A PRACTICAL GUIDE TO MACRO QUOTING AND WORKING WITH MACRO VARIABLES

Edinburgh – PhUSE 2017

Dawid Militowski, HTA Statistical Programming

Agenda

- Introduction
- Masking tokens
- Macro quoting functions
- Data step
- Examples
- Conclusion
- Questions

Introduction

- Whenever macro variables or macros are used there is often a need to mask tokens
- Best to mask all tokens and not only some
- Multiple functions to choose from
- Different tools
- A simplified view on macro quoting
- Recommendations

Tokens

- Tokens are divided into 3 categories:
- Type I:
  - Operators {+ - * / < > = ¬ ^ | ~}
  - Mnemonics {AND OR NOT EQ NE LE LT GE GT IN}
  - Miscellaneous {blank , ; “” ‘’ () #}
- Type II: Unmatched Characters {“ ‘ ( )}
- Type III: Macro Triggers {& %}
- Avoid unexpected results when value passed in a macro variable or a macro parameter

Compilation and execution

- Compilation

```
%macro <macro name>;
    <macro code>
%mend <macro name>;
```

- Execution

```
% <macro name>;
```


MSD INVENTING FOR LIFE

Macro quoting functions

- %NRSTR() – used to mask tokens in compilation phase
- %NRBQUOTE() – used to mask tokens in execution phase
- %SUPERQ() – used to mask tokens until they are explicitly unmasked
- %UNQUOTE() – used to explicitly unmask a macro variable
- %QSCAN() – used to scan a macro variable with or without tokens
- %QSUBSTR() – used to extract a sub string from a macro variable with or without tokens

Data step

- A powerful tool for processing macro variables
- SYMGET() – used to load a macro variable into a dataset variable
- CALL SYMPUTX() – used to create a macro variables in a data step
- RESOLVE() – used to unmask tokens within a data step - similar to %UNQUOTE() in macro language
- Beyond those 3 functions, values can be processed in the same way as any variable in a dataset

Macro functions, Example 1

```
%macro test(var=);
   %if &var=%NRSTR(New & old ‘s ) %then %put %NRSTR(%increase);
   %else %put Wrong result;
%mend test;

%test(var=%NRBQUOTE(New & old ‘s));
```

>> %increase

```
%test(var=%BQUOTE(New & old ‘s));
%test(var=%STR(New & old ‘s));
```

>> ERROR: A character operand was found in the %EVAL function or %IF condition where a numeric operand is required. The condition was: &var=New & old ‘s

ERROR: The macro TEST will stop executing.

Macro functions, Example 2

```
%macro test(var=);
   %if &var=%STR(New & old ‘s ) %then %put %NRSTR(%increase);
   %else %put Wrong result;
%mend test;

%test(var=%NRBQUOTE(New & old ‘s));
```

>> ERROR: A character operand was found in the %EVAL function or %IF condition where a numeric operand is required. The condition was: &var=New & old ‘s

>> ERROR: The macro TEST1 will stop executing.

Example 3, main differences

```
data _null_;
    CALL SYMPUTX('test1','%CI and &x1');
run;
```

```
%macro test3(var=);
      %put &var;
%mend test3;
```

```
%test3(var=%NRSTR(&test1));
```

>>&test1

```
%test3(var=%NRBQUOTE(&test1));
```

>>WARNING: Apparent invocation of macro CI not resolved.

>>WARNING: Apparent symbolic reference X1 not resolved.

```
%test3(var=%SUPERQ(test1));
```

>>%CI and &x1

Example 4, data step

```
%let x2=abc;

data _null_;
   CALL SYMPUTX('test1','%CI and &x1');
   CALL SYMPUTX('test2',"%CI and &x1");
   CALL SYMPUTX('test3','%CI and'||"&x2");
   test='%CI and &x1';
   CALL SYMPUTX('test4',test,'G');
run;
```

As in any case when using macro variables in a data step, double quotes will cause the macro triggers to resolve and produce warnings

To pass both triggers while only a macro variable is to be resolved, more complicated code needs to be used

The value stored in a variable can be passed into a macro variable

Specifying the third parameter in CALL SYMPUTX as 'G' ensures that the macro variable is created as a global macro variable

Example 5, advantages of using data step

```
data _null_;
      CALL SYMPUTX('test1','% CI and "( . , &x1');
run;
```

```
%let x1=abc;
%let new=%SUPERQ(test1);
```

```
%put &new;
```
→ *>>% CI and "(. , &x1*

```
%put %UNQUOTE(&new);
```
→ *>>*

```
 data _null_;
      length test test1 $80;
      test = SYMGET('new');
      put test;
      test1 = RESOLVE(test);
      put test1;
  run;
```
→ *>>% CI and "(. , &x1*

→ *>>% CI and "(. , abc*

Conclusions

- Use the newest macro language functions %NRSTR(), %NRBQUOTE() and %SUPERQ()
- Understand in detail what the differences between those three functions
- For complex operations use a data step

THANK YOU

Questions

Send comments or suggestions to
dawid.militowski@merck.com