SESUG Paper 118-2018

Reducing the Length of Character Variables in a SAS ® Data Set

Bruce Gilsen, Federal Reserve Board, Washington, DC

ABSTRACT

In the big data era, reducing the defined length of character variables to their actual maximum length in any observation is one way to reduce disk storage use and improve processing time with no loss of information. A simple way to do this is with the FIXLENG macro, which determines the maximum length for each character variable, and if any character variables have a maximum length smaller than their defined length does the following:

1. Prints a summary report listing the defined length and maximum length for each character variable
2. Based on the value of a user-specified parameter either (1) reduces the size of all character variables to their maximum length, or (2) generates and displays SAS ® code that would reduce the size of all character variables to their maximum length

FIXLENG provides a simple alternative to compression, which is commonly used to reduce disk storage. One advantage of the FIXLENG macro is that it is only executed once, whereas a data set must be compressed every time it is written and uncompressed every time it is read.

EXAMPLE USED IN THIS PAPER: DATA SET AND MACRO INVOCATION

Data set ONE contains one numeric variable, X, and four character variables, whose lengths were defined with the following statement.

```
length itemid $10 othervar1 $10 othervar2 $5 othervar3 $10;
```

Data set ONE has the following values.

```
Obs    x    itemid        othervar1    othervar2    othervar3
 1     1    abcdef          aaa          ddddd          a
 2     2    ghijklmnop      bbbbb        eeeee          b
 3     3    qrstuvwxyz      cccc         fffff          c
```

The character variables in ONE have the following characteristics.

Variable name	Defined length	Length in each observation versus defined length
ITEMID	10	Some observations equal to the length, some shorter
OTHERVAR1	10	Always shorter than the length, longest length=5
OTHERVAR2	5	Always equal to the length
OTHERVAR3	10	Always shorter than the length, longest length=1

The lengths of variables OTHERVAR1 and OTHERVAR3 can be shortened with no loss of information.

In this paper, we will invoke the FIXLENG macro to shorten the character variables in data set ONE in the WORK library as follows.

```
%fixleng(data=one, update=yes)
```

SUMMARY OF FIXLENG MACRO

The FIXLENG macro contains the following steps, which are described in the next several sections.

- Process the user-specified data set: separate it into libref (or WORK if omitted) and data set name in upper case.
- Copy character variable names and lengths to macro variables; terminate if no character variables.
- Determine the maximum length for each character variable.
- Determine if any character variables can be shortened. If not, then print a message and terminate; otherwise proceed.
- Print a summary report listing the defined length and maximum length for each character variable, and generate SAS code to reduce the length of the character variables that can be shortened.
- Based on a user-specified parameter, either submit or just display the code to reduce the length of the character variables.

PROCESS THE USER-SPECIFIED DATA SET

- The DATA= parameter is required so the macro terminates if it is not provided.
- The DICTIONARY.COLUMNS table, which is read in the next step, contains librefs and data set names in upper case so macro variable DATA is converted to upper case, then split it into libref (or WORK if omitted) and data set name.

```
%macro fixleng(data=, update=no);
  %local data_upcase library_name member_name character_vars
         length_character_vars num_charvars i;

  %if &data= %then %do;
    %put NOTE: Data set not specified, macro ends;
    %return;
  %end;
  %let update=%lowcase(&update); /* Make lowercase for simplicity */

    /* Convert data set name to upper case to match value in
       DICTIONARY table, then determine library and member name. */
  %let data_upcase = %upcase(&data);
  %let member_name=%scan(&data_upcase, -1, %str(.));
  %let library_name=%scan(&data_upcase, 1, %str(.));
  %if &library_name = &member_name
    %then %let library_name=WORK;
```

In our example, the macro variables have the following values.

```
MEMBER_NAME:                ONE
LIBRARY_NAME:               WORK
```

COPY CHARACTER VARIABLE NAMES AND LENGTHS TO MACRO VARIABLES; TERMINATE IF NO CHARACTER VARIABLES

The following macro variables are created.

- CHARACTER_VARS is a space-separated list of character variable names.
- LENGTH_CHARACTER_VARS is a space-separated list of the lengths of the character variables.
- NUM_CHARVARS is the number of character variables. It's copied from the automatic macro variable SQLOBS, which contains the number of rows selected by the last PROC SQL statement. SQLOBS is overwritten by the next PROC SQL statement, so copying it to NUM_CHARVARS preserves the value.

