SESUG Paper 142-2018

Reducing the space requirements of SAS® data sets without sacrificing any variables or observations

Stephen Sloan, Accenture

ABSTRACT

The efficient use of space can be very important when working with large SAS data sets, many of which have millions of observations and hundreds of variables. We are often constrained to fit the data sets into a fixed amount of available space. Many SAS data sets are created by importing Excel or Oracle data sets or delimited text files into SAS and the default length of the variables in the SAS data sets can be much larger than necessary. When the data sets don't fit into the available space, we sometimes need to make choices about which variables and observations to keep, which files to zip, and which data sets to delete and recreate later.

There are things that we can do to make the SAS data sets more compact and thus use our space more efficiently. These things can be done in a way that allows us to keep all the desired data sets without sacrificing any variables or observations.

SAS has compression algorithms that can be used to shrink the space of the entire data set. In addition, there are tests that we can run that allow us to shrink the length of different variables and evaluate whether they are more efficiently stored as numeric or as character variables. These techniques often save a significant amount of space; sometimes as much as 90% of the original space is recouped. We can use macros so that data sets with large numbers of variables can have their space reduced by applying the above tests to all the variables in an automated fashion.

INTRODUCTION

We often run into situations where we just don't have enough room to store the temporary or permanent SAS data set we are creating. It could be due to a crowded disk drive, very long variables, a large number of observations, new restrictions on space used by data sets, or some combination of the above.

This paper outlines a 4-step procedure that can reduce the space occupied by SAS data sets without changing any values.

If the data set is very large, it probably has a large number of variables. Since we don't want to have to hard-code the size reduction tests and implementations for each variable, we use the meta-data in the SAS file sashelp.vtable to identify the numeric and character variables and their length.

STEP 1 – INITIALIZE THE PROGRAM

We want to use an OPTIONS statement at the beginning of the program to have the program and the output files take up as little space as possible. We do the following to initialize the program to accomplish this:

- Set the compression to reduce the data set's footprint. We usually use COMPRESS=BINARY, although it is generally not helpful when the data sets have a smaller number of variables, and it causes the program to run longer. However, COMPRESS=BINARY will cause the data set to use less space when the observations have "several hundred" bytes or more, according to the SAS web site. COMPRESS=YES can be used when there are not enough variables to justify COMPRESS=BINARY.
- Set REUSE=YES to reduce the amount of memory used while the program is running.
- Set ERRORS=0 to reduce the size of the log file.
- The following OPTIONS statement accomplishes the above three goals:

- OPTIONS COMPRESS=BINARY ERRORS=0 REUSE=YES;

Now we use macro commands to do the following:

- Identify the input and output data sets:
  - %LET ds=inputfile;
  - %LET outds=outputfile;
  - LIBNAME LG 'input directory';
  - LIBNAME OUT 'output directory';
  - Initialize the tracking data so that the size can be compared after each step:

```
*** Initialize the tracker for the sizes and diagnostics ***;
%LET stat=0;
%LET diag=0;

*** Update the tracker for the sizes ***;
%LET stat=%EVAL(&stat+1);

*** Save the size of the data set ***;
DATA _NULL_;
           SET sashelp.vtable(WHERE=(libname='LG' AND
memname="&ds" AND typemem='DATA'));
           CALL SYMPUT("label&stat","Initial size");
           CALL SYMPUT("figure&stat",filesize);
RUN;
QUIT;
```

- Separate the input SAS data set into two data sets, one with numeric variables and one with character variables:

```
*** Split data into numeric and character variables ***;
DATA numeric(KEEP=_NUMERIC_) character(KEEP=_CHARACTER_);
   SET x;
     RUN;
```

STEP 2 – REDUCE THE SIZE OF THE NUMERIC VARIABLES

Numeric variables in SAS data sets have a default length of 8. However, integers can have a lower length if their absolute value permits it. What we need to do is identify the numeric variables that have every value as integer or missing and then take the highest absolute value. At that point we can use the chart in the second section below to determine the minimum required length for the variable.

