Using %SYMEXIST for eliminating repetitive words in a SAS® macro string - no data steps, no procedures.

Alexander M Feigin, Alex Feigin, Inc, Elkins Park, PA 19027

Abstract
The presented utility macro NONREP, which does not include any data or procedure steps, can be easily used for cleaning repeated words from long macro strings (without changing the original order of words), for identifying duplicated, triplicate, or quadruplicated words and creating the relevant lists of these words without redundant repetitions inside each of these lists. Starting from the first word, it checks one by one all words in a macro string, figuring out whether the macro variable with the name identical to the checked word already exists (SAS macro function %SYMEXIST does this particular job). If the relevant macro variable does not exist yet, the program creates it and also: i) creates another macro variable with a value equal to the checked word; ii) concatenates this word with the previously checked chain of distinct words, thus, making finally a new macro variable containing all distinct words from the original string placed in the original order without duplications.

Introduction
It is not very common but still a practical task for a SAS programmer to eliminate duplications in a macro variable string which can occur, for example, as a list of variables to be checked, or a list of concomitant medications used to treat a certain medical condition, etc. The standard approach to solve this problem includes conversion of this macro variable into a variable (column) in a new data set, then sorting the data set by this variable utilizing the NODUPKEY statement, creating another variable which contains all values of the sorted column as a string of words and conversion of this string into a final "sorted NODUPKEY" macro variable. However, it would be a real programming advantage to perform this operation with only macro code and without creating additional data steps or using any procedures. Moreover, according to Ronald Berry (Berry, 2008)1 'good' utility macro… avoids using data steps or procedures'. Our paper will present a short piece of code based on the use of the macro function %SYMEXIST, which eliminates duplications in a string of words (value of macro variable) without changing the original order of words. This code also can list the words which are repeated a certain number of times (for example, two or three). The code does not create any data sets or use any sorting procedures. Instead, starting from the first word, it checks one by one all the words in a macro string figuring out whether the macro variable with the name identical to the checked word already exists (SAS macro %SYMEXIST, which is available in version 9.1 and later, does this particular job). If the relevant macro variable does not exist yet, the program creates it and also: i) creates another macro variable with a value equal to the checked word; ii) concatenates this word with the previously checked chain of distinct words, thus, making finally a new macro variable containing all distinct words from the original string placed in the original order without duplications.

Task:

%let checked=DOMAIN STUDYID USUBJID STUDYID CECAT CESEV SUBJID CESEV CESPID EPOCH CESEV SPDYRLEP STDYRLEP CEDTC CEENDTC CEDURDD CEDURDDU CEENDTI CEGRPID CEDURDDU CEOCCUR CESEQ CEDECOD CETERM TAETORD CEDURDDU VISIT VISITNUM CESTDY CEENDY CEDURDDU CESTDTI;

The macro variable CHECKED contains 33 words, one of them: STUDYID is duplicated, one: CESEV is triplicated, and one: CEDURDDU is quadruplicated.

The task is:
1) To eliminate all repetitions, in other words, to obtain a macro variable NEWCHECKED with 27 words.
2) Create a macro variable NEWCHECKED2 containing the list of words at least doubled in CHECKED;
3) Create a macro variable NEWCHECKED 3 containing the list of words at least tripled in CHECKED;

Pure Macro Approach:

1. Creating a macro variable without any repetitions of words.

Macro %NONREP starts by counting the number of words in the macro variable CHECKED and placing this number into macro variable NUMVAR. Then a %DO loop is executed from 1 to &NUMVAR. Inside of this %DO loop, the statement "%if %symexist( %scan(&&mvar,&i)) = 0 ..." checks one by one, all of the words in the macro string figuring out whether the macro variable with the name identical to a checked word already exists (SAS macro %SYMEXIST, which is available in version 9.1 and later, does this particular job).

If the relevant macro variable does not exist yet, the program creates it and also: i) creates another macro variable with a value equal to the checked word (%let name&i=%scan(&&mvar,&i);); ii) concatenates this word with the previously checked chain of distinct words (%let long = &long &name&i;), thus, making finally a new macro variable NEWCHECKED containing all distinct words from original string placed in the original order without duplications.

%macro nonrep(mvar= , outvar= );
%global numvar;
%do z=1 %to 100;;
%if %scan(&&mvar,&z) ne %then %do;
 %let numvar=&z;
%end;
%end;
%put Number of variables = &numvar;
%global &outvar;
%do i=1 %to &numvar;
 %let j=%eval(&i-1);
%if %symexist(%scan(&&mvar,&i))=0 %then %do;
 %let %scan(&&mvar,&i)=1;
 %let name&i=%scan(&&mvar,&i);
%put Number &i : &&name&i;
%if &i=1 %then %let long = &&name1;
%else %let long = &long &&name&i;
%end;
%end;
%end;
2. Additional features: Creating lists of words repeated at least two or three times.

Macro NONREP2 is a further development of NONREP:

```
%macro nonrep2(mvar= , outvar= );
%global &outvar &outvar.2 &outvar.3;
%do z=1 %to 100;;
  %if %scan(&&&mvar,&z) ne %then %do;
    %let numvar=&z;;
  %end;
%end;
%put Number of variables = &numvar;
%local long long2 long3;
%do i=1 %to &numvar;
  %if %symexist(%scan(&&&mvar,&i))=0 %then %do;
    %let %scan(&&&mvar,&i)=1;
    %let name=%scan(&&&mvar,&i);
    %let long = &long &name;
  %end;
  %else %if %symexist(%scan(&&&mvar,&i)) = 1 %then %do;
    %let name=%scan(&&&mvar,&i);
    %if &&&name=1 %then %do;
      %let %scan(&&&mvar,&i)=2;
      %let name&i=%scan(&&&mvar,&i);
      %let long2 = &long2 &&name&i;
    %end;
    %else %if &&&name=2 %then %do;
      %let %scan(&&&mvar,&i)=3;
      %let name&i=%scan(&&&mvar,&i);
      %let long3 = &long3 &&name&i;
    %end;
  %end;
%end;
%let &outvar=&long;
%let &outvar.2=&long2;
%let &outvar.3=&long3;
%mend;
```

