TIPS AND TRICKS

FROM THE SNUG COMMITTEE

Scott Bass

Independent SAS Consultant

NSW Ministry of Health

I have built up a (semi) large collection of utility macros over the years

I have put them up on GitHub, and make them available for your use

Almost all are utility in nature, and not tied to any particular project

The macros I use ALL the time are:

**loop, seplist, parmv**,

check\_if\_empty, create\_format, dump\_mvars, fmtlist, get\_data\_attr, get\_lib\_attr,

hash\_define, hash\_lookup, kill, nobs, sendmail, squote, stp\_batch\_submit, varexist, varlist

<https://github.com/scottbass> , or

<https://github.com/scottbass/SAS/tree/master/Macro>

\* Note: works in SAS 9.3 ;

\* use the dlcreatedir option to create directories using the libname command ;

\* See also http://blogs.sas.com/content/sasdummy/2013/07/02/use-dlcreatedir-to-create-folders/ ;

**%let** root=\\my\root\dir;

options dlcreatedir;

\* you have to create each folder level separately ;

libname dummy "&root";

libname dummy "&root\Scott";

libname dummy "&root\Scott\DEV\_warehouse";

libname dummy "&root\Scott\DEV\_warehouse\02 Staging";

libname dummy "&root\Scott\DEV\_warehouse\02 Staging\one";

libname dummy "&root\Scott\DEV\_warehouse\02 Staging\one\two";

libname dummy clear;

\* however, using the pathname function makes this a little easier ;

libname dummy "&root";

libname dummy "%sysfunc(pathname(dummy))\Scott";

libname dummy "%sysfunc(pathname(dummy))\DEV\_warehouse";

libname dummy "%sysfunc(pathname(dummy))\02 Staging";

libname dummy "%sysfunc(pathname(dummy))\one";

libname dummy "%sysfunc(pathname(dummy))\two";

libname dummy clear;

\* which means you could use the %loop macro to shorten the code ;

**%let** path=Scott\DEV\_warehouse\02 Staging\one\two;

%macro ***code***;

libname dummy "%sysfunc(pathname(dummy))\&word";

%mend;

libname dummy "&root";

%***loop***(&path,dlm=\)

libname dummy clear;

SAS LOG:

24 \* which means you could use the %loop macro to shorten the code ;

25 %let path=Scott\DEV\_warehouse\02 Staging\one\two;

26 %macro code;

27 libname dummy "%sysfunc(pathname(dummy))\&word";

28 %mend;

29 libname dummy "&root";

NOTE: Libref DUMMY was successfully assigned as follows:

Engine: V9

Physical Name: \\my\root\dir

30 %loop(&path,dlm=\)

NOTE: Libref DUMMY was successfully assigned as follows:

Engine: V9

Physical Name: \\my\root\dir\Scott

NOTE: Libref DUMMY was successfully assigned as follows:

Engine: V9

Physical Name: \\my\root\dir\Scott\DEV\_warehouse

NOTE: Libref DUMMY was successfully assigned as follows:

Engine: V9

Physical Name: \\my\root\dir\Scott\DEV\_warehouse\02 Staging

NOTE: Libref DUMMY was successfully assigned as follows:

Engine: V9

Physical Name: \\my\root\dir\Scott\DEV\_warehouse\02 Staging\one

NOTE: Libref DUMMY was successfully assigned as follows:

Engine: V9

Physical Name: \\my\root\dir\Scott\DEV\_warehouse\02 Staging\one\two

31 libname dummy clear;

NOTE: Libref DUMMY has been deassigned.

