Reassigning Variables length for Efficiency and Consistency

Mahipal Vanam Srinivas Vanam Phaneendhar Vanam

Percept Pharma Services, Bridgewater, NJ.

ABSTRACT

The macro presented in this paper is extension of a previously presented macro at the "Western Users of SAS Software 2014" titled "*Reassigning Character Variables length across Multiple datasets for Efficiency and Consistency*". The logic to reassign numeric variables length has been added to the older macro. In addition, the concept of numeric variables storage is briefly discussed. Since PROC FCMP has been used to create a user-defined function, this macro works only with SAS v9.2 or above. However, an alternate solution has been provided for SAS versions prior to 9.2 within the macro.

INTRODUCTION

Character Variables:

Reducing character variables length is a fairly straightforward concept. All we need to do is to find the length of the largest value present within a variable and assign it to that variable. Character variables store one character per byte.

Example:

```
data test ;
    length sname $20 ;
    sname = "Iron Man" ;
run;
```

In the TEST dataset, the variable SNAME variable will be created with a length of 20. But the text “Iron Man” is comprised of 9 characters which needs 9 bytes. So the additional 11 bytes is padded with blank spaces which is unused. These 11 bytes can be saved by reducing the length of SNAME to 9.

Numeric Variables:

To understand the process of reducing numeric variables length, it is required to have a little background knowledge on the storage of Numeric Variables. The numbers within the SAS system are stored as several digits per byte. In the following section we briefly describe the concept of storage of numeric variables.

NUMERIC VARIABLE STORAGE IN SAS

There are two types of Computer systems in general, namely Mainframes and Non-mainframes. The storage of numeric variables in these systems is different from each other. Numbers are stored within in the SAS system as a bit sequence. The storage in Mainframes is slightly different when compared to

Non-mainframes systems. Since Non-mainframes is the widely used platform, we will restrict our focus to them. Once you get an idea on Non-Mainframes, you will be able to relate the same concept for Mainframes very easily.

Non-mainframe Systems:

The default length is 8 bytes (64 bits) which can vary from 3 to 8. Let us take a closer look at the bit sequence. The first bit is reserved for Sign bit whose value is 0 for a positive number and 1 for a negative number, which is followed by 11 Exponent bits. The rest of the bits (up to 52) are used for Mantissa. So the range of numbers that can be stored in a variable of particular length depends on the number of available mantissa bits.

Example:

The 64-bit sequence of number 100 is as follows,

```
0100001001100010000000000000000000000000000000000000000000000000
```

So, if we split the bits,

Sign Bit: [0]

Exponent: [1000010 0110]

Mantissa: [0010 00000000 00000000 00000000 00000000 00000000 00000000]

The first 3 bits of Mantissa are sufficient to store 100. So the length can be reduced to 3(1 Sign bit, 11 Exponent bits and 4 Mantissa bits).

Mainframe Systems:

The default length is 8 bytes (64 bits) which varies from 2 to 8. The first bit is reserved for Sign bit whose value is 0 for a positive number and 1 for a negative number, which is followed by 7 Exponent bits. The rest of the bits (up to 56) are used for Mantissa. Like in Non-mainframes, the range of numbers that can be stored in a variable of a particular length depends on the number of available mantissa bits. The base used in Mainframes is 16 where as the base used in Non-Mainframes is 2. The bit sequence created for a number is different in Mainframes and Non-mainframes.

BIT SEQUENCE

In order to find the 64-bit sequence for a given number programmatically, you can use **BINARYw.** format.

Example:

```
data _null_ ;
     x = 100 ;
     put x binary64. ;
run;
```

After running this code, the following bit sequence will be printed in the SAS Log:

```
1    data _null_ ;
2        x = 100 ;
3        put x binary64. ;
4    run;

0100000001011001000000000000000000000000000000000000000000000000
NOTE: DATA statement used (Total process time):
      real time           0.37 seconds
      cpu time            0.04 seconds
```

Since the above example is executed on a Non-mainframe system, the bit sequence is printed in the SAS log is of the Non-Mainframe system.

