

# CT03: This Data Step Has Been Running for Ages – Can You ‘Progress’ It?

Jean-Michel  
Sr Statistical  
Programmer

# Disclaimers

- The views expressed hereafter are mine and do not necessarily reflect those of my company.
- Due to the short timeframe, I won't explain the details of all the statements, functions, options, or procedures I will mention. Please refer to my paper, SAS Help or other reference materials for details.



# Motivation

Incoming  
Summary  
Tables

Text files

SAS datasets




**VIEWTABLE: page10000**

Table 2.1.1  
Demographics and Baseline Characteristics by Period 2 Treatment Group  
Analysis Set: Safety Set

| Variable    | Statistic | PSD/PSD | IND 2.5mg QM | IND 4.5mg QM | IND 2.5mg QM + QM | IND 4.5mg QM + QM | IND 2.5mg QM + QM + QM | IND 4.5mg QM + QM + QM | All Subjects |
|-------------|-----------|---------|--------------|--------------|-------------------|-------------------|------------------------|------------------------|--------------|
| Height (cm) | n         | 40      | 40           | 40           | 40                | 40                | 40                     | 40                     | 40           |
|             | Mean      | 167.13  | 167.13       | 167.13       | 167.13            | 167.13            | 167.13                 | 167.13                 | 167.13       |
|             | SD        | 7.15    | 7.15         | 7.15         | 7.15              | 7.15              | 7.15                   | 7.15                   | 7.15         |
|             | Median    | 167.0   | 167.0        | 167.0        | 167.0             | 167.0             | 167.0                  | 167.0                  | 167.0        |
|             | Min       | 157.0   | 157.0        | 157.0        | 157.0             | 157.0             | 157.0                  | 157.0                  | 157.0        |
|             | Max       | 176.0   | 176.0        | 176.0        | 176.0             | 176.0             | 176.0                  | 176.0                  | 176.0        |

**VIEWTABLE: page10000**

Table 2.1.1  
Demographics and Baseline Characteristics by Period 2 Treatment Group  
Analysis Set: Safety Set

| Variable    | Statistic | PSD/PSD | IND 2.5mg QM | IND 4.5mg QM | IND 2.5mg QM + QM | IND 4.5mg QM + QM | IND 2.5mg QM + QM + QM | IND 4.5mg QM + QM + QM | All Subjects |
|-------------|-----------|---------|--------------|--------------|-------------------|-------------------|------------------------|------------------------|--------------|
| Height (cm) | n         | 40      | 40           | 40           | 40                | 40                | 40                     | 40                     | 40           |
|             | Mean      | 167.13  | 167.13       | 167.13       | 167.13            | 167.13            | 167.13                 | 167.13                 | 167.13       |
|             | SD        | 7.15    | 7.15         | 7.15         | 7.15              | 7.15              | 7.15                   | 7.15                   | 7.15         |
|             | Median    | 167.0   | 167.0        | 167.0        | 167.0             | 167.0             | 167.0                  | 167.0                  | 167.0        |
|             | Min       | 157.0   | 157.0        | 157.0        | 157.0             | 157.0             | 157.0                  | 157.0                  | 157.0        |
|             | Max       | 176.0   | 176.0        | 176.0        | 176.0             | 176.0             | 176.0                  | 176.0                  | 176.0        |

**VIEWTABLE: page10000**

Table 2.1.1  
Demographics and Baseline Characteristics by Period 2 Treatment Group  
Analysis Set: Safety Set

| Variable    | Statistic | PSD/PSD | IND 2.5mg QM | IND 4.5mg QM | IND 2.5mg QM + QM | IND 4.5mg QM + QM | IND 2.5mg QM + QM + QM | IND 4.5mg QM + QM + QM | All Subjects |
|-------------|-----------|---------|--------------|--------------|-------------------|-------------------|------------------------|------------------------|--------------|
| Height (cm) | n         | 40      | 40           | 40           | 40                | 40                | 40                     | 40                     | 40           |
|             | Mean      | 167.13  | 167.13       | 167.13       | 167.13            | 167.13            | 167.13                 | 167.13                 | 167.13       |
|             | SD        | 7.15    | 7.15         | 7.15         | 7.15              | 7.15              | 7.15                   | 7.15                   | 7.15         |
|             | Median    | 167.0   | 167.0        | 167.0        | 167.0             | 167.0             | 167.0                  | 167.0                  | 167.0        |
|             | Min       | 157.0   | 157.0        | 157.0        | 157.0             | 157.0             | 157.0                  | 157.0                  | 157.0        |
|             | Max       | 176.0   | 176.0        | 176.0        | 176.0             | 176.0             | 176.0                  | 176.0                  | 176.0        |



Identify column headers based on number & max length of "blank fields" (  $\geq 2$  spaces)

