



SAS Coding Efficiency

Developing a Performance Mindset



Richard Paterson



15th–19th November

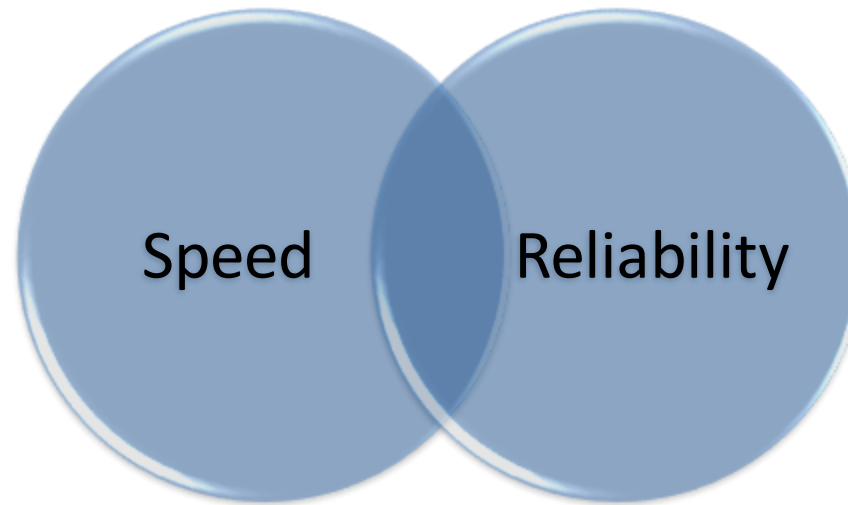
EU Connect 2021

The Clinical Data
Science Conference

What is Coding Efficiency?



- efficiency = speed of execution + reliability





What is a Performance Mindset?

- Begin with performance as main objective
- Build code with a long-term vision – maintainable, scalable
- Strive for more than 'good enough'



Is Coding Efficiency Still Relevant?

- Computer power is still increasing – Moore's Law, right?
- Storage space is seemingly endless
- Cloud computing

- Wirth's Law
software is getting slower more rapidly than hardware is becoming faster.
- Gates Law
The speed of software halves every 18 months



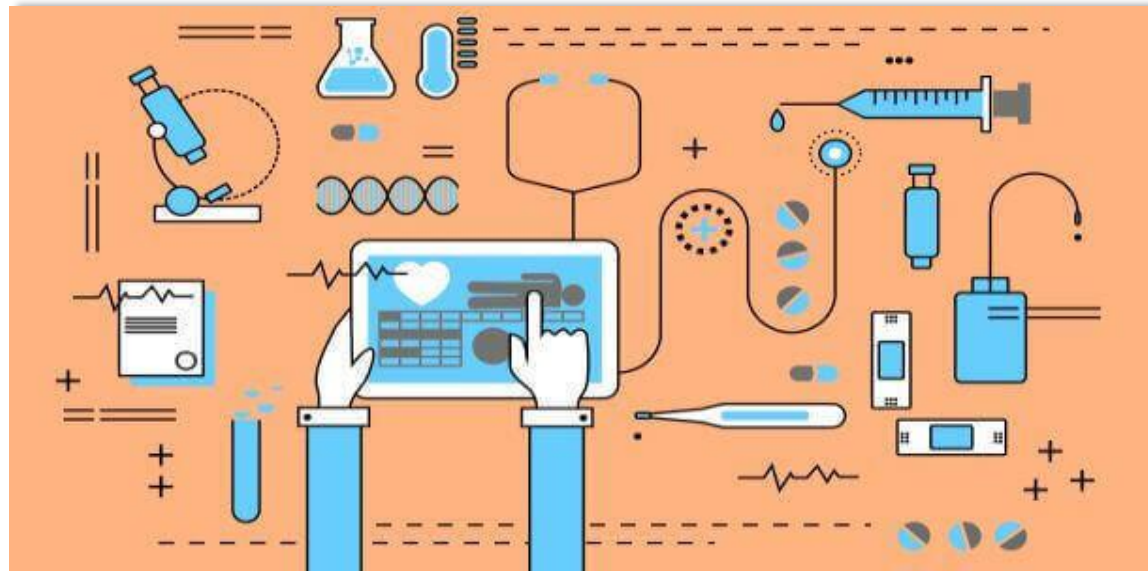
Life Sciences Challenges



“Data Deluge”

Increasing number of data sources

Growing pressures tighter deadlines



Performance Mindset #1: Control the data



Read Only What You Need



Questions before writing SAS code

- Do I need this dataset?
- Do I **need** to read **all** the rows in this dataset?
- Do I **need** to read **all** the columns in this dataset?
- Do I have this data in the current session? Can I reuse?
- Do I **need** to write **all** the rows to an output dataset?
- Do I **need** to write **all** the columns to an output dataset?



