



Time-Management: Programming Techniques to Positively Impact Runtimes.

PhUSE EU Connect - November, 2018

Overview



Key activities

- Identify big data
- Trimming the fat
- Starting small

Key outcome

- Work from the bare minimum

Key activities

- Sorting data
- Skipping steps
- Maintaining efficiencies

Key outcome

- Actively maintain and manage runtime

Key activities

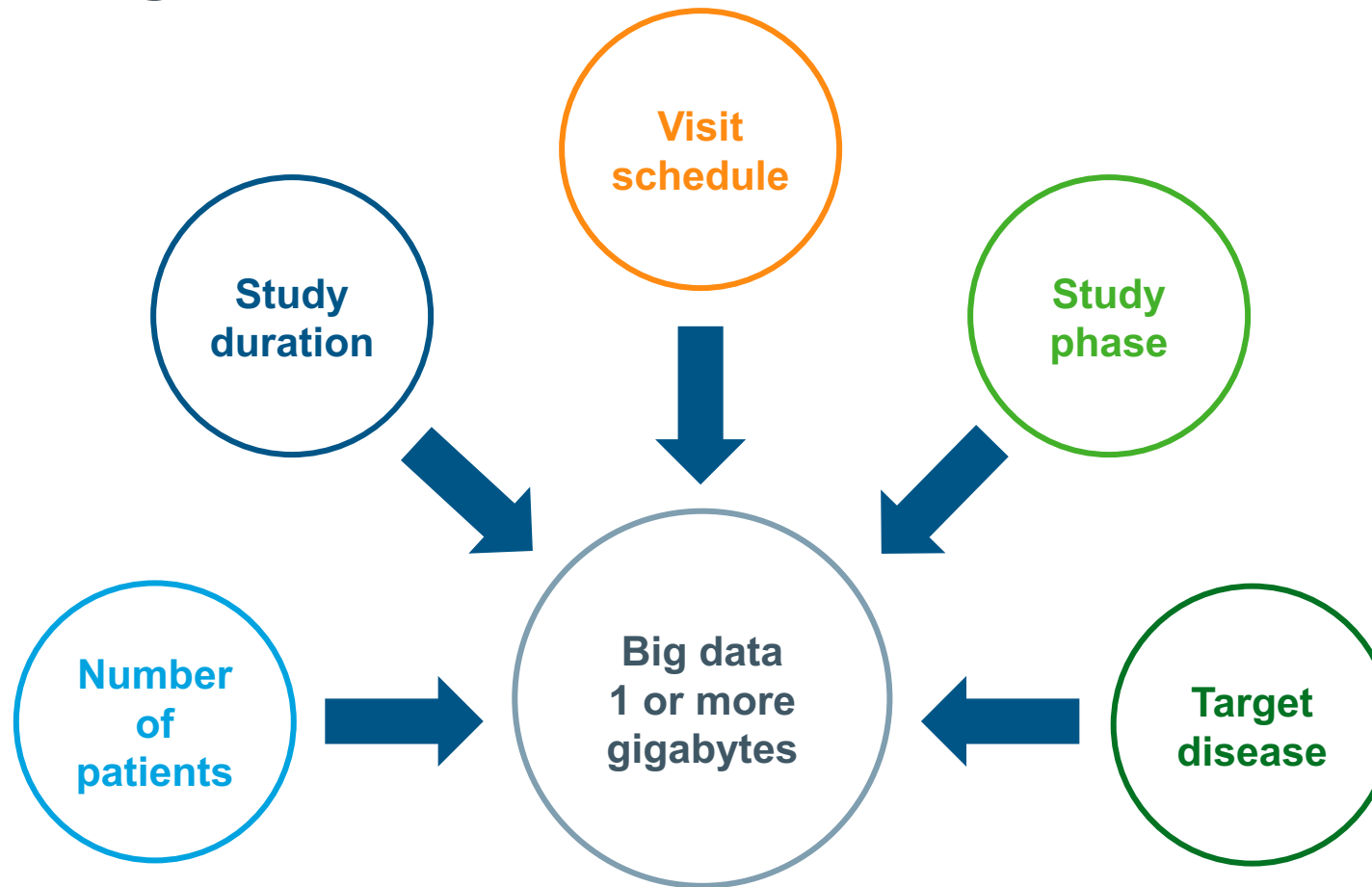
- Retracting steps
- Analyzing programs

Key outcomes

- Identify areas for improvement

Pre-Programming

Identify Big data



Pre-Programming

Trimming fat and Starting small

Trimming the fat

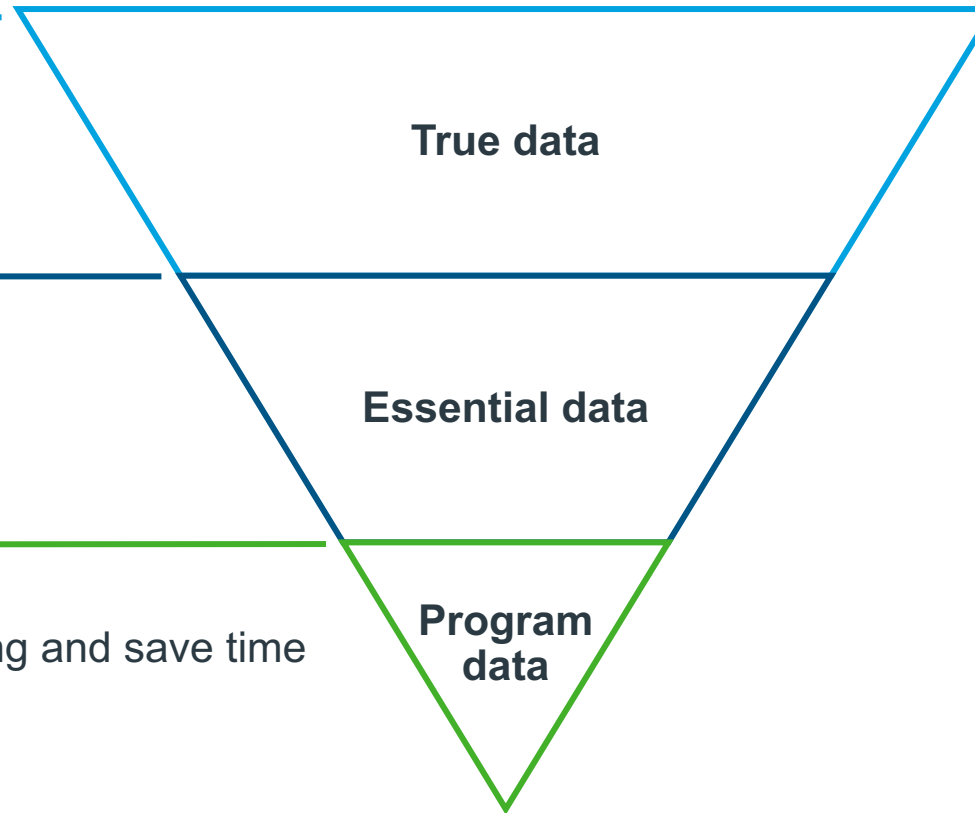
- Dead space (1 space = 1 byte)
- SQL, ALTER TABLE and MODIFY statments

Starting small

- List of variable, DROP / KEEP
- Subset of data

Outcome

- Minimum data before programming and save time during program development



Program development

Sorting data

Percentage Change in Runtime*

Changes to Code

Baseline

SORT-SORT and MERGE,
with KEEP on final data

```
*****Simple SORT-SORT and MERGE*****;
```

```
proc sort data=indat.small  
    out=small;  
    by trtgrps keyvar;  
run;
```

```
proc sort data=indat.large  
    out=large;  
    by trtgrps keyvar;  
run;
```

```
data outdat (keep=keyvar trtgrps largevar1-largevar10  
    smallvar1-smallvar4);  
    merge large  
        small;  
    by trtgrps keyvar;  
run;
```