| Nun<br>Pages | Line<br>num | Line                                                                                                                 | Block    | Sub<br>Block | Blank<br>Fields<br>Nun | Max<br>Blank<br>Field<br>Width | p |
|--------------|-------------|----------------------------------------------------------------------------------------------------------------------|----------|--------------|------------------------|--------------------------------|---|
| 10000        | 71          | BMI=Body Mass Index, BSA=Body Surface Area, IND=Investigational New Drug, IA=Interim Analysis, F=Female, M=Male,     | FOOTNOTE |              | 0                      | 0                              | 2 |
| 10000        | 72          | Max=Maximum, Min=Minimum, PBO=Placebo, Q4H=Every 4 Hours, Q8H=Every 8 Hours, SD=Standard Deviation, pctl=Percentile. | FOOTNOTE |              | 0                      | 0                              | 2 |
| 10000        | 73          | Note: Only data assessed during blinded treatment is presented in this table.                                        | FOOTNOTE |              | 0                      | 0                              | 2 |
| 10000        | 74          | References: Listing 2.1.1, Listing 2.3.1                                                                             | FOOTNOTE |              | 0                      | 0                              | 2 |
| 10000        | 75          |                                                                                                                      | FOOTNOTE | EMPTY        | 0                      | 0                              | 2 |
| 10000        | 76          | Program: t_dn211, 2019-09-01 at 14:24                                                                                | FOOTNOTE |              | 1                      | 69                             | 2 |
| 10000        | 77          | Page 2 of 10000                                                                                                      | FOOTNOTE | EMPTY        | 0                      | 0                              | 2 |
| 10000        | 78          |                                                                                                                      | FOOTNOTE |              | 0                      | 0                              | 2 |
| 10000        | 79          | SPONSOR                                                                                                              | TITLE    |              | 1                      | 103                            | 3 |
| 10000        | 80          | CONFIDENTIAL                                                                                                         | TITLE    |              | 1                      | 80                             | 3 |
| 10000        | 81          | INVESTIGATIONAL NEW DRUG / Indication                                                                                | TITLE    |              | 0                      | 0                              | 3 |
| 10000        | 82          | ST0012                                                                                                               | TITLE    |              | 0                      | 0                              | 3 |
| 10000        | 83          | Table 2.1.1                                                                                                          | TITLE    |              | 0                      | 0                              | 3 |
| 10000        | 84          | Demographics and Baseline Characteristics by Period 2 Treatment Group                                                | TITLE    |              | 0                      | 0                              | 3 |
| 10000        | 85          | Analysis Set: Safety Set                                                                                             | TITLE    |              | 0                      | 0                              | 3 |
| 10000        | 86          |                                                                                                                      | TITLE    | EMPTY        | 0                      | 0                              | 3 |
| 10000        | 87          | Period 2 Treatment Group                                                                                             | HEADER   |              | 0                      | 0                              | 3 |
| 10000        | 88          |                                                                                                                      | HEADER   | LINE         | 0                      | 0                              | 3 |
| 10000        | 89          |                                                                                                                      | HEADER   | COLHEADER    | 3                      | 4                              | 3 |
| 10000        | 90          |                                                                                                                      | HEADER   | COLHEADER    | 3                      | 9                              | 3 |
| 10000        | 91          |                                                                                                                      | HEADER   | COLHEADER    | 3                      | 4                              | 3 |
| 10000        | 92          |                                                                                                                      | HEADER   | COLHEADER    | 5                      | 10                             | 3 |
| 10000        | 93          |                                                                                                                      | HEADER   | COLHEADER    | 7                      | 21                             | 3 |
| 10000        | 94          |                                                                                                                      | HEADER   | EMPTY        | 0                      | 0                              | 3 |
| 10000        | 95          | Height (kg)                                                                                                          | BODY     |              | 0                      | 0                              | 3 |
| 10000        | 96          | Min - 33.3pctl                                                                                                       | BODY     |              | 12                     | 11                             | 3 |

# Pattern matching with CALL PRXNEXT

```
DATA text2;  
SET text1;  
pattern = '/(?<!\s)\s{2,}(?! \s)';
```

```
prx = PRXPARSE(pattern);  
start = 1; stop = lengthn(text); pos = 0; len = 0;  
matchnum = 0;  
maxlen = 0;  
if (stop > 0) then do until(pos = 0);  
    CALL PRXNEXT(prx, start, stop, text, pos, len);  
    if (len>0) then matchnum + 1; *- count the number of matches -*;  
    if (len>maxlen) then maxlen = len; *- store the max length -*;  
end;
```

```
RUN;
```

2 or more spaces, not immediately following a space, nor immediately followed by a space

PRXPARSE compiles a regular expression (pattern) that can then be used for pattern matching.

CALL PRXNEXT searches a string 'text' for a pattern match multiple times, between the 'start' and the 'stop' positions. When a match is found, the position 'pos' and length 'len' of the matched substring are updated; the 'start' for the next search is positioned after the end of the match. When no match is found, the position 'pos' and length 'len' are set to 0.



# PROC FCMP: Define custom functions (1)

```
PROC FCMP outlib=work.functions.prx;  
  function PRXNMATCH(pattern $, text $) ;  
  
    prx=PRXPARSE(pattern) ;  
    start = 1; stop = lengthn(text); pos = 0; len = 0;  
    matchnum = 0;  
  
    if (stop > 0) then do until(pos = 0);  
      CALL PRXNEXT(prx, start, stop, text, pos, len);  
      if (len>0) then matchnum+1; *- count the number of matches -*;  
    end;  
    RETURN(matchnum); *- return the number of matches -*;  
  
  endsub;  
  
RUN; QUIT; option cmplib = work.functions;
```



# PROC FCMP: Define custom functions (2)

```
PROC FCMP outlib=work.functions.prx;

function PRXMAXMATCHLEN(pattern $, text $) ;
    prx=PRXPARSE(pattern) ;
    start = 1; stop = lengthn(text); pos = 0; len = 0;

    maxlen = 0;
    if (stop > 0) then do until(pos = 0);
        CALL PRXNEXT(prx, start, stop, text, pos, len);

        if (len>maxlen) then maxlen = len; *- store the max length -*;
    end;

    RETURN(maxlen); *- return the maximum length of all matches -*;

endsub;
RUN; QUIT; option cmplib = work.functions;
```



# Data Step calling FCMP functions

```
DATA text2;  
  SET text1;
```

```
  *- number of 'blank' fields >=2 consecutive spaces in current LINE -*;  
  blankFieldsNum = PRXNMATCH('/(?<!\s)\s{2,}(?!\\s)/', cats(line));
```

```
  *- max width of all 'blank' fields of >=2 consecutive spaces -*;  
  maxBlFieldWidth = PRXMAXMATCHLEN('/(?<!\s)\s{2,}(?!\\s)/', cats(line));
```

```
RUN;
```

