



Make Your Macro Great from the Very Beginning

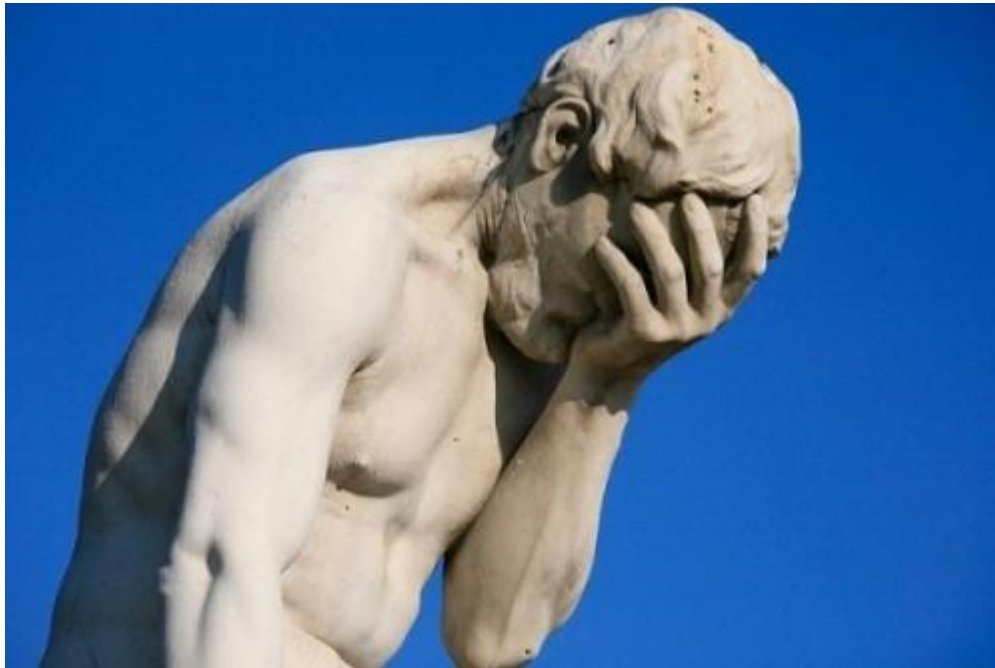
Yuliia Bahatska
Syneos Health

Vladlen Ivanushkin
DataFocus

PhUSE EU Connect 2018, Frankfurt

INTRODUCTION

This is me when I think of my first macro



THE COUNTW FUNCTION

```
%subgroup_analysis(data=data1, subgroup=subgroup1);  
%subgroup_analysis(data=data1, subgroup=subgroup2);  
%subgroup_analysis(data=data1, subgroup=subgroup3);  
...who knows how many more macro calls
```



```
%subgroup_analysis(data=data1, subgroup=s=subgroup1 subgroup2 subgroup3 ...);
```

THE COUNTW FUNCTION

```
%subgroup_analysis(data=data1, subgroup=subgroup1);  
%subgroup_analysis(data=data1, subgroup=subgroup2);  
%subgroup_analysis(data=data1, subgroup=subgroup3);  
...who knows how many more macro calls
```



```
%subgroup_analysis(data=data1, subgroup=s=subgroup1 subgroup2 subgroup3 ...);
```

```
%macro subgroup_analysis(data=, subgroups=);  
  %local subgroup i;  
  %do i=1 %to %sysfunc(countw(&subgroups));  
    %let subgroup=%scan(&subgroups, &i);
```

<Code that needs to be repeated for each value of subgroup.>

```
  %end;  
%mend subgroup_analysis;
```

