

Tip: How to use a 'quoted' SAS[®] macro variable to select observations

Gary Moore, Arkansas Foundation for Medical Care, Little Rock, AR

ABSTRACT

Often times we would like to repeat some process on a variable subset of a data set's observations. One way to derive a given subset is to use the SAS data step 'where' clause with the 'like' operator, e.g., "where upcase(drug) like '%ADVIL%'". Here we are using the wildcard character '%' to select all observations where the drug name variable contains the Advil drug label name. Rather than repeating the process' data step(s), etc., each time with a different drug name it would be desirable to form a looping construct in a macro program using a macro variable to derive the subset. In other words, it would be desirable to use a 'where' clause formed something like "where upcase(drug_name) like '%%upcase(&drug_name)%'". This statement doesn't work (for more than one reason). This paper describes different approaches on how to modify the 'where' clause so that it will be interpreted properly.

INTRODUCTION

This paper touches briefly on the subject of macro quoting and gives simple examples of how to quote the wildcard character '%' so that it will be interpreted in the proper context, i.e., interpreted as the wildcard character and not as the macro program invocation symbol. This all came about when a colleague wondered why the following 'where' clause — "where upcase(drug) like '%&drug%'" — didn't work. (I realized later on we could have used the 'contains' operator instead of the 'like' operator and therefore avoided the wildcard character and the single quotes in my colleague's code, but this would not have given us the chance to learn something about macro quoting.)

A BRIEF OVERVIEW OF MACRO QUOTING

The macro processor enables you to use various special characters as text; but because the macro language uses some of the same special characters to trigger specific macro processor activities, ambiguities often arise. Should the macro processor interpret a particular special character (e.g., & or %) or a mnemonic (e.g., GE or NOT) as text (literal characters) or as a symbol in the macro language? Macro quoting functions resolve these ambiguities by masking the significance of special characters so the macro processor does not misinterpret them, i.e., macro quoting functions tell the macro processor to interpret special characters and mnemonics as text rather than as part of the macro language.

The macro functions that mask special characters and mnemonic operators are called quoting functions because they behave like single quotation marks in SAS, i.e., the arguments to a macro quoting function are ignored as if they were enclosed in single quotation marks.

SAS CODE EXAMPLES

The first two code examples demonstrate different methods that correctly subset our initial data set but don't use macro quoting functions. (I list these results to baseline, if you will, the results I obtain in the code examples that do use macro quoting functions.)

NOTE: Copyright (c) 1999-2001 by SAS Institute Inc., Cary, NC, USA.

NOTE: SAS (r) Proprietary Software Release 8.2 (TS2M0)
Licensed to ARKANSAS FOUNDATION FOR MEDICAL CARE.

NOTE: This session is executing on the WIN_PRO platform.

NOTE: The data set WORK.DRUG_DATA has 4200 observations and 27 variables.

```
%let dsn01 = drug_data;
```

```
%let g1 = advil;  
%let g2 = aspirin;  
%let g3 = tylenol;
```

```
%let i = 1;
```

CODE EXAMPLE #1.

```
data &dsn01._d&i;  
  set &dsn01;  
  where upcase(nam_drug_label) like ('%' || "%upcase(&&g&i)" || '%');  
run;
```

NOTE: There were 62 observations read from the data set WORK.DRUG_DATA.

WHERE UPCASE(nam_drug_label) like '%||'ADVIL'||'%' ;

NOTE: The data set WORK.DRUG_DATA_D1 has 62 observations and 27 variables.

CODE EXAMPLE #2.

```
data &dsn01._d&i;  
  set &dsn01;  
  where upcase(nam_drug_label) contains "%upcase(&&g&i)";  
run;
```

NOTE: There were 62 observations read from the data set WORK.DRUG_DATA.

WHERE UPCASE(nam_drug_label) contains 'ADVIL';

NOTE: The data set WORK.DRUG_DATA_D1 has 62 observations and 27 variables.

CODE EXAMPLE #3.

```
data &dsn01._d&i;  
  set &dsn01;  
  where upcase(nam_drug_label) like '%%upcase(&&g&i)%';  
run;
```

NOTE: There were 0 observations read from the data set WORK.DRUG_DATA.

WHERE UPCASE(nam_drug_label) like '%%upcase(&&g&i)%';

NOTE: The data set WORK.DRUG_DATA_D1 has 0 observations and 27 variables.

This data step attempted to find drugs with the drug label name of (literally) '%%upcase(&&g&i)', i.e., the macro processor didn't resolve the value of the macro variable, 'g1' and certainly didn't perform the upper case operation. This code example demonstrates that single quotation marks cause the macro processor to ignore the special characters in the text string, i.e., the macro processor treats the text string as literal text.

