PharmaSUG China 2025 - Paper CC-138

SAS Code preprocessing tool development: accurate identification and removal of non-named text

Shulian Zhou, Sanofi China

ABSTRACT

In everyday SAS programming, we may encounter situations where we want to check if a variable exists in the code or if a dataset is referenced. However, we might face scenarios where these variables or datasets do not actually exist in the valid named code but do appear in comments, strings, or other non-named text. To ensure that the results of the check are accurate, I have developed a macro to remove these non-named texts. This article outlines the entire process of the problem from discovery to resolution, and I hope it will be effective for you.

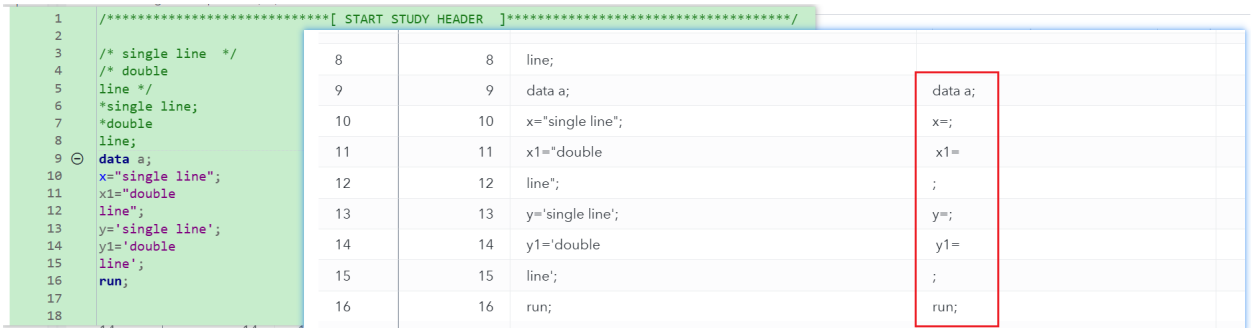
INTRODUCTION

I discovered this problem while developing a macro to check if all the variables in the metadata were present in the code. I faced multiple issues one after another while trying to resolve this, and after numerous modifications, I managed to solve most of the cases related to this problem. I will explain my thought process in the following.

MY THOUGHTS

FIRST ATTEMPT

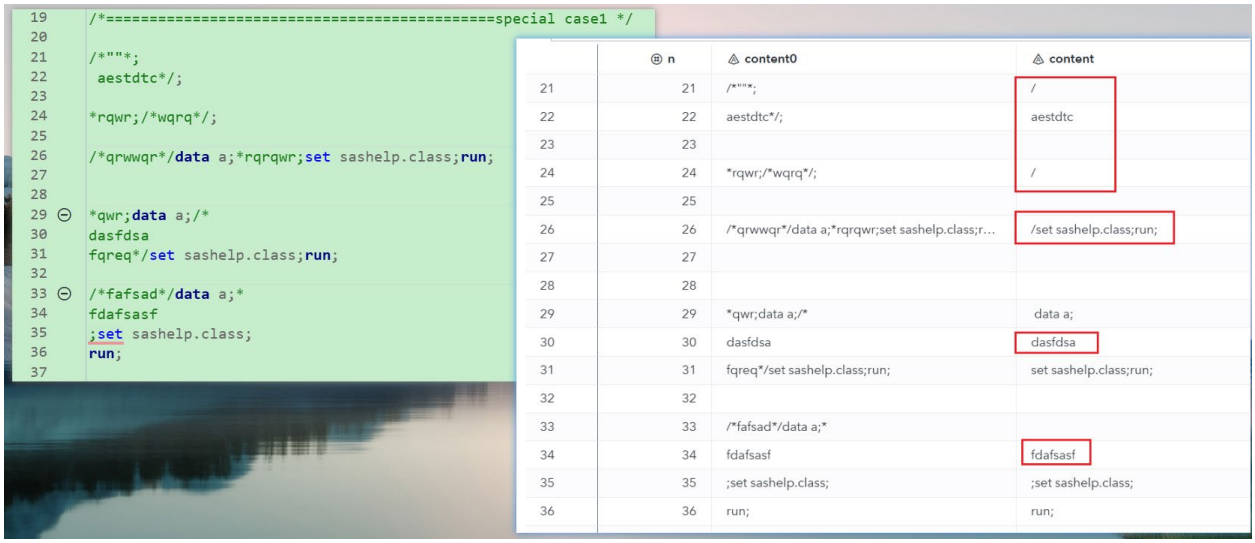
In my first attempt, I assumed that I would process single-line comments first, followed by multi-line comments, and then strings. The following figure illustrates this point clearly.



```
1  /*************************[ START STUDY HEADER  ]*********************************/
2
3  /* single line  */
4  /* double
5  line */
6  *single line;
7  *double
8  line;
9  data a;
10 x="single line";
11 x1="double
12 line";
13 y='single line';
14 y1='double
15 line';
16 run;
17
18
```

8	8	line;	
9	9	data a;	data a;
10	10	x="single line";	x=;
11	11	x1="double	x1=
12	12	line";	;
13	13	y='single line';	y=;
14	14	y1='double	y1=
15	15	line';	;
16	16	run;	run;

However, I later realized that this approach was flawed. These comments and strings are interconnected and can overlap with each other. For example, consider the following illustration. The comments were not deleted correctly.



SECOND ATTEMPT

In order to solve this problem, I scrutinized the relationship between / **/ *;‘’ and “”, which I will consider here as four groups, and which will also be referred to below by group.

For each row of records, I will identify the starting position of each group within the row and store these positions in an array called "starts." I will also check for the ending position of each group and store these in an array called "ends." Next, I will determine the smallest non-zero starting position among these groups.Finding the smallest position indicates we have identified non-named texts in a certain group, which need to be deleted.

After the deletion, re-check this line whether there is the beginning elements from any groups, if there is, repeat the previous operation; if you do not find the beginning elements from any groups, that is, there is no need to delete the non-named text, you can go to the next line, repeat the operation of the previous line.

