Ten tips for efficient SAS code
Host

Caroline Scottow
Presenter

Peter Hobart
Managing the webinar

• In Listen Mode

• Control bar opened with the white arrow in the orange box
Efficiency Overview

• Optimisation has four competing factors
  • CPU
  • Memory
  • I/O (disk and network)
  • Disk space
Efficiency
Overview

• Basic principles:
  • Don't do more work than you need to
  • Optimise for your environment
  • Go for the quick wins first
    - Jobs which take the most time
    - Jobs which are run most often
  • Benchmark after each change
Efficiency
Overview

Data Integration Studio produces detailed information on resources used and time taken by each step.

In base SAS and SAS Enterprise Guide use `options fullstimer` to get more information on the log.
Efficiency
Overview

• NOTE: The data set WORK.SAMPLE2 has 317223 observations and 27 variables.
• NOTE: DATA statement used (Total process time):
  • real time 2.10 seconds
  • cpu time 0.28 seconds

• NOTE: The data set WORK.SAMPLE2 has 317223 observations and 27 variables.
• NOTE: DATA statement used (Total process time):
  • real time 0.27 seconds
  • cpu time 0.28 seconds

• NOTE: The data set WORK.SAMPLE2 has 317223 observations and 27 variables.
• NOTE: DATA statement used (Total process time):
  • real time 0.27 seconds
  • cpu time 0.28 seconds

On the second and third runs the input data set was cached so the runtime was much lower. On a larger data set this effect will be reduced.

Note the CPU time remained constant.
Efficiency
Programmer time vs. Program time

Simple:
Reads every row,
Outputs every 3\textsuperscript{rd} row

```
Data sample1;
    set orion.customer_orders;
    if mod(_n_, 3)=1;
run;
```

Efficient:
Reads every 3\textsuperscript{rd} row and outputs

```
Data sample2;
    do i = 1 to rows by 3;
        set orion.customer_orders
            point=i
        nobs=rows;
        output;
    end;
    stop;
run;
```

Both produce a 1/3 sample
Tip 1
Subset early

```sas
data profits;
  set orion.customer_orders;
  sale = quantity*retailPrice;
run;

proc means data=profits nonobs maxdec=2;
  where month=12;
  class continent;
  var sale;
run;
```

```sas
data profits;
  set orion.customer_orders;
  sale = quantity*retailPrice;
run;

proc means data=profits nonobs maxdec=2;
  where month=12;
  class continent;
  var sale;
run;
```
Tip 2
select input columns with drop and keep

```
data profits;
  set orion.customer_orders;
  where month=12;
  sale = quantity*retailPrice;
run;
```

```
data profits;
  set orion.customer_orders
    (keep= month quantity retailPrice continent);
  where month=12;
  sale = quantity*retailPrice;
run;
```

```
proc means data=profits nonobs maxdec=2;
  class continent;
  var sale;
run;
```

```
proc means data=profits nonobs maxdec=2;
  class continent;
  var sale;
run;
```
Tip 3
select output columns

```sas
data profits;
  set orion.customer_orders
    (keep= month quantity
     retailPrice continent);
  where month=12;
  sale = quantity*retailPrice;
run;

proc means data=profits nonobs maxdec=2;
  class continent;
  var   sale;
run;
```

```sas
data profits (keep=sale continent);
  set orion.customer_orders
    (keep= month quantity
     retailPrice continent);
  where month=12;
  sale = quantity*retailPrice;
run;

proc means data=profits nonobs maxdec=2;
  class continent;
  var   sale;
run;
```
Tip 4
where vs IF

Read everything then select

```sas
data profits;
  set orion.customer_orders;
  sale = quantity*retailPrice;
  if month=12;
run;
```

Read only the required rows

```sas
data profits;
  set orion.customer_orders;
  sale = quantity*retailPrice;
  where month=12;
run;
```
Tip 4

where vs IF

Read everything then select

```sas
data profits;
  set orion.customer_orders;
  sale = quantity*retailPrice;
  if sale > 200;
run;
```