CODE EXAMPLE #4.

```
data &dsn01._d&i;  
  set &dsn01;  
  where upcase(nam_drug_label) like "%%upcase(&&g&i)";  
run;
```

NOTE: There were 0 observations read from the data set WORK.DRUG_DATA.

WHERE UPCASE(nam_drug_label) like '%%upcase(advil)%';

NOTE: The data set WORK.DRUG_DATA_D1 has 0 observations and 27 variables.

Using double quotation marks allowed the macro processor to resolve the value of the macro variable, but unfortunately, the macro processor didn't perform the upper case operation. In other words, after resolving the value of the macro variable the updated text string was treated as literal text.

CODE EXAMPLE #5.

```
data &dsn01._d&i;  
  set &dsn01;  
  where upcase(nam_drug_label) like "% %upcase(&g&i)";  
run;
```

NOTE: There were 14 observations read from the data set WORK.DRUG_DATA.

```
WHERE UPCASE(nam_drug_label) like '% ADVIL%';
```

NOTE: The data set WORK.DRUG_DATA_D1 has 14 observations and 27 variables.

With a space character between the first two percent signs, the macro processor interpreted the first percent sign as the wildcard character '%' (as intended) and performed the upper case function on the resolved macro variable, 'g1', but as we see the number of observations found is not correct.

CODE EXAMPLE #6.

```
data &dsn01._d&i;  
  set &dsn01;  
  where upcase(nam_drug_label) like "%nrstr(%)%upcase(&g&i)%nrstr(%)";  
run;
```

NOTE: There were 62 observations read from the data set WORK.DRUG_DATA.

```
WHERE UPCASE(nam_drug_label) like '%ADVIL%';
```

NOTE: The data set WORK.DRUG_DATA_D1 has 62 observations and 27 variables.

Now with the wildcard character '%' appropriately quoted, we obtain the correct subset. (Note the two percent signs as the argument to the %nrstr macro quoting function. If there was only one percent sign argument, the %nrstr quoting function would interpret the closing parenthesis as the special character being quoted and therefore the expected (now missing) closing parenthesis for the macro program invocation would cause a compilation error.)

The following example demonstrates that there is more than one macro quoting function we could have used, e.g., %nrstr(%). In order to determine which macro quoting function is most appropriate one needs to know if the resolved character(s) must be masked at macro compilation or at macro execution. (For more details, see the references listed below.)

CODE EXAMPLE #7.

```
data &dsn01._d&i;  
  set &dsn01;  
  where upcase(nam_drug_label) like "%nrstr(%)%upcase(&g&i)%nrstr(%)";  
run;
```

NOTE: There were 62 observations read from the data set WORK.DRUG_DATA.

```
WHERE UPCASE(nam_drug_label) like '%ADVIL%';
```

NOTE: The data set WORK.DRUG_DATA_D06 has 62 observations and 27 variables.

CODE EXAMPLES #8, #9, AND #10.

The following code examples demonstrate three different ways to use macro programs to subset our original drug data set based on drug label names. To keep these macro programs simple and somewhat easier to read, I left out error checking and validation code, i.e., the macro programs don't verify that the 'drug_data' data set or the 'nam_drug_label' data set variable exist.

```
%let dsn01 = drug_data;
```

```
%let g1 = advil;
%let g2 = aspirin;
%let g3 = tylenol;
```

The following macro program will subset the original drug data set into as many data sets as you have macro variable drug names.

```
%macro drug_filter(dsn=, varname_drug=, imax=);
  %local i;

  %do i = 1 %to &imax;
    data &dsn._d&i;
      set &dsn;
      where upcase(&varname_drug) like "%nrstr(%%)%upcase(&&g&i)%nrstr(%%)";
    run;
  %end;
%mend drug_filter;
```

For example, using the three different drug name macro variables labeled above, we would invoke the macro program as follows:

```
%drug_filter(dsn=&dsn01, varname_drug=nam_drug_label, imax=3);
```

And obtain the following results:

```
NOTE: There were 62 observations read from the data set WORK.DRUG_DATA.
      WHERE UPCASE(nam_drug_label) like '%ADVIL%';
NOTE: The data set WORK.DRUG_DATA_D1 has 62 observations and 27 variables.

NOTE: There were 3040 observations read from the data set WORK.DRUG_DATA.
      WHERE UPCASE(nam_drug_label) like '%ASPIRIN%';
NOTE: The data set WORK.DRUG_DATA_D2 has 3040 observations and 27 variables.

NOTE: There were 470 observations read from the data set WORK.DRUG_DATA.
      WHERE UPCASE(nam_drug_label) like '%TYLENOL%';
NOTE: The data set WORK.DRUG_DATA_D3 has 470 observations and 27 variables.
```