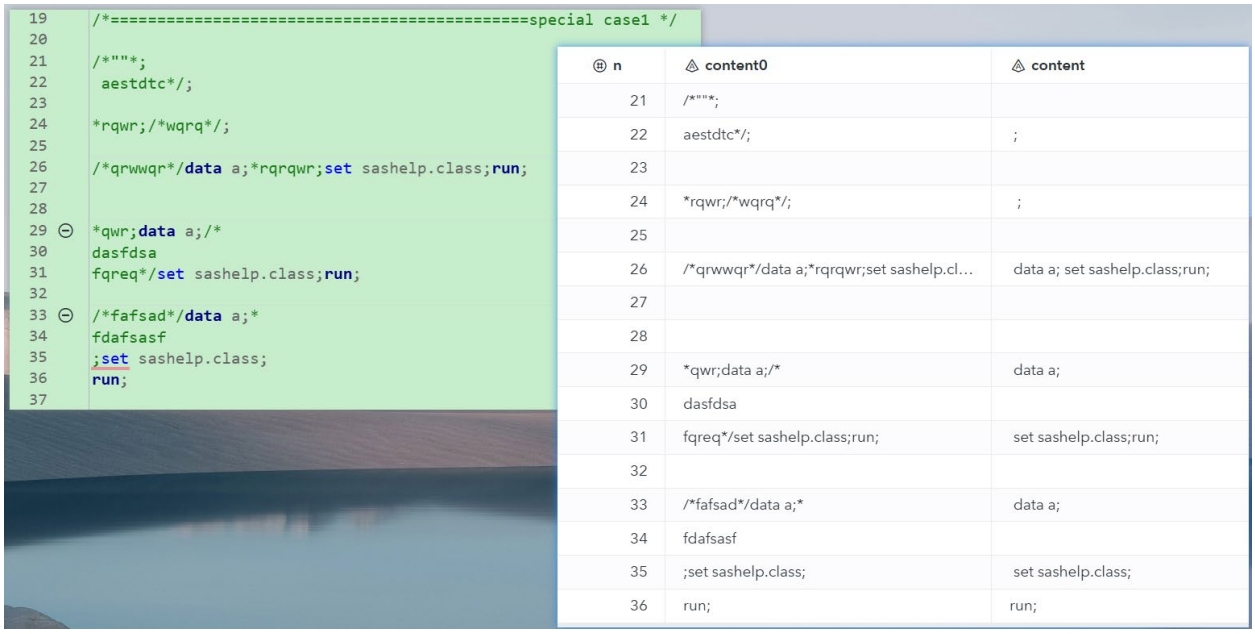
If you identify the starting element of a group but cannot find the ending element, it indicates that you have discovered non-named text that spans multiple lines. In this case, you should keep the text that appears before the starting element and then move to the next line.

The count variable must be initialized before moving on to the next line. Please note the following:

1. If you encounter a single line of unnamed text, set the count to 1.
2. If the unnamed text spans multiple lines, increment the count by 1 for each additional line before proceeding to the next line.

Now count>1, you only need to search for the ending element that corresponds to the previously identified beginning element within the same group. If you do not find the ending element, the current line should be deleted, and you should continue searching on the next line for the ending element. If you find the ending element, delete the text that appears before it, and then return to the previous single-line operation.

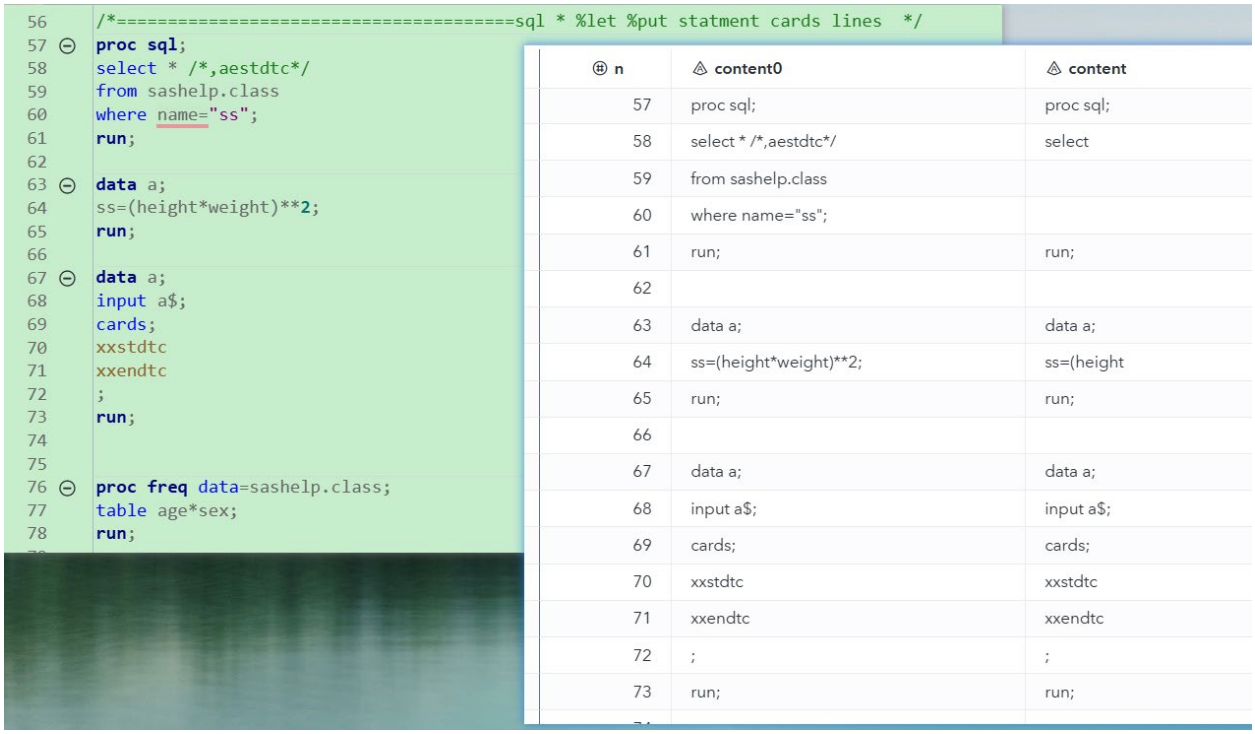
With the optimization implemented above, the previous issue has been resolved. As shown in the figure below.



```
/*==============================================special case1 */

/*"""*;
 aestdtc*/;

*rqwr;/*wqrq*/;

/*qrwwqr*/data a;*rqrqwr;set sashelp.class;run;


*qwr;data a;/*
dasfdsa
fqreq*/set sashelp.class;run;

/*fafsad*/data a;*
fdafsasf
;set sashelp.class;
run;