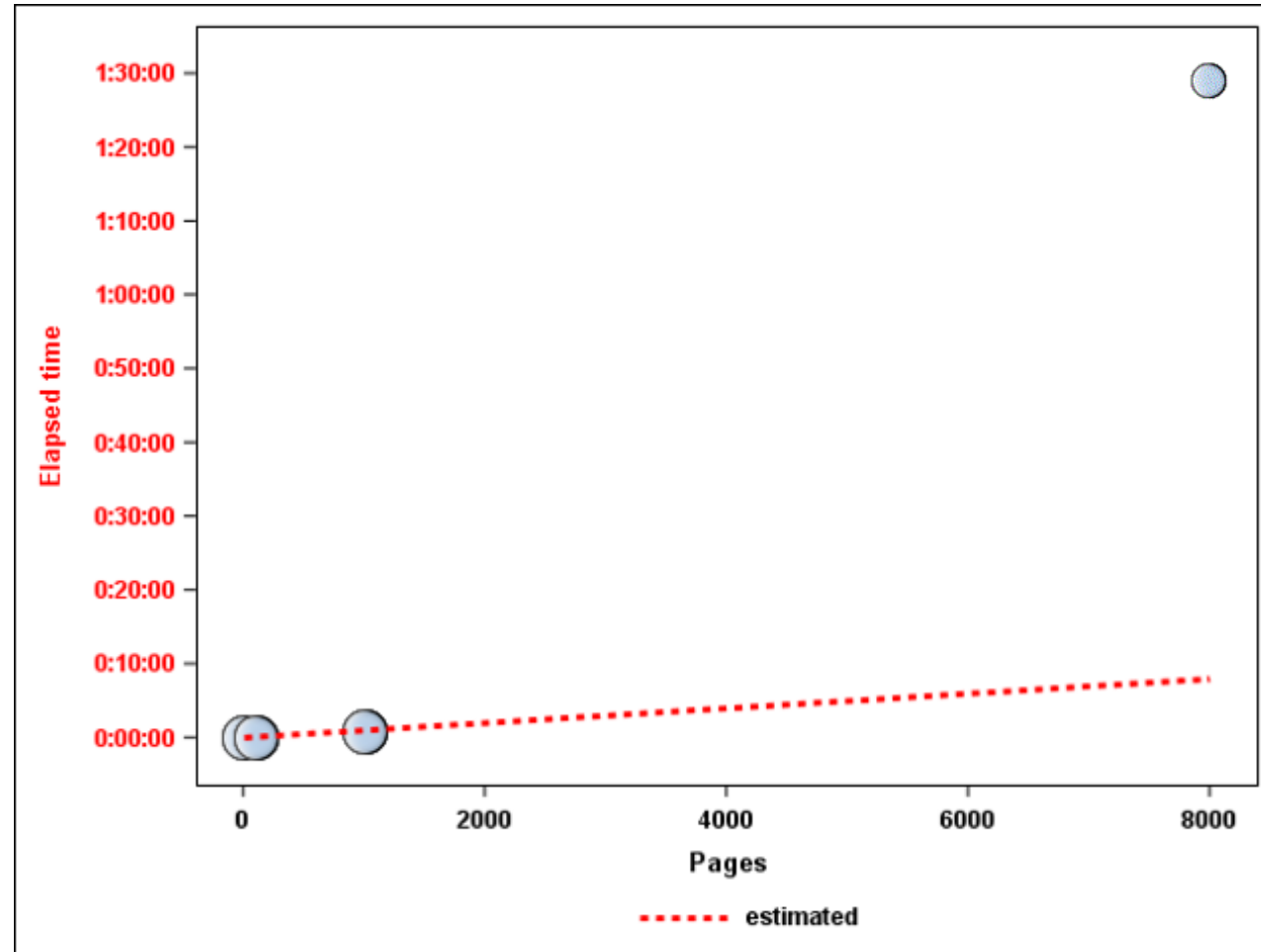
The use of FCMP functions is advocated in SAS Online Help as a way to keep code blocks **independent, reusable** and making a program **easier** to read, write, and modify.

Most **DATA step statements and functions** can be used in PROC FCMP functions and routines. PROC FCMP has however some (*scarcely documented*) **limitations**.



# Problem: Scalability

Run times increase  
unexpectedly



## Initial run

- 150 pages
- < 1 sec

## Intermediate run

- 1000 pages
- < 1 min

## Final run

- 8000 pages
- > 1 h, still running!

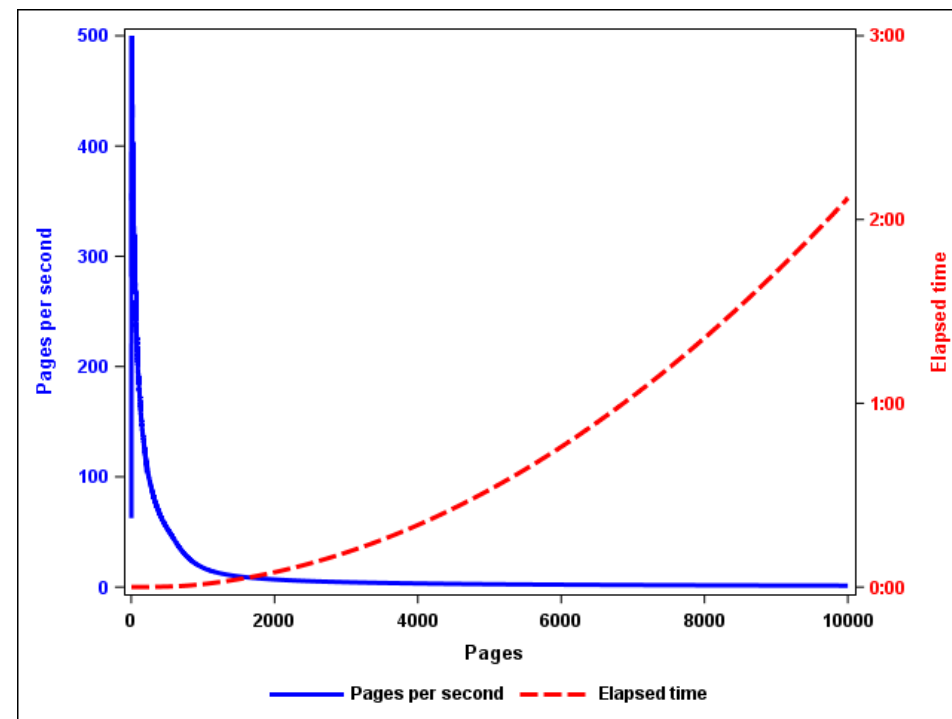
# Minimum Reproducible Example

Code on PhUSE wiki:  
[https://www.phusewiki.org/wiki/index.php?title=SA\\_S\\_macro\\_%25progress](https://www.phusewiki.org/wiki/index.php?title=SA_S_macro_%25progress)