\* You can use a single concatenated library ;

\* which makes a cleaner SAS log ;

libname dummy (

"&root\Scott",

"&root\Scott\DEV\_warehouse",

"&root\Scott\DEV\_warehouse\02 Staging",

"&root\Scott\DEV\_warehouse\02 Staging\one",

"&root\Scott\DEV\_warehouse\02 Staging\one\two",

);

libname dummy clear;

SAS LOG:

24 \* you can use a single concatenated library, ;

25 \* which makes a cleaner SAS log ;

26 libname dummy (

27 "&root\Scott",

28 "&root\Scott\DEV\_warehouse",

29 "&root\Scott\DEV\_warehouse\02 Staging",

30 "&root\Scott\DEV\_warehouse\02 Staging\one",

31 "&root\Scott\DEV\_warehouse\02 Staging\one\two",

32 );

NOTE: Libref DUMMY was successfully assigned as follows:

Levels: 5

Engine(1): V9

Physical Name(1): \\my\root\dir\Scott

Engine(2): V9

Physical Name(2): \\my\root\dir\Scott\DEV\_warehouse

Engine(3): V9

Physical Name(3): \\my\root\dir\Scott\DEV\_warehouse\02 Staging

Engine(4): V9

Physical Name(4): \\my\root\dir\Scott\DEV\_warehouse\02 Staging\one

Engine(5): V9

Physical Name(5): \\my\root\dir\Scott\DEV\_warehouse\02 Staging\one\two

33 libname dummy clear;

NOTE: Libref DUMMY has been deassigned.

\* You can also macroize this as well, although its a bit trickier ;

%macro ***code***;

**%global** concat; %\* this needs to be global to build up the string ;

**%if** (&\_\_iter\_\_ eq 1) **%then**

**%let** concat=&root\&word;

**%else**

**%let** concat=&concat\&word;

&concat|

%mend;

**%put** %loop(&path,dlm=\);

**%put** %seplist(%loop(&path,dlm=\),indlm=|,nest=qq);

SAS LOG:

24 \* however, you can also macroize this as well, ;

25 \* although its a bit trickier ;

26 %macro code;

27 %global concat; %\* needs to be global to build up the string ;

28 %if (&\_\_iter\_\_ eq 1) %then

29 %let concat=&root\&word;

30 %else

31 %let concat=&concat\&word;

32 &concat|

33 %mend;

34

35 %put %loop(&path,dlm=\);

\\my\root\dir\Scott| \\my\root\dir\Scott\DEV\_warehouse| \\my\root\dir\Scott\DEV\_warehouse\02 Staging|

\\my\root\dir\Scott\DEV\_warehouse\02 Staging\one| \\my\root\dir\Scott\DEV\_warehouse\02 Staging\one\two|

36

37 %put %seplist(%loop(&path,dlm=\),indlm=|,nest=qq);

"\\my\root\dir\Scott",

"\\my\root\dir\Scott\DEV\_warehouse",

"\\my\root\dir\Scott\DEV\_warehouse\02 Staging",

"\\my\root\dir\Scott\DEV\_warehouse\02 Staging\one",

"\\my\root\dir\Scott\DEV\_warehouse\02 Staging\one\two"

\* putting it all together ;

%macro concat\_libref(root,path,dlm=\,libref=dummy);

%macro ***code***;

**%global** concat;

**%if** (&\_\_iter\_\_ eq 1) **%then**

**%let** concat=&root\&word;

**%else**

**%let** concat=&concat\&word;

&concat|

%mend;

**%let** dlcreatedir=**%sysfunc**(getoption(dlcreatedir));

options dlcreatedir;

libname &libref (%***seplist***(%***loop***(&path,dlm=&dlm),indlm=|,nest=qq));

libname &libref clear;

options &dlcreatedir;

**%symdel** concat / nowarn;

%mend;

options nodlcreatedir;

%***concat\_libref***(&root,&path);

**%put** **%sysfunc**(getoption(dlcreatedir));

SAS LOG:

42 options nodlcreatedir;

43 %concat\_libref(&root,&path);

NOTE: Libref DUMMY was successfully assigned as follows:

Levels: 5

Engine(1): V9

Physical Name(1): \\my\root\dir\Scott

Engine(2): V9

Physical Name(2): \\my\root\dir\Scott\DEV\_warehouse

Engine(3): V9

Physical Name(3): \\my\root\dir\Scott\DEV\_warehouse\02 Staging

Engine(4): V9

Physical Name(4): \\my\root\dir\Scott\DEV\_warehouse\02 Staging\one

Engine(5): V9

Physical Name(5): \\my\root\dir\Scott\DEV\_warehouse\02 Staging\one\two

NOTE: Libref DUMMY has been deassigned.