```

n	content0	content
21	/*"""*;	
22	aestdtc*/;	;
23		
24	*rqwr;/*wqrq*/;	;
25		
26	/*qrwwqr*/data a;*rqrqwr;set sashelp.cl…	data a; set sashelp.class;run;
27		
28		
29	*qwr;data a;/*	data a;
30	dasfdsa	
31	fqreq*/set sashelp.class;run;	set sashelp.class;run;
32		
33	/*fafsad*/data a;*	data a;
34	fdafsasf	
35	;set sashelp.class;	set sashelp.class;
36	run;	run;

I've encountered another issue recently. I'm struggling to properly handle asterisks in PROC SQL statements. The asterisk serves two purposes: it marks the beginning of a comment and also signifies multiplication. Additionally, non-named text, such as inputs following SAS cards statements, and the %let and %put statements are not being processed correctly either. As shown in the figure below.



```
/*=======================================sql * %let %put statment cards lines  */
proc sql;
select * /*,aestdtc*/
from sashelp.class
where name="ss";
run;

data a;
ss=(height*weight)**2;
run;

data a;
input a$;
cards;
xxstdtc
xxendtc
;
run;


proc freq data=sashelp.class;
table age*sex;
run;
```

n	content0	content
57	proc sql;	proc sql;
58	select * /*,aestdtc*/	select
59	from sashelp.class	
60	where name="ss";	
61	run;	run;
62		
63	data a;	data a;
64	ss=(height*weight)**2;	ss=(height
65	run;	run;
66		
67	data a;	data a;
68	input a$;	input a$;
69	cards;	cards;
70	xxstdtc	xxstdtc
71	xxendtc	xxendtc
72	;	;
73	run;	run;

```
74
75
76  proc freq data=sashelp.class;
77  table age*sex;
78  run;
```

76	proc freq data=sashelp.class;	proc freq data=sashelp.class;
77	table age*sex;	table age
78	run;	run;

```
107  %put fasfaf***;
108
109  %put fasfaf***
110  wqrqr;
111
112  %let a=aestdtc**wqr***;
113  %put
114  fsaf
115  ;
116
117  %let
118  a
119  =
```

107	%put fasfaf***;	%put fasfaf
108		
109	%put fasfaf***	%put fasfaf
110	wqrqr;	
111		
112	%let a=aestdtc**wqr***;	%let a=aestdtc
113	%put	%put
114	fsaf	fsaf
115	;	;

THIRD ATTEMPT

I have reconsidered the use of the asterisk. I believe that if I can confirm that the first non-blank character before the asterisk is a semicolon, then the asterisk indicates the beginning of a comment. If the first non-blank character is not a semicolon, then the asterisk does not serve the purpose of marking a comment. Additionally, please ensure that the non-blank characters before and after the cards are also semicolons.

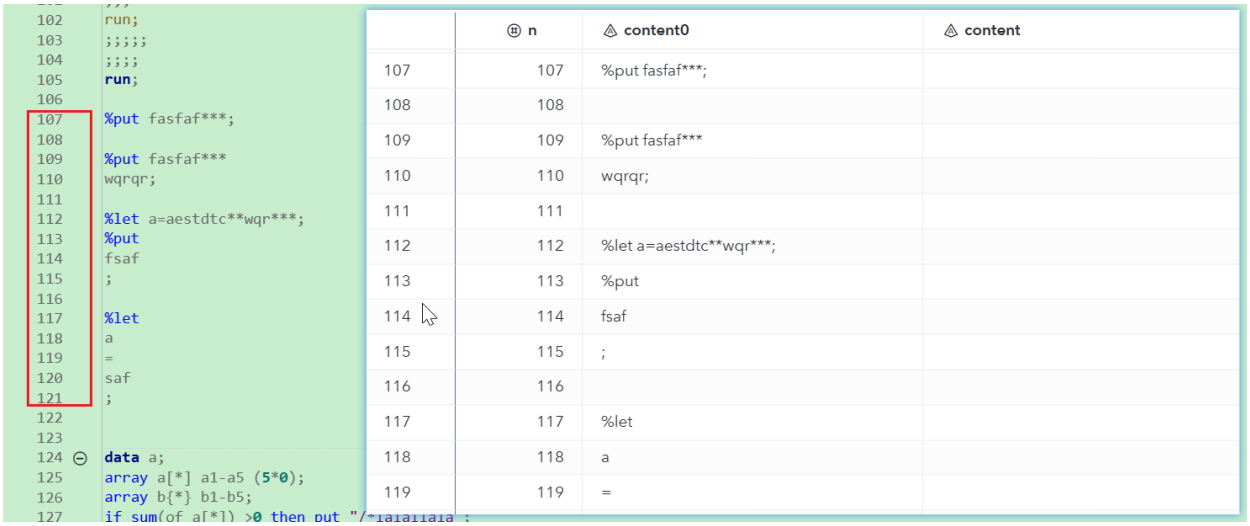
With the optimization implemented above, the previous issue has been resolved. As shown in the figure below.

```
56  /*=======================================sql * %let %put statment cards lines  */
57  proc sql;
58  select * /*,aestdtc*/
59  from sashelp.class
60  where name="ss";
61  run;
62
63  data a;
64  ss=(height*weight)**2;
65  run;
66
67  data a;
68  input a$;
69  cards;
70  xxstdtc
71  xxendtc
72  ;
73  run;
74
75
76  proc freq data=sashelp.class;
77  table age*sex;
78  run;
79
80  data a;
81  input a$;
82  cards;
83  fasf
84  ***
85  /*fa
86  fafa*/
87  ;
88  run;
89
```

n	content0	content
57	proc sql;	proc sql;
58	select * /*,aestdtc*/	select *
59	from sashelp.class	from sashelp.class
60	where name="ss";	where name= ;
61	run;	run;
62		
63	data a;	data a;
64	ss=(height*weight)**2;	ss=(height*weight)**2;
65	run;	run;
66		
67	data a;	data a;
68	input a$;	input a$;
69	cards;	cards;
70	xxstdtc	
71	xxendtc	
72	;	;
73	run;	run;

```
74
75
76  proc freq data=sashelp.class;
77  table age*sex;
78  run;
```

