PharmaSUG 2021 - Paper EP-105

Restriction of Macro variable length – Dynamic approach to overcome

Kalyani Telu, The Henry M. Jackson Foundation

ABSTRACT

Macro variables are one of the most powerful tools available in SAS with the maximum length of 65534 characters. Programmers sometimes face the situation to store data larger than this allotted limit, and it is difficult to overcome. This paper discusses the scenarios where the user needs to store large data and dynamic approaches to circumnavigate the maximum limitation.

INTRODUCTION

A macro variable is a symbolic variable that stores the data. It can be used anywhere in the program and as many times as needed to resolve the stored data. Often, SAS users need to search for a huge set of data points or manipulate several hundreds of variables. In such scenarios, the entire list can be stored as a macro variable by reading from a dataset, and later we can use it in the program as many times as required by simply calling the macro variable. However, the major constraint is how much data can be stored in each variable. Maximum character length of a single macro variable is 65534 characters. Every time, macro variable length exceeds this limit, SAS gives an error. To avoid this error, programmers need to partition the entire code which is repetitive and very time consuming. As such, an innovative approach or a compiled macro is highly useful to maximize the power of macros.

SCENARIO-1: IMPORTING TEXT FILE WITH A LARGE NUMBER OF VARIABLES WITH LAYOUT STORED IN SEPARATE FILE

Reading large text files with hundreds of variables with their formats and labels stored in a separate file is the first scenario. This is challenging as it requires hard coding, which often leads to human error. Steven Yan, Mu Hu published a paper with an idea of using the Select into: Clause of Proc SQL to host the macro variables for both variable attributes and variable labels. This is a very useful approach where the variable and associated format or labels were stored in a macro variable and simply use that macro variable in regular data statements to read the text file. However, this approach will not work when variable labels are large and macro variable with character length exceeding 65534 characters. Example: public survey data.

Below is the code to assign labels to large data with macro variable exceeding character limit:

```
* Generate layout with labels;
data layout(keep=variable label);
   length variable $10 label $100;
   do i=1 to 10000;
      do j=1 to 10;
         substr(variable,j)=byte(int(65+26*ranuni(0)));
      do k=1 to 100;
           substr(label,k)=byte(int(65+26*ranuni(0)));
      end;
      end;
      output;
   end;
run;

*Generate the survey data;
```

For the purpose of demonstration of the program, only data structure without any data is created.

```
 data survey;
   length variable $10;
   do i=1 to 10000;
```

```
        do j=1 to 10;
           substr(variable,j)=byte(int(65+26*ranuni(0)));
        end;
        output;
      end;
  run;

  * Use Select Into: of Proc SQL to host variable Labels;
  proc sql;
  select variable || ' ' || label
  into :varLBL separated by ''
  from layout;
  quit;
```

```
ERROR: The length of the value of the macro variable VARLBL (65540) exceeds the maximum length
       (65534). The value has been truncated to 65534 characters.
```

```
*Assign the labels to each variable;

data survey;

  set survey;

  label &varLBL;

run;
```

NEW APPROACH TO BYPASS THE ERROR:

One approach to bypass this error is to write a program using ‘put’ statements which writes another program that executes data labelling.

```
filename survey 'H:\sas global forum\2021\survey_label.sas' ;

 data _null_ ;

   set layout end=eof;

   file survey ;

   if _n_ = 1 then do;

   put "proc datasets /*lib=work */nolist;" /

   " modify survey;";

   end;

    if indexc(label,"'")=0 then

       put " label " variable " = '" label "';";

    else if indexc(label,'"')=0 then

      put " label " variable ' = "' label '";';

    if eof then put "quit;";

 run;

 %include survey;

 filename survey clear;
```

SCENARIO-2: SEARCHING DATA POINTS

Searching handful of data points through a large data set is simple if the data points are stored in a macro variable and later resolved with a conditional query. With the same macro variable character limitation, it is a tedious task to search a large data set. The above described approach can also be applied in this scenario.

```
*Generate search list;
data search_list;
  set layout (keep=variable obs=6000);
run;
filename search 'H:\sas global forum\2021\search_list.sas' ;
data null;
  set search_list end=eof;
  file search;
  if _n_=1 then do;
  put "data found; set layout; if variable in ('";
  end;
  put  variable "','";
 if eof then put " 0'); run;";
run;
%include search_list;
```

Though this approach works, there may be other scenarios where creating macro variable could be the ultimate solution. Since macro variables have the character limitation, we can combine more than one macro variable, and append into one master macro variable, and use it in query. The complication comes with this proposal is to determine how many individual macros are needed for current program. To simplify this complexity, below is the proposed macro that counts the number of macro variables dynamically and merges into a single macro variable.

```
/*************************************************************/
%macro max_macro(data=,var=,macro_var=);
*Determine the number of individual submacros needed;
data &data.2;
  set &data end=eof;
  length_var=length(compress(&var));
    if _n_=1 then sum_length=length_var;
    else sum_length+length_var;
  group=floor(sum_length/50000)+1;
    if eof then call symput('num_submacro',put(group,best.));
run;
%put Number of sub macros needed= %cmpres(&num_submacro);
*Macrotize the submacros;
%do i= 1 %to &num_submacro;
```

```
%global &macro_var submacro&i;
proc sql noprint;
  select cat("'", &var, "'") into:submacro&i
  separated by ","
  from &data.2
  where group =&i;
quit;
%end;
*Append the submacros into single master macro;
data submacro;
  do i = 1 to &num_submacro.;
  submacro = cats('&submacro',i);
  output;
  end;
run;
proc sql noprint;
  select submacro into:&macro_var separated by ' ' from submacro;
quit;
%mend max_macro;
%max_macro(data=search_list,var=variable,macro_var=master_macro);
/*************************************************************/
```

This master macro can be used in the program to query the large dataset.

```
data found;
  set layout;
  if variable in (&master_macro.);
run;
```

CONCLUSION

This paper demonstrates approaches to overcome the macro variable limitation error. There are other techniques that programmers use to search the datasets and import the data labels, but the reasoning behind this paper is to develop new methods of coding which could assist in improving SAS skills.

REFERENCES

Steven Yan, Mu Hu. “How to Import a Text File with a Large Number of Variables But The Layout Stored in a Separate File?”

CONTACT INFORMATION

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

Kalyani Telu
kalyani.telu.ctr@usuhs.edu