Don't quote me.
Almost having fun with quotes and macros

By Mathieu Gaouette

Plan

- Introduction
- The problem
- Avoiding the problem
- Tackling the problem
- Acknowledgements

Introduction

The problem

- Lets look at how all of it starts…
- Bob created a report.
- The title of the report was put in a macro variable on the top of the program.

```
%let report_title = Super amazing report ;

title &report_title. ;

proc freq data=sashelp.cars ;
    table make ;
quit ;
```

The problem

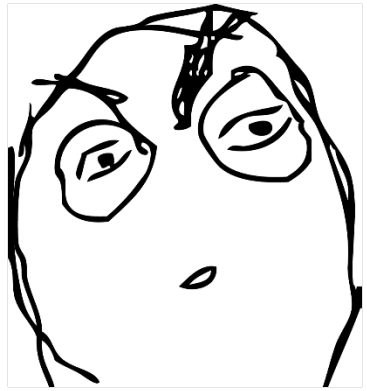
- Bob really likes his report.
- He wants to make sure that everyone knows it’s his report.

```
%let report_title = Bob's Super amazing report ;

title &report_title. ;

proc freq data=sashelp.cars ;
    table make ;
quit ;
```

Log

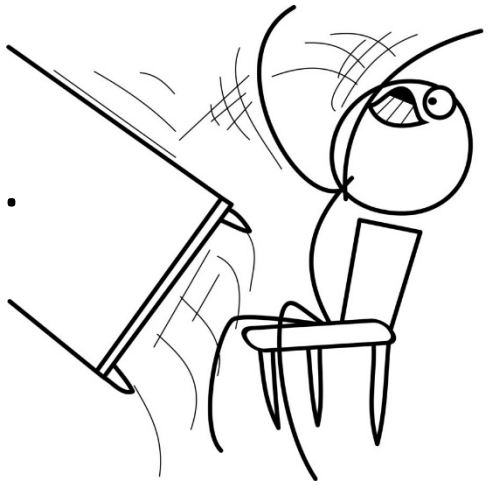
```
65 %let report_title = Bob's Super amazing report ;
66
67 title &report_title. ;
68
69 proc freq data=sashelp.cars ;
70 table make ;
71 quit ;
72
```

No report!



The problem

- Bob is really going overboard with his title.
- He would like people to know that this is just a joke.



```
%let report_title = Bob's "Super amazing" report ;

title &report_title. ;

proc freq data=sashelp.cars ;
    table make ;
quit ;
```

Log

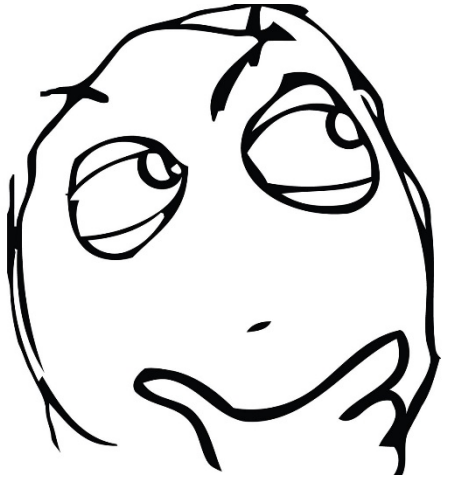
```
65 %let report_title = Bob's "Super amazing" report ;
66
67 title &report_title. ;
68
69 proc freq data=sashelp.cars ;
70 table make ;
71 quit ;
72
```

Still no report!

The root of the problem

- SAS uses quotes and double quotes as text delimeters.
- SAS expects quotes to comes in pairs.
- This pair of quotes delimits a chain of “characters” that can’t be divided.
- Why isn’t any processing happening when submitting such code?

SAS is simply waiting for you to submit the rest of the code that ends the quote “block”.



The root of the problem

- Lets look back at our last code:

```
%let report_title = Bob's "Super amazing" report ;

title &report_title. ;

proc freq data=sashelp.cars ;
    table make ;
quit ;
```

- The quote block has a starting point but no end.