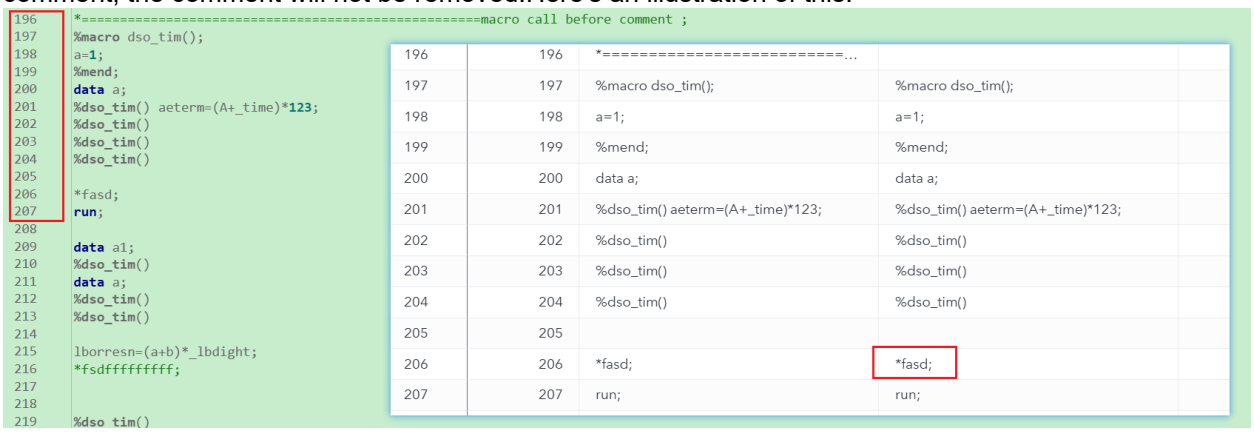
76	proc freq data=sashelp.class;	proc freq data=sashelp.class;
77	table age*sex;	table age*sex;
78	run;	run;



```
run;
;;;;;
;;;;
run;

%put fasfaf***;

%put fasfaf***
wqrqr;

%let a=aestdtc**wqr***;
%put
fsaf
;

%let
a
=
saf
;


data a;
array a[*] a1-a5 (5*0);
array b{*} b1-b5;
if sum(of a[*]) >0 then put "/~aaaaaaa ;
```

	n	content0	content
107	107	%put fasfaf***;	
108	108		
109	109	%put fasfaf***	
110	110	wqrqr;	
111	111		
112	112	%let a=aestdtc**wqr***;	
113	113	%put	
114	114	fsaf	
115	115	;	
116	116		
117	117	%let	
118	118	a	
119	119	=	

In most cases, the problem can be solved. However, it was later discovered that macros can be called in code without requiring a semicolon. This means that if a macro is called without a semicolon and is followed by a comment, the comment will not be removed.Here's an illustration of this.

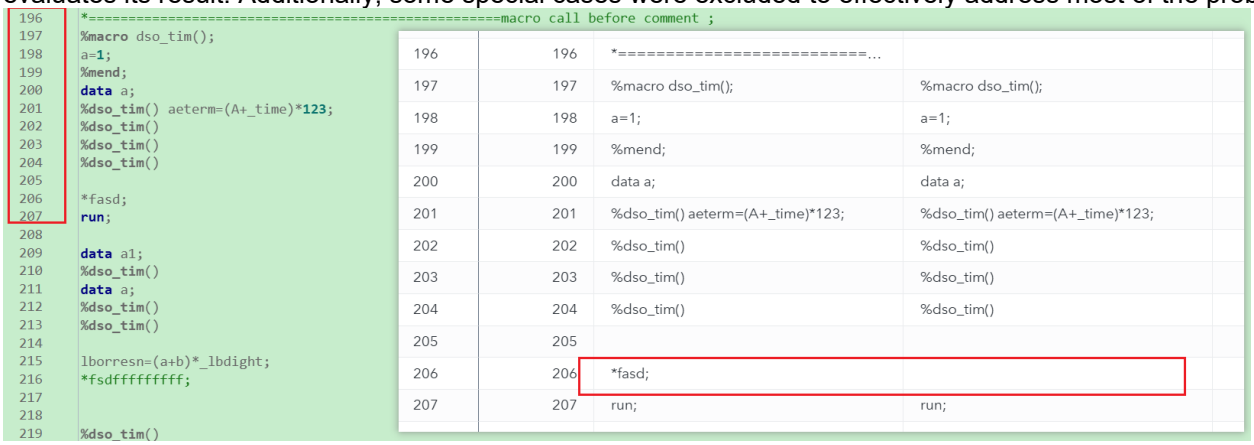


```
*==========macro call before comment ;
%macro dso_tim();
a=1;
%mend;
data a;
%dso_tim() aeterm=(A+_time)*123;
%dso_tim()
%dso_tim()
%dso_tim()

*fasd;
run;

data a1;
%dso_tim()
data a;
%dso_tim()
%dso_tim()

lborresn=(a+b)*_lbdight;
*fsdfffffffff;


%dso_tim()
```

196	196	*==========...	
197	197	%macro dso_tim();	%macro dso_tim();
198	198	a=1;	a=1;
199	199	%mend;	%mend;
200	200	data a;	data a;
201	201	%dso_tim() aeterm=(A+_time)*123;	%dso_tim() aeterm=(A+_time)*123;
202	202	%dso_tim()	%dso_tim()
203	203	%dso_tim()	%dso_tim()
204	204	%dso_tim()	%dso_tim()
205	205		
206	206	*fasd;	*fasd;
207	207	run;	run;