If the data set has no character variables, notify the user and terminate.

```
proc sql noprint;
   select name, length
     into :character_vars separated by ' ',
          :length_character_vars separated by ' '
     from dictionary.columns
     where libname="&library_name"
       and memname="&member_name"
       and type = "char";
   %let num_charvars=&sqlobs;
quit;
%if &num_charvars=0 %then %do;
   %put NOTE: No character variables in data set &data, macro ends;
   %return;
%end;
```

In our example, the macro variables have the following values. Note that the lengths of variables OTHERVAR1 and OTHERVAR3 can be shortened.

```
CHARACTER_VARS:            itemid othervar1 othervar2 othervar3
LENGTH_CHARACTER_VARS:     10 10 5 10
NUM_CHARVARS:              4
```

DETERMINE THE MAXIMUM LENGTH FOR EACH CHARACTER VARIABLE

The following arrays are defined.

- CHAR_VARS contains the values of the character variables in the data set. The array elements in our example are as follows:

  ```
  ITEMID OTHERVAR1 OTHERVAR2 OTHERVAR3
  ```

- CHAR_VARS_LENGTH has the defined length of the character variables in array CHAR_VARS. The array values in our example are as follows:

  ```
  10 10 5 10
  ```

- CHAR_VARS_MAX_LENGTH has the maximum length of each character variable found so far. All values are initialized to 1.

In the DO loop, whenever the length of a character variable in the current observation is larger than the largest length found so far, the appropriate element of CHAR_VARS_MAX_LENGTH is updated with the new maximum length.

```
data _null_;
  set &data end=last;
  array char_vars (&num_charvars) &character_vars;
  array char_vars_length (&num_charvars) _temporary_
        (&length_character_vars);
  array char_vars_max_length (&num_charvars) _temporary_
        (&num_charvars*1);
  length varname $32;
  do i = 1 to dim(char_vars);
    char_vars_max_length(i) =
    max(length(char_vars(i)), char_vars_max_length(i));
  end;
```

In our example, when this code has executed for all three observations, the four elements of the arrays contain the following values. OTHERVAR1 and OTHERVAR3 can be shortened.

	For ITEMID	For OTHERVAR1	For OTHERVAR2	For OTHERVAR3
CHAR_VARS_LENGTH:	10	10	5	10
CHAR_VARS_MAX_LENGTH:	10	5	5	1

DETERMINE IF ANY CHARACTER VARIABLES CAN BE SHORTENED. IF NOT, PRINT A MESSAGE AND TERMINATE, OTHERWISE PROCEED

After reading the entire data set, we loop through the length arrays, and one of two things occurs.

- We stop when we find a character variable whose maximum length and defined length are not equal. In the IF statement after the DO loop, the loop index, I, will be less than or equal to the size of the CHAR_VARS array, indicating that one or more character variables can be shortened.
- We finish looping through all array elements without finding any character variables whose maximum length and defined length are not equal. In the IF statement after the DO loop, the loop index, I, will be one greater than the size of the CHAR_VARS array. We notify the user that there is no action to take and terminate.

```
if last then do;
  do i = 1 to dim(char_vars) while
      (char_vars_length(i) = char_vars_max_length(i));
  end;
if i = dim(char_vars)+1
  then put "NOTE: all variables are minimum length, no action taken";
  else do;
```

In our example, I is 2 because the maximum length and defined length differ for the second variable, OTHERVAR1, causing the loop to stop executing.

PRINT A CHARACTER VARIABLE LENGTH REPORT AND GENERATE SAS CODE TO REDUCE LENGTH OF CHARACTER VARIABLES

In this section, we print a summary report listing the defined length and maximum length for each character variable to the SAS log and generate SAS code to reduce the length of character variables that can be shortened.

```
          /* Print header for length report */
        put @1 "Variable length for all character variables in data set
  &data:" //
          @1 "Variable" @34 "Variable" @44 "Longest length in"
             @63 "Shorten?"/
          @34 "length" @44 "any observation";

        ii=0; /* count number of LENGTH statements */
        do i=1 to dim(char_vars);
          varname = vname(char_vars(i));

            /* Print length report */
          if char_vars_max_length(i) = char_vars_length(i)
          then put @1 varname @34 char_vars_length(i) 5. -r
                    @44 char_vars_max_length(i) 5. -r;
          else do;
              /* Current variable will be shortened */
            put @1 varname @34 char_vars_length(i) 5. -r
                @44 char_vars_max_length(i) 5. -r @63 "shorten";
            length_statement=cat("length ",varname," $",
                char_vars_max_length(i),";");

              /* Generate macro variables containing LENGTH statements to
                 shorten variables for subsequent execution or display. */
            ii=ii+1; /* count number of LENGTH statements */
            call symput("lengthtrim" || compress(put(ii,5.)),
              compbl(length_statement));

          end; /* of else do*/
        end; /* of do for do i = dim(char_vars) */
      end; /* of else do for if i = dim(char_vars)+1 */

      call symputx("num_lengthtrim",ii); /* how many LENGTH statements */
    end; /* of if last then do; */
  run;
```