Avoiding the problem

- Sometimes, it’s faster to simply avoid the problem.
- Common case are apostrophes such as **Bob's**.
- You can use a similar character as the single quote that won’t trigger a quoting block.
- Possible characters are:
  - The apostrophe (‘)

    drawbacks: *Is it even on the keyboard!*
  - The grave accent (`)

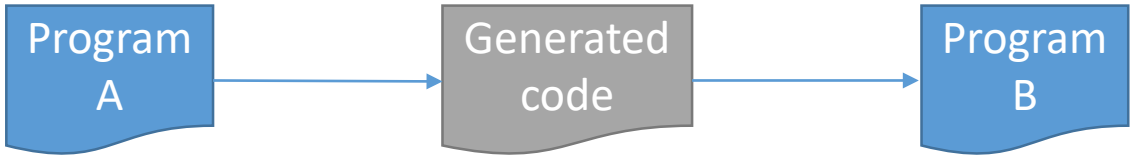
    drawbacks: *Sometimes looks funny depending on font.*

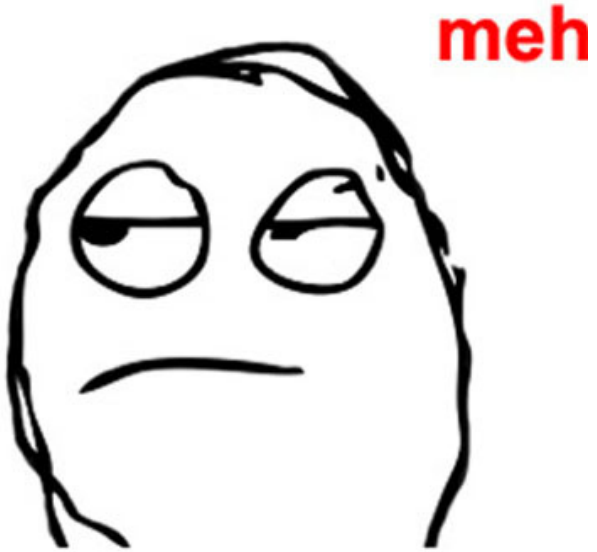
Avoiding the problem

- Ok, ok, not all use of quotes are for apostrophes.
- Some other common cases are:
  - Setting a date between quotes for later use (such as a date for a DBMS query)
  - Setting a fully qualified filename (path and name)
- Besides, if you want to put a single quote inside a macro variable, it’s your right!

Avoiding the problem

- Another way would be to generate the code.
- Program A writes code in a text file (data step).
- Then, program B includes that text file as code (%include).





- Pros: It works in most cases
- Cons: Looks messy and can be hard to support

Lets tackle the problem

- In order to quote quotes, we'll need to mask them.
- %str allows you to mask items during the compilation of a program (or macro).
- %str will allow resolution of macro variables and macro program within.
- %nrstr goes one step further and also masks & and %.
- You can think of nrstr as **N**ot **R**esolving str.

Lets tackle the problem

- A simple way of looking at this is seeing %str and %nrstr as ways of removing “powers” or abilities to special characters and keywords.
- So %str and %nrstr are like kryptonite.
- They remove the abilities of the following…

“superheroes”:

blank	)	=	NE
;	(	|	LE
¬	+	#	LT
^	--	AND	GE
~	*	OR	GT
, (comma)	/	NOT	
'	<	IN	
"	>	EQ	

Lets tackle the problem

- The double quote and single quote are not masked as easily.
- Like a famous dynamic duo, they always go in pairs, even within %str and %nrstr.
- If you must use only one, you need to prefix it with %.
- If you want to use a % within a %str function, you put %%.
- This is the only way of telling SAS to consider them separated.

Example: if you want %', you need to use %str(%%%')

Lets tackle the problem

- Getting back to our initial problem:

```
%let report_title = Bob's "Super amazing" report ;

title &report_title. ;

proc freq data=sashelp.cars ;
    table make ;
quit ;
```

```
%let report_title = %str(Bob%'s "Super amazing" report) ;

title &report_title. ;

proc freq data=sashelp.cars ;
    table make ;
quit ;
```

Lets tackle the problem

- Easy enough, right?
- Not quite.
- Look at this code:

```
%let val = aaa;
%let testval = %str(%'&val%');
%put TRACE: &testval. ;

data _null_;
  val = &testval;
run;
```

Lets tackle the problem

Program:

```
%let val = aaa;
%let testval = %str(%'&val%');
%put TRACE: &testval. ;

data _null_;
  val = &testval;
run;
```

Log:

```
13  28
14  29          %let val = aaa;
15  30          %let testval = %str(%'&val%');
16  31          %put TRACE: &testval. ;
17  TRACE: 'aaa'
18  32
19  33          data _null_;
20  34            val = &testval;
21  NOTE: Line generated by the macro variable "TESTVAL".
22  34           'aaa'
23                _
24               386
25                _
26               200
27  ERROR 386-185: Expecting an arithmetic expression.
28
29  ERROR 200-322: The symbol is not recognized and will be ignored.
30
31  35          run;
32
33  NOTE: The SAS System stopped processing this step because of errors.
34  NOTE: DATA statement used (Total process time):
35        real time           0.00 seconds
36        cpu time            0.00 seconds
37
```

Lets tackle the problem

- Apparently, quotes and double quotes loose their meaning when you mask them.
- If you wish to give them their “powers” back, you need to unmask them.
- A SAS macro function does just that.
- It is called…

  %unquote.