THE COUNTW FUNCTION

```
%subgroup_analysis(data=data1, subgroup=subgroup1);  
%subgroup_analysis(data=data1, subgroup=subgroup2);  
%subgroup_analysis(data=data1, subgroup=subgroup3);  
...who knows how many more macro calls
```



```
%subgroup_analysis(data=data1, subgroup=subgroup1 subgroup2 subgroup3 ...);  
  
%macro subgroup_analysis(data=, subgroups=);  
  %local subgroup i;  
  %do i=1 %to %sysfunc(countw(&subgroups));    counts words in &subgroups  
    %let subgroup=%scan(&subgroups, &i); gets the &ith word from &subgroups  
    <Code that needs to be repeated for each value of subgroup.>  
  %end;  
%mend subgroup_analysis;
```

THE GETOPTION FUNCTION



THE GETOPTION FUNCTION

```
%local l_pagesize;
```

```
%let l_pagesize= %sysfunc(getoption(PAGESIZE));
```

save option

Do whatever you need with it, e.g.:

```
options pagesize=60;
```

Change it back:

```
options pagesize=&l_pagesize;
```



THE GETOPTION FUNCTION

```
%local l_pagesize;
```

```
%let l_pagesize= %sysfunc(getoption(PAGESIZE));
```

save option

Do whatever you need with it, e.g.:

```
options pagesize=60;
```

Change it back:

```
options pagesize=&l_pagesize;
```

```
%if &delete_temp_datasets=1 %then %do;
```

```
proc datasets memtype=data nolist nowarn;
```

```
delete _temp: <names of all macro-created temporary datasets>;
```

```
quit;
```

```
%end;
```



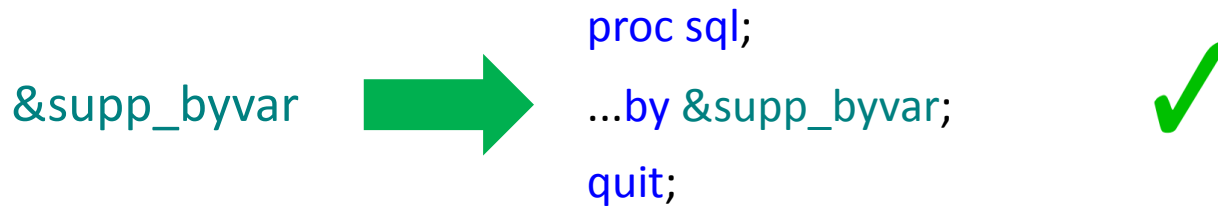
THE REGULAR EXPRESSIONS

CHANGING DELIMITERS IN A STRING

byvar=var1 var2 var3_2 var23



%let supp_byvar=%sysfunc(prxchange(s/\s+/%str(,)/, -1, &byvar));
supp_byvar=var1,var2,var3_2,var23



THE REGULAR EXPRESSIONS

SEARCHING WORDS IN A STRING

```
%if %sysfunc(findw(&string, WORD1)) or %sysfunc(findw(&string, WORD2)) or ...  
or %sysfunc(findw(&string, WORDN)) ...
```

OR

```
%let words=WORD1 WORD2 ... WORDN;  
  
%do counter=1 %to %sysfunc(countw(&words));  
  %let word=%scan(&words, &counter);  
  %if %sysfunc(findw(&string, &word)) %then %let wrd_ex_loop=YES;  
%end;
```



```
%if %sysfunc(prxmatch(/\b(WORD1|WORD2|...|WORDN)\b/, &string))
```

THE REGULAR EXPRESSIONS

SEARCHING WORDS IN A STRING

```
%if %sysfunc(findw(&string, WORD1)) or %sysfunc(findw(&string, WORD2)) or ...  
or %sysfunc(findw(&string, WORDN)) ...
```

OR

```
%let words=WORD1 WORD2 ... WORDN;  
  
%do counter=1 %to %sysfunc(countw(&words));  
  %let word=%scan(&words, &counter);  
  %if %sysfunc(findw(&string, &word)) %then %let wrd_ex_loop=YES;  
%end;
```



```
%if %sysfunc(prxmatch(/\b(WORD1|WORD2|...|WORDN)\b/, &string))
```

