Best Practices for Efficiency and Code Optimization in SAS® programming

Jay Iyengar, Data Systems Consultants LLC

Michigan SAS Users Group – February 22, 2024

Introduction

Standard ways of measuring efficiency

- Processing time
- Input/Output (I/O)
- Memory
- Storage space

Alternate ways of measuring efficiency

- Code Development time\Maintenance time

Metrics used to measure efficiency

- Processing time- CPU time, Real time
- Memory - RAM in Bytes (KB/MB/GB)
- Storage Space - Filesize in Bytes (KB/MB/GB)
- Input/Output (I/O) - Hard to measure
- Code Development time\Maintenance time - Lines of code

Conditional Logic

DATA STEP IF-THEN Logic used for Recoding Variables

```
Data TESTING2;
          Length Age_Group $8;
 Set TESTING;

   Age = Round((Today()-BirthDate_Text__c)/365.25, 1);

   If _N_<=20 Then Put Age=;

         /* Create age group variables for Age Cohorts */
   If 0<=Age<=9 Then age_group="0-9";
   If 10<=Age<=19 Then age_group="10-19";
   If 20<=Age<=29 Then age_group="20-29";
   If 30<=Age<=39 Then age_group="30-39";
   If 40<=Age<=49 Then age_group="40-49";
   If 50<=Age<=59 Then age_group="50-59";
   If 60<=Age<=69 Then age_group="60-69";
   If 70<=Age<=79 Then age_group="70-79";
   If 80<=Age<=89 Then age_group="80-89";
   If 90<=Age<=99 Then age_group="90-99";
   If Age>=100 Then age_group="100+";
   If Age=. Then age_group="Unknown";

Run;
```

Conditional Logic (con't)

Use ELSE Keyword in IF-THEN Logic

ELSE Condition only processed if prior condition is false

```
Data TESTING2;
          Length Age_Group $8;
  Set TESTING;

    Age = Round((Today()-BirthDate_Text__c)/365.25, 1);

    If _N_<=20 Then Put Age=;

                              /* Create age group variables */
    If 0<=Age<=9 Then age_group="0-9";
    Else If 10<=Age<=19 Then age_group="10-19";
    Else If 20<=Age<=29 Then age_group="20-29";
    Else If 30<=Age<=39 Then age_group="30-39";
    Else If 40<=Age<=49 Then age_group="40-49";
    Else If 50<=Age<=59 Then age_group="50-59";
    Else If 60<=Age<=69 Then age_group="60-69";
    Else If 70<=Age<=79 Then age_group="70-79";
    Else If 80<=Age<=89 Then age_group="80-89";
    Else If 90<=Age<=99 Then age_group="90-99";
    Else If Age>=100 Then age_group="100+";
    Else If Age=. Then age_group="zPending further info";
Run;
```

Subsetting –WHERE and IF

WHERE and IF statement/options

```
DATA NDF_MD1;
SET NEWFILE.NDF;
IF STATE = 'MD';
RUN;
```

NOTE: There were **1048575** observations read from the data set NEWFILE.NDF.

WHERE is more efficient than IF

```
DATA NDF_MD2;
SET NEWFILE.NDF;
WHERE STATE = 'MD';
RUN;
```

NOTE: There were **22518** observations read from the data set NEWFILE.NDF. WHERE STATE='MD';

Subsetting –WHERE/IF (con't)

Place IF statement at the top of the DATA STEP

```
DATA MedU16;
    SET MedUtiliz16;

        IF CITY='Bethesda';

        IF PROVIDER_CITY='Baltimore' THEN PROVIDER_STATE='MD';
        ELSE IF PROVIDER_CITY='Arlington' THEN PROVIDER_STATE='VA';

        IF BILLTYPE = 13 THEN BILLTYPEDSC = 'Hospital Outpatient';
        ELSE IF BILLTYPE = 11 THEN BILLTYPEDSC = 'Hospital Inpatient';
        ELSE IF BILLTYPE = 33 THEN BILLTYPEDSC = 'Home Health Agency';

RUN;
```

Using Indexes

An Index is a file which is attached to a data set.

