Good Programming Practice – Coding Efficiency

Introduction

Efficiency, as it applies to programming, means obtaining the correct results while minimizing the need for human and computer resources. We have broken the various aspects of programming efficiency into four major components and will discuss each below.

- Central processing unit (CPU) time
- Data storage (disk space)
- I/O time
- Programming time

CPU Time

Compiling and executing programs take up time and space. The required time the CPU spends to perform the operations that are assigned in the statements determine the complexity of the program. In order to make the program efficient and to reduce CPU time, we should

- execute only the necessary statements
- reduce the number of statements executed
- execute calculations only for the necessary observations.
- reduce the number of operations performed in a particular statement
- keep desired variables by using KEEP = or DROP = data set options
- create and use indexes with large data sets
- use IF-THEN/ELSE statements to process data
- avoid unnecessary sorting
- use CLASS statements in procedures
- use a subset of data set to test code before production
- consider the use of nested functions
- shorten expressions with functions

Data Storage

Data storage is primarily concerned with temporary datasets generated during program execution which can become very large and slow down processing. Here are some ways to reduce the amount of temporary data storage required by a program:

- Create a data set by reading long records from a flat file with an input statement with keeping the selected records with a needed incoming variables (?)
- Process and store only the variables that you need by using KEEP/DROP= data set options (or KEEP/DROP statements) to retain desired variables when reading or creating a SAS data set
- Create a new SAS data set by reading an existing SAS data set with a SET statement with keeping selected observations based on the values of only a few incoming variables
- Create as many data sets in one DATA step as possible with OUTPUT statements
- Use LENGTH statements to reduce variable size
- Read in as many SAS data sets in one DATA step as possible (SET or MERGE statement).
- Use data compression strategies by using COMPRESS=CHAR|YES for data sets that have long character data that contain many blanks and COMPRESS=BINARY for data that has long observation length
- Define numeric variables as character rather than numeric if they are not going to be used in arithmetic operations and are less than 8 bytes
- Use DATA _NULL_ steps when no output SAS data set is needed
- Use PROC DATASETS with DELETE statement to delete unwanted data sets
- Use a small data set page size to minimize wasted disk space when creating a small SAS data set or a SAS data set that will be accessed in a sparse random pattern using an index or the POINT= SET statement option

I/O Time

I/O time is the time the computer spends on data input and output (reading and writing data). Input refers to moving data from disk space into memory for work. Output refers to moving the results out of memory to disk space or a display device such as a terminal or a printer. To save I/O time, the following tips can be used:

- Read only data that is needed by subsetting data with WHERE or IF statement (or WHERE= data step option) and using KEEP/DROP statement (or KEEP=/DROP= data set option) instead of creating several datasets
- Avoid rereading data if several subsets are required
- Use data compression for large datasets
- Use the DATASETS procedure COPY statement to copy datasets with indexes
- Use the SQL procedure to consolidate code
- Store data in temporary SAS work datasets, not external files
- Assign a value to a constant only once (employ retain with initial values)

Programming Time

- Reducing I/O time and CPU usage are important, but using techniques which are efficient in terms of the programming time it takes to develop, debug, and validate code can be even more valuable. Much efficiency can be gained by following the good programming practices for readability and maintainability of code as discussed in this guide.
- utilise macros for redundant code
- use the SQL procedure to consolidate the number of steps

References

SAS Programming Efficiencies: http://www.ssc.wisc.edu/sscc/pubs/4-3.pdf

Gilsen, Bruce. SAS ® PROGRAM EFFICIENCY FOR BEGINNERS: http://www.ats.ucla.edu/stat/sas/library/nesug00/bt3005.pdf

Lafler, Kirk Paul. Efficient SAS® Programming Techniques: http://www2.sas.com/proceedings/sugi25/25/hands/25p146.pdf

Langston, Rick. Efficiency Considerations Using the SAS® System: http://www2.sas.com/proceedings/sugi30/002-30.pdf

Carpenter, Arthur L. Getting More For Less: A Few SAS® Programming Efficiency Issues: http://www.caloxy.com/papers/35-CC199.PDF

SAS Institute Inc. SAS® Programming Tips: A Guide to Efficient SAS Processing. 155pp. 1990.