***Median of Percent change from Baseline ($n=10$)** - Based on smaller dataset of ~10,000 observations and 22 variables, merged with big dataset of ~500,000 observations and 52 variables

Program development

Sorting data

Percentage Change in Runtime*

Changes to Code

Baseline

SORT-SORT and MERGE,
with KEEP on final data

-3.4%

Move KEEP to data going into
the MERGE

```
*****SORT-SORT and MERGE*****;  
***  with KEEP on in-data of MERGE  ***;  
proc sort data=indat.small  
    out=small;  
    by trtgrps keyvar;  
run;  
  
proc sort data=indat.large  
    out=large;  
    by trtgrps keyvar;  
run;  
  
data outdat;  
    merge large (keep=keyvar trtgrps largevar1-largevar10)  
           small (keep=keyvar trtgrps smallvar1-smallvar4);  
    by trtgrps keyvar;  
run;
```

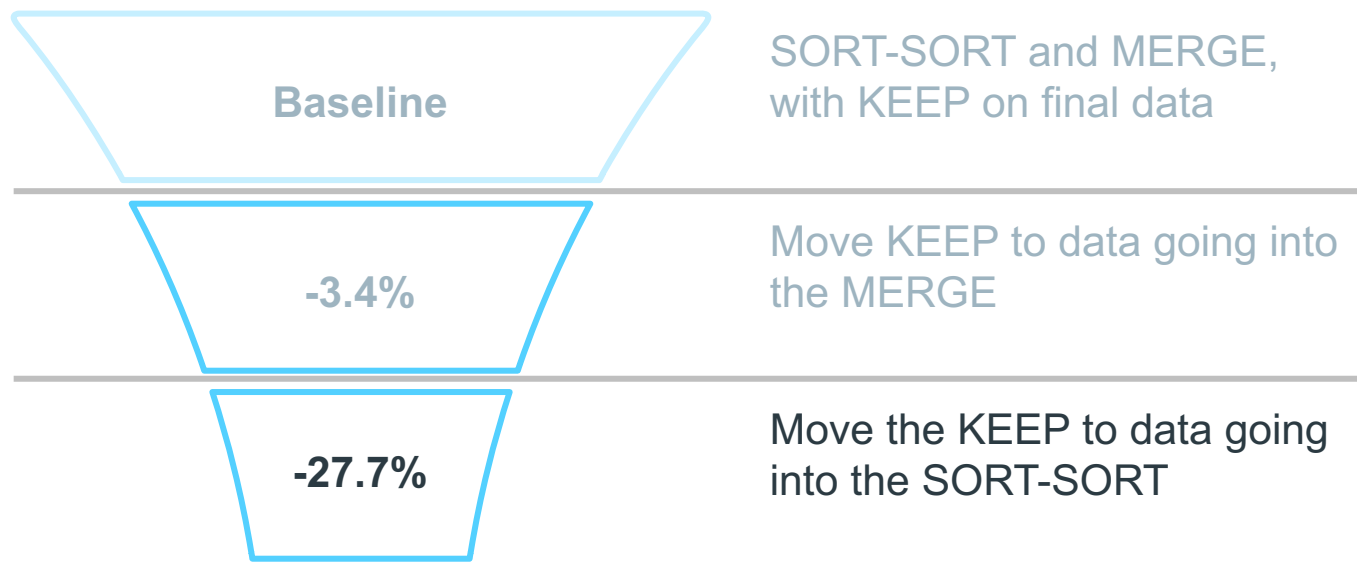
***Median of Percent change from Baseline (n=10)** - Based on smaller dataset of ~10,000 observations and 22 variables, merged with big dataset of ~500,000 observations and 52 variables