44 %put %sysfunc(getoption(dlcreatedir));

NODLCREATEDIR

\* Finally, the SAS Administrator can restrict this functionality ;

\* Google "SAS documentation Restricted Options" ;

\* Lists which options you can restrict ;

**proc** options listrestrict;

**run**;

\* Lists which options are restricted ;

**proc** options restrict;

**run**;

\* discarding non unique keys just got easier in base SAS ;

\* Note: works in SAS 9.3 ;

**data** test;

infile cards;

input id @@;

cards;

1 2 3 3 3 4 5 6 6 7 8 8 9 10

;

**run**;

\* This requires the data to already be sorted ;

**data** test\_unique1 test\_nonunique1;

set test;

by id;

if ^(first.id and last.id) then output test\_nonunique1;

else output test\_unique1;

**run**;

\* This sorts the data and creates the desired output ;

\* in the same step ;

**proc** sort data=test out=test\_nonunique2 uniqueout=test\_unique2 nouniquekey;

by id;

**run**;

options nocenter;

title "Unique V1";

**proc** print data=test\_unique1;

**run**;

title "Unique V2";

**proc** print data=test\_unique2;

**run**;

title "NonUnique V1";

**proc** print data=test\_nonunique1;

**run**;

title "NonUnique V2";

**proc** print data=test\_nonunique2;

**run**;

title;





Note: This tip used with permission by Michael Raithel:

<https://www.linkedin.com/pulse/hack-323-sorting-character-variables-have-leading-numbers-raithel?articleId=6193756909741309952>

<http://michaelraithel.blogspot.com.au/2016/10/hack-323-sorting-character-variables.html>

\* create sample data ;

**data** have;

do num=15 to 1 by -1;

char=put(num,3.-L);

file=cats("FILE",char);

output;

end;

**run**;

options nocenter;

title "want1: by num";

**proc** sort data=have out=want1;

by num;

**run**;

**proc** print;**run**;

title "want2: by char";

**proc** sort data=have out=want2;

by char;

**run**;

**proc** print;**run**;

title "want3: by char linguistic";

**proc** sort data=have out=want3 sortseq=linguistic(numeric\_collation=on);

by char;

**run**;

**proc** print;**run**;

title "want4: by file";

**proc** sort data=have out=want4;

by file;

**run**;

**proc** print;**run**;

title "want5: by file linguistic";

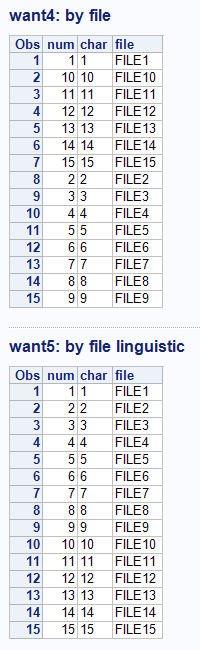
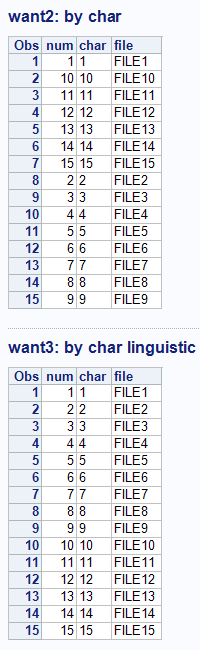
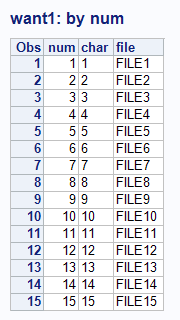
**proc** sort data=have out=want5 sortseq=linguistic(numeric\_collation=on);

by file;

**run**;

**proc** print;**run**;

title;



\* macro variables often need to be single quoted (apostrophe) ;

\* when working with RDBMS's ;

\* here are a few approaches... ;

\* first some sample data ;

%***libname\_sqlsvr***(libref=TMP,database=MYDB,schema=tmp)

\* drop the SQL Server table ;

**proc** datasets lib=tmp nowarn nolist;

delete class;

**quit**;

**data** tmp.class work.class;

set sashelp.class;

**run**;

\* option 1: macro quote the apostrophe ;

**%let** q=**%str**(%');

**%let** name=Jane;

\* this does not work! ;

**data** test;

set class;

where name=&q&name&q;

**run**;

\* but this does ;

**data** test;

set class;

where name=**%unquote**(&q&name&q);

**run**;

SAS LOG:

44 \* this does not work! ;

45 data test;

46 set class;

47 where name=&q&name&q;

NOTE: Line generated by the macro variable "Q".

