PhUSE 2007

Paper TU04

Quoting Macro Variable References

Shan Lee, GlaxoSmithKline, Harlow, United Kingdom

ABSTRACT

The aims of this paper are:

1. To give examples of situations in which quoted SAS® macro variable references do not resolve as one might expect.
2. To explain the behaviour of the macro variables in these examples.

INTRODUCTION

In this paper, I will discuss scenarios in which the following occur:

1. Macro variable references enclosed within double quotes do not resolve.
2. Macro variable references enclosed within single quotes do resolve.

This paper is **not** intended to be an introduction to macro quoting functions; this paper is targeted at SAS programmers who already understand the concepts of macro quoting, in particular:

1. The distinction between compile-time macro quoting functions (%str, %nrstr) and execution-time macro quoting functions (%bquote, %nrbquote).
2. The distinction between macro quoting functions that mask the meaning of ampersand and percent (i.e. %nrstr, %nrbquote) and those that do not (%str, %bquote).

QUOTING MACRO VARIABLES USING %NRSTR AND %NRBQUOTE

Let us suppose that we wish to construct a macro with the following requirements:

1. The macro shall have a parameter for specifying a text string.
2. The macro shall create a macro variable that stores a list of adverse event preferred terms.
3. If the text string (requirement 1) contains a reference to the macro variable storing preferred terms (requirement 2), then this macro variable reference will be resolved during macro execution.

Suppose we attempt to meet these requirements by constructing a macro similar to the following:

```
%macro test1 (first_row = );

  %* The following statement could be replaced with code to generate a list of preferred
terms based on your study data;

  %let aepts = headache or dizziness;
  %let first_row = %nrbquote(&first_row);
  %put first_row = &first_row;

%mend test1;
```

FIRST APPROACH - THIS DOES NOT WORK!

Let us consider what would happen if the macro *%test1* is invoked as follows:

```
%test1 (first_row = %nrstr("Number of subjects experiencing &aepts"))
```

The value of *&aepts* is not known before the start of macro execution, so the %nrstr function has been applied to avoid generating a WARNING message concerning the unresolved macro variable reference.

The macro variable reference *&aepts* is enclosed within double quotes, so one might expect that it would be resolved during macro execution.

When the `%let first_row = %nrbquote(&first_row)` statement executes, one might expect the value of first_row to be assigned in the following sequence (please note that this is **not** what actually happens):

Value assigned to first_row	Incorrect (but reasonable) Explanation
`%nrbquote(&first_row)`	This is the expression to the right of the equals sign in the %let statement.
`%nrbquote("Number of subjects experiencing &aepts")`	&first_row resolves to the value that was assigned during macro invocation.
`%nrbquote("Number of subjects experiencing headache or dizziness")`	**THIS EXPLANATION IS INCORRECT: &aepts resolves because it is enclosed within double quotes.**
`"Number of subjects experiencing headache or dizziness"`	Macro quoting will be applied to any special characters in the final value, because of the execution-time macro quoting function, %nrbquote.

However, when the macro executes, the message generated in the log by the `%put first_row = &first_row` statement is:

```
first_row = "Number of subjects experiencing &aepts"
```

The reason for this behaviour is that during macro invocation, macro quoting is applied to the macro variable reference *&aepts*, because of the %nrstr function. During macro execution, *&aepts* does not resolve, because macro quoting has already been applied to its ampersand before the macro began to execute.

When the `%let first_row = %nrbquote(&first_row)` statement executes, the value of first_row is assigned in the following sequence:

Value assigned to first_row	Explanation
`%nrbquote(&first_row)`	This is the expression to the right of the equals sign in the %let statement.
`%nrbquote("Number of subjects experiencing &aepts")`	&first_row resolves to the value that was assigned during macro invocation.
Value has not changed since the previous step: `%nrbquote("Number of subjects experiencing &aepts")`	**THIS EXPLANATION IS CORRECT: &aepts will not resolve, even though it is enclosed within double quotes, because macro quoting was**

	applied to its ampersand by %nrstr, before the macro started to execute. Therefore, &aepts is seen as text during macro execution.
"Number of subjects experiencing &aepts"	Macro quoting will be applied to any special characters in the final value, because of the execution-time macro quoting function, %nrbquote.

SECOND APPROACH - THIS WORKS

In order to assign a meaningful value to *first_row*, we could use single quotes instead of double quotes when the *%test1* macro is invoked:

```
%test1 (first_row = %nrstr('Number of subjects experiencing &aepts'))
```

The macro variable reference *&aepts* is enclosed within single quotes, so one might expect that it would not be resolved during macro execution.