FIND THE NUMERIC VARIABLES THAT ARE ALL INTEGERS OR MISSING

The following code uses the INT function to determine whether a numeric variable value is an integer. We set a flag to indicate whether the value is an integer:

- IF var=INT(var) THEN flag /* for this variable */ =1;
- ELSE flag=0;

We can now use PROC SUMMARY with the MIN function and only reduce the size of those numeric variables whose flag has a minimum value of 1.

REDUCE THE SIZE OF THE NUMERIC VARIABLES THAT ARE ALL NUMERIC OR MISSING

The following chart is from the following link in support.sas.com:

http://support.sas.com/documentation/cdl/en/lrcon/69852/HTML/default/viewer.htm#p0ji1unv6thm0dn1gp4t01a1u0g6.htm

It shows the minimum length necessary for different absolute values:

Largest Integer That Can Be Safely Stored in a Given Length		
When Variable Length Equals ...	**Largest Integer z/OS**	**Largest Integer Windows/UNIX**
2	256	not applicable
3	65,536	8,192
4	16,777,216	2,097,152
5	4,294,967,296	536,870,912
6	1,099,511,627,776	137,438,953,472
7	281,474,946,710,656	35,184,372,088,832
8 (default)	72,057,594,037,927,936	9,007,199,254,740,992

Figure 1. Space occupied by numeric variables.

- Take the absolute values of the variables identified in the previous section as numeric or missing.
- Use PROC SUMMARY with the MAX function to get the highest absolute value.
- If the highest absolute value would provide a length less than 8, use the LENGTH statement to reduce the length of the variable.

CHANGE SMALL NUMERIC VARIABLES TO CHARACTER VARIABLES

Since 2 bytes is the smallest length for OS and 3 bytes is the smallest length for Windows or Unix, we can save even more space if we have integer variables whose value ranges from 0-9 in an OS system or 0-99 in a Windows or Unix system. We can change the variables with values from 0-9 to one-byte character variables and, in Windows or Unix, we can change the variables with values from 10-99 to 2-byte character variables to save space.

The DATA step code to change a variable's value from numeric to character length 1 is:

- `LENGTH old_value $ 1; /* Set the length and the character value */`
- `SET ds(RENAME=old_value=new_value); /* Rename the numeric variable */`
- `old_value=new_value; /* This changes the numeric to the character variable */`
- `DROP new_value;  /* Clean-up */`

STEP 3 – REDUCE THE SIZE OF THE CHARACTER VARIABLES

Many character variables do not use all of the space allocated to them. This is especially true when SAS data sets are imported from other platforms like Excel and Oracle, where many character variables default to very long lengths when imported into SAS.

CALCULATE THE MAXIMUM LENGTH OF EACH CHARACTER VARIABLE AND REDUCE THE SIZE

- Use the LENGTH function to get the length of each value of each character variable
- Use PROC SUMMARY with the MAX function to get the largest length for each character variable.
- Use the LENGTH statement to re-set the length of any character variables whose maximum length is less than the existing length.

CONVERT ALL-DIGIT CHARACTER VARIABLES TO NUMERIC VARIABLES

As we can see above, all-digit variables take up less space (have a smaller length) as numeric variables than as character variables. For example, the value 5000 takes up 4 bytes as a character variable but only 3 bytes as a numeric variable in Unix or Windows.

We should only do this if the character variable has length >=4 in Windows or Unix or length >=3 in z/OS, because 3 and 2 are the are the minimum lengths for numeric variables in Windows/Unix or z/OS, respectively.

- See which character variables contain only digits. We can do this by removing all digits from the variables using the COMPRESS function with the parameter d, and then looking at the variables that have length 0
    - `no_digits=COMPRESS(var,d);`
    - `length_without_digits=LENGTH(no_digits);`
    - Use PROC SUMMARY with MAX. If the max of length_without_digits is 0, the variable is an all-digit character variable.
- Convert the all-digit character variables to numeric variables
- The DATA step code to change a variable's value from character to numeric length 3 is:

```
LENGTH old_value 3; /* Set the length of the numeric variable */
SET ds(RENAME=old_value=new_value); /* Rename the character
variable */
old_value=new_value; /* This changes the character to the numeric
variable */
DROP new_value;  /* Clean-up */
```

STEP 4 – CONCATENATE THE DATA SETS WITH NUMERIC AND CHARACTER VARIABLES

Before we concatenate the SAS data sets with the character and numeric variables we need to test the conversions from numeric to character and from character to numeric. Since COMPRESS=BINARY compresses blank character values to a lower length than missing numeric values, it is possible that converting to a lower length will result in a larger data set if the variable in question has a large number of missing values. I have not quantified the exact cutoff for this but it would be an interesting follow-up to this paper.