47 'Jane'

\_

22

\_

76

ERROR: Syntax error while parsing WHERE clause.

ERROR 22-322: Syntax error, expecting one of the following: a name, a quoted string, a numeric constant,

a datetime constant, a missing value, (, \*, +, -, :, INPUT, NOT, PUT, ^, ~.

ERROR 76-322: Syntax error, statement will be ignored.

48 run;

NOTE: The SAS System stopped processing this step because of errors.

\* but if you are doing explicit passthrough, ;

\* SAS unquotes the value for you ;

**proc** sql;

connect using tmp;

create table test as

select \* from connection to tmp (

select \* from tmp.class where name=&q&name&q

);

**quit**;

SAS LOG:

24 \* but if you are doing explicit passthrough, SAS unquotes the value for you ;

25 proc sql;

26 connect using tmp;

ODBC: AUTOCOMMIT is YES for connection 2

27 create table test as

28 select \* from connection to tmp (

29 select \* from tmp.class where name=&q&name&q

30 );

ODBC\_13: Prepared: on connection 2

select \* from tmp.class where name='Jane'

ODBC\_14: Executed: on connection 2

Prepared statement ODBC\_13

NOTE: Compressing data set WORK.TEST increased size by 100.00 percent.

Compressed is 2 pages; un-compressed would require 1 pages.

NOTE: Table WORK.TEST created, with 1 rows and 5 columns.

\* option 2: use the %bquote function ;

\* same characteristics as before ;

\* this does not work! ;

**data** test;

set class;

where name=**%bquote**('&name');

**run**;

\* but this does ;

**data** test;

set class;

where name=**%unquote**(**%bquote**('&name'));

**run**;

\* but if you are doing explicit passthrough, ;

\* SAS unquotes the value for you ;

**proc** sql;

connect using tmp;

create table test as

select \* from connection to tmp (

select \* from tmp.class where name=**%bquote**('&name')

);

**quit**;

\* option 3: use the %tslit macro ;

\* Note: works in SAS 9.3 ;

\* I hate this hideous name, and the symbolgen output is "messy"... ;

\* but it works ;

\* See http://blogs.sas.com/content/sgf/2014/08/15/macro-quoting-made-easy/ ;

\* and https://communities.sas.com/t5/SAS-Communities-Library/Not-All-Macro-Language-Elements-Are-Supported-by-the-Macro/ta-p/223904 ;

options symbolgen;

**data** test;

set class;

where name=%***tslit***(&name);

**run**;

**proc** sql;

connect using tmp;

create table test as

select \* from connection to tmp (

select \* from tmp.class where name=%***tslit***(&name)

);

**quit**;

SAS LOG:

24 options symbolgen;

25 data test;

26 set class;

27 where name=%tslit(&name);

SYMBOLGEN: Macro variable NAME resolves to Jane

SYMBOLGEN: Macro variable VALUE resolves to Jane

SYMBOLGEN: Macro variable S1 resolves to '"

SYMBOLGEN: Some characters in the above value which were subject to macro quoting have been unquoted for printing.

SYMBOLGEN: Macro variable S2 resolves to "'

SYMBOLGEN: Some characters in the above value which were subject to macro quoting have been unquoted for printing.

SYMBOLGEN: Macro variable V1 resolves to Jane

SYMBOLGEN: Some characters in the above value which were subject to macro quoting have been unquoted for printing.

SYMBOLGEN: Macro variable V2 resolves to "Jane"

SYMBOLGEN: Some characters in the above value which were subject to macro quoting have been unquoted for printing.

SYMBOLGEN: Macro variable S2 resolves to "'

SYMBOLGEN: Some characters in the above value which were subject to macro quoting have been unquoted for printing.