Program development

Sorting data

Percentage Change in Runtime*

Changes to Code



```
*****SORT-SORT and MERGE*****;  
*** with KEEP on in-data of SORTs ***;  
proc sort data=indat.small (keep=keyvar trtgrps  
                             smallvar1-smallvar4)  
                             out=small;  
    by trtgrps keyvar;  
run;  
  
proc sort data=indat.large (keep=keyvar trtgrps  
                             largevar1-largevar10)  
                             out=large;  
    by trtgrps keyvar;  
run;  
  
data outdat;  
    merge large  
          small;  
    by trtgrps keyvar;  
run;
```

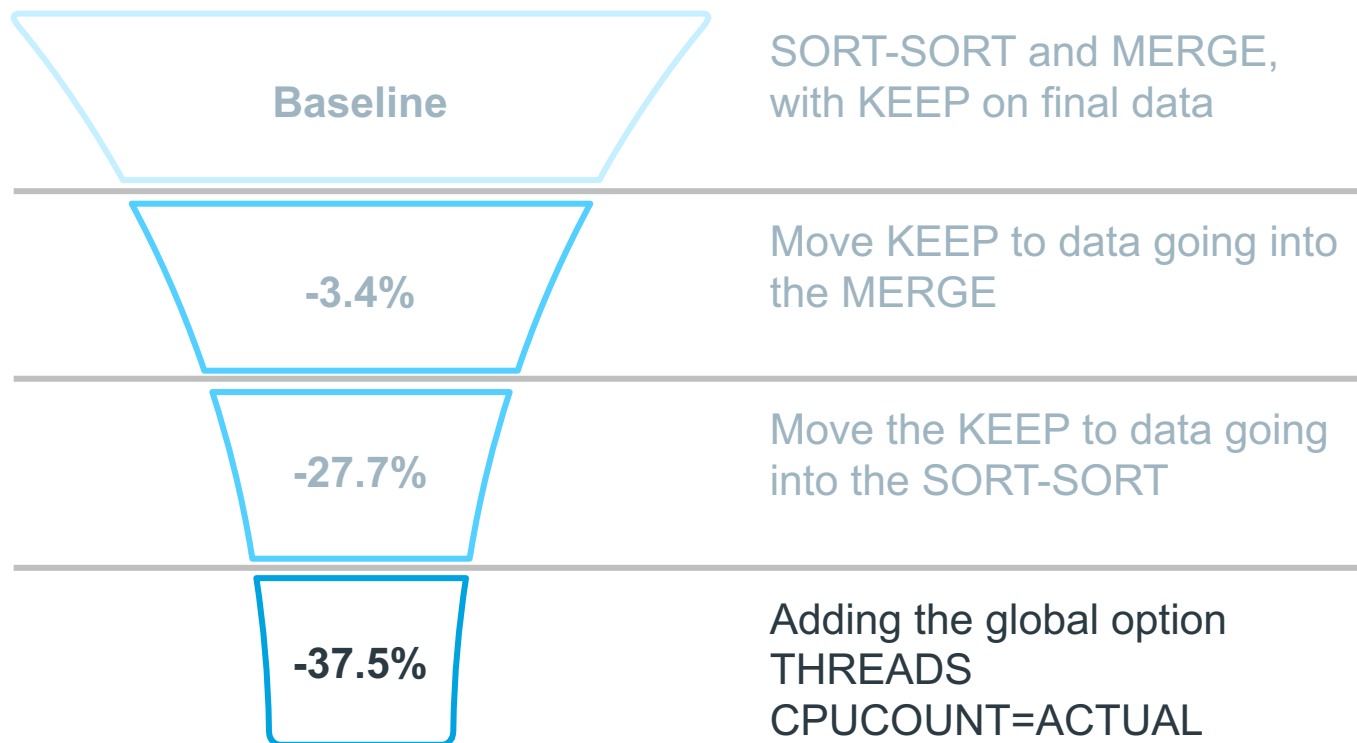
***Median of Percent change from Baseline (n=10)** - Based on smaller dataset of ~10,000 observations and 22 variables, merged with big dataset of ~500,000 observations and 52 variables

