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).
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
## 1.2 Process Time

<table>
<thead>
<tr>
<th>Jobs</th>
<th>201604</th>
<th>201607</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Real time</td>
<td>CPU time</td>
</tr>
<tr>
<td>M1_DFC_Patient_Lists.sas</td>
<td>4:16:02.04</td>
<td>4:04:10.54</td>
</tr>
<tr>
<td>M2_DFC_Patient_Lists.sas</td>
<td>16:30.83</td>
<td>13:40.53</td>
</tr>
<tr>
<td>M3_DFC_Patient_Lists.sas</td>
<td>1:39.45</td>
<td>18.93</td>
</tr>
<tr>
<td>total</td>
<td>19:26:14.02</td>
<td>17:08:55.50</td>
</tr>
</tbody>
</table>
II. Code Analysis

- Code Logic
- Code Design/Structure
- SAS Features
II.1 Code Listing (part 1)

```sas
proc sort data=faclib.facinfo &lookupdt. out= facinfo (keep= facid network provname provcity state) where DFC_report=1;
by facid;
run;

%macro print_list(data, measure, name);
%if &measure=M5 %then %do;
%let vars=firsts dialysis_90days age_ge_18 calcium_uncorrected in_facility modality elig_pm avg_3mo
%let vars=labels='First service date' dialysis_90_days='Dialysis age=90 days' age_ge_18='Patie
xt age 18' calcium_uncorrected='Facility calcium requirement' modality='Meets modality requirement' elig_pm='Eligible' pat

%let hypercalcia=gt10.2="Hypercalcemia gt=10.2";
%end;
%put &vars;
%put &oars;
%put &oarslabels:
***--------------------------------------------------------------------------------------------------------------------------;
***--------------------------------------------------------------------------------------------------------------------------;
-------------------------------------------------------------------------------------------------------------------------------;
proc sort data=mlib.&data. out-temp(keep=patient facid &vars year month quarter);
by patid;
run;

/****** Merge individual measure files with patients to get patient identifiers\\*****/
data safile.M5_plist &dateit.;
merge temp (in=a) safile.patients (keep-patid surname first_name m_initial ssn);
by patid;
if a; Name=trim(first_name)||' '||trim(m_initial);
Patient_id_n;
ssn=ssn+0;
**** Note: In this step, a small percentage of pts have characters in their SSN. This ***;
* causes warning messages in the log file because ssn cannot be calculated, and in the x;
* final patient list they will have a missing SSN. Since the SSNs are not numeric, we ******;
* assume they are not valid, so having missing SSN is not a problem.**
run;

/****** Merge with facinfo to obtain provider name, city, state, etc----------\\*****/
data safile.M5_plist &dateit.:
by facid;
run;
```
II.1 Code Listing (part 2)

```sas
data &measure_ptlist;
merge sailib.MS_plist &dateit.(in=a) facinfo (in=infacinfo);
by facid;
  if a and infacinfo;
  facility=trim(provname) || ' ' ||trim(provcity) || ' ' ||state;
  format ssn1 ssn1.;
  report_period=strip(year) || ' ' ||strip(month) || ' ' ||strip(quarter);
run;

proc sort data= &measure_ptlist;
by network facid facility surname first_name ;
run;

proc sql;
  select count(distinct facid) into: numprovs
  from &measure_ptlist;
quit;
%put &numprovs;

data null;
length numprovs $9.:
numprovs=strip(&numprovs);
call symput('numprovs', numprovs);
run;

proc sql;
  select distinct facid into :prov1 =prov &numprovs notrim
  from &measure_ptlist;
quit;
%do i=1 %to &numprovs:
  %put '**********
  data &provlevel ;
  set &measure_ptlist;
  where facid="&prov&i";
  call symput('facility', compress(facility,"")) ;
run;
  %put '**********

ods listing file="&outfile";
title "CONFIDENTIAL: Patients included in the &name. measure reported in the";
title2 "Quarterly Dialysis Compare-Preview for &month., &year. report.";
title2 "NM# Certification Number-&prov&i Facility-&facility";
options ls=max ps=85;
proc print data=&provlevel noobs split='') uniform;
```

#MWSUG2016#BB18 MSUG FEB 2017 9
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.
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 (+o), 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

NOTE: There were 2819969 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 to the current date.
MPRINT(PRINT_LIST): proc sort data=SAFLIB.
MPRINT(PRINT_LIST): by proofs;
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.
Figure 2.2 Log Snapshot Two

Large DATA step MERGERING and PROC SORTING were fast.
III.2 Estimation

Figure 2.3 Log Snapshot Three

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

Figure 2.4 Log Snapshot Four

It used a few seconds or so per facility.
III.3 Log Analysis Utility
III.3 Log Analysis Utility

```plaintext
********** STEP 2**************;
** usage: ----------------------------------;
** %log_10_data(log= [results txt file from step 1 above].txt, 
   doc=[results txt file].txt);

**********************************************************************;

%macro log_10_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):')
```