FOURTH ATTEMPT

To resolve this issue, I developed a custom function that eliminates the macro call before the asterisk and evaluates its result. Additionally, some special cases were excluded to effectively address most of the problems.



```
*==========macro call before comment ;
%macro dso_tim();
a=1;
%mend;
data a;
%dso_tim() aeterm=(A+_time)*123;
%dso_tim()
%dso_tim()
%dso_tim()

*fasd;
run;

data a1;
%dso_tim()
data a;
%dso_tim()
%dso_tim()

lborresn=(a+b)*_lbdight;
*fsdfffffffff;


%dso_tim()
```

196	196	*==========...	
197	197	%macro dso_tim();	%macro dso_tim();
198	198	a=1;	a=1;
199	199	%mend;	%mend;
200	200	data a;	data a;
201	201	%dso_tim() aeterm=(A+_time)*123;	%dso_tim() aeterm=(A+_time)*123;
202	202	%dso_tim()	%dso_tim()
203	203	%dso_tim()	%dso_tim()
204	204	%dso_tim()	%dso_tim()
205	205		
206	206	*fasd;	
207	207	run;	run;

REMIANING PROBLEMS

If a macro call's code is written across multiple lines and a single quote is present, it will not accurately remove the quote, affecting subsequent deletions.

```
%test(%nrstr(age*/%'
sex));

%test(%nrstr(*));
```

CONCLUSION

This article primarily discusses the author's step-by-step approach to deleting non-named text in order to address various challenges encountered along the way. Although the current code does not fully resolve all situations, it can handle most cases effectively. The author's inspiration for this approach stemmed from their implementation of a function that checks for the presence of all SDTM variables in batch code files. It is hoped that this information will be helpful to readers.

ACKNOWLEDGMENTS

Thanks to Xiuchuang Chen Sanofi R&D and Longfei Li Sanofi R&D for their support and help to my idea.

