A Time Saver for All: A SAS Macro Toolbox

Philip Jou
System Administrator
Office of Institutional Research
Baylor University

What Are Macros

- Macro Facility: Concepts to Understand
  - Macro Variable
    - Used to repeat values by using a single variable name
    - Scope can be either Local or Global
  - Macro Definition
    - Used to repeat custom code
  - Macro Function (Pre-Defined)
    - Built-in functions that can be utilized at any time
  - Positional/Keyword Parameters
    - Positional: Parameters passed into a macro definition in a specific order
    - Keyword: Explicit parameters that can be set to a value

What Are Macros

- Macro Definition: How to create/declare

```
%macro <name of macro>;
      /*Type in desired SAS code*/
%mend <name of macro>;
%<name of macro>; /*Ending ';' is not necessary*/
```

What Are Macros

- Positional Parameters
  - In this example var1, var2, and var3 are parameters that must be submitted in the order specified

```
%macro <name of macro>(var1,var2,var3);
      /*to use positional parameters call it as you would
      any macro variable using &var1, &var2, and &var3*/
%mend <name of macro>;
%<name of macro>(testVar1,testVar2,testVar3);
```

What Are Macros

- Keyword Parameters
  - In this example var1, var2, and var3 are parameters that have default values when the macro is run. A user can change the values by submitting the macro with <variable name>=<variable value>

```
%macro <name of macro>(var1=Y,var2=HI,var3=LOW);
      /*to use keyword parameters call it as you would
      any macro variable using &var1, &var2, and &var3*/
%mend <name of macro>;
%<name of macro>(var1=N,var2=UP,var3=DOWN);
```

CONVERT_ALLVAR_NUM_TO_CHAR

- Convert all numerical variables to character variables
- Proc Contents of Sample Dataset Created:

Variable	Type
ID	Num
col1/col5	Char
col2/col6	Char
col3/col7	Num
col4/col8	Num

CONVERT_ALLVAR_NUM_TO_CHAR

```
/*convert all numeric variables*/
%CONVERT_ALLVAR_NUM_TO_CHAR(DSET_CONVERT_NUM_TO_CHAR1);
```

Variable	Type (Before)	Type (After)
ID	Num	Char
col1	Char	Char
col2	Char	Char
col3	Num	Char
col4	Num	Char

CONVERT_ALLVAR_NUM_TO_CHAR

```
/*convert all numeric variables except ID*/
%CONVERT_ALLVAR_NUM_TO_CHAR(
      DSET_CONVERT_NUM_TO_CHAR2,IGNORE=ID);
```

Variable	Type (Before)	Type (After)
ID	Num	Num
col1	Char	Char
col2	Char	Char
col3	Num	Char
col4	Num	Char

CONVERT_ALLVAR_NUM_TO_CHAR

```
/*convert all numeric variables except ID & COL3 and create
copy of dataset with SUFFIX 'NEW'*/
%CONVERT_ALLVAR_NUM_TO_CHAR(
      DSET_CONVERT_NUM_TO_CHAR3,IGNORE=ID COL3,SUFFIX=NEW);
```

Variable	Type (Before)	Type (After)
ID	Num	Num
col1	Char	Char
col2	Char	Char
col3	Num	Num
col4	Num	Char

CONVERT_ALLVAR_NUM_TO_CHAR

```
/*process multiple datasets and convert all numeric variables except
ID, COL3, & COL8 and create copy of datasets with SUFFIX 'NEW'*/
%CONVERT_ALLVAR_NUM_TO_CHAR(DSET_CONVERT_NUM_TO_CHAR4
DSET_CONVERT_NUM_TO_CHAR5,IGNORE=ID COL3 COL8,SUFFIX=NEW);
```

DSET_CONVERT_NUM_TO_CHAR4			DSET_CONVERT_NUM_TO_CHAR5		
Variable	Type (Before)	Type (After)	Variable	Type (Before)	Type (After)
ID	Num	Num	ID	Num	Num
col1	Char	Char	col5	Char	Char
col2	Char	Char	col6	Char	Char
col3	Num	Num	col7	Num	Char
col4	Num	Char	col8	Num	Num

DELETE_EMPTY_COLUMNS

- Delete any column from the dataset that contains all missing values
- Sample Dataset created with two empty columns:

TWO_EMPTY_COLUMNS					
Obs #	ID (num)	col1 (char)	col2 (char)	col3 (num)	col4 (num)
1	1	1		1	.
2	2	2		2	.
3	3	3		3	.
4	4	4		4	.