We have four data sets to consider:

- Data set 1: The SAS data set with numeric variables before any were converted to character.
- Data set 2: The SAS data set with numeric variables after some or all were converted to character.

- Data set 3: The SAS data set with character variables before any were converted to numeric.
- Data set 4: The SAS data set with character variables after some or all were converted to numeric.

There are four ways to concatenate the numeric and character variables:

- Data set 1 and data set 3.
- Data set 1 and data set 4.
- Data set 2 and data set 3.
- Data set 2 and data set 4.

Whichever concatenation produces the smallest data set will be the output data set produced by the program.

OTHER ASPECTS OF THE PROGRAM

USE MACRO VARIABLES FOR ALL CALCULATIONS

There are often hundreds of variables in the large SAS data sets that are being reduced, so it is not practical to run the tests identified above for the variables by name. Instead we take the variables from the numeric and character data sets and assign them macro variable names with sequential numeric suffixes. Then, when running these variables through different tests, and assigning the appropriate MAX and MIN values for testing the reduction capabilities, we create new macro variables with the same numeric suffixes, so that they can be compared to the initial variables.

Here is an example where we test the maximum length for all-digit character variables to see if we gain space by converting from character to numeric:

```
*** Calculate the maximum value of the numeric variable ***;
      PROC SUMMARY DATA=character2_numeric NWAY MISSING MAX;
                        VAR
                  %DO I=1 %TO &ccount; /* number of character variables */
            %IF &&character_digit_max&i=0 %THEN %DO;
      /* Test for size without digits */
                        /* Test for length long enough to be reduced */
                  %IF &&character_length_max&i>=4 %THEN
                              %DO;
                                    &&cv&i
                              %END;
                  %END;
            %END;
            ;
            OUTPUT OUT=character_numeric_max (DROP=_TYPE_ _FREQ_) MAX=
                  %DO I=1 %TO &ccount;
                        %IF &&character_digit_max&i=0 %THEN %DO;
                              %IF &&character_length_max&i>=4 %THEN %DO;
                                    character_value_max&i
                              %END;
                        %END;
                  %END;
            ;
```

TRACK THE REDUCTION IN SIZE FROM EACH STEP OF THE PROCESS

The program stores the size of the data set and keeps a running spread sheet of the size that occurs after each iteration. At the end of the program a final spread sheet is produced that shows the size after each step. An example is shown below.

Data set	**Size in bytes**
Initial size of the SAS data set	57,747,456,000
Size of the SAS data set after COMPRESS=BINARY	4,788,715,520
Numeric variables	3,762,552,832
Character variables	1,027,080,192
Revised Numeric variables	3,424,649,216
Revised Character variables	821,428,224
Revised Concatenation of Numeric and Character variables	4,242,669,568
Revised Numeric variables after Character Conversion	2,842,296,320
Revised Character variables after Numeric Conversion	826,015,744
Size of Final Data Set	3,641,311,232
Reduction	**93.69%**

Figure 2 – Statistics on space reduction from a sample file

Note that, in this case, we achieved a 92% reduction by applying binary compression and then a further 24% reduction by applying the tests described above. Also, note that the character data set after the numeric conversions is larger than the original character data set, so we used the earlier character data set when concatenating to the numeric data set. This discrepancy is due to the reason mentioned above, the fact that blank character variables can use less space than missing numeric variables after binary compression.

REFERENCES

SAS Institute. 2010. SAS(R) 9.2 Companion for Windows, Second Edition. “Numeric Variables”. Accessed August 13, 2018. http://support.sas.com/documentation/cdl/en/hostwin/63285/HTML/default/viewer.htm#numvar.htm.

CONTACT INFORMATION

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

Stephen Sloan
Accenture
917-375-2937
Stephen.B.Sloan@accenture.com