Paper 84-26

SAS® Programming Solutions for Summarizing Character Variables

Yefim Gershteyn, Takeda Pharmaceuticals America, Inc.

ABSTRACT

Although there is no formal operation for summarizing character variables, creating a character variable with the value that combines values of other character variables is a very common task. The concatenation operator is frequently used for that purpose. However, restrictions on the length of a character variable, especially when using SAS Version 6.12 and lower, need to be considered. This might result in a necessity of creating more than one summary variable. A macro is presented, which automatically creates a minimum number of summary variables to summarize within observations and defines the values of these variables in a way that the original words are not truncated or interrupted. Summarizing the values of a character variable across observations is more complicated. The paper discusses an approach to that using PROC SQL and describes a respective macro. The presentation concerns SAS/BASE and is intended for intermediate and advanced SAS users.

INTRODUCTION

SAS provides many ways to calculate a sum of numeric variables within an observation or across observations. The "+" operator or SUM function, RETAIN statement, PROC MEANS, PROC SUMMARY, PROC UNIVARIATE, PROC SQL provide the necessary capabilities. Summarizing character variables, however, is a different case.

The concatenation operator is a normal way of combining character values within the observation. With some additional manipulations (e.g. using RETAIN statement), it can also be applied to summarize across the observations. Meanwhile, this operation need some additional code, even if there are no real concerns or specifications about the length of the combined variable (e.g. when SAS Version 7 or higher is used, which allows for the length of character variable to be up to 32,767). The additional code would define delimiters in the combined variable and secure that the values are not truncated. The task becomes more complicated when there is a reason to expect that the combined length will exceed a limit (e.g. 200 characters for SAS Version 6.12). In this case, more than one summary variable is needed. Then, you would like to minimize the number of summary variables, i.e. allocate the values of the initial variables most efficiently. Also, you would like to preserve the initial values (words) uninterrupted, so that a word would belong to either a previous or next summary variable rather than would be broken between them.

Another way of summarizing across observations is using PROC SQL, specifically, INTO clause to create a macro variable containing all of the initial values. Although the macro variable can absorb practically unlimited number of characters, values from macro variable resolution to be passed back into the DATA step might still have the same length limitations. Because of that, you would still like to allocate the initial variables most efficiently, and also preserve the initial values (e.g. words) uninterrupted.

