DATA: TIME real time          0.00 seconds   6,554,484    WORK.M5_PTLIST ---ntime=0:00.00
SQL : TIME real time          6.75 seconds   6,554,484    WORK.M5_PTLIST ---ntime=0:06.75
DATA: TIME real time          5.21 seconds       1,380    WORK.PRVLEVEL ---ntime=0:05.21
PRNT: TIME real time          0.01 seconds       1,380    WORK.PRVLEVEL ---ntime=0:00.01
DATA: TIME real time          5.21 seconds         720    WORK.PRVLEVEL ---ntime=0:05.21
PRNT: TIME real time          0.00 seconds         720    WORK.PRVLEVEL ---ntime=0:00.00
```

III.4 Log Analysis: Output 3

	dsn	PROCDAT	obs	ctime	ntime
1	WORK.FACINFO	SORT:	6553	0.74	0.74
2	WORK.TEMP	SORT:	6554484	19.53	19.53
3	SAFLIB.M5_PLIST_201607	DATA:	6554484	27.00	27
4	SAFLIB.M5_PLIST_201607	SORT:	6554484	18.59	18.59
5	WORK.M5_PTLIST	DATA:	6554484	26.34	26.34
6	WORK.M5_PTLIST	SORT:	6554484	29.15	29.15
7	WORK.M5_PTLIST	SQL :	6554484	7.20	7.2
8	WORK.M5_PTLIST	DATA:	6554484	0.00	0
9	WORK.M5_PTLIST	SQL :	6554484	6.75	6.75

III.5 Log Analysis: Statistics

PROCDAT	dsn	N Obs	Variable	Mean	Maximum	Minimum	Sum
DATA:	SAFLIB.M5_PLIST_201607	1	ntime	27.0000000	27.0000000	27.0000000	27.0000000
			obs	6554484.00	6554484.00	6554484.00	6554484.00
	WORK.M5_PTLIST	2	ntime	13.1700000	26.3400000	0	26.3400000
			obs	6554484.00	6554484.00	6554484.00	13108968.00
	WORK.PRVLEVEL	6426	ntime	6.1020090	20.9000000	5.1600000	39211.51
			obs	1019.99	6240.00	12.0000000	6554484.00
PRNT:	WORK.PRVLEVEL	6426	ntime	0.0066355	0.0700000	0	42.6400000
			obs	1019.99	6240.00	12.0000000	6554484.00
SORT:	SAFLIB.M5_PLIST_201607	1	ntime	18.5900000	18.5900000	18.5900000	18.5900000
			obs	6554484.00	6554484.00	6554484.00	6554484.00
	WORK.FACINFO	1	ntime	0.7400000	0.7400000	0.7400000	0.7400000
			obs	6553.00	6553.00	6553.00	6553.00
	WORK.M5_PTLIST	1	ntime	29.1500000	29.1500000	29.1500000	29.1500000
			obs	6554484.00	6554484.00	6554484.00	6554484.00
	WORK.TEMP	1	ntime	19.5300000	19.5300000	19.5300000	19.5300000
			obs	6554484.00	6554484.00	6554484.00	6554484.00
SQL :	WORK.M5_PTLIST	2	ntime	6.9750000	7.2000000	6.7500000	13.9500000
			obs	6554484.00	6554484.00	6554484.00	13108968.00

6,429 DATAs
3 large, 6,426 small.
Large: a few mins.
6,426 small, >10 hrs: 39,211/(60*60) secs =10.89 hrs.
PROCs: < 1 minute.

III.6 Areas to Improve

- Reduce the number of steps
  - Data steps, procs
  - %macro
  - Data sorting
- Reduce the notes in log file
  - Macro related
  - Invalid data errors

IV. Redeveloping the Application

- with Efficiency in Mind

IV.1 New Design & Structure

- **Redesign the process**
  - One data step: Avoid/eliminate macro & unnecessary sorting
  - Create view instead of data set
  - Use a simple (but advanced traditional, and powerful) technique
- **New code structure**
  - One PROC SQL VIEW + one DATA step
  - Code outline

```
PROC SQL CREATE VIEW
     . . .
  QUIT;
  DATA . . ;
     . . .
     BY FACID; ...;
     _FN=...FACID...;
     FILE WRITEOUT FILEVAR=_FN;
     . . .
     PUT
     . . .
  RUN;
```

IV.2 New Code (part 1)

Programmer's File Editor - [M5_DFC_Patient_Lists_quinn_MWSUG.sas]

File Edit Options Template Execute Macro Window Help