Indexes directly access observations and bypass sequential processing.

```
412  Proc Datasets Library=Work;
413     Modify Testing;
414      Index Create STATUS / NOMISS;
NOTE: Simple index Status has been defined.
415  Quit;

NOTE: MODIFY was successful for WORK.TESTING.DATA.
NOTE: PROCEDURE DATASETS used (Total process time):
       real time           20.87 seconds
       cpu time            14.37 seconds
```

Using Indexes (con't)

Indexes reduce processing time and CPU Time

```
Data Testing_Subset_ac;
   Set Testing
     (Where=(STATUS='Administratively Converted'));
Run;

NOTE: There were 66836 observations read from the
data set WORK.TESTING.
      WHERE STATUS='Administratively Converted';
NOTE: The data set WORK.TESTING_SUBSET_AC has 66836
observations and 67 variables.
NOTE: DATA statement used (Total process time):
      real time           3.80 seconds
      cpu time            1.03 seconds
```

```
Data Testing_Subset;
   Set Testing
     (Where=(STATUS='Administratively Converted'));
Run;

NOTE: There were 66836 observations read from the data set
WORK.TESTING.
      WHERE STATUS='Administratively Converted';
NOTE: The data set WORK.TESTING_SUBSET has 66836
observations and 67 variables.
NOTE: DATA statement used (Total process time):
      real time           21.93 seconds
      cpu time            12.68 seconds
```

Testing Code

- Limit the number of observations on large data sets
- OBS= Global and data set option
- Execute your code without reading in any observations.

```
OPTIONS OBS=0;
```

- Read in small sample of observations.

```
OPTIONS OBS=1000;
```

Testing Code

Run multiple tests using different sample sizes

```
OPTIONS OBS=10000;
      ........
OPTIONS OBS=100000;
```

Use FIRSTOBS= with OBS= to read from the middle and end of the data set.

```
OPTIONS FIRSTOBS=500000 OBS=10000;
OPTIONS FIRSTOBS=990000 OBS=10000;
```

Keep and Drop

KEEP / DROP statements/options used to select variables

```
Data DeathsDemo2;
 Set DeathsDemo;
      Keep ALF Age Any_Cong_Setting Case County Date_of_Birth
           Date_of_Death Epi_Report_Date Ethnicity Facility_Name
           First_Name GH Intake_LTCF_NH  Last_Name Location_Name
           Location_Type Notes Number Race Sex Staff_or_Resident
           Town ZCTA_D;
run;
```

```
Data DeathsDemo2;
   Set DeathsDemo
       (Keep=ALF Age Any_Cong_Setting Case County Date_of_Birth
            Date_of_Death Epi_Report_Date Ethnicity Facility_Name
            First_Name GH Intake_LTCF_NH Last_Name Location_Name
            Location_Type Notes Number Race Sex
            Staff_or_Resident Town ZCTA_D);
run;
```

Keep and Drop

Use KEEP= instead of KEEP statement

- KEEP/ DROP input data set option

- KEEP statement and output data set option

Concatenating data sets

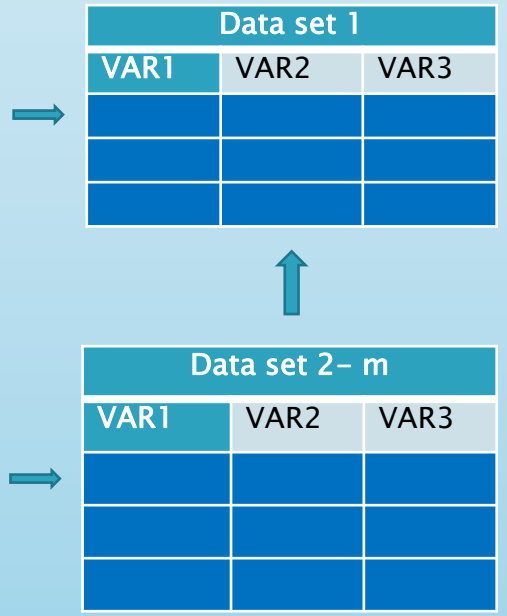
Concatenating-
Stacking/ Appending

Multiple methods for concatenating SAS data sets

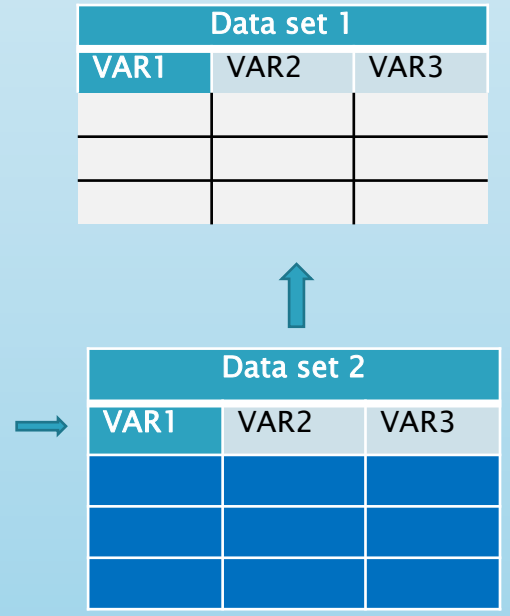
DATA STEP SET Statement

PROC APPEND

DATA STEP



PROC APPEND



Concatenating data sets (con't)

PROC APPEND reads less data and is more efficient.

