



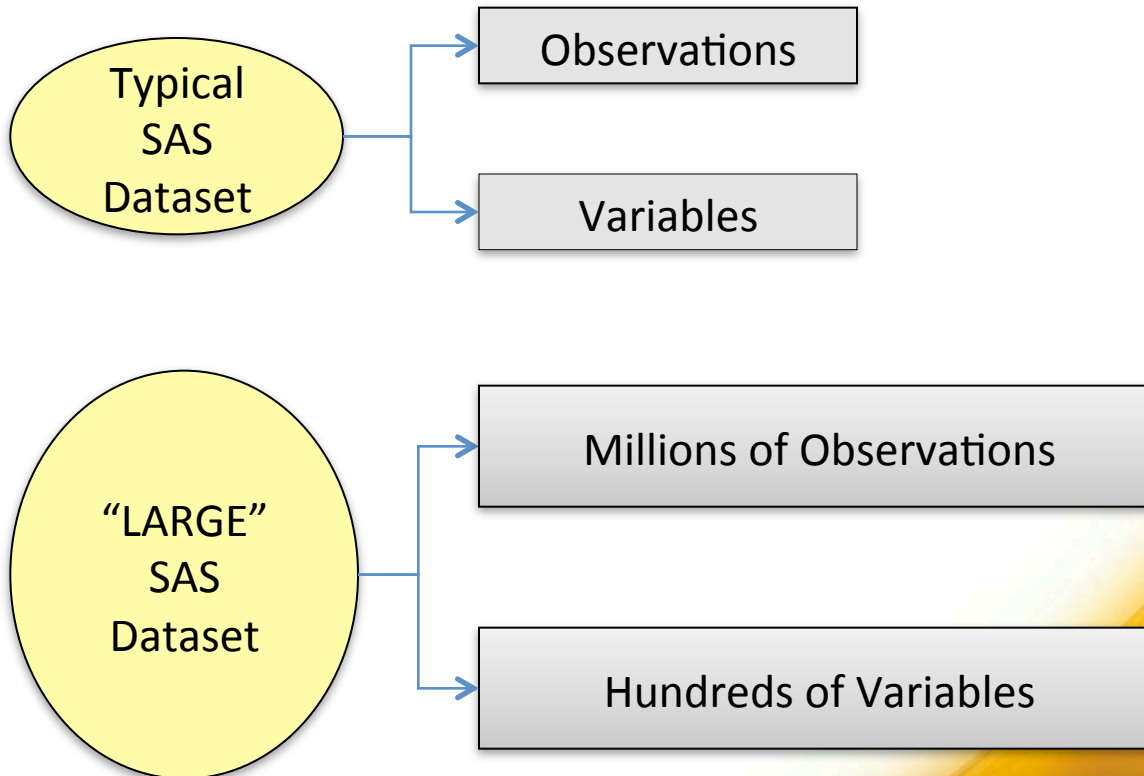
Quanticate

CONFIDENCE COMES FROM QUALITY

Working Efficiently with Large SAS® Datasets

Vishal Jain
Senior Programmer

Introduction



'Large' SAS dataset is a subjective term as it largely depends on how a user perceives it to be and also on the available resources.

Challenges with Large SAS datasets

- Storage:
 - Disk Space
 - Memory
- Time:
 - Real Time
 - CPU time
- There are several ways by which one can handle these challenges.
- One of the most common approach is to make LARGE datasets smaller without losing any of its information.

Agenda

- Techniques to reduce or compress the LARGE SAS datasets
- Programming tips to make working with LARGE SAS datasets efficient.

LENGTH STATEMENT

- Can be used to set or control the number of bytes required to store a SAS variable.
- Advantages:
 - Reduces the storage space required by variables.
- Drawbacks:
 - Requires extra programming time to reduce length of variables.
 - Might lead to wrong results when used incorrectly.
 - Reducing the length of fractional numbers or numbers with decimals might result in a loss of accuracy due to truncation.
 - It requires not more than 4 bytes to store a reasonable SAS date.
 - There are no warnings or errors are issued when the specified length in the LENGTH statement results in the truncation of data.

%SQUEEZE macro

- Originally developed by Ross Bettinger* can be used to find the minimum lengths required for both numeric and character variables in a SAS dataset.
- These minimum lengths can be assigned to the variables to reduce their storage space and hence reducing the size of the SAS dataset.

* Please see the references slide at the end of presentation for more details.

COMPRESS Dataset Option

- Compression is a process that reduces the number of bytes required to represent each observation.
 - Dataset options `COMPRESS= BINARY|CHAR` can be used to carry out the compression of observations in output SAS datasets.
 - The resultant compressed dataset obtained from this compression technique requires less storage space and fewer I/O operations to read or write data, however additional CPU resources might be needed to access that file.
- * `COMPRESS= NO` disables compression.

