Paper 097-2010

Efficient Statistical Programming? Reducing SAS® Run Time with WHERE Setup

Keiko I. Powers, Ph.D., J. D. Power and Associates, Westlake Village, CA

ABSTRACT

When statisticians handle time-consuming statistical analysis runs, setting up SAS® code for efficient processing is critical for the overall project success. In preparing a SAS program, it is obviously important to ensure that the selection and setup for the statistical PROCs are correct. When the overall run-time efficiency is concerned, equally important is how the SAS DATA steps are set up for the subsequent statistical analysis. Creating temporary partitioned data sets for various analysis steps can often lead to an unnecessarily long run time. If a temporary partitioned data set is used for only one PROC, it is often the case that the parenthesis-based WHERE coding scheme -- e.g., PROC REG DATA=A(WHERE = (FLAG = 1)); etc., -- works more efficiently and reduces the overall run time substantially. Systematic comparisons of the SAS execution time between the partitioned temporary data setup and the parenthesis-based WHERE setup were performed based on several simulation runs. Based on the investigation results, recommendations are provided on how to set up SAS data steps for efficient statistical analysis.

INTRODUCTION

When conducting statistical analysis for various projects, we often deal with time constraints due to a tight deadline, and for these instances it is particularly important to handle analysis as efficiently as possible to ensure the project is completed on time. If the focal point of efficiency is the overall completion time, the time spent on data processing often becomes more critical than the time for the statistical steps, especially because statistical analysis can nowadays be completed very quickly with SAS. For this reason, paying attention to and continuously addressing time efficiency with respect to data processing steps should be a high-priority objective for statisticians/SAS programmers.

This paper illustrates how a small difference in data processing steps can cause a big difference in overall SAS run time. More specifically, it shows two different data steps – one creating a partitioned dataset for subsequent analysis steps and the other using the original dataset with the WHERE parenthesis clause. These two setups were run multiple times with the same large dataset to compare and contract the overall run time. The results are helpful for a better understanding on how and why it is important to always try to find a more efficient way to process data for statistical analysis.

DATA

The data used for the simulation analysis is actual survey-related data with the data size over 400MB with approximately 500 variables and 80,000 subjects. There are many long character fields and as a result, just reading in the data typically takes over 100 CPU seconds with SAS for PC. To demonstrate how the two setups – the partitioned data setup versus the WHERE-clause setup – affect the overall run time in a more general scheme, four data setups are prepared, each having a different number of observations as shown below.

Setup 1 – 25% of the original sample size (i.e., # of subjects is about 80000/4 = 20000)

Setup 2 – 50% of the original sample size

Setup 3 – 75% of the original sample size

Setup 4 – 100% of the original same size

METHOD

For each data setup, three PROCs were included – PROC FREQ, PROC MEANS, and PROC REG. When these procedures were run, they were handled in two data processing setups – based on the partitioned data and based on the WHERE clause.

1. SAS codes based on the partitioned data approach.

```
proc printto log="E:\Project\sasruntime25p.log" new; run;

%macro runtestp;

%do i = 1 %to 10;

%put !!!!! with extra data step: Iteration &i;

data temtem; set og; if respond='keep'; run;  /* data partitioning here */

proc freq data=temtem;
table osat;
title "with extra data step: Iteration &i";
run;

proc means data=temtem;
var osat; run;

proc reg data=temtem;
model extsub = ext01-ext06/tol;
run; quit;

%end;

%mend;
%runtestp

proc printto; run;
```

2. SAS codes based on the WHERE clause approach.

```
proc printto log="E:\Project\sasruntime25w.log" new; run;

%macro runtestw;
%do i = 1 %to 10;

%put !!!!! with no data step: Iteration &i;

proc freq data=og(where = (respond='keep')); /*data partitioning by ‘where’ */
table osat;
title "with no data step: Iteration &i";
run;

proc means data=og(where = (respond='keep'));/*data partitioning by ‘where’ */

var osat; run;

proc reg data=og(where = (respond='keep')); /*data partitioning by ‘where’ */

model extsub = ext01-ext06/tol;
run; quit;

%end;

%mend;
%runtestw

proc printto; run;
```

As shown in the codes, these two processing setups were run 10 times for each of the four data setups (i.e., the size of the data OG in SAS setup was adjusted to 25%, 50% etc., by random sampling) to observe the SAS run time with respect to the CPU time and the real time. The average run time was derived over 10 iterations to compare the time differences associated with the two processing steps.