DELETE_EMPTY_COLUMNS

```
/*remove all empty columns and create dataset with
SUFFIX 'DEL'*/
%delete_emtpy_columns(two_empty_columns,SUFFIX=DEL);
```

TWO_EMPTY_COLUMNS_DEL			
Obs #	ID (num)	col1 (char)	col3 (num)
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

DELETE_EMPTY_COLUMNS

```
/*remove all empty columns except COL2 and create
dataset with SUFFIX 'DEL_IGNORE'*/
%delete_emtpy_columns(
      two_empty_columns,IGNORE=COL2,SUFFIX=DEL_IGNORE);
```

TWO_EMPTY_COLUMNS_DEL_IGNORE				
Obs #	ID (num)	col1 (char)	col2 (char)	col3 (num)
1	1	1		1
2	2	2		2
3	3	3		3
4	4	4		4

DELETE_EMPTY_COLUMNS

```
/*remove all empty columns except numeric columns and
create dataset with SUFFIX 'DEL_CHAR'*/
%delete_emtpy_columns(
      two_empty_columns,CHARONLY=Y,SUFFIX=DEL_CHAR);
```

TWO_EMPTY_COLUMNS_DEL_CHAR				
Obs #	ID (num)	col1 (char)	col3 (num)	col4 (num)
1	1	1	1	.
2	2	2	2	.
3	3	3	3	.
4	4	4	4	.

DELETE_EMPTY_COLUMNS

```
/*remove all empty columns except character columns and
create dataset with SUFFIX 'DEL_NUM'*/
%delete_emtpy_columns(
      two_empty_columns,NUMONLY=Y,SUFFIX=DEL_NUM);
```

TWO_EMPTY_COLUMNS_DEL_NUM				
Obs #	ID (num)	col1 (char)	col2 (char)	col3 (num)
1	1	1		1
2	2	2		2
3	3	3		3
4	4	4		4

DELETE_MACRO_VARIABLES

- Delete macro variables from a space-delimited list
- Macro will use pre-defined functions %SYMEXIST and %SYMDEL
- Global Macros Created

```
%let testMacro1=1;
%let testMacro2=2;
%let testMacro3=3;
```

SASHELP.VMACRO		
SCOPE	NAME	VALUE
GLOBAL	TESTMACRO1	1
GLOBAL	TESTMACRO2	2
GLOBAL	TESTMACRO3	3

DELETE_MACRO_VARIABLES

```
/*delete macro variables testMacro1 and
testMacro3*/
%delete_macro_variables(testMacro1 testMacro3);
```

SASHELP.VMACRO		
SCOPE	NAME	VALUE
GLOBAL	TESTMACRO2	2

DIR_EXIST

- Determine if a File Directory Exists under a Windows Operating System
- Example Code:

```
%put Does the Directory Exist: 1=Yes, 0=No;
%put Does the Directory 'C:\WINDOWS\SYSTEM'
Exist: %DIR_EXIST(C:\WINDOWS\SYSTEM);
%put Does the Directory 'C:\NOT_REAL_DIRCTORY'
Exist: %DIR_EXIST(C:\NOT_REAL_DIRCTORY);
```