SUMMARIZING USING CONCATENATION OPERATOR

You can summarize variables using a DO loop to add a new variable to a cumulative string with each iteration. You can also insert a delimiter between the values of the initial variables. An example is given below.

```
** CREATE A DATA SET TO WORK WITH **;
❶ data names;
      length name1-name20 $21;
      infile cards delimiter=',' ;
      input name1 - name20 $;
cards;
Abracham Abbott,Benjamin Barnett,Carol Carlton
Daniel Durbin, Elizabeth Etheridge, Fransis Farehghait
Gerthrude George,Hazel Hawns,Ivan Ivanov,Jerome Jerome
Katherine Kohl, Lucy Lowell,Max MacIntosh,Nadine Novell
Orrin Orrington,Peter Peters,Quin Quincy, Roger Rogers
Samuel Stevens,Tara Target
Abracham1 Abbott1,Benjamin1 Barnett1,Carol1 Carlton1
Daniel1 Durbin1,Elizabeth1 Etheridge1, Fransis1 Farehghait1
Gerthrude1 George1,Hazel1 Hawns1,Ivan1 Ivanov1
Jerome1 Jerome1,Katherine1 Kohl1,Lucy1 Lowell1
Max1 MacIntosh1,Nadine1 Novell1,Orrin1 Orrington1
Peter1 Peters1,Quin1 Quincy1,Roger1 Rogers1
Samuel1 Stevens1,Tara1 Target1
;
run;
**CREATE SUMMARY VARIABLES **;
data names;
      set names;
**CALCULATE HOW MANY VARIABLES AT MAXIMUM
CAN BE CREATED GIVEN THE LENGTH OF INITIAL
VARIABLES **;
❷ %let maxvar=%sysfunc(ceil((21*20 + 1*19)/100));
❸     length listnu1- listnu&maxvar $100;
      keep listnu1 - listnu&maxvar;
      array name(20) $;
      array listnu(*) $ listnu1 - listnu&maxvar;
      array stopnu(*)  stopnu1 - stopnu&maxvar;
**ORGANIZE LOOPS FILLING EACH OF THE
VARIABLES LISTNU(I), IF APPLICABLE. SOME OF THE
VARIABLES LISTNU(I) CAN BE BLANK DEPENDING ON
THE DATA, AS THE REAL LENGTH OF SOME
VARIABLES CAN BE LESS THAN MAXIMUM
DECLARED LENGTH.
MAKE A CUMULATIVE SUMMARIZING STRING FOR
EACH OF THE SUMMARY VARIABLES TO FIRST
OBTAIN THE COUNT OF INITIAL WORDS TO BE
INCLUDED IN EACH OF THE SUMMARY VARIABLES **;
```

```
❹  do i=1 to &maxvar;
*START WITH FIRST INITIAL VARIABLE **;
        listnu1 = name1;
        if i=1 then oops=0;
        else listnu(i)=name(oops+1);
          do j=(oops+2) to 20;
❺             listnu(i) =
              left(trim(listnu(i)))||','||left(trim(name(j)));
**CHECK IF SAS TRUNCATES THE LATEST VARIABLE
APPENDED.  REVERSE AND MAKE SURE THE LAST
WORD HAS BEEN CONCATENATED IN FULL: THE
LENGTH OF CUMULATIVE STRING IS STILL LESS OR
EQUAL TO SPECIFIED OR MAXIMUM LENGTH.  IF THE
LATEST WORD IS NOT APPENDED IN FULL, THAN GO
BACK AND RESTRICT THE CUMULATIVE VARIABLE
TO ONE LESS WORD **;
❻             if index(reverse(left(trim(listnu(i)))),
                reverse(left(trim(name(j))))) ne 1 then do;
                    oops=j-1;
                    stopnu(i)=oops;
                    listnu(i)=name(j);
❼                   leave;
              end;
              else stopnu(i) = 20;
          end;
     end;
❽**VARIABLE IND DEFINES HOW MANY INITIAL
WORDS SHOULD BE INCLUDED IN EACH SUMMARY
VARIABLE**;
     ind=1;
     do i=1 to &maxvar ;
          if stopnu(i) ne 20 then ind +1;
     end;
❾**RESET VARIABLES LISTNU(I), AS THEY
GENERALLY CONTAIN WRONG VALUES AT THAT
POINT  AND REDEFINE THE CUMULATIVE STRING FOR
EACH SUMMARY VARIABLE USING VARIABLE IND**;
     do i=1 to &maxvar;
          listnu(i) = ' ';
     end;
     n=0;
❿    do i=1 to &maxvar while (n < ind);
          if i=1 then do;
               start=2; listnu1=name1;
          end;
          else do ;
               if (n < ind) then do;
                    start=stopnu(i-1) +2;
                    listnu(i)=name(start -1);
               end;
          end;
          do j=start to stopnu(i);
               listnu(i)=
               left(trim(listnu(i)))||','||left(trim(name(j)));
          end;
          n+1;
     end;  run;
**KEEP ONLY SUMMARY VARIABLES WITH AT LEAST
ONE NON-BLANK VALUE* *;
❶❶ data _null_;
     set names;
     array filled(&maxvar);
     array listnu(*) $ listnu1 - listnu&maxvar;
     do i=1 to &maxvar;
          filled(i) = 0;
          if listnu(i) ne ' ' then filled(i) + 1;
          varname = "var"||left(put(i, 3.));
          call symput(varname, left(put(filled(i), 3.)));
     end; run;

%macro dropit;
     data names;
          set names;
          array listnu(*) $ listnu1 - listnu&maxvar;
          drop
          %do i=1 %to &maxvar;
          %if &&var&i = 0 %then listnu&i;
          %end; ;
     run;
%mend dropit;
%dropit
```

The final data set NAMES has the following summary variables:

OBS	LISTNU1	LISTNU2	LISTNU3	LISTNU4
1	Abracham Abbott,Benjamin Barnett,Carol Carlton,Daniel Durbin,Elizabeth Etheridge,Fransis Farehghait	Gerthrude George,Hazel Hawns,Ivan Ivanov,Jerome Jerome,Katherine Kohl,Lucy Lowell,Max MacIntosh	Nadine Novell,Orrin Orrington,Peter Peters,Quin Quincy,Roger Rogers,Samuel Stevens,Tara Target	
2	Abracham1Abbott1,Benjamin1 Barnett1,Carol1 Carlton1,Daniel1 Durbin1,Elizabeth1 Etheridge1	Fransis1 Farehghait1,Gerthrude1 George1,Hazel1 Hawns1,Ivan1 Ivanov1,Jerome1 Jerome1,Katherine1 Kohl1	Lucy1 Lowell1,Max1 MacIntosh1,Nadine1 Novell1,Orrin1 Orrington1,Peter1 Peters1,Quin1 Quincy1	Roger1 Rogers1,Samuel1 Stevens1,Tara1 Target1

Notes:

❶ Create a data set to work with.

❷ Calculate the maximum number of summary variables assuming that all initial variables could have the maximum declared length (in this example, 21), and the summary variables will have a specified length limit (100 in the example). A macro variable *&maxvar* holds this number. On the other hand, for the sake of simplicity, we used the explicit values for the lengths of the initial and summary variables. To generalize the code, we could have used macro variables holding these values, e.g. through the use of VARLEN function or through the use of a data set created by a PROC CONTENTS procedure.

❸ Declare length of summary variables *listnu1-listnu&maxvar* ($100) and arrays.