CONTACT INFORMATION

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

Shulian Zhou
Sanofi R&D
Shulian.zhou@sanofi.com

APPENDIX A: MACRO AND CUSTOM FUNCTION

```
/* delete comment , string, %let statement %put statment  */
%macro delete_comment(inds=,outds=);
    data &inds.;
    set &inds.;
    /* delete the ( or )  in  %nrstr or %str  */
    /* just for the %nrstr|%str|%quote|%nrquote|%bquote|%nrquote|%superq in the
single line */
    content=prxchange('s/((%nrstr|%str|%quote|%nrquote)\([^\(\)]*?)(%\()([^\(\)]*
?\))/$1 $4/io',-1,content);
    content=prxchange('s/((%nrstr|%str|%quote|%nrquote)\([^\(\)]*?)(%\))([^\(\)]*
?\))/$1 $4/io',-1,content);
    content=prxchange('s/((%bquote|%nrquote|%superq)\([^\(\)]*?)(\()([^\(\)]*?\))
/$1 $4/io',-1,content);
    content=prxchange('s/((%bquote|%nrquote|%superq)\([^\(\)]*?)(\))([^\(\)]*?\))
/$1 $4/io',-1,content);



    /* delete the * or / or ' or " or ;  in  %nrstr or %str  */
    /* just for the %nrstr|%str|%quote|%nrquote|%bquote|%nrquote|%superq in the
single line */
    content=prxchange('s/((%nrstr|%str|%quote|%nrquote|%bquote|%nrquote|%superq)\
([^\(\)]*?)(\*)([^\(\)]*?\))/$1 $4/io',-1,content);
    content=prxchange('s/((%nrstr|%str|%quote|%nrquote|%bquote|%nrquote|%superq)\
([^\(\)]*?)(\/)([^\(\)]*?\))/$1 $4/io',-1,content);
    content=prxchange('s/((%nrstr|%str|%quote|%nrquote|%bquote|%nrquote|%superq)\
([^\(\)]*?)(;)([^\(\)]*?\))/$1 $4/io',-1,content);

    content=prxchange('s/((%nrstr|%str|%quote|%nrquote)\([^\(\)]*?)(%")([^\(\)]*?
\))/$1 $4/io',-1,content);
    content=prxchange('s/((%nrstr|%str|%quote|%nrquote)\([^\(\)]*?)(%'')([^\(\)]*
?\))/$1 $4/io',-1,content);


    content=prxchange('s/((%bquote|%nrquote|%superq)\([^\(\)]*?)(")([^\(\)]*?\))/
$1 $4/io',-1,content);
    content=prxchange('s/((%bquote|%nrquote|%superq)\([^\(\)]*?)('')([^\(\)]*?\))
/$1 $4/io',-1,content);


    /* delete the % in %nrstr|%nrquote|%nrbquote|%superq */
    /* just for the %nrstr|%nrquote|%nrbquote|%superq in the single line */
    content=prxchange('s/(%nrstr\([^\(\)]*?)(%%)([^\(\)]*?\))/$1 $3/io',-
1,content);
    content=prxchange('s/((%nrquote|%nrquote|%superq)\([^\(\)]*?)(%)([^\(\)]*?\))
/$1 $4/io',-1,content);
    run;

    /* delete the comment and quotation mark and relational part */
    /* delete the comment and quotation mark and relational part */
    data &outds.;
    set &inds.;
    array starts{*} star1num slash1num  sing1num doub1num put1num let1num;
    array ends{*} star2num slash2num sing2num doub2num put2num let2num;
    retain star1num slash1num  sing1num doub1num put1num let1num 0;
```

```
    retain star2num slash2num sing2num doub2num put2num let2num 0;
    retain count cardnum 0;
    length contentlag $5.;
    retain contentlag "";
    length constat constat0 constat1 constat2 $200.;
    retain constat "";

    /* variable tmpcontent and tmpcontent1 are generated just for modifing the
value of cardnum */
    tmpcontent=prxchange('s/\/\*(.*?)\*\// /io',-1,content);
    tmpcontent1=prxchange('s/\*(.*?);/*;/io',-1,tmpcontent);
    drop tmpcontent tmpcontent1;
    if cardnum=1 and prxmatch('/\**?;/io',compress(tmpcontent1))=1 then
cardnum=0;
    else if cardnum=2 and find(compress(content),";;;;")=1 then cardnum=0;
    else if cardnum in (0.5,1.5) then /* let program execute and delete the
comment*/;
    else if cardnum in (1,2) then content="";

    loop:
    if sum(of starts[*])=0 then do;
          if find(content,"/*") and slash1num=0 then do;
                /* confirm the /* not in the %macro */
                slash1num=find(content,"/*");
          end;


          if find(content,"'") and sing1num=0 then do;
                sing1num=find(content,"'");
          end;

          if find(content,'"') and doub1num=0 then do;
                doub1num=find(content,'"');
          end;

          if prxmatch('/%put\s+|%put$|%put;/io',content) and put1num=0 then do;
                      put1num=prxmatch('/%put\s+|%put$|%put;/io',content);
          end;


          if prxmatch('/%let\s+|%let$/io',content) and let1num=0 then do;
                      let1num=prxmatch('/%let\s+|%let$/io',content);
          end;


          /* search the start of comment or ""''  */
          if find(content,"*") and star1num=0 then do;
                pos=0;
                poscom=0;
                call missing(constat1,constat0,bracket);
                /*confirm the * in the statement and not in the start of
statement */
                /* 1.* in the start of obsevation*/
                if find(strip(content),"*")=1 then do;
                      /* the end of last obs is semicon(;) */
                      if contentlag in (";","") then do;
                            star1num=find(content,"*");
```

```
                        end;
                        else do;
                              /* check whether there is a macro call before * sign
*/
                              /* delete the  macro call from the beginning of the
statement*/
                              constat0=deleteMacro(constat,bracket);
                              /* if parentheses match perfectly and the statement
before the * sign is empty,then * sgin is the beginning of the comment */
                              if bracket=0 and missing(strip(constat0)) then
star1num=find(content,"*");
                        end;
                  end;

                  else do ss=1 to count(content,"*");
                        /* get the  postion of the ss *. */
                        pos=find(content,"*",sum(pos,1));
                        poscom=find(compress(content),"*",sum(poscom,1));

                        /* 2.* in the middle of the observation */
                        if find(compress(content),";*",poscom-1)=(poscom-1) then
do;
                              star1num=pos;
                              leave;
                        end;
                        else if find(compress(content),"%*",poscom-1)=(poscom-1)
then do;
                              star1num=pos-1;
                              put n= content= pos=;
                              leave;
                        end;
                        else do;
                              /* just when find * sgin and the positon of *sign is
the minmum value */
                              if pos>0 and

    ((pos<min(slash1num,sing1num,doub1num,put1num,let1num) and sum(of
starts[*])^=0)
                                                      or sum(of starts[*])=0
                                          ) then do;
                                    /* check whether there is a macro call before *
sign  */
                                    /* delete the  macro call from the beginning of
the statement*/
                                    call missing(constat1,constat0,bracket);
                                    constat1=strip(constat)||" "||substr("
"||content,1,pos);
                                    constat0=deleteMacro(constat1,bracket);
                                    /* if parentheses match perfectly and the
statement before the * sign is empty,then * sgin is the beginning of the comment
*/
                                    if bracket=0 and missing(strip(constat0)) then
do;
                                          star1num=pos;
                                          leave;
                                    end;
                              end;
```

```
                    end;
              end;
        end;

        if sum(of starts[*])>0 then count+1;
    end;

    if sum(of ends[*])=0 then do;
        /* search the end  of comment and "" '' */