TRUNC() FUNCTION

Decoding the bit sequence and finding the minimum length required for each value of a variable is a time consuming process. Fortunately, SAS provides us with a function called TRUNC(num,len) which does not directly return the minimum length, but helps in finding the minimum length with a simple logic. The function TRUNC(num,len) returns a value that is obtained by representing the bit sequence of "num" in the "len" number of bytes. If the returned value is equal to "num", then "len" is the sufficient length. If the returned value is not equal to “num”, then “len” is not sufficient for “num”. In the following section, the TRUNC() function is included within an algorithm that returns the minimum length.

ALGORITHM TO FIND MINIMUM LENGTH

1. Consider a number N and a variable L which iterates from 8 to 3.
2. Initially set L = 8.
3. If TRUNC(N,L)=N then go to step 4. If TRUNC(N,L)^=N then return (L+1) and exit.
4. If L^=3 then decrement L by 1 and go to step 3. If L=3, then return 3 and exit.

USER-DEFINED MINLEN() FUNCTION

Including the above algorithm at more than one instance makes the program look complicated. So it is a good programming practice to separate this algorithm from the actual macro. A user-defined function called MINLEN(num) has been written using PROC FCMP by incorporating the algorithm to find the minimum length for a particular number. Here is the code:

```
proc fcmp outlib = work.myfuncs.mylen ;
    function minlen(num);
        x = num;
        do L = 8 to 3 by -1 ;
            if x ne trunc(x,L) then do;
                return(L+1);
            end;
        end;
        return(3);
    endsub;
```

```
run;
options cmplib = work.myfuncs ;
data _null_ ;
    x=5;          mlen=minlen(x); put x= mlen=; /*x=5          mlen=3*/
    x=8192;       mlen=minlen(x); put x= mlen=; /*x=8192       mlen=3*/
    x=8193;       mlen=minlen(x); put x= mlen=; /*x=8193       mlen=4*/
    x=3.625;      mlen=minlen(x); put x= mlen=; /*x=3.625      mlen=3*/
    x=3.2;        mlen=minlen(x); put x= mlen=; /*x=3.2        mlen=8*/
    x=999999999; mlen=minlen(x); put x= mlen=; /*x=999999999 mlen=6*/
run;
```

%REASSIGN(TYPE=, FUNC=)

The macro REASSIGN() takes 2 arguments.

1. **TYPE=:** The possible values are N, C, B.

   N: reassigns only numeric variables length.

   C: reassigns only character variables length.

   B: reassigns both numeric and character variables length.

2. **FUNC=:** The possible values are E, C, B.

   E: performs only Efficiency (i.e., reassigns the variables length at each dataset level).

   C: performs only Consistency (i.e., reassigns the variables length at library level).

   B: performs both Efficiency and Consistency (i.e., reassigns at each dataset level first and then reassign at library level).

The macro **%REASSIGN(TYPE=,FUNC=)** internally calls 2 other macros based on the values of TYPE & FUNC arguments. They are:

1. **%EFFICIENCY(TYPE=):** The macro %EFFICIENCY() reassigns the variables length at each dataset level. The possible values for the argument TYPE are N, C, B which specify the macro to reassign only Numeric, only Character, both Numeric and Character Variables respectively.

2. **%CONSISTENCY():** The macro %CONSISTENCY reassigns the variables length at library level to have a consistency. Having consistency at library level, helps in avoiding a WARNING message **[Multiple lengths were specified for the variable <var> by input data set(s). This may cause unexpected results.]** when datasets that have common variables are merged. As per the Industry standards, it is a good practice to have variable attributes consistent across the datasets of a Study.

CONCLUSION

The section 2.4 of FDA Study Data Specifications v2.0 specifies the following text:

2.4 Sizing of Columns

For all datasets, in order to significantly reduce dataset file sizes, the allotted character column length/size for each column should be the maximum length used. Lengths/sizes of columns should not arbitrarily be set to 200. For example, if USUBJID has a maximum length of 18, the USUBJID's column size should be set to 18, not 200. An inclusion of a small amount of padding to column width may be acceptable as long as this doesn't result in significant increases in file size.