Read only the required rows

```sas
data profits;
  set orion.customer_orders;
  sale = quantity*retailPrice;
  where sale > 200;
run;
```

- **sale** is calculated then evaluated
- **sale** does not exist in the source table

But...we could use

```sas
where quantity*retailPrice > 200
```
Tip 5
Use indexes

No index

```sas
data profits;
    set orion.customer_orders;
    sale = quantity*retailPrice;
    where month=12;
run;
```

Index on month

```sas
data profits;
    set orion.customer_orders;
    sale = quantity*retailPrice;
    where month=12;
run;
```

NOTE: DATA statement used (Total process time):
real time  2.43 seconds
cpu time  0.26 seconds

NOTE: DATA statement used (Total process time):
real time  0.24 seconds
cpu time  0.07 seconds
Tip 6
Build indexes efficiently

Recreates the data and builds an index

```sas
data orion.customer_orders
  (index=(month));
  set orion.customer_orders;
run;
```

Reads the data and builds an index

```sas
proc datasets lib=orion noprint;
  modify customer_orders;
  index create month;
run;
quit;
```
Tip 7
You can subset during a sort

```sas
proc sort data=orion.customer_orders
   out=orders_sorted
   (drop=continent);
   by month;
   where continent="Europe";
run;
```

Danger!
Sorting a data set without specifying `out=` can result in data loss
Tip 8
optimise Conditional logic

Tests every condition

```sas
data profits;
  set orion.customer_orders;
  if quantity = 1 then order="Sml";
  if quantity = 2 then order="Med";
  if quantity = 3 then order="Lrg";
  if quantity > 3 then order="XXL";
run;
```

Tests only until a condition is TRUE

```sas
data profits;
  set orion.customer_orders;
  if quantity = 1 then order="Sml";
  else if quantity = 2 then order="Med";
  else if quantity = 3 then order="Lrg";
  else if quantity > 3 then order="XXL";
run;
```

All four tests are performed on every row

If on the current row quantity=1 then only one test is performed
Tip 9
Code with your data in mind

Most orders in this data contain 2 items.
By testing for this value first, less tests will be performed overall
Tip 10
Consider compression

• Compressing data:
  • Reduces disk space used
  • Increases CPU
  • Decreases IO
  • May introduce changes in behaviour in "edit in place"

• SAS default compression is deliberately "light touch"
  • Long character variables will usually compress well
  • Numbers will not compress with the default algorithm
  • Compression introduces a space overhead: the compression achieved should outweigh this
  • Look at the log to check the compression achieved
  • Poor candidates for compression can grow instead
  • Check bufsize is not excessively large, if compression is less than expected.
Tip 10
Consider compression

```
data customer_orders
  (compress=yes);
  set orion.customer_orders;
run;
```

NOTE: Compressing data set WORK.CUSTOMER_ORDERS decreased size by 41.70 percent. Compressed is 3603 pages; un-compressed would require 6180 pages.

NOTE: DATA statement used (Total process time):
  real time 2.57 seconds
  cpu time 1.59 seconds

Compress = yes or compress= char uses run length encoding
Tip 10
Consider compression

data customer_orders
  (compress=\texttt{binary});
  
  set orion.customer_orders;

run;

\textbf{NOTE:} Compressing data set WORK.CUSTOMER_ORDERS decreased size by 46.21 percent. Compressed is 3324 pages; un-compressed would require 6180 pages.

\textbf{NOTE:} DATA statement used (Total process time):
  \begin{itemize}
    \item real time \hspace{1cm} 2.62 seconds
    \item cpu time \hspace{1cm} 2.27 seconds
  \end{itemize}

Compress = binary uses Ross data compression
- Slightly better compression
- Slightly increased CPU
And finally...
Examine the log!

- Check the results are from a successful run, not a cached data set
- Every site has a different mix of constraints
- Measurements can be distorted by other running jobs competing for resources
- Benchmark changes by running several times in real-world situations
- Test each change in isolation
Questions?
Resources
More information and sources of help

SAS customer loyalty


Links to hundreds of free resources
Thankyou