SYMBOLGEN: Macro variable S1 resolves to '"

SYMBOLGEN: Some characters in the above value which were subject to macro quoting have been unquoted for printing.

SYMBOLGEN: Macro variable V3 resolves to 'Jane'

SYMBOLGEN: Some characters in the above value which were subject to macro quoting have been unquoted for printing.

28 run;

NOTE: There were 1 observations read from the data set WORK.CLASS.

WHERE name='Jane';

NOTE: The data set WORK.TEST has 1 observations and 5 variables.

\* option 4: use %squote macro ;

\* See https://communities.sas.com/t5/SAS-Communities-Library/Not-All-Macro-Language-Elements-Are-Supported-by-the-Macro/ta-p/223904 ;

\* and https://github.com/scottbass/SAS/blob/master/Macro/squote.sas ;

**data** test;

set class;

where name=%***squote***(&name);

**run**;

**proc** sql;

connect using tmp;

create table test as

select \* from connection to tmp (

select \* from tmp.class where name=%***squote***(&name)

);

**quit**;

options nosymbolgen;

SAS LOG:

24 data test;

25 set class;

26 where name=%squote(&name);

SYMBOLGEN: Macro variable NAME resolves to Jane

27 run;

NOTE: There were 1 observations read from the data set WORK.CLASS.

WHERE name='Jane';

\* directly from https://support.sas.com/rnd/base/datastep/perl\_regexp/regexp-tip-sheet.pdf ;

**data** phone\_numbers;

length first last phone $ 16;

input first last phone & $16.;

datalines;

Thomas Archer (919)319-1677

Lucy Barr 800-899-2164

Tom Joad (508) 852-2146

Laurie Gil (252)152-7583

;

data invalid;

set phone\_numbers;

where not prxmatch("/\([2-9]\d\d\) ?[2-9]\d\d-\d\d\d\d/",phone);

\* this would also work: \([2-9]\d{2}\) ?[2-9]\d{2}-\d{4} ;

**run**;

**data** \_null\_;

input;

\_infile\_ = prxchange('s/</&lt;/', -1, \_infile\_);

put \_infile\_;

datalines;

x + y < 15

x < 10 < y

y < 11

;

**run**;

/\* See:

https://regex101.com/

http://regexr.com/

and others...Google is your friend!

\*/

\* you no longer need to specify an upper bound for creating multiple macro variables in proc sql ;

\* Note: works in SAS 9.3 ;

**proc** sql noprint;

select distinct name

into :name1-

from sashelp.class;

**quit**;

**%let** myobs=&sqlobs;

**%put** &=myobs;

%***dump\_mvars***(name1 name2 name18 name19 name20 name21)

\* these are the respondent's answers to the three questions ;

**data** answers;

input id q1 $ q2 $ q3 $;

cards;

5 a b e

6 c d b

8 c b a

9 d a b

10 a a e

;

**run**;

\* this is the scoring table, with the score for each question/answer combination ;

**data** scoring (index=(qa=(q ans)));

input q ans $ score;

if missing(score) then delete;

cards;

1 a 1

1 b 2

1 c 3

1 d 4

2 a 4

2 b 3

2 c 2

2 d 1

3 a 1

3 b 2

3 c 3

3 d 4

;

**run**;

\* use do over arrays + index key lookup to match ;

\* the respondent's answers with the scoring table ;

\* Note: do over arrays is deprecated ;

\* See https://communities.sas.com/t5/General-SAS-Programming/array-and-do-over-question/m-p/301174, answer # 6 ;

**data** scored;

set answers;

array qarray (q) q1-q3;

array scarray (q) sc1-sc3;

do over qarray;

ans=qarray;

put q= ans=;

set scoring key=qa / unique;

if \_error\_=1 then do;

call missing(scarray);

put 'Score not found ' id= q= ans=;

\_error\_=0;

end;

else do;

scarray=score;

end;

end;

output;

drop score q ans;

**run**;

\* explicit rather than implicit arrays ;

**data** scored;

set answers;

array qarray (\*) q1-q3;

array scarray (\*) sc1-sc3;

do q=1 to dim(qarray);

ans=qarray(q);

put q= ans=;

set scoring key=qa / unique;

if \_error\_=1 then do;

call missing(scarray(q));

put 'Score not found ' (id q ans) (=);