FDA does not have any specification on numeric variable length. However, this macro would be useful in those scenarios where reducing all variables length is required.

REFERENCES

1. "***Reassigning Character Variables length across Multiple datasets for Efficiency and Consistency***" by Mahipal Vanam, Phaneendhar Vanam, Srinivas Vanam.
2. *"**Numeric Precision in SAS Software**"* from SAS website. [http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a000695157.htm].

CONTACT INFORMATION

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

Mahipal Vanam mahipal.vanam@perceptservices.com

Srinivas Vanam srinivas.vanam@gmail.com

Phaneendhar Vanam phaneendhar.v@gmail.com

ACKNOWLEDGEMENTS

- Percept Pharma Services (www.perceptservices.com)
  1031, US Highway 22W, Suite 304,
  Bridgewater, NJ. 08807

- I would like to take this opportunity to thank my Manager at Bristol-Myers Squibb, **Mr. David Jurek** for his constant support and encouragement. – Srinivas Vanam.

- I would like to take this opportunity to thank my Director at Pharmacyclics, **Ms. Linda Gua** for her constant support and encouragement during my Project. – Phaneendhar Vanam.

TRADEMARK INFORMATION

APPENDIX: SOURCE CODE

```
/*************************************************************************************************
 ** Project:                  South Central SAS Users Group (SCSUG 2015)                        **
 **                                                                                             **
 ** Title:                    Reassigning variable lengths for Efficiency and Consistency       **
 **                                                                                             **
 ** Program:                  reassign.sas                                                      **
 **                                                                                             **
 ** Programmers:              Mahipal Vanam                                                     **
 **                           Phaneendhar Vanam                                                 **
 **                           Srinivas Vanam                                                    **
 **                                                                                             **
 ** Date:                     14-May-2015                                                       **
 **                                                                                             **
 ** Modification History:                                                                       **
 **                                                                                             **
 **                                                                                             **
 **                                                                                             **
 **                                                                                             **
 *************************************************************************************************/


/*************************************************************************************************
                    Creating a User-Defined Function MINLEN(num)

 ** If this program is executed on a SAS version prior to 9.2,                                  **
 ** Please comment this section of the program, as it will not recognize FCMP procedure.       **
 *************************************************************************************************/
proc fcmp outlib = work.myfuncs.mylen ;
  function minlen(num);
    x = num;
      do L = 8 to 3 by -1 ;
      if x ne trunc(x,L) then do;
        return(L+1);
      end;
    end;
    return(3);
  endsub;
run;
options cmplib = work.myfuncs ;


/*************************************************************************************************
                                        Preprocessing
 *************************************************************************************************/

/* Cleaning up the LOG and OUTPUT windows */
dm "log; clear; lst; clear;" ;

/* Setting Global options */
options symbolgen mprint ;
options minoperator mindelimiter="," ;

/* VARLENCHK option supresses the WARNING message from being printed in the SAS Log */
options varlenchk=nowarn;

/* Assigning Library references to SOURCE and OUTPUT directories */
libname source "C:\Users\vanams\Desktop\SUG Papers\reassign\source" ;
libname output "C:\Users\vanams\Desktop\SUG Papers\reassign\output" ;

/* Getting the count and list of INPUT datasets */
proc sql noprint ;

  select count(*) into :dscount