DIR_EXIST

- Log Output

Does the Directory Exist: 1=Yes, 0=No

Does the Directory ‘C:\WINDOWS\SYSTEM’ Exist: 1

Does the Directory ‘C:\NOT_REAL_DIRECTORY’ Exist: 0

GET_FILEPATH

- Returns the file path of the program the macro is run in and stores the value into a global macro variable
- Utilizes SAS Environment Variables SAS_EXECFILEPATH and SAS_EXECFILENAME
- Example Code and Log Output:

```
%get_filepath;
%put &=filepath;
%get_filepath(myMacroVar);
%put &=myMacroVar;
```

```
FILEPATH=E:\SAS Programs\Sample Code
MYMACROVAR=E:\SAS Programs\Sample Code
```

NUM_OBS

- Return the number of observations in a dataset
- Code is a play off of SAS Sample 24671
- Example Dataset and Code
  - Dataset created with 10 observations
  - Macro Call: %NUM_OBS(DSET_TEN_OBS)
  - Log Output: DSET_TEN_OBS has 10 Observations

VAR_EXIST

- Determines if a variable exists in a submitted dataset.
- Returns 1=Yes, 0=No
- Sample dataset created:

VAR_EXIST_DSET				
ID	col1 (char)	col2 (char)	col3 (num)	col4 (num)
1	1	1	1	1

VAR_EXIST

- Sample dataset created:

VAR_EXIST_DSET				
ID	col1 (char)	col2 (char)	col3 (num)	col4 (num)
1	1	1	1	1

- Sample Code and Log Output

```
%put Does the variable exist? 1=Yes, 0=No;
%put COL2: %var_exist(var_exist_dset,col2);
%put COL5: %var_exist(var_exist_dset,col5);
%put COL2: %var_exist(var_exist_dset2,col2);/*dataset does not exist,
warning will print to log and macro will return 0*/
```

```
Does the variable exist? 1=Yes, 0=No
COL2: 1
COL5: 0
COL2: 0
```

SYSTEM_OPTIONS

- Disable/Reset and set default system options that relate to printing to the log
- System Options Controlled:
  - SYMBOLGEN, MPRINT, MLOGIC, MERROR, SERROR, QUOTELENMAX, SOURCE, SOURCE2, NOTES, and VARLENCHK
- 3 Modes (DEFAULT, DISABLE, RESET)
  - DEFAULT: will set SAS to default environment settings
  - DISABLE: turn off System Options
  - RESET: set system options to state prior to DISABLE call

SYSTEM_OPTIONS

- Macro Function %SYSMEXECDEPTH is used to create unique global variable
- IMPORTANT NOTE: if a call to MODE=DISABLE is made, MODE=RESET should be called at some point at the same execution depth

SYSTEM_OPTIONS

Example Code:

```
OPTIONS MLOGIC MPRINT SYMBOLGEN; /*Enable system options*/
%macro example_sys_options;
       %let test1=TEST1;
       %put &test1;
       %SYSTEM_OPTIONS(DISABLE);
       %let test2=TEST2;
       %put &test2;
       /*Notice that SYMBOLGEN was not printed for resolving macro variable
       test2*/
       %SYSTEM_OPTIONS(RESET);
       %let test3=TEST3;
       %put &test3;
%mend example_sys_options;
%example_sys_options;
OPTIONS NOMLOGIC NOMPRINT NOSYMBOLGEN;
```