        if find(content,";") and star2num=0 then do;
              if star1num>0 and count=1 then
star2num=find(content,";",star1num);
              else star2num=find(content,";");
        end;

        if find(content,"*/") and slash2num=0 then do;
              slash2num=find(content,"*/");
        end;


        if find(content,"'") and sing2num=0 then do;
              if sing1num>0 and find(substr(content,find(content,"'")+1),"'")
then do;
                    /* count =1 means the first line of comment or ""''*/
                    if count=1 then
sing2num=find(substr(content,find(content,"'")+1),"')+find(content,"'");
                    else if count>1 then sing2num=find(content,"'");
              end;
              else if sing1num>0 and
find(substr(content,find(content,"'")+1),"')=0 then do;
                    if count=1 then do;end;
                    else if count>1 then sing2num=find(content,"'");
              end;
        end;

        if find(content,'"') and doub2num=0 then do;
              if doub1num>0 and find(substr(content,find(content,'"')+1),'"')
then do;
                    if count=1 then
doub2num=find(substr(content,find(content,'"')+1),'"')+find(content,'"');
                    else if count>1 then doub2num=find(content,'"');
              end;
              else if doub1num>0 and
find(substr(content,find(content,'"')+1),'"')=0 then do;
                    if count=1 then do;end;
                    else if count>1 then doub2num=find(content,'"');
              end;
        end;

        if find(content,";") and put2num=0 then do;
              put2num=find(content,";");
        end;

        if find(content,";") and let2num=0 then do;
              let2num=find(content,";");
        end;
    end;
```

```
    /* chioce the min start */
    min=999;
    do i=1 to dim(starts);
          if starts[i]<min and starts[i]^=0  then min=starts[i];
    end;

    startnum=whichn(min,of starts[*]);


    /* ========================process the content============================
*/
    /* startnum=0 means that there no the start of comment or ""''. therefore,do
not change the content */
    if startnum=0 then do;
          count=0;
                do x=1 to dim(starts);
                      starts[x]=0;
                      ends[x]=0;
                end;
          goto exit;
    end;
    /* there exist the first of comment or ""'',but no the end of the first start
of comment */
    else if min<999 and ends[startnum]=0 then do;
          if count=1 then do;
                if startnum=1 and starts[startnum]>1 then do;
                      content=substr(content,1,star1num-1);
                end;
                else if startnum=2 and starts[startnum]>1 then do;
                      content=substr(content,1,slash1num-1);
                end;
                else if startnum=3 and starts[startnum]>1 then do;
                      content=substr(content,1,sing1num-1);
                end;
                else if startnum=4 and starts[startnum]>1 then do;
                      content=substr(content,1,doub1num-1);
                end;
                else if startnum=5 and starts[startnum]>1 then do;
                      content=substr(content,1,put1num-1);
                end;
                else if startnum=6 and starts[startnum]>1 then do;
                      content=substr(content,1,let1num-1);
                end;
                else content="";
          end;
          else if count>1 then do;
                content="";
          end;
          do j=1 to dim(starts);
                if j^=startnum then starts[j]=0;
                ends[j]=0;
          end;
          count+1;
    end;

    /* there exist the first and end of comment or ""'' */
```

```
    else do;
          if count=1 then do;
                if startnum=1 and star2num>0 then do;
    /*                  content=prxchange('s/(\*)(.*?)(;)/ /io',1,content); */
                        /* skip the star in the normal statement  */
                        content=substr(" "||content,1,star1num)
                                    ||prxchange('s/(%\*)(.*?)(;)|(\*)(.*?)(;)/
/io',1,substr(content,star1num));
                        star1num=0;
                        star2num=0;
                  end;
                  if startnum=2 and slash2num>0 then do;
                        content=prxchange('s/(\/\*)(.*?)(\*\/)/ /io',1,content);
                        slash1num=0;
                        slash2num=0;
                  end;
                  if startnum=3 and sing2num>0 then do;
                        content=prxchange("s/(\')(.*?)(\')/ /io",1,content);
                        sing1num=0;
                        sing2num=0;
                  end;
                  if startnum=4 and doub2num>0 then do;
                        content=prxchange('s/(\")(.*?)(\")/ /io',1,content);
                        doub1num=0;
                        doub2num=0;
                  end;
                  if startnum=5 and put2num>0 then do;
                        content=prxchange('s/%put( )+(.*?);/ /io',1,content);
                        put1num=0;
                        put2num=0;
                  end;
                  if startnum=6 and let2num>0 then do;
                        content=prxchange('s/%let( )+(\w)+( )*=(.*?);/
/io',1,content);
                        let1num=0;
                        let2num=0;
                  end;
                  count=0;
                  do j=1 to dim(starts);
                        starts[j]=0;
                        ends[j]=0;
                  end;
                  goto loop;
          end;
          else if count>1 then do;
                  if startnum=1 and  ends[startnum]>0 then do;
                        content=" "||substr(content,star2num+1);
                  end;
                  else if startnum=2 and ends[startnum]>0 then do;
                        content=" "||substr(content,slash2num+2);
                  end;
                  else if startnum=3 and ends[startnum]>0 then do;
                        content=" "||substr(content,sing2num+1);
                  end;
                  else if startnum=4 and ends[startnum]>0 then do;
                        content=" "||substr(content,doub2num+1);
                  end;
```

```
                else if startnum=5 and ends[startnum]>0 then do;
                      content=" "||substr(content,put2num+1);
                end;
                else if startnum=6 and ends[startnum]>0 then do;
                      content=" "||substr(content,let2num+1);
                end;
                count=0;
                do m=1 to dim(starts);
                      starts[m]=0;
                      ends[m]=0;
                end;
                goto loop;
          end;
    end;
    exit:

    /* check the  whether the remaining semicolons are found*/
    /* 1.the remaining semicolons has found */
    if cardnum =0.5 and strip(content)=";" then cardnum=1;
    /* 2.the remaining semicolons has not found and the text is not missing */
    else if cardnum=0.5 and not missing(strip(content)) then cardnum=0;
    else if cardnum =1.5 and strip(content)=";" then cardnum=2;
    else if cardnum=1.5 and not missing(strip(content)) then cardnum=0;

    /* check the end of the obs wheather or not is cards; lines; datalines;
cards4; lines4; datalines4; */

    /* cards */
    if prxmatch('/cards(?!4)/io', content) then do;
          /* 1.cards; in the middle of statement     */
          if prxmatch('/(;|\))cards;/io',compress(content))
                or prxmatch('/ cards *;/io',strip(content)) then cardnum=1;