Program development

Sorting data

Percentage Change in Runtime*

Changes to Code



```
*****Addition of THREADED option*****;  
OPTIONS THREADS CPUCOUNT=ACTUAL;  
proc sort data=indat.small (keep=keyvar trtgrps  
                             smallvar1-smallvar4)  
    out=small;  
    by trtgrps keyvar;  
run;  
  
proc sort data=indat.large (keep=keyvar trtgrps  
                             largevar1-largevar10)  
    out=large;  
    by trtgrps keyvar;  
run;  
  
data outdat;  
    merge large  
          small;  
    by trtgrps keyvar;  
run;
```

***Median of Percent change from Baseline (n=10)** - Based on smaller dataset of ~10,000 observations and 22 variables, merged with big dataset of ~500,000 observations and 52 variables

Program development

Skipping steps

PRESORTED option on SORT procedure

- Allows SAS to determine if sort is needed
- Can be used as standard for all sort with minimal drawbacks

• Example: `***Using PRESORTED option***;`
`proc sort data=indat presorted;`
`by var1 var2 var3;`
`run;`

Using an INDEX

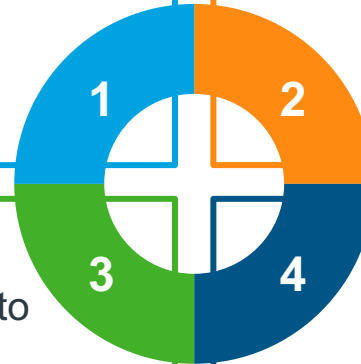
- Reference card for specified variable(s) in ascending order
- Allows SAS to retrieve and read data in order without a SORT using WHERE and BY statements
- **Must** be used multiple times to get benefit
- Target for index should be less than 25% the size of the initial dataset

HASH object merging

- Puts dataset in to internal memory with built-in index to directly access observations from there
- Effectively construct your dataset within a DATA step as already merged
- Replaces SORT-SORT and MERGE with DATA step
- Best used as a MACRO to prevent syntax confusion
- Tests showed **~40% runtime reduction** compared to standard

Using VIEWS

- Create a small file information SAS needs to resolve a DATA step, without resolving
- Best used on the DATA steps where simple mapping, merging or before simple procedures
- Example: `***Using a VIEW***;`
`data mydata / view=mydata;`
`set indat;`
`run;`



Program development

Maintaining efficiencies

Manage data size

- Regular use of DROP/KEEP statements
- Create temporary flags as FLAG="01" – "nn" rather than FLAG=1 – n
- Map or merge description variables (i.e. --TEST) late in the program

Smart logic

- Use WHERE for sub-setting: WHERE = applied as data is read / IF = applied as data is written
- Exclude observations not relevant to derivations using IF-THEN, ELSE statements

Outcome

- Reduce read/write cycle time
- Keep data streamlined
- Focus CPU time on essential tasks
- Maintain efficiency gains

