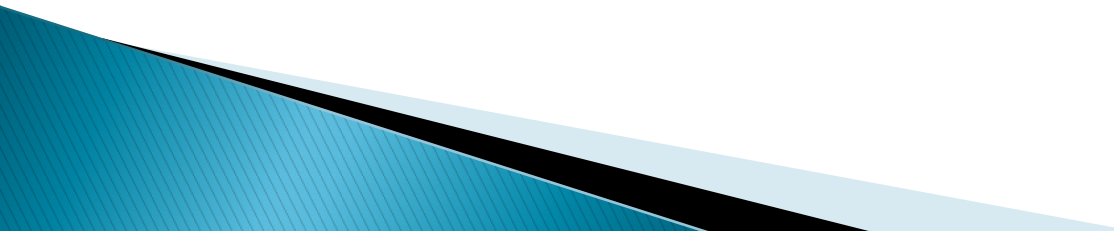


Macro Variables

and the Many Ways to Create Them

Sarah Berenbrinck
Coding Tips & Tricks CT13
PhUSE 2018

Things we will cover

- ▶ Macro variables
 - %let, %put, call symput, proc sql
 - ▶ Macro statements
 - positional parameters, keyword parameters
 - ▶ Local/Global
 - ▶ Usage
- 

Also in the paper

- ▶ There is more in the paper than can be presented in 8 minutes so I won't present all of my slides
- ▶ You can refer to the paper or come find me if I miss something that interested you.

Setup

► Create a dataset

```
data phuse1 ;  
    length year 8 city $9 country $8 ;  
    input year city $ country $ ;  
datalines ;  
2016 Barcelona Spain  
2017 Edinburgh Scotland  
2018 Frankfurt Germany  
;  
run ;
```

	year	city	country
1	2016	Barcelona	Spain
2	2017	Edinburgh	Scotland
3	2018	Frankfurt	Germany

Macro variables – %let

- ▶ Create a macro variable

```
%let testmacvar1=Frankfurt ;
```

Macro variables – %put

► View macro variable

```
%put &testmacvar1 ;
```

```
454      +%put &testmacvar1 ;  
Frankfurt
```

```
%put The problem value is &testmacvar1 ;
```

```
530      +%put The problem value is &testmacvar1 ;  
The problem value is Frankfurt
```

Macro variables – call symput

► Create a macro variable

```
data _null_ ;  
    call symput('testmacvar2', "Frankfurt") ;  
run ;
```

```
%put &testmacvar2 ;
```

```
611          +%put &testmacvar2 ;  
Frankfurt
```

Macro variables – call symput

► Use a dataset

```
data _null_ ;  
    set phuse1 ;  
    call symput(city, country) ;  
run ;  
  
%put &Barcelona &Edinburgh &Frankfurt ;
```

```
2630      +%put &Barcelona &Edinburgh &Frankfurt ;  
Spain      Scotland Germany
```

Macro variables – call symput

► Trim trailing spaces

```
data _null_ ;  
    set phuse1 ;  
    call symput(city, trim(country)) ;  
run ;
```

```
2630      +%put &Barcelona &Edinburgh &Frankfurt ;  
Spain      Scotland Germany
```

```
data _null_ ;  
    set phuse1 ;  
    call symputx(city, country) ;  
run ;
```

```
%put &Barcelona &Edinburgh &Frankfurt ;
```

```
2712      +%put &Barcelona &Edinburgh &Frankfurt ;  
Spain Scotland Germany
```

Macro variables – proc sql

► Create a macro variable

```
proc sql noprint ;  
    select city into :testmacvar4  
    from phuse1  
    where year=2018 ;  
quit ;
```

```
%put &testmacvar4 ;
```

```
987          +%put &testmacvar4 ;  
Frankfurt
```

Macro variables – proc sql

► Create a multiple macro variables

```
proc sql noprint ;  
    select city into :testmacvarsql1-:testmacvarsql3  
    from phuse1  
quit ;  
  
%put &testmacvarsql1 &testmacvarsql2 &testmacvarsql3 ;
```

```
1069      +%put &testmacvarsql1 &testmacvarsql2 &testmacvarsql3 ;  
Barcelona Edinburgh Frankfurt
```

Macro variables – proc sql

- ▶ Create a macro variable containing many values

```
proc sql noprint ;  
    select city into :testmacvarsql separated by ", "  
    from phuse1  
quit ;
```

```
%put &testmacvarsql ;
```

```
1151      +%put &testmacvarsql ;  
Barcelona, Edinburgh, Frankfurt
```