**To Read What You Need
You must first know your data**

Get Acquainted with Your Data



```
Proc contents data=db.patients;  
Run;
```

- Physical size
- Number of records
- Length of variables
- Indexes
- Compression
- Sorted

The CONTENTS Procedure

Data Set Name	HOME.DEMO	Observations	100000
Member Type	DATA	Variables	106
Engine	V9	Indexes	0
Created	07/06/2021 16:13:18	Observation Length	40808
Last Modified	07/06/2021 16:13:18	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information

Data Set Page Size	245760
Number of Data Set Pages	16667
First Data Page	1
Max Obs per Page	6
Obs in First Data Page	5
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\home\demo.sas7bdat
Release Created	9.0401M7
Host Created	X64_10PRO
Owner Name	OCS\vp1
File Size	4GB
File Size (bytes)	4096327680

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format
102	char1	Char	8000	\$8000.
103	char2	Char	8000	\$8000.
104	char3	Char	8000	\$8000.
105	char4	Char	8000	\$8000.
106	char5	Char	8000	\$8000.



Use the OBS Option

```
OPTIONS OBS=1000;
```

For all records. Turnoff with OBS=MAX.

```
DATA sample;  
  SET source (OBS=1000);  
  Other SAS Statements...  
RUN;
```

Allows for quickly checking of code.

Allow quick check of disk space used

Is your program slow with reduced records then something maybe wrong...



Use KEEP and DROP



```
PROC SQL;  
  create table demo as  
  select * FROM bigtable;  
QUIT;
```

```
proc sql noprint;  
  create table keep1 as  
  select *  
  from demo(keep=char1-char5 num1-num5);  
quit;
```

real time

20.20 seconds

```
proc sql noprint;  
  create table keep1 as  
  select char1, char2, char3, char4,char5,  
  num1, num2, num3, num4, num5  
  from demo(keep=char1-char5 num1-num5);  
quit;
```

real time

15.03 seconds



Use KEEP and DROP

- Using the KEEP= data set option instructs the SAS System to load only the specified variables into the program data vector (PDV), eliminating all other variables from being loaded.
- Use only the variables that you need.
- Using DROP= and KEEP= on the input data sets is more efficient than using them as program statements because they don't bring the unneeded variables into the workspace

```
DATA keep1;  
  SET demo (KEEP = char: num1-num10);  
RUN;
```

real time
20.87 seconds

```
DATA sample;  
  SET source;  
  KEEP char: num1-num10;  
RUN;
```

real time
35.52 seconds

OPTIONS FULLSTIMER



```
NOTE: The data set WORK.INCLUSION01 has 6 observations and 3 variables.
NOTE: Compressing data set WORK.INCLUSION01 increased size by 100.00 percent.
      Compressed is 2 pages; un-compressed would require 1 pages.
NOTE: DATA statement used (Total process time):
      real time           0.05 seconds
      user cpu time       0.00 seconds
      system cpu time     0.03 seconds
      memory             1187.00k
      OS Memory          8536.00k
      Timestamp           02/02/2021 08:13:58 PM
      Step Count          1      Switch Count    0
```

Statistic	Description
Real-Time	the amount of time spent to process the SAS job.
User CPU Time	the CPU time spent to execute SAS code.
System CPU Time	the CPU time spent to perform operating system tasks
Memory	the amount of memory required to run a step.
OS Memory	the maximum amount of memory that a step requested from the System.

If **realtime** is much higher than user **cpu time** are you reading in too much data?