- ✓ more convenient
- ✓ more flexible
- ✓ less code
- ✓ easier to understand

THE REGULAR EXPRESSIONS

SUBTRACTION OF TWO STRINGS

```
%let text=WORD1 WORD2 WORD3 WORD4 WORD5;
```

```
%let words_to_ignore=WORD4 WORD1;
```

THE REGULAR EXPRESSIONS

SUBTRACTION OF TWO STRINGS

```
%let text=WORD1 WORD2 WORD3 WORD4 WORD5;
```

```
%let words_to_ignore=WORD4 WORD1;
```

```
/*Insert | symbols to use in regular expression*/
```

```
%let words_to_ignore_del=%sysfunc(prxchange(s/\s+|/, -1, &words_to_ignore));
```

THE REGULAR EXPRESSIONS

SUBTRACTION OF TWO STRINGS

```
%let text=WORD1 WORD2 WORD3 WORD4 WORD5;
```

```
%let words_to_ignore=WORD4 WORD1;
```

```
/*Insert | symbols to use in regular expression*/
```

```
%let words_to_ignore_del=%sysfunc(prxchange(s/\s+//, -1, &words_to_ignore));
```

```
%put NOTE: Result - %cmpres(%sysfunc(prxchange(s/\b(&words_to_ignore_del)\b//, -1, &text)));
```

Exclude all words given in &words_to_ignore from &text

THE REGULAR EXPRESSIONS

SUBTRACTION OF TWO STRINGS

```
%let text=WORD1 WORD2 WORD3 WORD4 WORD5;
```

```
%let words_to_ignore=WORD4 WORD1;
```

```
/*Insert | symbols to use in regular expression*/
```

```
%let words_to_ignore_del=%sysfunc(prxchange(s/\s+//, -1, &words_to_ignore));
```

```
%put NOTE: Result - %cmpres(%sysfunc(prxchange(s/\b(&words_to_ignore_del)\b//, -1, &text)));
```

Exclude all words given in &words_to_ignore from &text

```
1      %let text=WORD1 WORD2 WORD3 WORD4 WORD5;
```

```
2      %let words_to_ignore=WORD4 WORD1;
```

```
3  
4      /*Insert | symbols to use in regular expression*/
```

```
5      %let words_to_ignore_del=%sysfunc(prxchange(s/\s+//, -1, &words_to_ignore));
```

```
6
```

```
7      %put NOTE: Result - %cmpres(%sysfunc(prxchange(s/\b(&words_to_ignore_del)\b//, -1, &text)));
```

```
NOTE: Result - WORD2 WORD3 WORD5
```



COMPLEX MACRO PARAMETERS

Sort several datasets in the same way with some exception, for instance:

- Datasets DATA1-DATA5 need to be sorted  datasets = DATA1 DATA2 DATA3 DATA4 DATA5
- DATA1, DATA2, DATA4 by VAR1 VAR2  sort_vars = VAR1 VAR2
- DATA3 by VAR1 VAR3  exception = DATA3: VAR1 VAR3#DATA5: VAR2 VAR4
- DATA5 by VAR2 VAR4

%sort_variables(

```
datasets = DATA1 DATA2 DATA3 DATA4 DATA5,  
sort_vars = VAR1 VAR2,  
exception = DATA3: VAR1 VAR3#DATA5: VAR2 VAR4)
```

COMPLEX MACRO PARAMETERS

```
%macro sort_variables(datasets=, sort_vars=, exception=);  
  <For each single word in &datasets:>  
    %if %index(&exception, &dataset. :) %then %do;  
      <If "word:" is contained in &exception then get respective sorting variables from &exception>  
  
      %let _sort_except=%sysfunc(prxchange(s/.*&dataset.://, 1, &exception));  
  
      %let sort_except=%scan(&_amp;sort_except, 1, #);  
  
    %end;  
  %else %let sort_except=;
```