Macro statements

– positional parameters

► Create a macro

```
%macro printj(city, number);  
  title "output &number" ;  
  proc print data=phuse1(where=(city="&city")) ;  
  run ;  
%mend printj ;
```

```
%printj(Barcelona, 1) ;  
%printj(Edinburgh, 2) ;  
%printj(Frankfurt, 3) ;
```

output 1			
<u>Obs</u>	year	city	country
1	2016	<u>Barcelona</u>	Spain

Macro statements

– positional parameters

```
%print(4, Barcelona) ;
```



```
1982      +%print(4, Barcelona) ;
```

```
NOTE: No observations were selected from data set WORK.PHUSE1.
```

```
NOTE: There were 0 observations read from the data set WORK.PHUSE1.  
      WHERE city='4';
```

Macro statements

– keyword parameters

► Create a macro

```
%macro printk(city=, number=);  
  title "output &number" ;  
  proc print data=phuse1(where=(city="&city")) ;  
  run ;  
%mend printk ;
```

```
%printk(city=Barcelona, number=1) ;  
%printk(city=Edinburgh, number=2) ;  
%printk(city=Frankfurt, number=3) ;
```

output 1			
<u>Obs</u>	year	city	country
1	2016	<u>Barcelona</u>	Spain

Macro statements

– keyword parameters

```
%printk(number=4, city=Barcelona) ;
```



output 4			
<u>Obs</u>	year	city	country
1	2016	<u>Barcelona</u>	Spain

```
%printk(city=Barcelona) ;
```

output			
<u>Obs</u>	year	city	country
1	2016	<u>Barcelona</u>	Spain

Macro statements

– keyword parameters

► Define default values for parameters

```
%macro printm(city=Frankfurt, number=x) ;  
  title "output &number" ;  
  proc print data=phuse1 (where=(city="&city"))  
  ;  
  run ;  
%mend printm ;
```

output x			
<u>Obs</u>	year	city	country
3	2018	<u>Frankfurt</u>	Germany

output x			
<u>Obs</u>	year	city	country
1	2016	<u>Barcelona</u>	Spain

output 2			
<u>Obs</u>	year	city	country
2	2017	<u>Edinburgh</u>	Scotland

```
%printm() ;  
%printm(city=Barcelona) ;  
%printm(city=Edinburgh, number=2) ;
```

Macro statements

– mixed parameters

- ▶ If both positional and keyword parameters appear in a macro definition, positional parameters must come first

Local/Global

- ▶ Create a macro variable

```
%global testmacvar5 ;
```

```
%let testmacvar5=Amsterdam ;
```

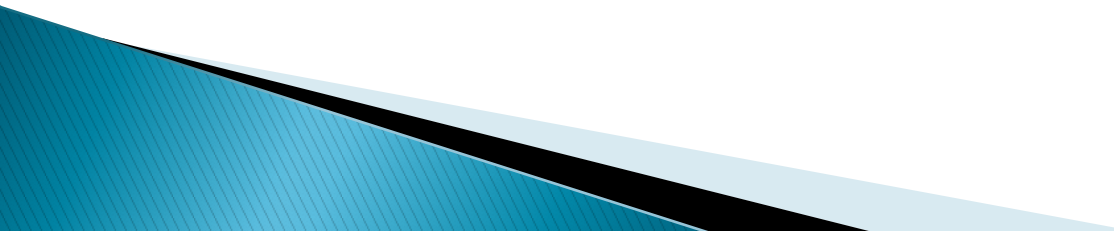
Local/Global

- ▶ Define a macro variable as global

```
%global testmacvar5 ;
```

```
%let testmacvar5=Amsterdam ;
```

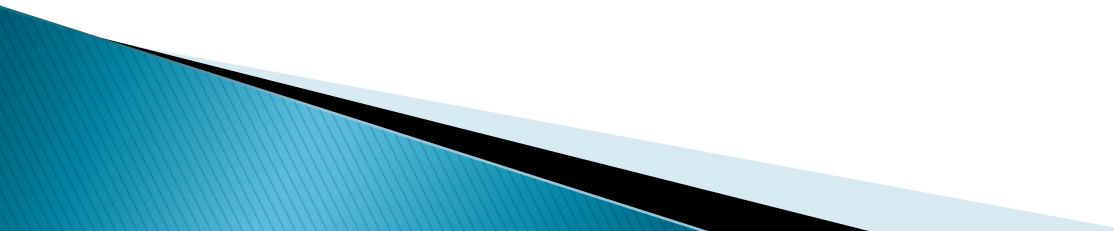
Usage – Macro Variables

- ▶ More dynamic code by creating macro variables to populate headers and footnotes.
 - Counting number of subjects into a macro variable and using this to populate headers.
 - Putting MedDRA version in a macro variable and using that in the footnote.
 - Putting a list of PT terms into a macro variable to add to a footnote.
- 