## Create dataset with a dummy page

Expand to 10,000 p

Run on N pages  
Collect run times



## Plot the data

[illegible]

**Expanding to 10,000 pages**

```

* duplicate the page data 10,000 times by re-reading the page dataset repeatedly -*;
data page10000(drop = start elapsed);
  label p = 'page number' l = 'line number';
  put nob5;
  start = time();
  do p = 1 to 10000; /* loop over the desired number of pages */
    elapsed = time()-start;
    if mod(p, 1000) = 0 then put = elapsed= time12.3;
    do l = 1 to nob5; /* loop over each line of the page */
      set page key = l nob5 = nob5; /* the key option allows reading the matching observation using the dataset index */
      if l = nob5 then substr(line, 105) = put(cat(' ', 'Page', p, 'of', 10000), $20.-r); /* replace "Page x of y" by the actual numbers */
    end;
  end;
  stop;
run;

```

[illegible]

# Real-time feed-back in the Data Step

```
%macro progress(counter=_n_, check_cond=1, every_sec = 10, total=., ref=.);  
  retain _start_time _check_time _check_count _check_speed _remain_time;  
  if _start_time = . then do;  
    _start_time = time();    _check_time = time() - &every_sec +1;  
    _check_count = &counter; _check_speed = .X;  
  end;  
  _elapsed_time = time() - _start_time;  
  if (time() > _check_time + &every_sec) and (&check_cond) then do;  
    _check_speed = round((&counter - _check_count) / (time() - _check_time), .1);  
    _check_time = time(); _check_count = &counter; _msg_now = "Y";  
  end;  
  if &total>0 and _check_speed>0  
    then _remain_time = (&total - &counter) / _check_speed;  
  _msg_ = catx(" ", "&counter:", &counter, "/", &total, "- Elapsed:",  
               put(_elapsed_time, time8.), "- Remains:", put(_remain_time,  
               time8.), "- Speed:", _check_speed, "/s", "- Ref:", &ref);  
  if getoption("DMS") = "DMS" then do;  
    window progresswin rows=1 columns=100 #1 @3 _msg_;  
    display progresswin noinput;  
  end; else if _msg_now="Y" then put _msg_;  
%mend progress;
```

By default, the **speed** is recomputed and messages are written to the log at most **every 10 seconds**.

Interactive **Display Manager** allows **real-time feed-back**

**SAS Batch, Studio, EG** only allow log messages



# Real-time feed-back: Display Manager

The screenshot shows the SAS Display Manager interface. At the top, a taskbar lists system metrics: Image Name (sas.exe), User Name (E618655), CPU (17), Memory (Private Working Set) (144,308 K), Page Faults (6,935,410), PF Delta (88,108), and Description (SAS 9.4 for Windows). Below this is the SAS menu bar (File, Edit, View, Tools, Run, Solutions, Window, Help) and a toolbar. The main window is divided into several panes. The 'Log - (Untitled)' pane shows system statistics: memory (22065.78k), OS Memory (35168.00k), Timestamp (10/14/2019 08:10:06 AM), and Step Count (6 Switch Count 0). Below the log, a code editor shows a data step: 

```
172 %process(pages = 1000);
```

```
206 %let pages=1000;
```

```
207 %let pages=1000;
```

```
208 data text2;
```

```
209   set page10000 (where = ( p <= &pages ));
```

```
210   by p 1;
```

```
211   blankFieldsNum = PRXNMATCH('/(?<!\s)\s{2,}(?!\\s)/', cats(line));
```

```
212   maxBlFieldWidth = PRXMAXMATCHLEN('/(?<!\s)\s{2,}(?!\\s)/', cats(line));
```

```
213   %progress(counter = p, check_cond = last.p, total = &pages );
```

```
214 run;
```

The 'CT03 - Minimum Reproducible Example.sas DATA STEP running' pane shows the same code. The bottom status bar indicates 'NOTE: 316 Lines Submitted.' and 'Ln 213, Col 6'.

Log - (Untitled)

```
memory          22065.78k
OS Memory       35168.00k
Timestamp       10/14/2019 08:10:06 AM
Step Count      6 Switch Count 0
```

```
172 %process(pages = 1000);
```

CT03 - Minimum Reproducible Example.sas DATA STEP running

```
206 %let pages=1000;
```

```
207 %let pages=1000;
```

```
208 data text2;
```

```
209   set page10000 (where = ( p <= &pages ));
```

```
210   by p 1;
```

```
211   blankFieldsNum = PRXNMATCH('/(?<!\s)\s{2,}(?!\\s)/', cats(line));
```

```
212   maxBlFieldWidth = PRXMAXMATCHLEN('/(?<!\s)\s{2,}(?!\\s)/', cats(line));
```

```
213   %progress(counter = p, check_cond = last.p, total = &pages );
```

```
214 run;
```

NOTE: 316 Lines Submitted. Ln 213, Col 6

In Interactive **Display Manager**, Data Step statements **WINDOW** and **DISPLAY** (or macro statements **%WINDOW** and **%DISPLAY**) allow status messages to be displayed in a specific [window](#).

If at the same time the special data step variable **\_msg\_** (or macro-variable **&sysmsg**) is defined, its value is displayed in the [status line](#).

Both can be updated at each iteration.