#MWSUG2016#BB18   MSUG FEB 2017 19
III.3 Log Analysis Utility

```sas
input /
ctime = scan(_infile_, ".")
if index(ctime, ":") then do
  if countc(ctime, ":")=1 then ctime='0:\|ctime
ntime=Input(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 ' &060 DSN "---" ntime=mms8.2; *optional;
end;
else input;
end;
else input;
end;
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;
％end;
option mprint;
％let log1:"MWSUG16_BB18\MS_DFC_Patient_Lists_MWSUG.txt;
％let doc1:"MWSUG16_BB18\MS_DFC_Patient_Lists_MWSUG2.txt;
％log10 data(log1&log, doc1&doc);
** STEP 3 ***********************************
** summarize the results;
** ******************************************
proc means data=log_runtime mean max min sum;
var ntime obs;
types dsn procdat procdat*dsn;
run;
** ENDSAS **;
```
III.4 Log Analysis: Output 1

```
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

```plaintext
| 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.FACINFO -- ntime=0:19.53 |
| DATA: TIME | real time | 27.80 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_PLIST -- ntime=0:26.34 |
| SORT: TIME | real time | 29.15 seconds | 6,554,484 | WORK.M5_PLIST -- ntime=0:29.15 |
| SQL: TIME  | real time | 7.29 seconds  | 6,554,484 | WORK.M5_PLIST -- ntime=0:07.20 |
| DATA: TIME | real time | 0.00 seconds  | 6,554,484 | WORK.M5_PLIST -- ntime=0:00.00 |
| SQL: TIME  | real time | 6.75 seconds  | 6,554,484 | WORK.M5_PLIST -- ntime=0:06.75 |
| DATA: TIME | real time | 5.21 seconds  | 1,380  | WORK.PRULEVEL -- ntime=0:05.21 |
| PRINT: TIME| real time | 0.01 seconds  | 1,380  | WORK.PRULEVEL -- ntime=0:00.01 |
| DATA: TIME | real time | 5.21 seconds  | 720    | WORK.PRULEVEL -- ntime=0:05.21 |
| PRINT: TIME| real time | 0.00 seconds  | 720    | WORK.PRULEVEL -- ntime=0:00.00 |
```
### III.4 Log Analysis: Output 3

<table>
<thead>
<tr>
<th>dsn</th>
<th>PROCDAT</th>
<th>obs</th>
<th>ctime</th>
<th>ntime</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK.FACINFO</td>
<td>SORT:</td>
<td>6553</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td>WORK.TEMP</td>
<td>SORT:</td>
<td>6554484</td>
<td>19.53</td>
<td>19.53</td>
</tr>
<tr>
<td>SAFLIB.M5_PLIST_201607</td>
<td>DATA:</td>
<td>6554484</td>
<td>27.00</td>
<td>27</td>
</tr>
<tr>
<td>SAFLIB.M5_PLIST_201607</td>
<td>SORT:</td>
<td>6554484</td>
<td>18.59</td>
<td>18.59</td>
</tr>
<tr>
<td>WORK.M5_PTLIST</td>
<td>DATA:</td>
<td>6554484</td>
<td>26.34</td>
<td>26.34</td>
</tr>
<tr>
<td>WORK.M5_PTLIST</td>
<td>SORT:</td>
<td>6554484</td>
<td>29.15</td>
<td>29.15</td>
</tr>
<tr>
<td>WORK.M5_PTLIST</td>
<td>SQL:</td>
<td>6554484</td>
<td>7.20</td>
<td>7.2</td>
</tr>
<tr>
<td>WORK.M5_PTLIST</td>
<td>DATA:</td>
<td>6554484</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>WORK.M5_PTLIST</td>
<td>SQL:</td>
<td>6554484</td>
<td>6.75</td>
<td>6.75</td>
</tr>
</tbody>
</table>
III.5 Log Analysis: Statistics

<table>
<thead>
<tr>
<th>PROCDAT</th>
<th>dsn</th>
<th>N Obs</th>
<th>Variable</th>
<th>Mean</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA:</td>
<td>SAFLIB.M5.PLIST_201697</td>
<td>1</td>
<td>rtime</td>
<td>27.000000</td>
<td>27.000000</td>
<td>27.000000</td>
<td>27.000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cbs</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
</tr>
<tr>
<td></td>
<td>WORK.M5_PTLIST</td>
<td>2</td>
<td>rtime</td>
<td>13.170000</td>
<td>26.340000</td>
<td>0</td>
<td>26.340000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cbs</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
</tr>
<tr>
<td></td>
<td>WORK.PRLEVEL</td>
<td>6426</td>
<td>rtime</td>
<td>6.1020900</td>
<td>20.900000</td>
<td>5.160000</td>
<td>39211.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cbs</td>
<td>1019.99</td>
<td>6240.00</td>
<td>12.000000</td>
<td>6554484.00</td>
</tr>
<tr>
<td>PRNT:</td>
<td>WORK.PRLEVEL</td>
<td>6426</td>
<td>rtime</td>
<td>0.0006355</td>
<td>0.0700000</td>
<td>0</td>
<td>42.640000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cbs</td>
<td>1019.99</td>
<td>6240.00</td>
<td>12.000000</td>
<td>6554484.00</td>
</tr>
<tr>
<td>SORT:</td>
<td>SAFLIB.M5.PLIST_201697</td>
<td>1</td>
<td>rtime</td>
<td>18.590000</td>
<td>18.590000</td>
<td>18.590000</td>
<td>18.590000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cbs</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
</tr>
<tr>
<td></td>
<td>WORK.FACINFO</td>
<td>1</td>
<td>rtime</td>
<td>0.740000</td>
<td>0.740000</td>
<td>0.740000</td>
<td>0.740000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cbs</td>
<td>6553.00</td>
<td>6553.00</td>
<td>6553.00</td>
<td>6553.00</td>
</tr>
<tr>
<td></td>
<td>WORK.M5_PTLIST</td>
<td>1</td>
<td>rtime</td>
<td>29.150000</td>
<td>29.150000</td>
<td>29.150000</td>
<td>29.150000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cbs</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cbs</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
</tr>
<tr>
<td>SQL:</td>
<td>WORK.M5_PTLIST</td>
<td>2</td>
<td>rtime</td>
<td>6.9750000</td>
<td>7.2000000</td>
<td>6.7500000</td>
<td>13.950000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cbs</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>6554484.00</td>
<td>13108968.00</td>
</tr>
</tbody>
</table>

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
    ```plaintext
    PROC SQL CREATE VIEW
    QUIT;
    DATA . . ;
    BY FACID; ...;
    _FN=...FACID...;
    FILE WRITEOUT FILEVAR=_FN;
    PUT
    RUN;
    ```
One PROC SQL view to put 3 data sets together.