COMPRESS option and statement

```
options compress=yes;
```

- Reduced disk space
- Reduce time for reading data – huge benefits for large datasets.
- Requires more CPU to uncompress.
- More overhead for smaller datasets

```
data demo_compress(compress=yes);  
  set demo;  
run;
```

#	Name	Member Type	File Size	Last Modified
1	DEMO	DATA	4GB	07/07/2021 21:19:58
2	DEMO_COMPRESS	DATA	89MB	07/07/2021 21:48:54
3	_PRODSAVAIL	DATA	128KB	07/07/2021 20:13:05



Length statement

- Sometimes character variables imported into SAS from other systems, like Oracle or Excel, have very large lengths.
- Use the LENGTH command to define the length of character and numeric variables. This can achieve a significant reduction in the space used by the program.

```
data demo_filter;  
  length char1 char2 char3 char4 char5 $20;  
  set demo;  
run;
```

#	Name	Member Type	File Size	Last Modified
1	BIG1	DATA	4GB	07/08/2021 13:17:37
2	BIG1_FILTER	DATA	88MB	07/08/2021 13:17:47

COMPRESS datasets



```
data democompress (compress=yes) ;  
set demo;  
run;
```

```
data test1;  
  set demo;  
  where group=5;  
run;  
real time      7.96 seconds
```

```
data test2;  
  set democompress;  
  where group=5;  
run;  
real time      1.28 seconds
```

Use INDEXES



Check for Indexes. PROC CONTENTS.

```
proc datasets library=work;  
modify demo;  
index create group;  
run;
```

```
OPTIONS MSGLEVEL=I;  
data test;  
  set demo;  
where group=5;  
run;
```

Check if being used. OPTIONS MSGLEVEL=I;

INFO: Index group selected for WHERE clause optimization.



WHERE vs IF?

```
data if;  
  set demo(keep=group char1 num:);  
  if group=1;  
run;
```

NOTE: There were **100000 observations** read from the data set WORK.DEMO.

NOTE: The data set WORK.IF has 10177 observations and 102 variables.

```
data where;  
  set demo (keep=group char1 num:);  
  where group=1;  
run;
```

NOTE: There were **10177 observations** read from the data set WORK.DEMO.
WHERE group=1;

NOTE: The data set WORK.IF has 10177 observations and 102 variables.



Monitor and delete work tables

- Although WORK SAS data sets will be deleted at the end of the program, they occupy space while the program is running. Permanent and WORK SAS data sets that are no longer needed can be deleted while the program is running by using PROC DELETE or PROC DATASETS.

```
proc datasets lib=work details;  
run;
```

```
proc datasets lib=work details;  
Delete table1 table2;  
run;
```

```
proc datasets library=work kill;  
run;
```

#	Name	Member Type	Obs, Entries or Indexes	Vars	Label	File Size	Last Modified
1	BIG1_SELECT	DATA	1000000	3		38MB	07/06/2021 12:05:01
2	SASGOPT	CATALOG	0			5KB	07/06/2021 11:16:48
3	SASMAC3	CATALOG	0			13KB	07/06/2021 12:12:27
4	SASMACR	CATALOG	22			125KB	07/06/2021 11:22:38
5	TEST2	DATA	1000000	1		31MB	07/06/2021 12:08:35
6	_PRODSAVAIL	DATA	8	6		128KB	07/06/2021 11:16:47

- This space is immediately recycled and available for other users.**



SORT and Subset in 1 Step

Sorting is **THE** most intensive operation in data processing

Always subset using the WHERE and KEEP statements

Sort only the variables that are required

Always use the OUT= option

```
PROC SORT data = data.transactions(KEEP=vars.... WHERE= (status = 'ACTIVE')) OUT = active;  
    BY customerid trans_datum;  
RUN
```



Avoid the Sort by using CLASS

Use procedures that support the CLASS statement to take advantage of group processing without sorting. Summary, Means, Univariate

<pre>Proc sort data=demo; by group; proc summary data=demo; by group; var num1; output out=stats max= min=; run;</pre>	3.8GB Workspace 10.8GB	Sort = 1min 8 secs Summary = 12 secs
<pre>proc summary data=demo; CLASS group; var num1; output out=stats max= min=; run;</pre>	3.8GB Workspace 3.8GB	Sort = 2.68 seconds

Additional Techniques



- PROC SQL Pass-Through
- HASH techniques
- PROC FORMAT