Macro **%progress** is called at the end of the data step

# Stop-and-restart loop

```
%macro progress(counter=_n_, check_cond=1, every_sec = 10, total=.,  
  stop_pct_speed = 10, stop_statement = %quote(stop;), stop_counter = stop_n);  
  retain _start_time _check_time _check_count _check_speed _remain_time  
         _max_speed _pct_max_speed;  
  [... keep previously described code ...]  
  if (time() > _check_time + &every_sec) and (&check_cond) then do;  
    [... keep previously described code ...]  
    if _check_speed > _max_speed then _max_speed = _check_speed;  
    _pct_max_speed = 100 * _check_speed / _max_speed;  
  
  end;  
  [... keep previously  
    described code ...]  
  if (&check_cond)  
    then call symputx("&stop_counter", &counter.);  
  if .Z < _pct_max_speed <= &stop_pct_speed then do;  
    put "Not" "ice: Stopping at " _msg_;  
    %unquote(%superq(stop_statement));  
  end;  
%mend progress;
```

Retain the maximum speed observed **\_max\_speed**.  
Compute **\_pct\_max\_speed** in % of the maximum speed observed.

Save the counter value to the macro-variable specified by **stop\_counter**.

Execute the **stop\_statement** as soon as the current speed drops below the **&stop\_pct\_speed** threshold.



# Stop-and-restart loop

```
%macro stop_and_restart(pg = 10000);  
  %let compl_pages=0; %let iter=0; %let start_t=%sysfunc(time());  
  proc datasets nolist; delete text2; quit; *- first append to empty dataset -*;  
  %do %until(&compl_pages >= &pg);  
    %let iter = %eval(&iter + 1);  
    data __text2;  
      set page10000 (where=(&compl_pages< p<=&pg));  
      by p 1;  
      blankFieldsNum = PRXNMATCH('/(?<!\s)\s{2,}(?!\\s)/', cats(line));  
      maxBlFieldWidth = PRXMAXMATCHLEN('/(?<!\s)\s{2,}(?!\\s)/', cats(line));  
      output; *- ouput last obs before stop -*;  
      %progress(counter=p, check_cond=last.p, total=&pg, stop_pct_speed=30  
        ,stop_counter=compl_pages, ref=&iter);  
    run;  
    proc append base=text2 data=__text2; run;  
  %end;  
  %let eltm=%sysevalf(%sysfunc(time())-&start_t); %let spd=%sysevalf(&pg/&eltm);  
  %put Processed &pg pages in &iter iterations, &eltm sec, speed: &spd p/sec.;  
%mend; %stop_and_restart;
```

Start data step with first page not yet processed.

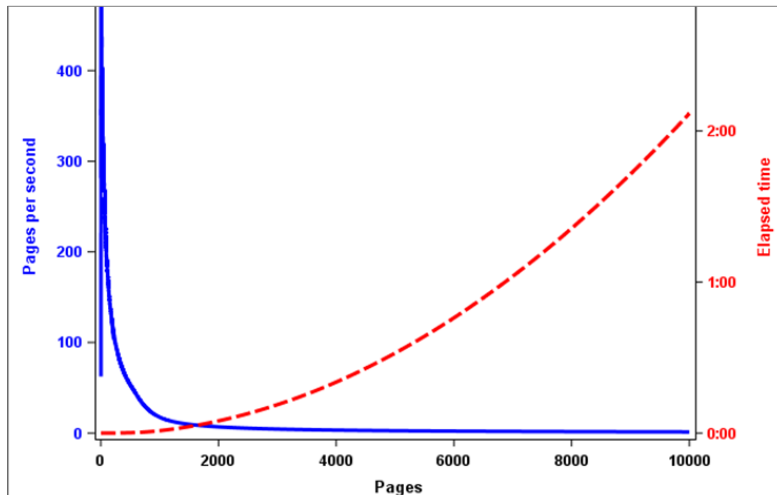
Stop when speed drops < 30% of max;  
append processed pages; iterate and  
restart with first page not yet processed.

Report overall duration and speed.



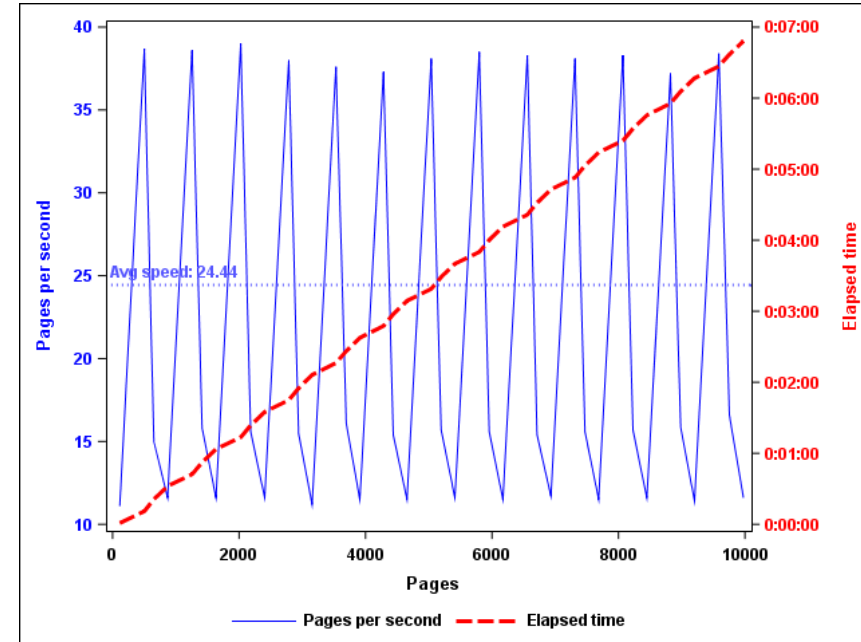
## Solution

# Stop-and-restart loop: results



Originally:

10,000 pages  
in 02:06:56  
Avg: 1.3 p./sec



Now in 00:06:48  
14 iterations  
18.7 x faster  
Avg: 25 p. / sec

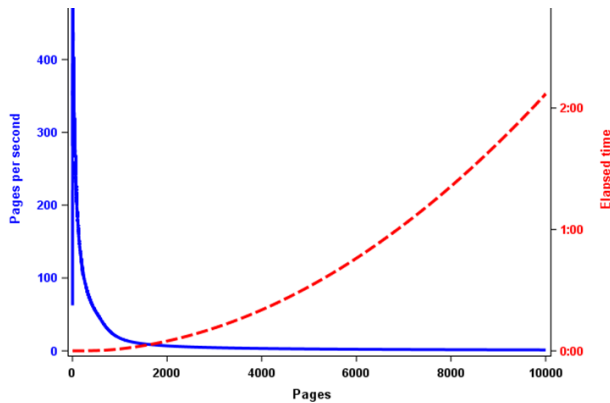
Speed is high at start of each iteration,  
then decreases

Avg speed over iterations does  
not decrease over time



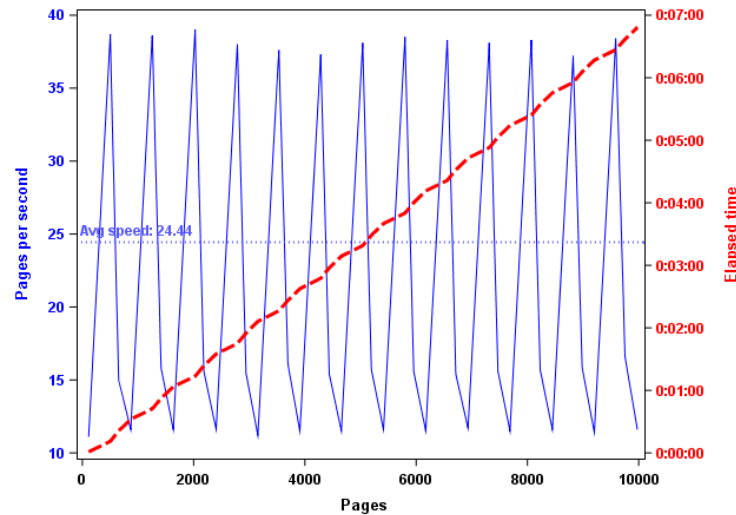
## Optimized Solution

# Optimized FCMP functions: results



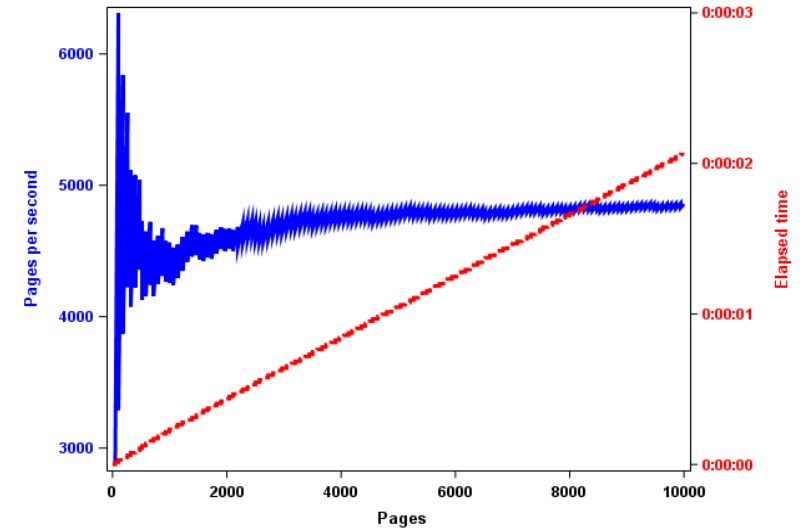
### Originally

- 10,000 pages
- in 02:06:56
- Avg: 1.3 p./sec



### Stop-and-Restart loop

- 14 iterations
- Avg: 25 p./sec
- 18.7 x faster
- Avg speed (per iteration) stable over time



### Optimized FCMP functions

- Avg: 4854 p./sec
- 198 x faster than Stop-and-Restart loop
- 3700 x faster than originally
- Avg speed stable over time



- Monitoring the progress of SAS execution can be done
  - In **Data Step**, **Macro** (loop), and a few **Procedures** supporting FCMP functions.
  - **Real-time feed-back** can be provided, especially with the interactive Display Manager.
- Monitoring the execution speed in a data step within a **Stop-and-Restart loop** can lead to a large **gain in performance**
  - when the original processing speed *decreases over time*
  - without identifying the *root cause* nor optimizing the code.
- Identifying **the Root Cause** and **Optimizing the code** are still worth doing
  - when the process has to be repeated time and again.
- **PROC FCMP functions**, though powerful, are *NOT as much optimized for ease of use* as the data step equivalent code
  - they have *scarcely documented limitations*
  - some *improvements* by the SAS Institute would be much appreciated.

# THANK YOU

**ROI<sup>2</sup>**  
return on investment  
through  
return on information



• Jean-Michel Bodart

 **Business & Decision**  
Life Sciences





# Q&

# A



# Backup Slides



# Live Feed-Back from SAS PROCs through FCMP functions

- in **PROC SQL** *SELECT STATEMENT*
- in **PROC REPORT** *COMPUTE BLOCK*
- Write to the LOG
  - using **PUT** statement - caution:
    - *limited functionalities* (compared to Data Step)
    - *undocumented*: **'%'** signs in strings must be doubled (otherwise they get replaced by unpredictable text)
- Write to an External File or Device<sup>1</sup>
  - using **FILENAME()**, **FOPEN()**, **FPUT()**, **FWRITE()** and **FCLOSE()** functions
  - device type 'TEMP' (temporary file)
    - viewed in external window using
      - **"tail -f"** command (\*Unix)
      - Powershell cmdlet **"Get-Content -wait"** (Windows)
  - device type 'TERMINAL'
    - this opens a window titled **"SAS MESSAGE LOG"** with the last 20 lines visible (700-800 previous lines visible by scrolling back)

<sup>(1)</sup> specify 'device type' as 3<sup>rd</sup> argument to FILENAME function