Lets tackle the problem

- Fixed program:

Program:

```
%let val = aaa;
%let testval = %str(%'&val%');
%put TRACE: &testval. ;

title %unquote(&testval.) ;
proc freq data=sashelp.cars ;
    table make ;
quit ;

data _null_;
  val = %unquote(&testval.);
run;
```

Log:

```
14  29         %let val = aaa;
15  30         %let testval = %str(%'&val%');
16  31         %put TRACE: &testval. ;
17  TRACE: 'aaa'
18  32
19  33         title %unquote(&testval.) ;
20  34         proc freq data=sashelp.cars ;
21  35             table make ;
22  36         quit ;
23
24  NOTE: There were 428 observations read from the data set SASHELP.CARS.
25  NOTE: PROCEDURE FREQ used (Total process time):
26        real time           0.01 seconds
27        cpu time            0.00 seconds
28
29
30  37
31  38         data _null_;
32  39           val = %unquote(&testval.);
33  40         run;
34
35  NOTE: DATA statement used (Total process time):
36        real time           0.00 seconds
37        cpu time            0.00 seconds
38
39
```

Quiz!

Program:

```
%let val = aaa;
%let testval = %str(%'&val%');
%put TRACE: &testval. ;

title %unquote(&testval.) ;
proc freq data=sashelp.cars ;
    table make ;
quit ;

data _null_;
  val = %unquote(&testval.);
run;
```

- What is the title printed for the frequency report?

A) aaa

B) 'aaa'

C) I wasn't paying attention, could you start from slide one again?

Going the extra mile

- Lets almost have fun with a few more examples…

What we want:

```
%let val = M&M ;

data _null_;
  candy = "&VAL";
run;
```

What we need:

```
%let val = M%nrstr(&)M ;
        Or: M%str(&)M ;
data _null_;
  candy = "&VAL";
run;
```

Going the extra mile

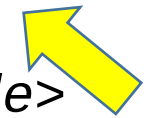
- Lets almost have fun with a few more examples…

What we want:

```
%let fname = Marie-Eve ;
…
/* within a macro pgm */
%if &fname. NE <nothing>
%then %do ;
      <some fancy code>
%end ;
…
```

What we need:

```
%let fname = Marie%str(-)Eve ;
…
/* within a macro pgm */
%if &fname. NE %str( ) %then
%do ;
      <some fancy code>
%end ;
…
```



Going the extra mile

- Lets almost have fun with a few more examples…

<u>What we want:</u>

```
%let somevar = TST ;
%let pushlimits = '%)*&somevar. /* ;

Data test ;
  nonsense = "&pushlimits." ;
Run ;
```

<u>What we need:</u>

```
%let somevar = TST ;
%let pushlimits =
    %str(%'%%%)*&somevar. /)%str(*) ;

Data test ;
  nonsense = "&pushlimits." ;
Run ;
```

Bonus: Executive summary

- If you want to use any of these “superheroes” in a macro context, you might need to mask them.
- Masking anything else does nothing.
- %str will mask and let macro variable be resolved.
- %nrstr will mask without resolving macros.
- If you want to use a quote or double quote alone, precede it with a %.
- If you wish to use a quoted string, make sure it is not masked (use unquote).

blank	)	=	NE
;	(	|	LE
¬	+	#	LT
^	--	AND	GE
~	*	OR	GT
, (comma)	/	NOT	
'	<	IN	
"	>	EQ	

Acknowledgements

- Arthur L. Carpenter:

  Quotes within Quotes: When Single (‘) and Double (“) are not Enough

  https://support.sas.com/resources/papers/proceedings15/2221-2015.pdf

- SAS support:

  Macro Quoting: %str and %nrstr functions

  http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a001061290.htm

  Macro Functions: %unquote

  http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a000543618.htm