          /* 2.cards; in the beginning of statement */
          else if find(lowcase(compress(content)),"cards;")=1  then cardnum=1;
          /*3. cards in the beginning of statement,but there is a character
before the cards       */
          else if prxmatch('/(;|\))cards$/io',compress(content)) or
prxmatch('/^cards$/io',compress(content)) then cardnum=0.5;
    end;
    /* lines */
    if prxmatch('/(?<!data)lines(?!4)/io', content) then do;
          if prxmatch('/(;|\))lines;/io',compress(content))
                or prxmatch('/ lines *;/io',strip(content))  then cardnum=1;
          else if find(lowcase(compress(content)),"lines;")=1 then cardnum=1;
          else if  prxmatch('/(;|\))lines$/io',compress(content)) or
prxmatch('/^lines$/io',compress(content)) then cardnum=0.5;
    end;
    /* datalines */
    if prxmatch('/datalines(?!4)/io', content) then do;
          if prxmatch('/(;|\))datalines;/io',compress(content))
                or prxmatch('/ datalines *;/io',strip(content)) then cardnum=1;
          else if find(lowcase(compress(content)),"datalines;")=1 then cardnum=1;
    /*    else if
find(lowcase(compress(content)),";datalines")=(length(compress(content))-9) then
cardnum=0.5; */
          else if prxmatch('/(;|\))datalines$/io',compress(content)) or
prxmatch('/^datalines$/io',compress(content)) then cardnum=0.5;
```

```
    end;
    /* cards4 */
    if prxmatch('/cards4/io',content) then do;
          if prxmatch('/(;|\))cards4;/io',compress(content))
                or prxmatch('/ cards4 *;/io',strip(content)) then cardnum=2;
          else if find(lowcase(compress(content)),"cards4;")=1 then cardnum=2;
          else if prxmatch('/(;|\))cards4$/io',compress(content)) or
prxmatch('/^cards4$/io',compress(content)) then cardnum=1.5;
    end;
    /* lines4 */
    if prxmatch('/(?<!data)lines4/io', content) then do;
          if prxmatch('/(;|\))lines4;/io',compress(content))
                or prxmatch('/ lines4 *;/io',strip(content)) then cardnum=2;
          else if find(lowcase(compress(content)),"lines4;")=1 then cardnum=2;
          else if prxmatch('/(;|\))lines4$/io',compress(content)) or
prxmatch('/^lines4$/io',compress(content)) then cardnum=1.5;
    end;
    /* datalines4 */
    if prxmatch('/datalines4/io',content) then do;
          if prxmatch('/(;|\))datalines4;/io',compress(content))
                or prxmatch('/ datalines4 *;/io',strip(content)) then cardnum=2;
          else if find(lowcase(compress(content)),"datalines4;")=1 then
cardnum=2;
          else if prxmatch('/(;|\))datalines4$/io',compress(content)) or
prxmatch('/^datalines4$/io',compress(content)) then cardnum=1.5;
    end;

    /* Get the end of last processed observation but not missing. */
    if not missing(strip(content)) then
          /* just for the assii character */
          contentlag=substr(strip(content),length(strip(content)));

    /* Keep the last sas statment   */
    if not missing(strip(content)) then do;
          /* just for the assii character */
          if find(content,";")  then do;
                /* get the  postion and length of the last semicolon*/
                constat2=strip(constat)||" "||strip(content);
                con_start=1;
                con_len=length(constat2);
                regid=prxparse('/;/io');
                call prxnext(regid,con_start,con_len,constat2,position,statlen);
                do while(position >0);
                      constat=substr(constat2,position+1);
                      call
prxnext(regid,con_start,con_len,constat2,position,statlen);
                end;
                call missing(position,constat2);
          end;
          else if find(content,";")=0 then do;
                constat=strip(constat)||" "||strip(content);
          end;
    end;



/*  if not missing(content); */
```

```
    run;
%mend delete_comment;

/* define a function to delete the macro call before * symbol */
proc fcmp outlib=work.funcs.math  trace;
function deleteMacro(inputString0 $,depth0)$;
    outargs depth0;
    attrib inputString inputString1 outString length=$100;
    depth0=0;
    count_f1=1;
    count_f2=1;
    count_f3=1;
    inputString=inputString0;
    do while(max(count_f1,count_f2,count_f3)=1);
          if prxmatch('/(%if|%do|%put|%let)(?=[^A-Za-z0-
9_\(])/io',strip(inputString))=1 then do;
                count_f1=0;count_f2=0;count_f3=0;
          end;
          else if prxmatch('/(%if|%do|%put|%let)$/io',strip(inputString))=1 then
do;
                count_f1=0;count_f2=0;count_f3=0;
          end;
          else if prxmatch('/%[A-Za-z_]\w* *\(/io',strip(inputString))=1 then do;
                inputString1=prxchange('s/(%[A-Za-z_]\w*)( *)(\()/
$3/io',1,inputString);
                len=length(inputString1);
                 pos_a = 1;
          depth = 0;

              do while(pos_a <= len);
                  char = substr(inputString1, pos_a, 1);
                  if char = '(' then do;
                      depth = depth + 1;
                      if pos_a=1 then inputString1 = "$"||substr(inputString1,
pos_a + 1);
                      else inputString1 = substr(inputString1, 1, pos_a -
1)||"$"||substr(inputString1, pos_a + 1);
                  end;
                  else if char = ')' then do;
                      depth = depth - 1;
                      if depth >= 0  then do;
                      if pos_a=1 then inputString1 = "$"||substr(inputString1,
pos_a + 1);
                      else inputString1 = substr(inputString1, 1, pos_a -
1)||"$"||substr(inputString1, pos_a + 1);
                      end;
                  end;
                      else if depth>0 or (depth=0 and char='') then do;
                      if pos_a=1 then inputString1 = "$"||substr(inputString1,
pos_a + 1);
                      else inputString1 = substr(inputString1, 1, pos_a -
1)||"$"||substr(inputString1, pos_a + 1);
                      end;
                  pos_a = pos_a + 1;
                      depth0=depth;
                      if depth=0 and char=")" then leave;
              end;
```

```
                inputString=prxchange('s/^ +\$+/ /io',1," "||inputString1);
                call
missing(inputString1,mpos,mlen,pos_a,depth,char,len,start0,start1);
                count_f1=prxmatch('/%[A-Za-z_]\w* *\(/io',strip(inputString))=1;
                count_f2=prxmatch('/%[A-Za-z_]\w*(?=[^A-Za-z0-
9_\(])/io',strip(inputString))=1;
                count_f3=prxmatch('/(%[A-Za-z_]\w*)$/io',strip(inputString))=1;
          end;

          else if prxmatch('/%[A-Za-z_]\w*(?=[^A-Za-z0-
9_\(])/io',strip(inputString))=1 then do;
                inputString=prxchange('s/(%[A-Za-z_]\w*)(?=[^A-Za-z0-9_\(])/
/io',1,inputString);
                count_f1=prxmatch('/%[A-Za-z_]\w* *\(/io',strip(inputString))=1;
                count_f2=prxmatch('/%[A-Za-z_]\w*(?=[^A-Za-z0-
9_\(])/io',strip(inputString))=1;
                count_f3=prxmatch('/(%[A-Za-z_]\w*)$/io',strip(inputString))=1;
          end;

          else if prxmatch('/(%[A-Za-z_]\w*)$/io',strip(inputString))=1 then do;
                inputString="";
                count_f1=prxmatch('/%[A-Za-z_]\w* *\(/io',strip(inputString))=1;
                count_f2=prxmatch('/%[A-Za-z_]\w*(?=[^A-Za-z0-
9_\(])/io',strip(inputString))=1;
                count_f3=prxmatch('/(%[A-Za-z_]\w*)$/io',strip(inputString))=1;
          end;

          else do;
                count_f1=0;
                count_f2=0;
                count_f3=0;
          end;
    end;
    outString=inputString;
    return(outString);
endfunc;
run;
quit;
```