# Live Feed-Back from SAS macro loops

PROC PRINTTO Log redirection temporarily disabled to display messages

The screenshot displays the SAS Studio environment. The main editor window shows a SAS macro with a loop. A callout box indicates that PROC PRINTTO log redirection is temporarily disabled to display messages. The macro code includes a PROC Gplot and a loop that uses %MESSAGE and %PUT to display debug information. A second callout box explains that %WINDOW and %DISPLAY write messages to the DMS status line. The status window on the right shows the execution progress, including the start time and the current step.

```
Log - (Untitled)
6238! visit) -*;

MPrint(T):  *- Redirecting log to a file to avoid log window fill
MPrint(T):  proc printto log="f_compk_ps3c_printto.log" new;
MPrint(T):  run;

f_compk_ps3c.sas  DATA STEP running

1317 * PROC Gplot *
1318 *-----*
1319 %let d=%sysfunc(putn("&sysdate9."d, e8601da.));
1320 %let t=%sysfunc(putn("&sysdate9."t, tod5.));
1321
1322
1323 %do i = 1 %to &nsubj ; /* subject do loop */
1324     %* _message(%sysfunc(datetime(), E8601DT.): Processing S
1325     %_message(%sysfunc(putn("&sysdate9."d, e8601da.));
1326     %PUT "DEBUG: SUBJECT &i of &nsubj: &&sub&i";
1327
1328     %if &i=1 %then %do; /*- JMB 2019-03-04 - this does not

STATUSUPDT - #LN00401
--- STATUSUPDT started on 2019-10-21T10:16:06 ---
LAST WARNING:
LAST ERROR:
-- 2019-10-21T10:16:06 -- Running program f_compk_ps3c
```

PowerShell cmdlet "Get-Content -wait" shows messages written to a TEMP external file, including last Warning and Error messages from &syserrortext and &syswarningtext

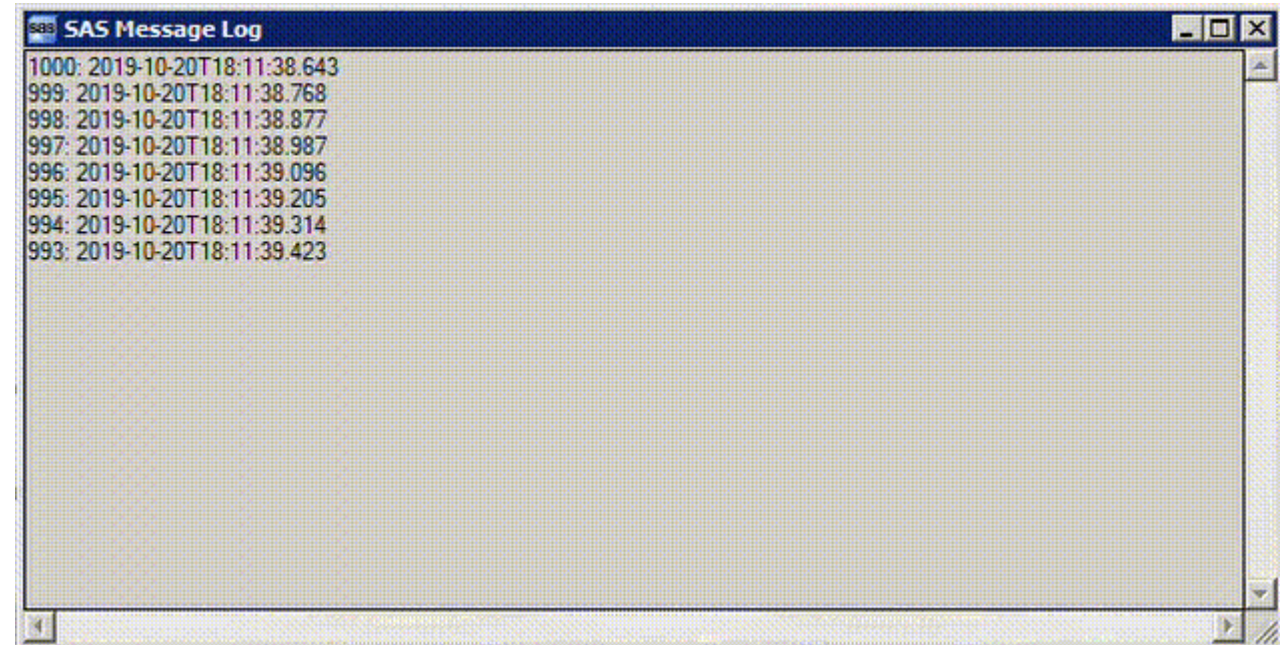
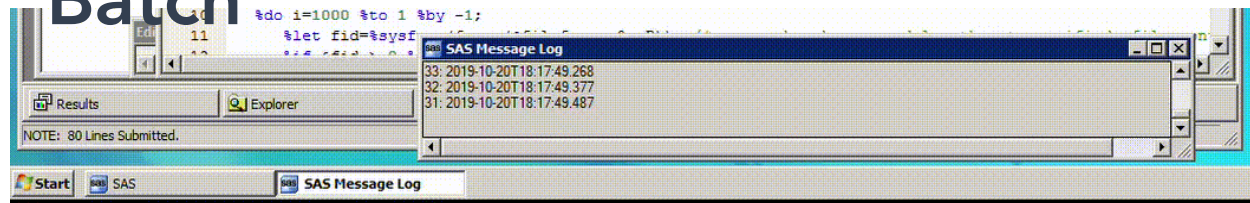
%WINDOW and %DISPLAY write messages to DMS status line



# Writing to device type 'TERMINAL'

- A window **separate** from the main SAS DMS window
- Can only **append** to it
- **Pops up** to the front when written to
- Contents **scroll up** when full
- Also with 'local' SAS