RESULTS

Table 1 shows the average run time for different data sizes and two processing steps. It can be clearly seen that the WHERE setup outperforms the partitioned data approach for all the setups, and the difference in time efficiency is substantial.

Table 1. Comparisons of overall SAS run time between two data setups

	Data Size	Partitioned-Data Approach	WHERE-Clause Approach	% Reduction in Run Time
Real time (sec.)	25%	6.4	0.5	93%
Real time (sec.)	50%	14.5	0.9	94%
Real time (sec.)	75%	31.2	11.9	62%
Real time (sec.)	100%	102.0	55.1	46%
CPU Time (sec.)	25%	1.2	0.4	63%
CPU Time (sec.)	50%	2.4	0.9	64%
CPU Time (sec.)	75%	3.7	2.0	47%
CPU Time (sec.)	100%	5.4	4.1	25%

The difference is mainly caused by the time for creating the partitioned data as shown below in the SAS LOG (lines 4 and 5 in bold type). It shows the breakdown of how much time was spent for each step using the 100% data-size case as an example. Other setups/iterations had a similar pattern with a considerable amount of time being spent on the data partition step.

```
!!!!! with extra data step: Iteration 1 (the partitioned data approach)

NOTE: There were 81530 observations read from the data set WORK.OG.
NOTE: The data set WORK.TEMTEM has 76529 observations and 483 variables.
NOTE: DATA statement used (Total process time):
      real time           1:32.28
      cpu time            3.70 seconds


NOTE: There were 76529 observations read from the data set WORK.TEMTEM.
NOTE: PROCEDURE FREQ used (Total process time):
      real time           0.52 seconds
      cpu time            0.52 seconds


NOTE: There were 76529 observations read from the data set WORK.TEMTEM.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.39 seconds
      cpu time            0.38 seconds



NOTE: PROCEDURE REG used (Total process time):
      real time           0.44 seconds
      cpu time            0.42 seconds


!!!!! with no data step: Iteration 1 (the where-clause approach)

NOTE: There were 76529 observations read from the data set WORK.OG.
```

```
      WHERE respond='keep';
NOTE: PROCEDURE FREQ used (Total process time):
      real time           22.32 seconds
      cpu time            0.93 seconds


NOTE: There were 76529 observations read from the data set WORK.OG.
      WHERE respond='keep';
NOTE: PROCEDURE MEANS used (Total process time):
      real time           21.20 seconds
      cpu time            1.00 seconds


NOTE: PROCEDURE REG used (Total process time):
      real time           25.23 seconds
      cpu time            2.03 seconds
```

CONCLUSION

The simple simulation shown in this paper clearly illustrates the importance of efficient data processing steps for the overall reduction of SAS run completion time. The results suggest that creating partitioned data should be avoided as much as possible unless the partitioned data is used multiple times for various PROCs. If the partitioned data is used only for one or two PROCs, it is very likely that the overall SAS run time is longer than necessary. On the other hand, if the partitioned data is used multiple times – at least by five or six PROCs - then it is likely that this approach is more efficient. As the SAS LOG above indicated, each PROC step takes less time to run with the partitioned data approach than with the WHERE clause approach. Differentiating these two setups and applying each to an appropriate situation will help statisticians minimize the overall run time of various SAS programs for statistical analysis.

Though the time difference shown in this paper seems to be small (e.g., 55 seconds versus 102 seconds for the 100% data size case), these differences could affect time efficiency considerably when the data become very large or if the same steps are run multiple times with Macro setup. For example, if the same procedure has to be run 1,000 iterations – instead of 10 iterations – for simulation, the real time difference can be 5,500 seconds versus 10,200 seconds (or 1.5 hours versus almost 3 hours!). The WHERE clause setup is very easy to use and knowing the pros and cons of each setup should help statisticians improve overall SAS programming efficiency.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Keiko I. Powers, Ph.D.
J. D. Power and Associates
2625 Townsgate Road
Westlake Village, CA 91361
Work Phone: 805-418-8114
Fax: 805-418-8241
E-mail Address: Keiko.Powers@jdpa.com