COMPLEX MACRO PARAMETERS

```
%macro sort_variables(datasets=, sort_vars=, exception=);
```

```
  <For each single word in &datasets:>
```

```
  %if %index(&exception, &dataset. :) %then %do;
```

```
    <If "word:" is contained in &exception then get respective sorting variables from &exception>
```

```
    %let sort_except=%sysfunc(prxchange(s/.*&dataset.://, 1, &exception));
```

```
    DATA3: VAR1 VAR3#DATA5: VAR2 VAR4 ➡ VAR1 VAR3#DATA5: VAR2 VAR4
```

```
    %let sort_except=%scan(&_sort_except, 1, #);
```

```
    VAR1 VAR3#DATA5: VAR2 VAR4 ➡ VAR1 VAR3
```

```
%end;
```

```
%else %let sort_except=;
```

COMPLEX MACRO PARAMETERS

```
%macro sort_variables(datasets=, sort_vars=, exception=);
```

```
  <For each single word in &datasets:>
```

```
  %if %index(&exception, &dataset. :) %then %do;
```

```
    <If "word:" is contained in &exception then get respective sorting variables from &exception>
```

```
    %let sort_except=%sysfunc(prxchange(s/.*&dataset.://, 1, &exception));
```

```
    DATA3: VAR1 VAR3#DATA5: VAR2 VAR4 ➡ VAR1 VAR3#DATA5: VAR2 VAR4
```

```
    %let sort_except=%scan(&_sort_except, 1, #);
```

```
    VAR1 VAR3#DATA5: VAR2 VAR4 ➡ VAR1 VAR3
```

```
%end;
```

```
%else %let sort_except=;
```

```
proc sort data=&dataset out=&dataset._srt;
```

```
  by %sysfunc(coalescec(&sort_except, &sort_vars));
```

```
run;
```

```
%mend sort_variables;
```

MACRO QUOTING/UNQUOTING

MACRO VARIABLE NOT RESOLVED AT MACRO EXECUTION

```
%macro print_note(data=, footnote=text from the dataset - %nrstr(&studyid));  
  proc sql;  
    select studyid into: studyid from &data;  
  quit;  
  
  /* UNQUOTE should be used to resolve the previously quoted macro parameter */  
  %put NOTE: %unquote(&footnote);  
%mend print_note;
```

MACRO QUOTING/UNQUOTING

MACRO VARIABLE NOT RESOLVED AT MACRO EXECUTION

```
%macro print_note(data=, footnote=text from the dataset - %nrstr (&studyid));  
  proc sql;  
    select studyid into: studyid from &data;  
  quit;  
  
  /* UNQUOTE should be used to resolve the previously quoted macro parameter */  
  %put NOTE: %unquote(&footnote);  
%mend print_note;
```

Not resolved yet

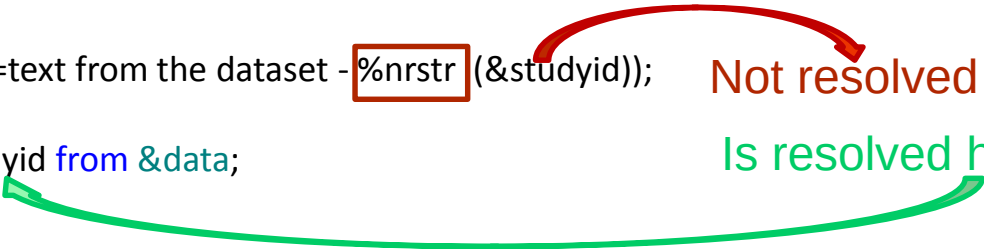
MACRO QUOTING/UNQUOTING

MACRO VARIABLE NOT RESOLVED AT MACRO EXECUTION

```
%macro print_note(data=, footnote=text from the dataset - %nrstr (&studyid));  
  proc sql;  
    select studyid into: studyid from &data;  
  quit;  
  
  /* UNQUOTE should be used to resolve the previously quoted macro parameter */  
  %put NOTE: %unquote(&footnote);  
%mend print_note;
```