❹ Organize loops filling each of the variables *listnu(i)*, if applicable. Some of the summary variables *listnu(i)* can be blank depending on the data, as the length of the values of some initial variables *name(i)* can be less than the maximum declared length of 21. First, you make a cumulative summarizing string ❺ for each of the summary variables to obtain the count of the initial words to be included in it. In order to know if the latest initial value that has been appended is truncated, a reverse cumulative string ❻ is formed and compared in each step with the reverse value of the latest initial variable *name(j)* that was concatenated. If the last word is concatenated in full, (meaning that the length of cumulative string is still less or equal to the specified or maximum possible length), then the program appends the next initial variable to the summary string. If the word is not appended in full, then the program stops filling in the particular summary variable *listnu(i)* ❼, marks the initial word on which the summary variable needs to be limited (variable *oops*), and starts filling the next summary variable *listnu(i +1).*

Thus, you obtain the maximum numbers of the initial words, which can fill each particular summary variable without being truncated - variable *ind* ❽.

❾ Variable *ind* defines how many words should be included in each variable.

❿ Reset variables *listnu(i),* as initially they could contain wrong values with truncated words. Then redefine the cumulative string for each summary variable using the variable *ind* – the number of the initial words in each summary string.

❶❶ Finally, in the case when we have a summary variable that is blank for all observations in the data set, we might want to drop this summary variable to keep only variables which have at least one non-blank value.

SUMMARIZING ACROSS OBSERVATIONS USING PROC SQL

A feature in PROC SQL, host variable INTO provides a convenient way to summarize character variables by creating a macro variable accommodating all initial values with a specified delimiter. One application of this host variable for determining the order of output was discussed in [1]. A similar approach with some extensions could be used for summarizing. An example is given below.

```
❶ data animals;
      length animal $40;
      input animal @@;
cards;
Cat000000000000000000   Dog00000000000000000000000000
Lion000000000000000000000000000000
Tiger0000000000000000    Mice0000000000000000000000000
Rabbit0000000000000000000000000000
Dragon000000000000000  Cow00000000000000000000000000
Horse00000000000000000000000000000
;
run;

❷ proc sql;
      select /*distinct*/  animal
      into :group separated by ","
      from animals;
quit;
```

```
❸ %macro combine;
      %let alllen = %length(&group);
      %let i=1;
      %if %length(&group) > 100 %then %do;
           %do %until (%length(&group) = 0);
             %let upto = 100;
             %let start=1;
             %let var&i = %substr(%quote(&group), &start,
             &upto);
             %let rev&i =
           %index(%quote(%sysfunc(reverse(%quote(&&var&i)
           ))),%str(,));
             %if %length(&group) > 100 %then %do;
                %let upto = %eval(100 -&&rev&i);
                %let word&i=  %substr(%quote(&group),
                                     &start, &upto);
             %end;
             %else %do;
                %let word&i = %quote(&group);
             %end;
             %let i= %eval(&i + 1);
             %let start = %eval(&start + &upto + 1);
             %let group = %substr(%quote(&group), &start);
           %end;
      %end;
      %else %do;
           %let word1 = %quote(&group);
      %end;

      data test;
           word1 = "&word1";
           word2= "&word2";
           word3="&word3";
      run;
      proc print;
%mend combine;
%combine
```

Data set *test* contains three summary variables *word1-word3* accommodating all the initial values of the variable *animal*.

Notes:

❶ Create a data set to work with.

❷ Use PROC SQL and INTO clause to pass the values of the variable *animal* to a macro variable named *group* and separate the values by a ','. Note, that the macro variable value is constructed alphabetically when you use the DISTINCT keyword. If you do not, then the order is the same as the order of the observations in the data set.

❸ Macro *%combine* breaks the resolved values of the macro variable *group* to accommodate the length specifications (the summary variable length of $100 in the example). Prototype code for separating the value of a macro variable into 40-character units and storing each unit in a separate variable was taken from [2].

OBS	WORD1	WORD2	WORD3
1	Cat000000000000000000,Dog00000000000000000000000000,Lion000000000000000000000000000000	Tiger0000000000000000,Mice0000000000000000000000000,Rabbit0000000000000000000000000000	Dragon000000000000000,Cow00000000000000000000000000,Horse00000000000000000000000000000

CONCLUSIONS

Summarizing character variables within the observation can be accomplished by using the concatenation operator. INTO clause in PROC SQL is a way to summarize character variable values across observations. To comply with restrictions on the length of a character variable, creation of more than one summary variable sometimes is needed. Macros, which create a minimum number of summary variables and define the values of these variables in a way that the original words are not truncated or interrupted can be used for that purpose.

REFERENCES

[1] Ray Pass (2000) "What We Really Need is a %BY Statement - V2," Proceedings of the Twenty-Fifth Annual SAS® Users Group International Conference, 25, Paper 83-25 , pp. 494-495.

[2] SAS Institute, Inc., SAS® Macro Language: Reference, First Edition, Cary, NC: SAS Institute Inc., 1997, 304 pp.

CONTACT INFORMATION

Yefim Gershteyn, Ph.D.
Takeda Pharmaceuticals America Inc.
475 Half Day Rd., Suite 500
Lincolnshire, IL 60069
Phone: (847) 383-3000
E-mail: **fgershteyn@takedapharm.com**

TRADEMARKS