In our example, the following report is written to the SAS log.

```
Variable length for all character variables in data set one:

 Variable                        Variable  Longest length in  Shorten?
                                 length    any observation
 itemid                             10         10
 othervar1                          10          5              shorten
 othervar2                           5          5
 othervar3                          10          1              shorten
```

The following macro variables are created in this step.

- LENGTHTRIM1, LENGTHTRIM2,… contain LENGTH statements that shorten character variables.
- NUM_LENGTHTRIM contains the number of variables to be shortened, which equals the number of LENGTHTRIM*n* macro variables created.

In our example, two variables, OTHERVAR1 and OTHERVAR3, can be shortened, and the macro variables have the following values.

```
NUM_LENGTHTRIM:          2
LENGTH_TRIM1:            length othervar1 $5;
LENGTH_TRIM2:            length othervar3 $1;
```

EITHER SUBMIT OR JUST DISPLAY THE CODE TO REDUCE THE LENGTH OF THE CHARACTER VARIABLES

If macro variable UPDATE is yes, generate and submit a DATA step that shortens the lengths of character variables whose maximum length is less than their defined length. Otherwise, generate and display but do not submit a DATA step that shortens the length of the character variables.

Note that the length of a character variable is determined from its first occurrence in the DATA step, and subsequent occurrences in the DATA step cannot change the length. Therefore, the LENGTH statements must precede the SET statement in the DATA step.

```
 %if &update=yes %then %do; /* Update the data set with shorter lengths */
   data &data;
   %do i= 1 %to &num_lengthtrim;
     &&lengthtrim&i;
   %end;
   set &data;
   run;
   %put NOTE: &num_lengthtrim character variables in data set &data
shortened;
 %end; /* of %if &update=yes %then %do; */

 %else %do; /* Display but do not submit code that reduces lengths */
   %put Data set &data not changed, DATA step statements to shrink
character variables are:;
   %put ;
   %put data &data %str(;);
   %do i= 1 %to &num_lengthtrim;
     %put &&lengthtrim&i;
   %end;
   %put set &data %str(;);
   %put run %str(;);
 %end; /* of %else %do for %if &update=yes %then %do; */
%mend fixleng;
```

In our example, the following DATA step is submitted if macro variable UPDATE is yes. Otherwise, the DATA step is displayed in the SAS log.

```
data one ;
   length othervar1 $5;
```

```
  length othervar3 $1;
  set one ;
run ;
```

CONCLUSION

This paper discussed the FIXLENG macro, which provides an easy way to reduce disk space usage by reducing the length of character variables to their actual maximum length. This one-time process is a simple alternative to compressing a data set every time it's output and uncompressing the data set whenever it's read.

REFERENCES

SAS Institute Inc. (2017), "Base SAS 9.4 Procedures Guide, Seventh Edition," Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2016), "SAS 9.4 Statements: Reference, Fifth Edition," Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2016), "SAS 9.4 Macro Language: Reference, Fifth Edition," Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2009), SAS Note 35230, "Shrinking character variables to minimum required length." <http://support.sas.com/kb/35/230.html>

ACKNOWLEDGMENTS

The following people contributed extensively to the development of this paper and their support is greatly appreciated: Peter Sorock, Heidi Markovitz, and Donna Hill at the Federal Reserve Board. Special thanks to Heidi for various improvements to the code. SAS Note 35230, which I discovered after writing the FIXLENG macro, takes a similar approach to the non-macro portion of FIXLENG, though it also updates the FORMAT of all character variables.