```
1  /*****************************************************************************************
2  Program Name: M5_PatLis.sas
3  Purpose      : Print faclity patient list for M5 measure for DFC
4  By           : lqliu@umich.edu 2016-04-25
5
6  Input        : 1. measure results- QDFC.M5_patient_list
7               : 2. patient info    - saflib.patients
8               : 3. facility info   - faclib.facinfo_&lookupdt
9
10 Output       : 1. facility patient lists:  &outpath\M5_PatList_999999.lst
11              : 2. SAS dataset -- saflib.M5_plist
12
13 Note         : A programmer with appropriate permissions must run this code
14              : Make sure output folder has been created.
15
16 *****************************************************************************************/
17
18 %include "\\disk\DFC\Code\dfc_dateparms.sas";
19 options  ls=max ps=85 nodate nonumber source;
20
21 *--Output for patient lists--*;
22 libname ptlists "\\disk\\QDFC\Patient_Lists\&refreshdt._release\&dateit.";
23 libname saflib "\\disk\saflib";
24
25 *%let outpath=\\disk\\QDFC\Patient_Lists\&refreshdt._release\&dateit.;
26 %let outpath=\\disk\\QDFC\Patient_lists\quinn_test_output; * testing path;
27
28
29 %let vars=firsts dialysis_90days age_ge_18 calcium_uncorrected in_facility modality elig_pm
30
31 *-- put Mesures, Facinfo, and Patinfo together;
32 proc sql;
33 create view M5_patlst as
34 select a.*,b.*,c.*
35 from QDFC.M5_patient_list (keep=patid facid &vars year month quarter) a
36      left join saflib.patients (keep=patid surname first_name m_initial ssn) b
37      on a.patid=b.patid
38      join faclib.facinfo_&lookupdt (keep=facid network provname provcity DFC_report state wh
39      on a.facid=c.facid
40 order by a.facid, b.surname, b.first_name, a.year, a.month;
41 quit;
```

One PROC SQL view to put 3 data sets together.

Figure 4.1.1

IV.2 New Code (part 2)

Programmer's File Editor - [M5_DFC_Patient_Lists_quinn_MWSUG.sas]

File Edit Options Template Execute Macro Window Help

```
42
43 *-- print out pat lists by facility along with some data formatting: fake patient id, formating
44 data saflib.M5_plist_&dateit.&runby;
45   set M5_patlst;
46   by facid;
47
48   patient_rec_id=_n_;
49   _fname=trim(first_name)||' '||trim(m_initial);
50   _facility=trim(provname)||', '||trim(provcity)||', '||state;
51   if month<10
52   else
53
54   * -- in ord
55   if prxmatch
56   format _ssn1
57
58   _fn = "&outpath\M5_PatList_" || TRIM(facid) || ".txt";
59   FILE writeout FILEVAR=_fn HEADER=newpage LINESLEFT=_remain LINESIZE=80 NOTITLES  N=ps        ;
60
61   * --if new page then put header lines---;
62   if _ramian<11 then put _page_;
63   put  @3 Patient_rec_id  @15 surname @49 _fname  @64 _ssn1 @80 _report_period @101 firsts @118
64   if last.facid then put "<---END of FILE--->";
65
66   drop _: network state provcity dfc_report provname;
67   return;
68   * --if new page then put header lines---;
69   newpage:
70   put  "CONFIDENTIAL: Patients included in the Adult Uncorrected serum calcium > 10.2 mg/dL meas
71   put  "Quarterly Dialysis Compare-Preview for July, 2016 report.";
72   put  "MMM Certification Number=" facid "Facility=" _facility;
73   put;
74   put   @127 "Patient";
75   put   @69 "Social" @105 "First" @130 "Age" @157 "Meets"  @172 "Meets" @184 "Eligible";
76   put   @1 "Patient ID" @67 "Security" @81 "Report" @88 "Period" @103 "service" @114 "dialysis_"
77   put   @3 "Number" @15 "Last Name"  @49 "First Name" @69 "Number"  @79 "Year  Month Quarter"  @
78   put;
79   return;
80 run;
81
82
83 ENDSAS::::::
```

Use FILE statement with option FILEVAR= and BY processing to write out multiple files in one DATA step.

IV.2 New Code: SAS Features

FILEVAR=**variable**

defines a variable whose change in value causes the FILE statement to close the current output file and open a new one the next time the FILE statement executes. The next PUT statement that executes writes to the new file that is specified as the value of the FILEVAR= variable.

Restriction:	The value of a FILEVAR= variable is expressed as a character string that contains a physical filename.
Interaction:	When you use the FILEVAR= option, the **file-specification** is just a placeholder, not an actual filename or a fileref that has been previously assigned to a file. SAS uses this placeholder for reporting processing information to the SAS log. It must conform to the same rules as a fileref.
Tip:	This variable, like automatic variables, is not written to the data set.
Tip:	If any of the physical filenames is longer than eight characters (the default length of a character variable), assign the FILEVAR= variable a longer length with another statement, such as a LENGTH statement or an INPUT statement.

IV.2 New Code: SAS features

HEADER=**label**

defines a statement label that identifies a group of SAS statements that you want to execute each time SAS begins a new output page.

Restriction:	The first statement after the label must be an executable statement. Thereafter you can use any SAS statement.
Restriction:	Use the HEADER= option only when you write to print files.
Tip:	To prevent the statements in this group from executing with each iteration of the DATA step, use two RETURN statements: one precedes the label and the other appears as the last statement in the group.

IV.2 New Code: SAS features

LINESLEFT=*variable*

defines a variable whose value is the number of lines left on the current page. You supply the variable name; SAS assigns the value of the number of lines left on the current page to that variable. The value of the LINESLEFT= variable is set at the end of PUT statement execution.

Alias	LL=
Tip	This variable, like automatic variables, is not written to the data set.
Example	Determining New Page by Lines Left on the Current Page

IV.2 New Code: SAS features

N=*available-lines*

specifies the number of lines that you want available to the output pointer in the current iteration of the DATA step. *Available-lines* can be expressed as a number (*n*) or as the keyword PAGESIZE or PS.

n

specifies the number of lines that are available to the output pointer. The system can move back and forth between the number of lines that are specified while composing them before moving on to the next set.

PAGESIZE

specifies that the entire page is available to the output pointer.

Alias	PS
Restrictions	N=PAGESIZE is valid only when output is printed.
	If the current output file is a file that is to be printed, *available-lines* must have a value of either 1 or PAGESIZE.
Interactions	In addition to use in the N= option to control the number of lines available to the output pointer, you can also use the #*n* line pointer control in a PUT statement.
	If you omit the N= option and no # pointer controls are used, one line is available. That is, by default, N=1. If N= is not used but there are # pointer controls, N= is assigned the highest value that is specified for a # pointer control in any PUT statement in the current DATA step.
Tip	Setting N=PAGESIZE enables you to compose a page of multiple columns one column at a time.

IV.3 New Process Time

Figure 4.2 New Log Snapshot One

I:\MWSUG16__BB18\M5_DFC_Patient_Lists_quinn_MWSUG.log