SYSTEM_OPTIONS

Log Output:

```
12   OPTIONS MLOGIC MPRINT SYMBOLGEN;

13   %example_sys_options;
MLOGIC(EXAMPLE_SYS_OPTIONS):  Beginning execution.
MLOGIC(EXAMPLE_SYS_OPTIONS):  %LET (variable name is TEST1)
MLOGIC(EXAMPLE_SYS_OPTIONS):  %PUT &test1
SYMBOLGEN:  Macro variable TEST1 resolves to TEST1
TEST1
MLOGIC(EXAMPLE_SYS_OPTIONS):  Beginning compilation of SYSTEM_OPTIONS using the autocall
      file [redacted]
MLOGIC(EXAMPLE_SYS_OPTIONS):  Ending compilation of SYSTEM_OPTIONS.
MLOGIC(SYSTEM_OPTIONS):  Beginning execution.
MLOGIC(SYSTEM_OPTIONS):  This macro was compiled from the autocall file
      [redacted]
MLOGIC(SYSTEM_OPTIONS):  Parameter MODE has value DISABLE
MLOGIC(SYSTEM_OPTIONS):  %LET (variable name is OPTIONS_LIST_VARLENGTH)
MPRINT(SYSTEM_OPTIONS):   options NOSYMBOLGEN NOMPRINT NOMLOGIC
TEST2
MPRINT(SYSTEM_OPTIONS):   MERROR SERROR QUOTELENMAX SOURCE NOSOURCE2 NOTES VARLENCHK=WARN;
MLOGIC(SYSTEM_OPTIONS):  Ending execution.
MPRINT(EXAMPLE_SYS_OPTIONS):   ;
MLOGIC(EXAMPLE_SYS_OPTIONS):  %LET (variable name is TEST3)
MLOGIC(EXAMPLE_SYS_OPTIONS):  %PUT &test3
SYMBOLGEN:  Macro variable TEST3 resolves to TEST3
TEST3
MLOGIC(EXAMPLE_SYS_OPTIONS):  Ending execution.
14   OPTIONS NOMLOGIC NOMPRINT NOSYMBOLGEN;
```

Notice that there is no ‘SYMBOLGEN: …’ statement as there is for TEST1 & TEST3

DSET_*LENGTH_MAX

- DSET_VAULELENGTH_MAX
  - Process a space-delimited list of datasets one by one and apply the maximum necessary “space” to store each variable
- DSET_VARLENGTH_MAX
  - Apply the maximum detected “space” to store matched variables from the comparison of datasets
  - Macro looks at the current space used by each variable
- DSET_LENGTH_MAX
  - Will call DSET_VALULELENTGH_MAX then DSET_VARLENGTH_MAX
- Default setting is to ONLY process character variables

DSET_*LENGTH_MAX

- Special Notes
  - All 3 macros use the SYSTEM_OPTIONS macro
  - There are system errors printed to the log if invalid options are submitted
  - Can be used to prep datasets for PROC COMPARE or DATA MERGE
    - Will eliminate any truncation of data concerns
  - Can save disk and/or memory space

DSET_*LENGTH_MAX

Example Code

```
data TEST_DSET_1;
        input ID col1 $ col2 $ col3 col4;
        datalines;
        1 1 1 1 1
        2 2 . 2 2
        3 3 3 3 .
        4 4 . 4 .
        ;
run;
data TEST_DSET_2;
        input ID col1 $ col2 $ col3 col4;
        datalines;
        1 111 1 11 1
        2 2 . 2 2
        3 33 3 33 .
        4 4 . 4 .
        ;
run;
```

TEST_DSET_1/TEST_DSET_2		
Variable	Type	Length
ID	Num	8
col1	Char	8
col2	Char	8
col3	Num	8
col4	Num	8

DSET_*LENGTH_MAX

```
%DSET_VALUELENGTH_MAX(
      TEST_DSET_1 TEST_DSET_2,SUFFIX=VALUE_DEFAULT);
```

			TEST_DSET_1	TEST_DSET_2
Variable	Type	Length (Before)	Length (After)	Length (After)
ID	Num	8	8	8
col1	Char	8	1	3
col2	Char	8	1	1
col3	Num	8	8	8
col4	Num	8	8	8

Reminder: Observation 1 for col1 has 3 characters (the max) in TEST_DSET_2

DSET_*LENGTH_MAX

```
%DSET_VALUELENGTH_MAX(TEST_DSET_1
      TEST_DSET_2,CHARONLY=N,SUFFIX=VALUE_ALL);
```

			TEST_DSET_1	TEST_DSET_2
Variable	Type	Length (Before)	Length (After)	Length (After)
ID	Num	8	3	3
col1	Char	8	1	3
col2	Char	8	1	1
col3	Num	8	3	3
col4	Num	8	3	3