%nonrep2(mvar=checked, outvar=newchecked);
%put &newchecked;
%put doubled &newchecked2;;
%put tripled &newchecked3;;
In order to create the lists of words repeated at least two or three times, in addition to what NONREP does, NONREP2 performs several things:

i) When it "sees" the word a second time (%let name=%scan(&&mvar,&i);%if &&&name=1 ...
   ... it brings the value equal to 2 to the macro variable with the name equal to this word:
   (%let %scan(&&mvar,&i)=2;)
   and adds the word doubled in CHECKED (for Example, CESEV) to the value of macro variable LONG2, thus creating a chain like
   STUDYID CESEV (%let long2 = &long2 &name;);

ii) When it "sees" the word a third time (%if &&&name=2 ...
    ... it brings the value equal to 3 to the macro variable with the name equal to this word (%let%scan(&&mvar,&i)
    =3;)
    and adds the word tripled in CHECKED (for example, CEDURDDU) to the value of macro variable LONG3, thus creating a chain like
    CESEV CEDURDDU (%let long3 = &long3 &name;).

So, if the word CEDURDDU happens to appear in the third time in CHECKED, it will not be added to LONG2 in a second time, because the value of macro variable CEDURDDU is equal to 2, and the macro NONREP2 would add the word to LONG2 only in case the relevant macro variable has a value equal to 1. However, the word CEDURDDU will be added to LONG3 due to the same fact - the value of macro variable CEDURDDU is equal to 2.

We obtain the value of each macro variable with name equal to a relevant word in CHECKED using the triple ampersand call for NAME (&&&NAME). If we call for NAME with double or single ampersand we will get the value of macro variable NAME, which is the word CEDURDDU (during execution of a relevant %do loop, in this particular case).

3. Results

The values of the newly created macro strings are:

- NEWCHECKED (distinct words in CHECKED): DOMAIN STUDYID USUBJID CECAT CESEV SUBJID CESPID EPOCH SPDYRLEP STDRLEP CEDTC CEENDTC CESTDTC CEDURDD CEDURDDU CEENDTI CEGRPID CECCUR CESEQ CEDECOD CETERM TAETORD VISIT VISITNUM CESTDY CEENDY CESTDTI
- NEWCHECKED2 (words that are duplicated in CHECKED): STUDYID CESEV CEDURDDU
- NEWCHECKED3 (words that are triplicate in CHECKED): CESEV CEDURDDU

The SAS macro function %SYMEXIST was already suggested for a similar purpose: finding duplicates in a string (Staum, 2006)² However, the code provided in that publication cannot obtain a distinct list of replicated words found in the original string, instead these words will appear in the list as many times as they appear in the original string. Staum states that a good alternative to this macro approach in identifying duplicates is a PROC SORT with the DUPOUT option, however, she notes, that the output list in this case still "... retains redundant duplicates". Our code presented in this paper definitely resolves the problem of redundant duplicates without help from data step or any sorting procedures and provides the non-repetitive lists of words duplicated in original string.

² Staum P.W. "Efficient Processing of Long Lists of Variable Names" - NESUG, 2006
Non-Macro Approach:

Certainly all three tasks previously described can be performed utilizing a completely different approach which uses data steps and SAS procedures. This approach is based on the transposing of the macro string into the variable (NAME) of the SAS data set (VARCHECK), where each word becomes an observation of this variable.

```sas
data varcheck(keep=name);
    length long $ 3000  name $ 25;
    long="&checked";
    do i=1 to 200;
        name=scan(long,i);
        if name ne ' ' then output;
    end;
run;
```

Then, using Proc Freq, a data set FREQO, can be created which contains only 27 observations (one observation per word) and counts for each word.

```sas
proc freq data=varcheck;
    tables name /out=freqo;
run;
```

At the next step we do the rest of the job by using the RETAIN statement and a set of conditions for adding words for retained variables: LONG, LONG2 and LONG3, standing for a string of all distinct words, string of all at least doubled words, and string of all at least tripled words, respectively. Using CALL SYMPUT, we send the values of these ‘long’ variables into the relevant macro variables.

```sas
data _null_;
    length long long2 long3 $ 3000 ;
    retain long long2  long3 ;
    set freqo;
    if _n_=1 then
        do;
            long=name;
            long2=' ';
            long3=' ';
        end;
    else do;
        long=left(trim(long))||' '||compbl(name);
        call symput('newchecked',left(trim(long)));
        if count ge 2 then long2=left(trim(long2))||' '||compbl(name);
        call symput('newchecked2',left(trim(long2)));
        if count ge 3 then long3=left(trim(long3))||' '||compbl(name);
        call symput('newchecked3',left(trim(long3)));
    end;
run;
```

```sas
%put &newchecked;
```
%put doubled &newchecked2;;
%put tripled &newchecked3;;

This approach is absolutely fine; however, it forces us to use several data steps and procedures, while with the macro approach, we did not utilize either one.

Conclusion

The presented utility macro, NONREP, demonstrates that operations, like cleaning replicated words from long macro strings, identification of duplicated, triplicate, or quadruplicated words and creating the relevant lists of these words without redundant repetitions inside each of these lists can be performed with only macro code, without any data or procedure steps.

ACKNOWLEDGMENTS
SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Please contact the author at: feiginy@aol.com