CONTACT INFORMATION

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

Bruce Gilsen
Federal Reserve Board, Mail Stop N-122, Washington, DC 20551
202-452-2494
bruce.gilsen@frb.gov

APPENDIX: THE FIXLENG MACRO

For ease of use, the entire FIXLENG macro is displayed in this section.

```
%macro fixleng(data=, update=no);
  %local data_upcase library_name member_name character_vars
         length_character_vars num_charvars i;

    /*****************************************************************
    Section One. Process the user-specified data set.
    *****************************************************************/

  %if &data= %then %do;
    %put NOTE: Data set not specified, macro ends;
    %return;
  %end;
  %let update=%lowcase(&update); /* Make uppercase for simplicity */

    /* Convert data set name to upper case to match value in
       DICTIONARY table, then determine library and member name. */
  %let data_upcase = %upcase(&data);
  %let member_name=%scan(&data_upcase, -1, %str(.));
  %let library_name=%scan(&data_upcase, 1, %str(.));
  %if &library_name = &member_name
    %then %let library_name=WORK;

      /*****************************************************************
      Section Two. Copy character variables and their lengths to macro
      variables; terminate if no character variables.
      *****************************************************************/

  proc sql noprint;
  select name, length
    into :character_vars separated by ' ',
         :length_character_vars separated by ' '
    from dictionary.columns
    where libname="&library_name"
    and memname="&member_name"
    and type = "char";
    %let num_charvars=&sqlobs;
  quit;
  %if &num_charvars=0 %then %do;
    %put NOTE: No character variables in data set &data, macro ends;
    %return;
  %end;

    /*****************************************************************
    Section Three. Process the user-specified data set.
    *****************************************************************/

  data _null_;
    set &data end=last;
    array char_vars (&num_charvars) $32 &character_vars;
    array char_vars_length (&num_charvars) _temporary_
         (&length_character_vars);
    array char_vars_max_length (&num_charvars) _temporary_
         (&num_charvars*1);
```

```
      length varname $32;
      do i = 1 to dim(char_vars);
         char_vars_max_length(i) =
           max(length(char_vars(i)), char_vars_max_length(i));
      end;

        /*****************************************************************
        Section Four. Determine if any character variables can be shortened:
        if not, print a message and terminate, otherwise proceed.
        *****************************************************************/

      if last then do;
        do i = 1 to dim(char_vars) while
            (char_vars_length(i) = char_vars_max_length(i));
        end;
        if i = dim(char_vars)+1
        then put "NOTE: all variables are minimum length, no action taken";
        else do;

            /*************************************************************
            Section Five. Print a character variable length report and
            generate SAS code to reduce length of character variables.
            *************************************************************/

            /* Print header for length report */
          put @1 "Variable length for all character variables in data set
&data:" //
              @1 "Variable" @34 "Variable" @44 "Longest length in"
                 @63 "Shorten?"/
              @34 "length" @44 "any observation";

          ii=0; /* count number of LENGTH statements */
          do i=1 to dim(char_vars);
            varname = vname(char_vars(i));

              /* Print length report */
            if char_vars_max_length(i) = char_vars_length(i)
            then put @1 varname @34 char_vars_length(i) 5. -r
                     @44 char_vars_max_length(i) 5. -r;
            else do;
                /* Current variable will be shortened */
              put @1 varname @34 char_vars_length(i) 5. -r
                  @44 char_vars_max_length(i) 5. -r @63 "shorten";
              length_statement=cat("length ",varname," $",
                  char_vars_max_length(i),";");

                /* Generate macro variables containing LENGTH statements to
                   shorten variables for subsequent execution or display. */
              ii=ii+1; /* count number of LENGTH statements */
              call symput("lengthtrim" || compress(put(ii,5.)),
                compbl(length_statement));

            end; /* of else do*/
          end; /* of do for do i = dim(char_vars) */
        end; /* of else do for if i = dim(char_vars)+1 */

        call symputx("num_lengthtrim",ii); /* how many LENGTH statements */
```

```
    end; /* of if last then do; */
  run;

    /*****************************************************************
    Section Six. Either submit or display (but do not submit) the code
    to reduce the length of the character variables.
    *****************************************************************/

  %if &update=yes %then %do; /* Update data set with shorter lengths */
    data &data;
      %do i= 1 %to &num_lengthtrim;
        &&lengthtrim&i;
      %end;
      set &data;
    run;
    %put NOTE: Character variables in data set &data shortened;
  %end; /* of %if &update=yes %then %do; */

  %else %do; /* Display but do not submit code that reduces lengths */
    %put Data set &data not changed, DATA step statements to shrink
character variables are:;
    %put ;
    %put data &data %str(;);
    %do i= 1 %to &num_lengthtrim;
      %put &&lengthtrim&i;
    %end;
    %put set &data %str(;);
    %put run %str(;);
  %end; /* of %else %do for %if &update=yes %then %do; */
%mend fixleng;
```