LENGTH & COMPRESS Results

Original Data = 114.3 MB			
No. of Char Variables = 38		No. of Num variables = 7	
Length Reduction (Using %SQUEEZE)	Resultant Size expressed as % of Original data size with/without COMPRESS Option		
	No	COMPRESS= BINARY	COMPRESS= CHAR
No	0	21%	23%
Yes	40%	19%	21%

Original Data = 928.8 MB			
No. of Char Variables = 36		No. of Num variables = 19	
Length Reduction (Using %SQUEEZE)	Resultant Size expressed as % of Original data size with/without COMPRESS Option		
	No	COMPRESS= BINARY	COMPRESS= CHAR
No	0	13%	15%
Yes	24%	11%	13%

Original Data = 131.5 MB			
No. of Char Variables = 23		No. of Num variables = 42	
Length Reduction (Using %SQUEEZE)	Resultant Size expressed as % of Original data size with/without COMPRESS Option		
	No	COMPRESS= BINARY	COMPRESS= CHAR
No	0	16%	18%
Yes	25%	14%	17%

Original Data = 5.1 GB			
No. of Char Variables = 49		No. of Num variables = 90	
Length Reduction (Using %SQUEEZE)	Resultant Size expressed as % of Original data size with/without COMPRESS Option		
	No	COMPRESS= BINARY	COMPRESS= CHAR
No	0	16%	22%
Yes	20%	14%	17%

Results Summary

- LENGTH Statement when used in combination with COMPRESS= BINARY option yields the best results with respect to reduction of data size.
- The resultant data was around **15 %** of the original data size.
- SAS programs were executed using these reduced datasets and it was observed that the performance was faster than running them through corresponding original datasets.
- This observation may not be true with all SAS datasets as it largely depends on the size and structure of the dataset, number/lengths of the variables, SAS job to be performed and the operating environment.

DROP= and KEEP= Dataset options

- SAS dataset may contain many variables that are either completely blank or not required for report generation.
- Dropping such missing variables might make a huge difference when the size of the dataset is large.
- The DROP= and KEEP= option or DATA step statements DROP and KEEP can be used to select variables from a SAS dataset.
- DROP and KEEP dataset options are always flexible, efficient and advantageous to use over DROP and KEEP DATA step statements as:
 - They can be exclusively applied to the variables of input and/or output dataset
 - They can be used in Procedures

Summary I

- Points discussed:
 - Techniques to reduce the size of Large SAS datasets.
 - Advantages and drawbacks of these techniques.
- Next section covers few programming tips and techniques to make working with LARGE SAS datasets more efficient.

Tip # 1 – Create Variables

- While working on analysis datasets, one usually tends to create flag variables based on certain criteria.
- Since, the default length of numeric variable is 8 bytes and its minimum length cannot be less than 3 bytes, instead one can think of creating a character variable to store such flags.
- For example, if a flag contains either 0 or 1, then it can be stored as a character variable having a length of 1 byte.

Tip # 2 – PROC DATASETS

- To efficiently utilize the available storage space, one can delete the SAS datasets in the WORK library, other temporary space, or permanent space when they are no longer needed.

This can be done using the **DATASETS** procedure as shown in the example below:

```
/* To clear ALL work datasets */
```

```
PROC DATASETS LIBRARY=WORK KILL NOLIST;  
QUIT;
```

Tip # 3 – **_NULL_**

- **_NULL_** can be used as a dataset name in the DATA statement when we need to execute a DATA step but do not want to create a SAS dataset.
- This is particularly useful while creating macro variables based on values obtained from an input SAS data set.
- For example:

```
/* To count number of observations in dataset */;
```

```
DATA _NULL_;  
    SET raw.testdrug END =eof;  
    IF eof THEN CALL SYMPUT('nobs', left(put(_N_)));  
RUN;
```

Tip # 4 – SAS Views

- **SAS Views** are virtual SAS datasets that can be used as an alternative to SAS datasets.
- Advantages:
 - Avoid unnecessary reading or writing of temporary datasets.
 - Occupy very small amount of storage space as compared to the space required by its original SAS data set.
- Drawbacks:
 - It takes additional time to process the data defined by a SAS VIEW as compared to processing a regular SAS dataset.

Tip # 5 - Merging

- Merging of two or more datasets can be done by either using **SQL JOIN** or **DATA Step MERGE** depending on the data situation.
- Since DATA Step MERGE requires prior sorting, it might result in creation of temporary datasets.
- SQL JOIN can be used to avoid such requirement, thus saving the storage space.

Tip # 6 - Subsetting

- Subsetting or filtering any unwanted records from a dataset reduce the size of the data thus saves the storage space.
- For this purpose, either a **WHERE** statement or an **IF** statement can be appropriately used depending on the underlying task and data situation.
- In most cases, a WHERE statement proves to be efficient and faster in performance than an IF statement.

Tip # 7 - Sorting

- Sorting a large dataset with several key variables can take enormous time and it can cause insufficient disk space or memory error.
- One can perform Sorting and Subsetting in the same SORT procedure as:
 - Takes lesser amount of processing time than total time consumed to individually subset a data and then sort it in two separate steps.
 - Only one dataset created in the WORK library instead of two thus saving the storage space.
- The TAGSORT option in the PROC SORT statement can be used when there insufficient disk space is available to sort a large SAS data set.

Summary II

- While working with Large datasets, one might come across few challenges or constraints that are hardly encountered while working with smaller size SAS datasets.
- There are many techniques available to overcome these challenges.
- It is very important to be aware of the benefits and trade-offs of each of the available technique depending on the data situation, project requirements and available resources.

References

- SAS Support, <http://support.sas.com/documentation/>
- Paul Gorrell, NESUG 2007, Numeric Length: Concepts and Consequences
- Andrew H. Karp and David Shamlin, SUGI Paper 3-28, Indexing and Compressing SAS® Data Sets: How, Why and Why Not
- Sunil Gupta, SAS Global Forum, Paper 213-2007, WHERE vs. IF Statements: Knowing the Difference in How and When to Apply
- Selvaratnam Sridharma, NESUG 2006, How to Reduce the Disk Space Required by a SAS® Data Set
- SAS Support, Sample 24804: *%SQUEEZE-ing Before Compressing Data, Redux*

Questions ?

THANK YOU!!

Any comments or suggestions are welcomed on
vishal.jain@quanticate.com