When the `%let first_row = %nrbquote(&first_row)` statement executes, one might expect the value of first_row to be assigned in the following sequence (please note that this is **not** what actually happens):

Value assigned to first_row	Incorrect (but reasonable) Explanation
`%nrbquote(&first_row)`	This is the expression to the right of the equals sign in the %let statement.
`%nrbquote('Number of subjects experiencing &aepts')`	&first_row resolves to the value that was assigned during macro invocation.
`%nrbquote('Number of subjects experiencing &aepts')`	**THIS EXPLANATION IS INCORRECT:** **&aepts does not resolve because it is enclosed within single quotes; and, in any case, macro quoting was applied by %nrstr to the ampersand in &aepts, during macro invocation.**
`'Number of subjects experiencing &aepts'`	Macro quoting will be applied to any special characters in the final value, because of the execution-time macro quoting function, %nrbquote.

However, when the macro executes, the message generated in the log by the `%put first_row = &first_row` statement is:

```
first_row = 'Number of subjects experiencing headache or dizziness'
```

During macro invocation, macro quoting was not applied to &aepts, because it was enclosed within single quotes: everything within the single quotes was seen as text, including macro variable references, so %nrstr ignored the ampersand, and macro quoting was not applied to the ampersand.

During macro execution, everything (including single quotes) enclosed within the %nrbquote() is seen as text, except macro variable references, which will still resolve, so long as they have not previously been macro-quoted with %nrstr. Single quotes within %nrbquote are seen as text, and they do not prevent a macro variable reference from resolving.

When the `%let first_row = %nrbquote(&first_row)` statement executes, the value of first_row is assigned in the following sequence:

Value assigned to first_row	Explanation
`%nrbquote(&first_row)`	This is the expression to the right of the equals sign in the %let statement.
`%nrbquote('Number of subjects experiencing &aepts')`	&first_row resolves to the value that was assigned during macro invocation. Note that &aepts was seen as text during macro invocation, so macro quoting was never applied to it.
`%nrbquote('Number of subjects experiencing headache or dizziness')`	**THIS EXPLANATION IS CORRECT:** **&aepts will resolve, even though it is enclosed within single quotes: the single quotes are seen as text,**

	and they do not prevent &aepts from resolving.
'Number of subjects experiencing headache or dizziness'	Macro quoting will be applied to any special characters in the final value, because of the execution-time macro quoting function, %nrbquote.

EFFECT OF NESTING SINGLE AND DOUBLE QUOTES

Let us suppose that we wish to construct a macro that meets the following requirements:

1. The macro shall assign a value to a macro variable.
2. The value of the macro variable (requirement 1) shall consist of a text string enclose within single or double quotes.
3. The text string (requirement 2) shall itself contain a substring enclosed within single or double quotes.
4. The value of the substring shall be passed to the macro via a parameter of the macro.

NESTING DOUBLE QUOTES WITHIN SINGLE QUOTES – THIS APPROACH DOES NOT WORK!

One might try to meet these requirements by defining a macro like this:

```
%macro test2(aepts = );

  %let first_row = 'Number of subjects experiencing "&aepts"';

  %put first_row = &first_row;

%mend test2;

%test2(aepts = headache or dizziness)
```

One might expect that the above code would generate the desired result, because the ampersand in *&aepts* is enclosed within double quotes. However, when the code is run, the following message is generated in the log:

```
first_row = 'Number of subjects experiencing "&aept"'
```

The reason for this behaviour is that everything (including the double quotes and ampersand) within the single quotes is seen as text, so the macro variable reference *&aepts* does not resolve.

NESTING SINGLE QUOTES WITHIN DOUBLE QUOTES – THIS WORKS

In order to assign a meaningful value to *first_row*, we could amend the macro so that single quotes are nested within double quotes:

```
%macro test3 (aepts = );

  %let first_row = "Number of subjects experiencing '&aepts'";

  %put first_row = &first_row;

%mend test3;

%test3(aepts = headache or dizziness)
```

One might expect that the above code would not generate the desired result, because the ampersand in *&aepts* is enclosed within single quotes. However, when the code is run, the following message is generated in the log:

```
first_row = "Number of subjects experiencing 'headache or dizziness'"
```

The reason for this behaviour is that everything (including the single quotes) within the double quotes is seen as text. However, double quotes do not prevent macro variable references from resolving (and the single quotes have no effect because they are seen as text), so *&aepts* resolves to give the desired result.

CONCLUSION

We need to understand the effects of quoting macro variable references, so that we can develop reliable SAS macros. The following table summarises the various situations that have been considered in this paper, together with the effects of quoting macro variable references in each situation:

Situation	Has macro quoting previously been applied to &reference?	Effect
%nrstr("&reference")	No	Macro quoting will be applied to the ampersand.
%nrstr('&reference')	No	Macro quoting will not be applied to the ampersand.
%nrbquote("&reference")	Yes	Macro variable reference will not resolve.
%nrbquote('&reference')	No	Macro variable reference will resolve.
' "&reference" '	No	Macro variable reference will not resolve.
" '&reference' "	No	Macro variable reference will resolve.

CONTACT INFORMATION

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

Author's name: Shan Lee

Company name and address:

GlaxoSmithKline
New Frontiers Science Park
Third Avenue
Harlow
Essex
CM19 5AW
United Kingdom

Work phone: +44 (0)1279 646 802

Email: shan.2.lee@gsk.com

Web: www.gsk.com