Here's another macro program that will subset the original drug data set into a single data set based on the resolved value of the macro variable 'mvarname_drug'.

```
%macro drug_filter(dsn=, varname_drug=, mvarname_drug=);
  data &dsn._&mvarname_drug;
    set &dsn;
    where upcase(&varname_drug) like "%nrstr(%%)%upcase(&mvarname_drug)%nrstr(%%)";
  run;
%mend drug_filter;
```

We invoke this macro program three times as follows:

```
%drug_filter(dsn=&dsn01, varname_drug=nam_drug_label, mvarname_drug=advil);
%drug_filter(dsn=&dsn01, varname_drug=nam_drug_label, mvarname_drug=aspirin);
%drug_filter(dsn=&dsn01, varname_drug=nam_drug_label, mvarname_drug=tylenol);
```

And obtain the following results:

```
%drug_filter(dsn=&dsn01, varname_drug=nam_drug_label, mvarname_drug=advil);
```

NOTE: There were 62 observations read from the data set WORK.DRUG_DATA.
 WHERE UPCASE(nam_drug_label) like '%ADVIL%';
 NOTE: The data set WORK.DRUG_DATA_ADVIL has 62 observations and 27 variables.

```
%drug_filter(dsn=&dsn01, varname_drug=nam_drug_label, mvarname_drug=aspirin);
```

NOTE: There were 3040 observations read from the data set WORK.DRUG_DATA.
 WHERE UPCASE(nam_drug_label) like '%ASPIRIN%';
 NOTE: The data set WORK.DRUG_DATA_ASPIRIN has 3040 observations and 27 variables.

```
%drug_filter(dsn=&dsn01, varname_drug=nam_drug_label, mvarname_drug=tylenol);
```

NOTE: There were 470 observations read from the data set WORK.DRUG_DATA.
 WHERE UPCASE(nam_drug_label) like '%TYLENOL%';
 NOTE: The data set WORK.DRUG_DATA_TYLENOL has 470 observations and 27 variables.

This last macro program will subset the original drug data set into as many data sets as there are drug label names listed in a single macro variable delimited by, in this example, the space character. In order to make the code easier to read I created a macro variable 'name_drug' whose resolved value is the ith indexed drug label name in the list of drug label names.

```
%let drug_list = advil aspirin tylenol;

%macro drug_filter(dsn=, varname_drug=, list_drug=);
  %local i;
  %let i = 1;
  %let name_drug = %scan(&list_drug,&i,%str( ));

  %do %while(&name_drug ne);
    data &dsn._d&i;
      set &dsn;
      where upcase(&varname_drug) like "%nrstr(%%)%upcase(&name_drug)%nrstr(%%)";
    run;
    %let i = %eval(&i + 1);
    %let name_drug = %scan(&list_drug,&i,%str( ));
  %end;
%mend drug_filter;
```

We invoke this macro program as follows:

```
%drug_filter(dsn=&dsn01, varname_drug=nam_drug_label, list_drug=&drug_list);
```

And obtain the following results:

NOTE: There were 62 observations read from the data set WORK.DRUG_DATA.
 WHERE UPCASE(nam_drug_label) like '%ADVIL%';
 NOTE: The data set WORK.DRUG_DATA_D1 has 62 observations and 27 variables.

NOTE: There were 3040 observations read from the data set WORK.DRUG_DATA.
 WHERE UPCASE(nam_drug_label) like '%ASPIRIN%';
 NOTE: The data set WORK.DRUG_DATA_D2 has 3040 observations and 27 variables.

NOTE: There were 470 observations read from the data set WORK.DRUG_DATA.
 WHERE UPCASE(nam_drug_label) like '%TYLENOL%';
 NOTE: The data set WORK.DRUG_DATA_D3 has 470 observations and 27 variables.

You'll note that all three macro programs generate the same results (same number of variables and observations) although the names of the data sets are different.

CONCLUSION

Hopefully, I've shown that the macro quoting functions are important and essential even in very simple coding examples. You should now realize that you must think about the context of the special characters in your text strings that reference macro variables: should they be interpreted as ordinary text or as a symbol in the macro language? It's then just a matter of determining which macro (compilation- or execution-type) quoting function should be used. I also hope that the three macro programs give you an idea of how to iteratively partition a data set.

REFERENCES

SAS Institute Inc., *SAS[®] Macro Language: Reference, First Edition*, Cary, NC: SAS Institute Inc., 1997, 304 pp.
Burlew, Michele M., *SAS[®] Macro Programming Made Easy*, Cary, NC: SAS Institute Inc., 1998, 280 pp.

CONTACT INFORMATION

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

Gary Moore
Arkansas Foundation for Medical Care
401 West Capitol, Suite 508
Little Rock, AR 72201
Work Phone: (501) 375-5700
Fax: (501) 375-5705
Email: gmoore@afmc.org
Web: www.afmc.org

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 trademarks of their respective companies.