Usage – Macro Statements

- ▶ Increased efficiency and consistency by using macros to run bits of code multiple times.
 - Creating tables of the same structure and content on different subgroups.
 - Running counts or analysis on a number of different PT terms.
 - While debugging code you can wrap parts you don't want to run in a macro and not call that macro, *this should be removed from final code as it doesn't meet programming guidelines, as all unused code should be removed and macros like this are very confusing for reviewers.*

Usage – %put

- ▶ This can be used to check values assigned to macro variables.
 - ▶ You can also use %put to add notes to the log.
 - If you are running a long or complicated program you can flag certain points of the program to check timing or progress.
 - When using where or if statements you can write to the log details of records which don't meet expected criteria.
 - If you are having issues with how a program runs but struggle to identify where in the log, then you can add distinctive text to the log to help your search.
- 

Usage – %put

- ▶ This can be used to check values assigned to macro variables.
- ▶ You can also use %put to add notes to the log.
 - If you are running a long or complicated program you can flag certain points of the program to check timing or progress.
 - When using where or if statements you can write to the log details of records which don't meet expected criteria.
 - If you are having issues with how a program runs but struggle to identify where in the log, then you can add distinctive text to the log to help your search.
 - `%put cheese ;`

Usage – %put _all_ ;

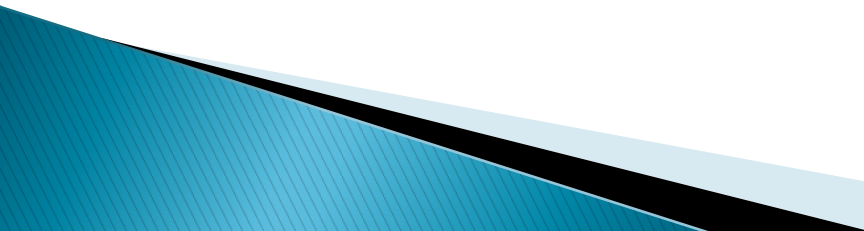
```
GLOBAL TESTMACROSQL1 Barcelona  
GLOBAL TESTMACROSQL2 Edinburgh  
GLOBAL TESTMACROSQL3 Frankfurt
```

```
AUTOMATIC SYSDATE 28SEP18  
AUTOMATIC SYSDATE9 28SEP2018  
AUTOMATIC SYSDAY Friday
```

```
AUTOMATIC SYSTEMTIME 10:19
```

```
AUTOMATIC SYSVER 9.4
```

```
SCSM SQLOBS 3
```



Usage – %put _all_ ;

```
GLOBAL TESTMACROSQL1 Barcelona  
GLOBAL TESTMACROSQL2 Edinburgh  
GLOBAL TESTMACROSQL3 Frankfurt
```

```
AUTOMATIC SYSDATE 28SEP18  
AUTOMATIC SYSDATE9 28SEP2018  
AUTOMATIC SYSDAY Friday
```

```
AUTOMATIC SYSTEMTIME 10:19
```

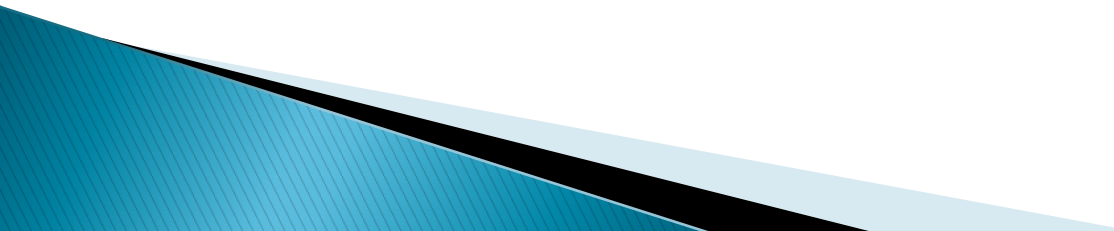
```
AUTOMATIC SYSVER 9.4
```

```
SCSM SQLOBS 3
```

```
SCSM _EXEC_USERFIRSTNAME Sarah
```

Conclusion

Macros and macro variables can be used in many ways to aid programming and efficiency. Care should be taken when using them as we work in an industry where traceability and reuse of code are important, so we need to take care that macros are well commented and easy to follow.



Any Questions



References

- ▶ <http://documentation.sas.com/?docsetId=mcrolref&docsetTarget=titlepage.htm&docsetVersion=9.4>
 - SAS 9.4 Macro Language: Reference,