Not resolved yet

Is resolved here



MACRO QUOTING/UNQUOTING

MACRO VARIABLE NOT RESOLVED AT MACRO EXECUTION

```
%macro print_note(data=, footnote=text from the dataset - %nrstr(&studyid));  
  proc sql;  
    select studyid into: studyid from &data;  
  quit;  
  
  /* UNQUOTE should be used to resolve the previously quoted macro parameter */  
  %put NOTE: %unquote(&footnote);  
%mend print_note;
```

Not resolved yet

Is resolved here

```
data data1;  
  studyid='12345';  
run;
```

```
%print_note(data=data1, footnote = text from the dataset - %nrstr(&studyid));
```

MACRO QUOTING/UNQUOTING

MACRO VARIABLE NOT RESOLVED AT MACRO EXECUTION

```
%macro print_note(data=, footnote=text from the dataset - %nrstr(&studyid));  
  proc sql;  
    select studyid into: studyid from &data;  
  quit;  
  
  /* UNQUOTE should be used to resolve the previously quoted macro parameter */  
  %put NOTE: %unquote(&footnote);  
%mend print_note;
```

Not resolved yet

Is resolved here

```
data data1;  
  studyid='12345';  
run;
```

```
%print_note(data=data1, footnote = text from the dataset - %nrstr(&studyid));
```

NOTE: text from the dataset – 12345

MACRO QUOTING/UNQUOTING

MACRO VARIABLE NOT RESOLVED AT MACRO EXECUTION

```
%macro print_note(data=, footnote=text from the dataset - %nrstr(&studyid));  
  proc sql;  
    select studyid into: studyid from &data;  
  quit;  
  
  /* UNQUOTE should be used to resolve the previously quoted macro parameter */  
  %put NOTE: %unquote(&footnote);  
%mend print_note;
```

Not resolved yet

Is resolved here

```
data data1;  
  studyid='12345';  
run;
```

```
%print_note(data=data1, footnote = text from the dataset - %nrstr(&studyid));
```

NOTE: text from the dataset – 12345

NOTE: text from the dataset - &studyid

MACRO QUOTING/UNQUOTING

MACRO VARIABLE NOT RESOLVED AT MACRO EXECUTION

```
%macro print_note(data=, footnote=text from the dataset - %nrstr(&studyid));  
  proc sql;  
    select studyid into: studyid from &data;  
  quit;  
  
  /* UNQUOTE should be used to resolve the previously quoted macro parameter */  
  %put NOTE: %unquote(&footnote);  
%mend print_note;
```

Not resolved yet
Is resolved here

```
data data1;  
  studyid='12345';  
run;
```

```
%print_note(data=data1, footnote = text from the dataset - %nrstr(&studyid));
```

NOTE: text from the dataset – 12345

NOTE: text from the dataset - &studyid

```
%print_note(data=data1, footnote = text from the dataset - %nrstr(&studyid));
```

WARNING: Apparent symbolic reference STUDYID not resolved.

MACRO PARAMETER CHECKS

❑ **In the beginning of the macro set some variable, i.e. param_error to 0:**

❑ **Perform the checks, i.e.**

Check whether a dataset/directory/library/format exists

Check whether a variable specified in a parameter is of required type. Like NUMBER, YESNO, CHAR etc.

Check if a variable exists in a dataset (if the source dataset is known) etc.

❑ **If a check fails then**

Provide informative message to the log for all failed checks

set param_error to 1 and

❑ **After all checks have run, see if any check failed and if yes, abort macro execution**

```
%if &param_error=1 %then %do;
```

```
    %put ERROR: The macro terminated due to errors in the macro parameters;
```

```
    %goto exit;
```

```
%end;
```

MACRO PARAMETER CHECKS

STANDARDIZATION OF MACRO PARAMETER VALUES

- ❑ Omitting not intended letter case issues ➡ %UPCASE
- ❑ Omitting redundant blanks ➡ %CMPRES
- ❑ Harmonizing values ➡ YES → Y, NO → N
- ❑ etc.

CONCLUSION