Program Review

Retrace and Analyze

Situation

- We've created a program or inherited a program that seem to be running slowly

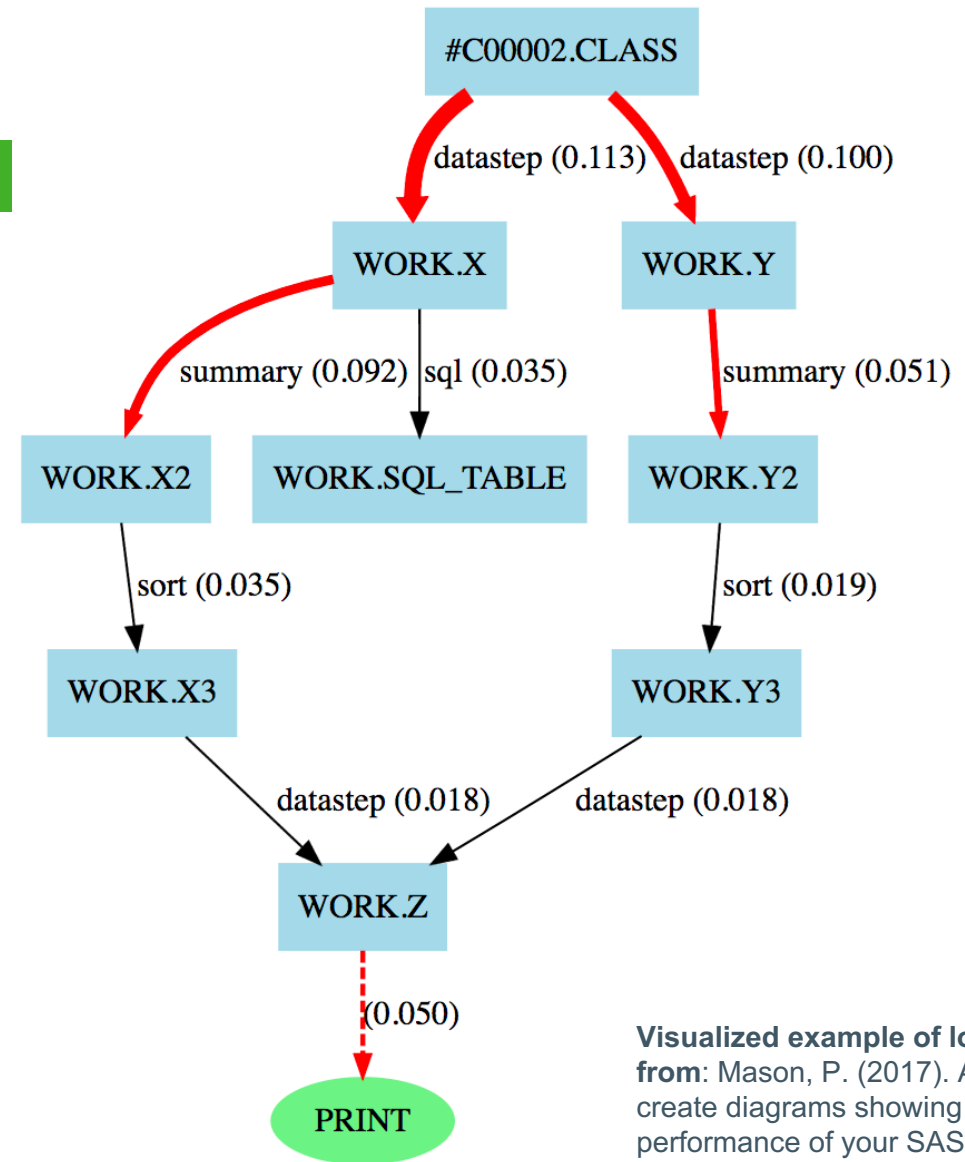
Solution

- Parse LOG file looking for:
 - Global options: THREADS
 - Procedure options: PRESORTED
 - Variables used from initial data for KEEP statement
- SCAPROC procedure:
 - Additional LOG file with; dataset size, access times, locations, variables and attributes
 - Easier to parse than normal LOG but needs to rerun the program to execute

Program Review

Outcome

- Summarize program runtimes
- Identify problem areas
- Track program flow



Visualized example of log information
from: Mason, P. (2017). Automatically
create diagrams showing the structure and
performance of your SAS code.

Summary

Keep data small early

- DROP / KEEP just what you need
- Remove dead spaces
- Map description late in the program

Boost SORT

- THREADS global option
- PRESORTED option
- Remove superfluous variables

Alternative methods

- VIEWS
- INDEXES
- HASH object merges

Smart logic

- IF-THEN ELSE
- WHERE vs. IF

Please Contact Me for More Information



Ian Sanderson

Senior Statistical Programmer - IQVIA

ian.sanderson@iqvia.com

+44 (0)118 450 1267