\_error\_=0;

end;

else do;

scarray(q)=score;

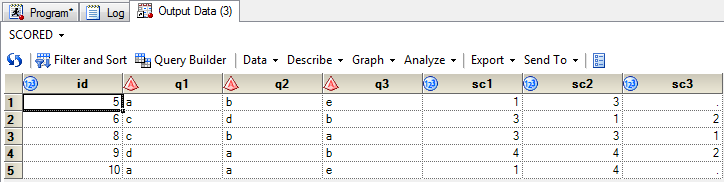
end;

end;

output;

drop score q ans;

**run**;



SAS LOG:

24 \* you no longer need to specify an upper bound ;

25 \* for creating multiple macro variables in proc sql ;

26 \* Note: works in SAS 9.3 ;

27 proc sql noprint;

28 select distinct name

29 into :name1-

30 from sashelp.class;

31 quit;

NOTE: PROCEDURE SQL used (Total process time):

real time 0.00 seconds

user cpu time 0.00 seconds

system cpu time 0.00 seconds

memory 508.87k

OS Memory 13860.00k

Timestamp 11/10/2016 07:32:57 PM

32 %let myobs=&sqlobs;

33 %put &=myobs;

MYOBS=19

34

35 %dump\_mvars(name1 name2 name18 name19 name20 name21)

================================================================================

name1 = Alfred

name2 = Alice

name18 = Thomas

name19 = William

name20 = \*\*\*MACRO VARIABLE UNDEFINED\*\*\*

name21 = \*\*\*MACRO VARIABLE UNDEFINED\*\*\*

================================================================================

\* specify dsoptions system option to get data step statistics ;

\* Note: undocumented option :-) ;

\* See http://www2.sas.com/proceedings/sugi29/040-29.pdf ;

\* and http://support.sas.com/resources/papers/proceedings16/2120-2016.pdf ;

options dsoptions = profile;

\* Note: this program stolen from http://support.sas.com/resources/papers/proceedings16/2120-2016.pdf ;

**data** test;

do \_n\_ = 1 to 1e7;

point\_row = 1+int(nobs\*ranuni(1));

set sampsio.empinfo nobs= nobs point= point\_row;

source= point\_row;

if index(lowcase(addr1), 'avenue') then do;

ila = 1;

fav = find(addr1, 'Avenue');

iav = index(addr1, 'Avenue');

fia = find(addr1, 'Avenue', 'i');

end;

if index(lowcase(addr1), 'street') then do;

ils = 1;

fst = find(addr1, 'Street');

ist = index(addr1, 'Street');

fis = find(addr1, 'Street', 'i');

end;

output;

end;

stop;

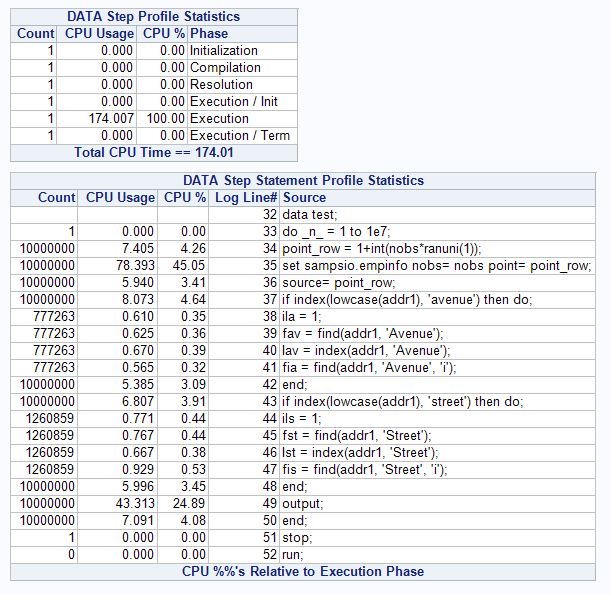
**run**;

options dsoptions=''; \* turn off dsoptions ;

\* list undocumented options with the undocumented INTERNAL option to proc options ;

**proc** options internal;

**run**;



# Merry Christmas