```sql
/*IV.2 New Code (part 1)*/

//**
1 /***************************************************************
2 Program Name: MS_PatList.sas
3 Purpose : Print facility patient list for MS measure for DFC
4 By : iqiluo@umich.edu 2016-04-25
5
6 Input : 1. measure results- QDFC.MS_patient_list
7 2. patient info - saflib.patients
8 3. facility info in _lookupdt
9
10 Output : 1. facility patient lists & output\MS_PatList_999999.1st
11 2. SAS dataset -- saflib.MS_plist
12
13 Note : A programmer with appropriate permissions must run this code
14 Make sure output folder has been created.
15
16 ***************************************************************/

%include "\disk\QDFC\code\qdfc_datedparam.sas";
options ls=80 ps=80 nodate nonumber source;

* Output for patient lists;
libname ptlists "\disk\QDFC\Patient_lists\refreshdt._release\dateit."
libname saflib "\disk\saflib";

*let output=\disk\QDFC\Patient_lists\refreshdt._release\dateit; 
*let output=\disk\QDFC\Patient_lists\refreshdt._release\dateit; 
*let output=\disk\QDFC\Patient_lists\refreshdt._release\dateit; 
*let output=\disk\QDFC\Patient_lists\refreshdt._release\dateit; 

-- put Mesures, Facinfo, and Patinfo together: 
proc sql;
create view MS_patlist as
select a.*,b.*,c.*
from QDFC.MS_patient_list (keep=patient_id facid &vars year month quarter) a
left join saflib.patients (keep=patient_id surname first_name m_initial ssn) b
on a.patient_id=b.patient_id 
join saflib.facinfo &lookupdt (keep=facid network prorown provcity DFC_report state wh
on a.facid=c.facid
order by a.facid, b.surname, b.first_name, a.year, a.month;
quit;
```
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.

<table>
<thead>
<tr>
<th>Restriction:</th>
<th>The value of a FILEVAR= variable is expressed as a character string that contains a physical filename.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction:</td>
<td>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.</td>
</tr>
<tr>
<td>Tip:</td>
<td>This variable, like automatic variables, is not written to the data set.</td>
</tr>
<tr>
<td>Tip:</td>
<td>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.</td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th>Restriction:</th>
<th>The first statement after the label must be an executable statement. Thereafter you can use any SAS statement.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restriction:</td>
<td>Use the HEADER= option only when you write to print files.</td>
</tr>
<tr>
<td>Tip:</td>
<td>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.</td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th>Alias</th>
<th>LL=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tip</td>
<td>This variable, like automatic variables, is not written to the data set.</td>
</tr>
<tr>
<td>Example</td>
<td><a href="#">Determining New Page by Lines Left on the Current Page</a></td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Alias</th>
<th>PS</th>
</tr>
</thead>
</table>

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
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
END
```
IV.5 Difference

New Code vs. Original Code
• 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

New Code vs. Original Code

- 79 lines vs. 150 lines
- 1 step vs. 6,384 steps
- 22,518,989 vs. 61,852,446 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