  from dictionary.tables
  where upcase(libname) = "SOURCE" ;
   select memname into :dsnames separated by " "
  from dictionary.tables
  where upcase(libname) = "SOURCE" ;
```

```
  %put >>> No. of Input Datasets: &dscount. <<< ;
  %put >>> List of Datasets:      &dsnames. <<< ;

quit;


/*****************************************************************************************
                    Macro to reassign the Variables length for Efficiency
 *****************************************************************************************/
%macro efficiency(type=);
%global nvars;
%global cvars;
%global cvarscnt;
%global nvarscnt;
  %do i = 1 %to &dscount ;
    %let dsn = %scan(&dsnames,&i);
      proc sql noprint;

        /* Character Variable # & list */
        select distinct(name) into: cvars separated by " "
        from sashelp.vcolumn
        where upcase(libname)="SOURCE" and upcase(memname)="&dsn." and upcase(type)="CHAR";

        select count(distinct name) into: cvarscnt separated by " "
        from sashelp.vcolumn
        where upcase(libname)="SOURCE" and upcase(memname)="&dsn." and upcase(type)="CHAR";

        /* Numeric Variable # & list */
        select distinct(name) into: nvars separated by " "
        from sashelp.vcolumn
        where upcase(libname)="SOURCE" and upcase(memname)="&dsn." and upcase(type)="NUM";

        select count(distinct name) into: nvarscnt separated by " "
        from sashelp.vcolumn
        where upcase(libname)="SOURCE" and upcase(memname)="&dsn." and upcase(type)="NUM";

        select name into: retain_&dsn. separated by " "
        from sashelp.vcolumn
        where upcase(libname)="SOURCE" and upcase(memname)="&dsn."
        order by varnum;

        %put [MESSAGE:] Dataset:              &dsn. ;
        %put [MESSAGE:] Char Variable #:      &cvarscnt. ;
        %put [MESSAGE:] Char Variable list:   &cvars. ;
        %put [MESSAGE:] Num Variable #:       &nvarscnt. ;
        %put [MESSAGE:] Num Variable list:    &nvars. ;
        %put [MESSAGE:] Variable order:       &&retain_&dsn. ;

      quit;

      data output.&dsn.;
        set source.&dsn.;
      run;


      /***********************
         Character Variables
       ***********************/
      %if &type. in (C,B) %then %do;
        %if &cvarscnt. ne 0 %then %do;
          %do j=1 %to &cvarscnt.;
            %let dsnvar&j.=%sysfunc(scan(&cvars.,&j.,' '));
            %put &&dsnvar&j.;

            proc sql noprint;
              /* Get the Largest value length */
              select max(length(&&dsnvar&j.)) into: mlen&j. from source.&dsn.;
            quit;

            data output.&dsn.;
```

```
        retain &&retain_&dsn ;
        length &&dsnvar&j.  $ &&mlen&j. ;
        set output.&dsn. ;
      run;
    %end;
  %end;
  %else %do;
    %put [MESSAGE:] No Character Variables in dataset: &dsn. ;
  %end;
%end;


/***********************
   Numeric Variables
 ***********************/
%if &type. in (N,B) %then %do;
  %if  &nvarscnt. ne 0 %then %do;
    %do j=1 %to &nvarscnt. ;
      %let dsnvar&j.=%sysfunc(scan(&nvars.,&j.,' '));
      %put &&dsnvar&j.;

    /***********************************
        For SAS versions 9.2 or later
     ***********************************/
    %if &sysver. in (9.2,9.3,9.4,9.5) %then %do ;
      proc sql noprint;
        /* Get the MAX of minimum lengths */
        select max(minlen(&&dsnvar&j.)) into: mlen&j. from source.&dsn.;
      quit;
    %end;

    /*********************************
      For SAS versions prior to 9.2
             Alternative code(not using FCMP)
     *********************************/
    %else %do;
      data temp ;
        set source.&dsn.;
        x = &&dsnvar&j.;
        if x ne . then do;
          do L = 8 to 3 by -1 ;
            if x NE trunc(x,L) then do;
                minlen = L + 1 ;
                output;
                return;
            end;
          end;
          if L = 2 then do ;
            minlen = 3 ;
            output;
            return;
          end;
        end;
        else do ;
          /* If the value is missing, take the min length as 3 */
          minlen = 3 ;
          output;
          return;
        end;
      run;

      proc sql noprint;
        /* Get the maximum length */
        select max(minlen) into: mlen&j. from temp;
      quit;
    %end;

      data output.&dsn.;
        retain &&retain_&dsn ;
        length &&dsnvar&j.  &&mlen&j. ;
        set output.&dsn. ;
```

```
            run;
          %end;
        %end;
        %else %do;
          %put [MESSAGE:] No Numeric Variables in dataset: &dsn. ;
        %end;
      %end;


      /***********************