Use PROC APPEND to concatenate SAS data sets

```
Data NDF_DSC;
 Set NDF_1994
     NDF_1995;
Run;

NOTE: There were 515155 observations read from the data set NDF_1994.
NOTE: There were 533420 observations read from the data set NDF_1995.
NOTE: DATA statement used (Total process time):
        real time  50.36 seconds
        cpu time   5.44 seconds

Proc Append Base=NDF_1994 Data=NDF_1995;
Run;

NOTE: Appending NDF_1995_AFTER to NDF_1994.
NOTE: There were 533420 observations read from the data set NDF_1995.
NOTE: 533420 observations added.
NOTE: PROCEDURE APPEND used (Total process time):
        real time   27.33 seconds
        cpu time    2.80 seconds
```

Merging data sets

- Horizontal as opposed to vertically combining data sets.
- Merging and joining data sets aka table lookups.
- DATA STEP Merge, PROC SQL Join, Hash Tables
- Different methods process merge differently.

Merging data sets (con't)

DATA STEP Merge usually requires PROC SORT

```
Proc Sort Data = Gdelt.Gdelt_All Out=Gdelt_All;
      By EventCode;
Proc Sort Data = Gdelt.Cameo_Event_Codes
           Out = Cameo_event_codes Nodupkey;
      By CameoEventCode;
Run;
```

```
NOTE: PROCEDURE SORT used (Total process time):
      real time           14.31 seconds
      cpu time            12.60 seconds
```

```
Data Gdelt_All_V2;
  Merge Gdelt_All(IN=A)
        Cameo_event_codes
          (Rename=(CameoEventCode=EventCode) IN=B);
     By EventCode;
  If A and B;
Run;
```

```
NOTE: DATA statement used (Total process time):
      real time           37.56 seconds
      cpu time            4.57 seconds
```

PROC SQL Join sorts data implicitly

Avoids PROC SORT

```
Proc Sql;
   Create Table gdelt_All_V2 as
   Select A.*, B.EventDescription
   From Gdelt.Gdelt_All as A,
        Gdelt.Cameo_Event_Codes as B
   Where A.EventCode=B.CameoEventCode;
Quit;
```

```
NOTE: PROCEDURE SQL used (Total process time):
      real time           14.56 seconds
      cpu time            6.46 seconds
```

Minimizing passes through the data

Each DATA STEP involves reading & writing of data

```
data ridoc4;
  merge SF1 (in=a) LL1 (in=b);
     by IDL_FirstName IDL_LastName IDL_DOB collection_date;
  format match $50.;
  if a and b then match="Both";
  if a and not b then match="SF Only";
  if b and not a then match="Linelist Only";
run;

data ridoc5 discrep SF2 LL2;
  set ridoc4;
    If match="Both" & ((IDL_CF_RorE in ("Resident","Unknown") & LL_RorE="Resident")
        OR IDL_CF_RorE in ("Employee","Unknown") & LL_RorE="Employee")) then output ridoc5;
    else if match="Both" then output discrep;
    else if match="SF Only" then output SF2;
    else If match="Linelist Only" then output LL2;
run;
```

Minimizing passes of the data (con't)

Fewer DATA STEPS means fewer data sets are created and stored.

Combining DATA STEPS results in reduced reading\writing of data.

```
Data ridoc4 discrep SF2 LL2;
   Merge SF1 (in=a) LL1 (in=b);

      By IDL_FirstName IDL_LastName IDL_DOB collection_date;

   If A and B and
       ((IDL_CF_RorE in ("Resident","Unknown") and LL_RorE="Resident") Or
       (IDL_CF_RorE in ("Employee","Unknown") AND LL_RorE="Employee")) then Output ridoc4;

   Else If A and B then Output discrep;
   Else If A and not B then output SF2;
   Else If B and not A then output LL2;
Run;
```

Data set compression

- Compression eliminates empty space in the variables in your data set., i.e. missing values.

```
OPTIONS COMPRESS=BINARY;
```

- Compression reduces data set size, and minimizes storage space requirements.

```
OPTIONS COMPRESS=NO;
```

- Overhead required to read compressed data sets.

Conclusion

- Advanced knowledge of efficiency and code optimization techniques is valuable.
- Programmers can measure most computing resources, CPU, memory, storage space.
- Perform testing to determine most efficient methods in your computing environment.

Contact Information

Feel free to contact me with any questions you have about my talk.

Jay Iyengar, Director

Data Systems Consultants LLC

Email: datasyscon@gmail.com

https://www.linkedin.com/in/datasysconsult/