```
10707 NOTE: The file WRITEOUT is:
10708       Filename=\\DISK\quinn_test_output\MBD_PatList_111111.txt,
10709       RECFM=V,LRECL=256,File Size (bytes)=0,
10710       Last Modified=28Apr2016:20:52:01,
10711       Create Time=28Apr2016:16:29:03
10712
10713 NOTE: The file WRITEOUT is:
10714       Filename=\\DISK\\quinn_test_output\MBD_PatList_222222.txt,
10715       RECFM=V,LRECL=256,File Size (bytes)=0,
10716       Last Modified=28Apr2016:20:52:01,
10717       Create Time=28Apr2016:16:29:03
```

IV.3 New Process Time

Figure 4.3 New Log Snapshot Two

I:\MWSUG16_BB18\M5_DFC_Patient_Lists_quinn_MWSUG.log

```
59500 NOTE: The data set SAFLIB.M5_PLIST_201607_QUINN has 6554484 observations and 19 variables.
59501 NOTE: Compressing data set SAFKECC.M5_PLIST_201607_QUINN decreased size by 42.20 percent.
59502       Compressed is 86102 pages; un-compressed would require 148966 pages.
59503 NOTE: DATA statement used (Total process time):
59504       real time           2:02.22
59505       cpu time            1:45.83
59506
59507
59508 196
59509 197
59510 198        ENDSAS;;;;;;
59511
59512 NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
59513 NOTE: The SAS System used:
59514       real time           2:24.67
59515       cpu time            1:46.53
```

IV.5 Difference

<u>New Code vs. Original Code</u>

- 00:02:30 vs. 11:02:17 (hh:mm:ss). Process time saved 99.62%.

It is 260x faster.

V. Conclusions

- Efficiency Awareness
- Log Analysis Utility
- Some Programming Tips

V.1 Efficiency Awareness

<u>New Code vs. Original Code</u>

- 79 lines vs. 150 lines
- 1 step vs. 6,384 steps
- **22,5**18,989 vs. **61,8**52,446 records processed
- 00:02:30 vs. 11:02:17 (hh:mm:ss).
  Process time saved 99.62%.

V.2 Suggestions

- Application Design
  - Understand the problem
  - Improve problem solving skills
  - Design the right algorithm
  - Knowledge base and skill sets

- Programming Tips
  - ✓ Use less steps if applicable
  - ✓ Avoid complex macro if you can
  - ✓ Process only the required variables and observations
  - ✓ Do not fall in love with your "hammer", know the right tool
  - ✓ Be machine, human and environment friendly

V.3 Another Example

Original vs. Optimized Application

- 1,100 lines vs. 480 lines
- 142 steps Vs. 30 steps
- 3,344 vs. 232 millions records processed
- >40 hrs vs. 5 hrs: about 87.5% of process time saved
- 10 GB Vs. 6 GB output

A claim processing application optimized in 2008

Thank You!

- Questions &/: Comments?
- Share Your Experiences/Tips, etc.?

"Everything should be as simple as it can be, but not simpler." says Einstein.

Name : Lingqun (Quinn) Liu
Organization: University of Michigan
Work Phone: 734-763-1603
E-mail: lqliu@umich.edu
Web: www.kecc.sph.umich.edu