          TYPE not in N,C,B
       ***********************/
      %if not (&type. in (N,C,B)) %then %do ;
        %put [MESSAGE:] The values for TYPE should be N/C/B only ;
      %end;
  %end;

%mend;


/*************************************************************************************************
              Macro to reassign the Variables length for Consistency
 *************************************************************************************************/
%macro consistency(func=);

  %if &func. = C %then %do ;
    %do i = 1 %to &dscount ;
      %let dsn = %scan(&dsnames,&i);

      data output.&dsn. ;
        set source.&dsn. ;
      run;
    %end;
  %end;
  %else %if &type. = B %then %do ;
     /* Do Nothing */
  %end;


  /*************************************************************************
          Finding the max.length for a variable across multiple datasets
   *************************************************************************/
  proc sql;
    create table var_attr as
      select memname,type,upcase(name) as name,length
      from dictionary.columns
      where upcase(libname)="OUTPUT"
    ;

    create table var_uniq as
      select type,name,max(length) as maxlen
      from var_attr
      group by type,name
    ;

    create table var_attr_max as
      select memname,a.type,a.name,length,maxlen
        from var_attr as a,var_uniq as b
        where a.type = b.type and a.name = b.name
        order by memname,type,name
    ;
  quit;


  /*************************************************************************
          Reassigning the maximum lengths for all variables
   *************************************************************************/
  %do i = 1 %to &dscount. ;

    %let dsn = %scan( &dsnames. , &i. ) ;
      proc sql noprint ;
```

```
        select strip(name) || " $ " || put(maxlen,4.) into :stmt1 separated by "  "
        from var_attr_max
        where memname = "&dsn." and upcase(type)="CHAR" ;

        select strip(name) || " " || put(maxlen,4.) into :stmt2 separated by "  "
        from var_attr_max
        where memname = "&dsn." and upcase(type)="NUM" ;

        select name into: retain_&dsn. separated by " "
        from sashelp.vcolumn
        where upcase(libname)="SOURCE" and upcase(memname)="&dsn."
        order by varnum;
      quit;

    data output.&dsn. ;
      retain &&retain_&dsn. ;
      length &stmt1. &stmt2. ;
      set output.&dsn. ;
    run;

    %put Dataset:      &dsn.           ;
    %put Var. lengths: &stmt1. &stmt2. ;

  %end;

%mend ;



%macro reassign(type=,func=) ;

  /*************************************************************************
                Calling the macros based on requirement
   *************************************************************************/
  %if &func. = E %then %do ;
    %efficiency(type=&type.) ;
  %end;
  %else %if &func. = C %then %do ;
    %consistency(func=C) ;
  %end;
  %else %if &func. = B %then %do ;
    %efficiency(type=&type.);
    %consistency(func=B) ;
  %end;
  %else %do ;
    %put [MESSAGE:] The values for type should be E/C/B only ;
  %end;

%mend;


/*****************************************************************************************
                                Calling only EFFICIENCY macro
 *****************************************************************************************/
%*reassign(type=N,func=E);
%*reassign(type=C,func=E);
%*reassign(type=B,func=E);


/*****************************************************************************************
                                Calling only CONSISTENCY macro
 *****************************************************************************************/
%*reassign(type=N,func=C);
%*reassign(type=C,func=C);
%*reassign(type=B,func=C);


/*****************************************************************************************
                          Calling both EFFICIENCY and CONSISTENCY macros
 *****************************************************************************************/
%*reassign(type=N,func=B);
```

```
%*reassign(type=C,func=B);
%reassign(type=B,func=B);



/** ****** REASSIGN MACRO PARAMETERS ******
 **
 ** TYPE = N => NUMERIC:     Reassign only Numeric Variables length
 ** TYPE = C => CHARACTER:   Reassign only Character variables length
 ** TYPE = B => BOTH:        Reassign both Numeric & Character variables length
 **
 ** FUNC = E => EFFICIENCY:  Perform only Efficiency(Dataset level)
 ** FUNC = C => CONSISTENCY: Perform only Consistency(Library level)
 ** FUNC = B => BOTH:        Perform both Efficiency & Consistency(Dataset & Library Level)
 **
 **/
```