## Batch



# Live Feed-Back from SAS macro (loops)

- Write to an External File or Device<sup>1</sup>
  - using `FILENAME()`, `FOPEN()`, `FPUT()`, `FWRITE()` and `FCLOSE()` functions called via `%sysfunc(FUNCTION())`
  - device type 'TEMP' (temporary file)
    - viewed in external window using
      - "tail -f" command (\*Unix)
      - Powershell cmdlet "Get-Content -wait" (Windows)
  - device type 'TERMINAL'
    - This opens a window titled "SAS MESSAGE LOG" with the last 20 lines visible (700-800 previous lines visible by scrolling back)
- Write to the LOG
  - using `%PUT` statement
    - Log redirection through `PROC PRINTTO` can be *temporarily disabled* to display the message, and then re-established
- Write to specific windows - only with **SAS Display Manager** (interactive "SAS base" environment)
  - Using `%WINDOW` and `%DISPLAY` statements

(<sup>1</sup>) specify 'device type' as 3<sup>rd</sup> argument of FILENAME function



# Useful additional live feedback

- (Last) **Error Message** issued
- Retrieved from automatic macro-variable **&syserrortext**
- **Not** in interactive **DM status line** – a multiline window is needed
- Only print *messages different from the previous one*
- (Last) **Warning Message** issued
- Retrieved from automatic macro-variable **&syswarningtext**

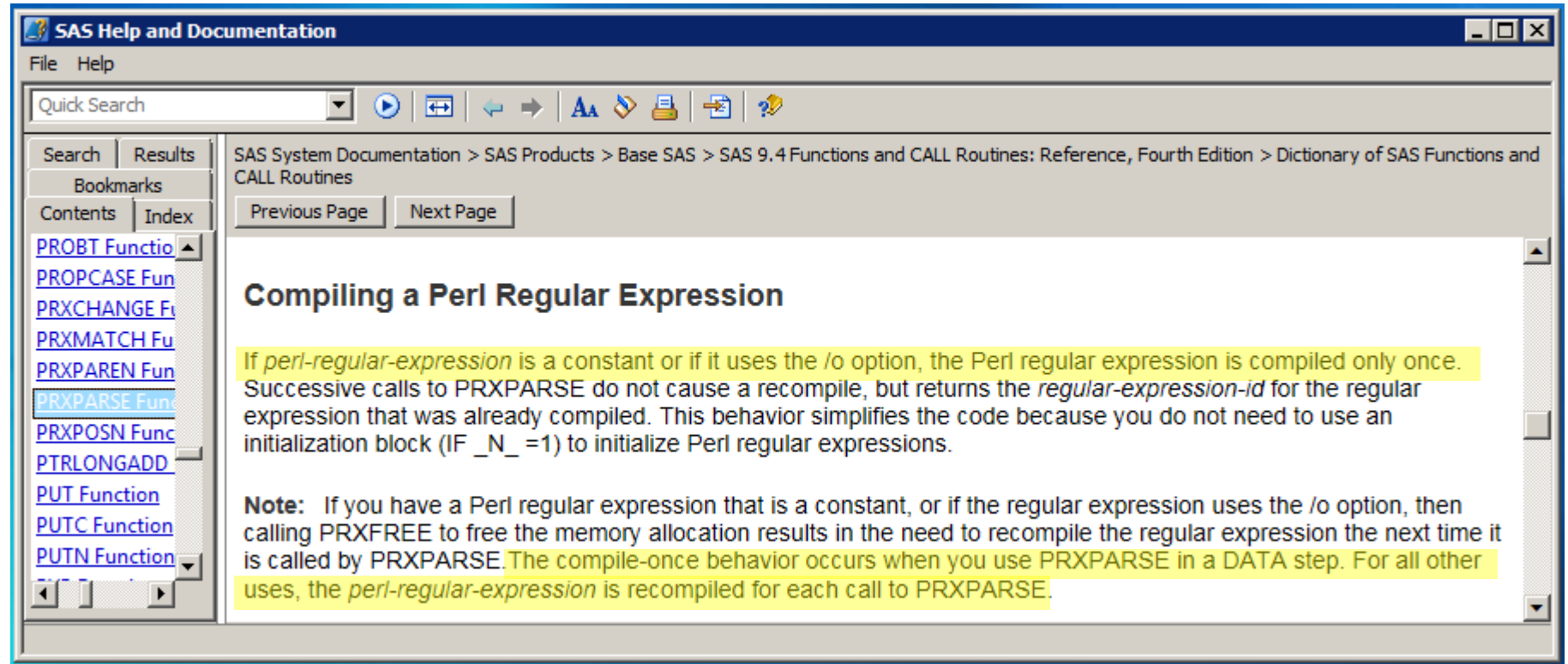
In interactive sessions it's possible to reset those macro-variables to a blank value before submitting a new program by printing ***an empty error or warning message*** to the log (after temporarily re-directing it to a temporary file - so the empty message does not show up in the 'main' log).

```
%put ER%bquote (ROR:  ) ;    *- >=2 spaces are necessary after ':' -*;
%put WAR%bquote (NING:  ) ;
```



# Root Cause

- **Compiling Perl Regular Expressions (RegEx) eats up memory**
- When short of memory, the system starts **swapping** memory blocks **to disk** – a much slower way of allocating memory



## Data Step

Constant RegEx compiled only once

- Memory is **freed automatically** at the end of the Data Step.  
*Note: in SAS Help, the example for PRXPARSE() does not include a call to PRXFREE() in the Data Step.*

## FCMP Function

RegEx passed as a **variable** (though its value remains constant)

- **Recompiled** at each call
- Memory is **never freed up** until end of SAS session (CALL PRXFREE not called)

## Optimized FCMP Function

- **previously used RegEx** (character) and corresponding compiled RegEx IDs (numeric) are stored in 2 **STATIC\* arrays** (\*: i.e. their values are retained across calls)
- At each call, the character RegEx passed as argument is **compared to the stored values** in the array; if found, the corresponding compiled RegEx ID is used; otherwise, the **new RegEx is compiled and added to both STATIC arrays** for reuse. When arrays are full, old RegEx entries are discarded and memory is released with CALL PRXFREE.
- max **number of RegEx stored in memory** = arrays size (e.g.  $n = 10$ )

# THANK YOU

**ROI<sup>2</sup>**  
return on investment  
through  
return on information



• Jean-Michel Bodart

 **Business & Decision**  
Life Sciences