Reminder: Observation 1 for col1 has 3 characters (the max) in TEST_DSET_2

DSET_*LENGTH_MAX

```
%DSET_VALUELENGTH_MAX(TEST_DSET_1
TEST_DSET_2,CHARONLY=N,NUMONLY=Y,SUFFIX=VALUE_NUMONLY);
/*Notice that CHARONLY=N since the default setting
within the macro function is CHARONLY=Y*/
```

			TEST_DSET_1	TEST_DSET_2
Variable	Type	Length (Before)	Length (After)	Length (After)
ID	Num	8	3	3
col1	Char	8	8	8
col2	Char	8	8	8
col3	Num	8	3	3
col4	Num	8	3	3

DSET_*LENGTH_MAX

```
%DSET_VARLENGTH_MAX(TEST_DSET_1
      TEST_DSET_2,SUFFIX=VAR_DEFAULT);
```

			TEST_DSET_1	TEST_DSET_2
Variable	Type	Length (Before)	Length (After)	Length (After)
ID	Num	8	8	8
col1	Char	8	8	8
col2	Char	8	8	8
col3	Num	8	8	8
col4	Num	8	8	8

DSET_*LENGTH_MAX

```
%DSET_VARLENGTH_MAX(TEST_DSET_1
      TEST_DSET_3,SUFFIX=VAR_DIFFLEN);
```

		TEST_DSET_1	TEST_DSET_3	TEST_DSET_1	TEST_DSET_3
Variable	Type	Length (Before)	Length (Before)	Length (After)	Length (After)
ID	Num	8	8	8	8
col1	Char	8	32	32	32
col2	Char	8	8	8	8
col3	Num	8	8	8	8
col4	Num	8	8	8	8

DSET_*LENGTH_MAX

```
%DSET_LENGTH_MAX(TEST_DSET_1
      TEST_DSET_2,SUFFIX=LENGTH_DEFAULT);
```

			TEST_DSET_1	TEST_DSET_2
Variable	Type	Length (Before)	Length (After)	Length (After)
ID	Num	8	8	8
col1	Char	8	3	3
col2	Char	8	1	1
col3	Num	8	8	8
col4	Num	8	8	8

Reminder: Observation 1 for col1 has 3 characters (the max) in TEST_DSET_2

DSET_*LENGTH_MAX

```
%DSET_LENGTH_MAX(TEST_DSET_1
TEST_DSET_2,CHARONLY=N,NUMONLY=N,SUFFIX=LENGTH_ALL);
```

			TEST_DSET_1	TEST_DSET_2
Variable	Type	Length (Before)	Length (After)	Length (After)
ID	Num	8	3	3
col1	Char	8	3	3
col2	Char	8	1	1
col3	Num	8	3	3
col4	Num	8	3	3

DSET_*LENGTH_MAX

```
%DSET_LENGTH_MAX(TEST_DSET_1
TEST_DSET_2,CHARONLY=N,NUMONLY=Y,SUFFIX=LENGTH_NUMONLY);
/*Notice that CHARONLY=N since the default setting
within the macro function is CHARONLY=Y*/
```

			TEST_DSET_1	TEST_DSET_2
Variable	Type	Length (Before)	Length (After)	Length (After)
ID	Num	8	3	3
col1	Char	8	8	8
col2	Char	8	8	8
col3	Num	8	3	3
col4	Num	8	3	3

Where to Find the Macro Toolbox

- Materials include Sample Program as well as in-depth documentation
- SUGIR Community (SAS Users Group for Institutional Research): https://communities.sas.com/t5/SUGIR-Community/SAS-Macro-Toolbox/gpm-p/257166 or https://goo.gl/ILXlnv
- Public Dropbox: https://goo.gl/SeqrZR

- Contact Information: Philip_Jou@baylor.edu