

P_MACRO

Parameters extraction from macros to SAS dataset

Eswara Gunisetti, PHUSE EU Connect 2024, Strasbourg
Stream: Coding Tips & Tricks (CT)
Presentation No. CT05



Disclaimer

The slides and views expressed in this presentation are those of speakers and are not representative of Cytel's views or opinions.



Agenda

- Key text definitions
- What does P_MACRO do?
- Why is needed?
- How to program?
- Limitations
- Conclusion
- Q & A





Key text Definitions

Macro :

Macros are compiled programs that you can call in a submitted SAS program

Macro parameters :

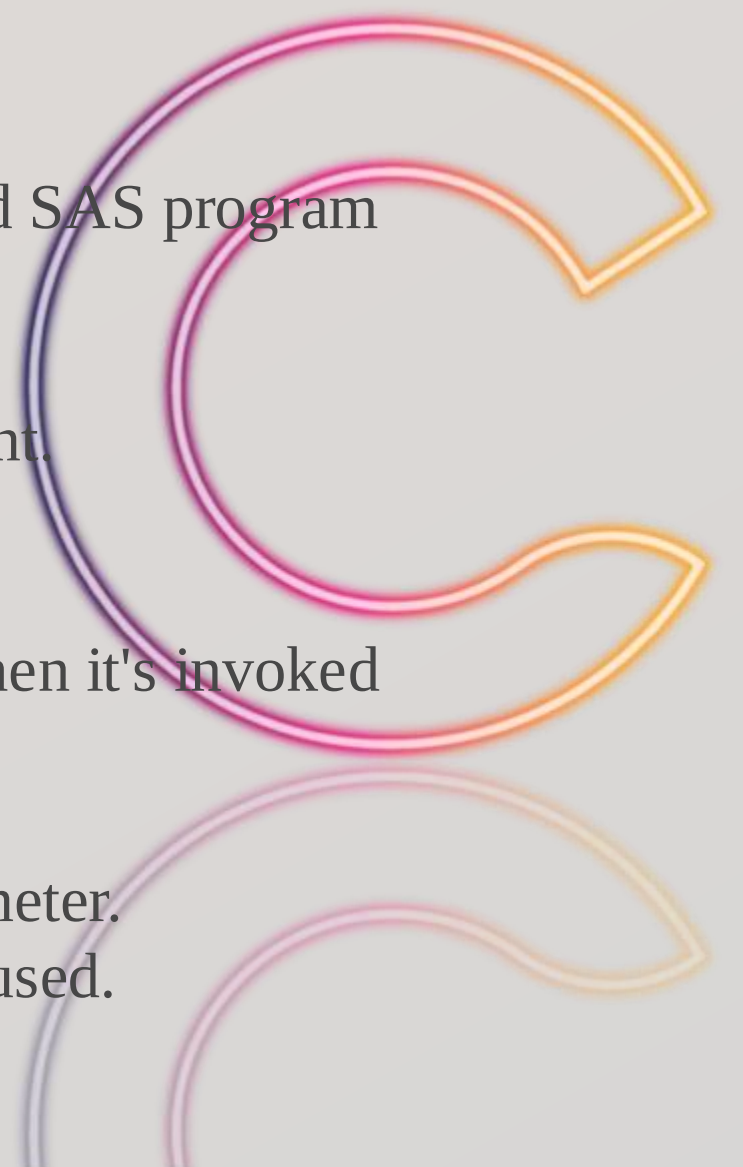
A variable defined in parentheses within a %MACRO statement.

Parameter use :

Macro parameters allow values to be passed into the macro when it's invoked

Default values :

A default value provided at the time of defining a macro parameter.
If not modified at the time of invocation default value will be used.



What does P_MACRO do?

- Extracts parameters from group of macro programs to SAS dataset
- Extracts Default values along with parameters if defined already
- Retains order of parameters
- Prioritizes main macro information over nested macros
- Provides a common text for the macros without parameters
- Generates automated macro call



Cytel What does P_MACRO do?

Program Files

Name	Date modified	Type	Size
CHRTONUM	10/7/2024 2:18 PM	SAS System Program	
DATE_IMPUTE	10/7/2024 2:18 PM	SAS System Program	
PRINT	10/7/2024 2:17 PM	SAS System Program	
QC_COMPARE	10/7/2024 2:17 PM	SAS System Program	
SORT	10/7/2024 2:18 PM	SAS System Program	

SAS Program

```
1 *-----*;
2 * Protocol: Macro_Testing
3 * Study : Testing
4 *
5 * Program name: QC_COMPARE.sas
6 * Programmer name : Eswara G
7 * SAS Version : 9.4
8 *
9 *-----*;
10
11 %macro qc_compare(prod=, qc=, criterion=%str(0.0001), whr =);
12
13 ***** Body of Macro *****;
14
15 %mend qc_compare ;
```

SAS Dataset

Table: WORK.P_MACRO View: Column labels Filter: (none)

Total rows: 4 Total columns: 4

	Macro Name	Macro Parameter Name	Macro Parameter Default Value	Parameters Order as defined in Macro
1	QC_COMPARE	prod		1
2	QC_COMPARE	qc		2
3	QC_COMPARE	criterion	%str(0.001)	3
4	QC_COMPARE	whr		4

Cytel

Why is needed ?

Query 1,

Is it mandatory to include all parameters while invocation?

```
1 %macro sort(DSN =%str(sashelp.cars), OUT =, OPT = , WHR =, VAR = ).  
2   proc sort data = &dsn out = &out. &opt.;  
3       &whr.;  
4       by &var.;  
5   run;  
6 %mend;
```

Not mandatory as default value already provided

Not mandatory to keep while invocation

```
CALL 1: %sort(DSN =%str(sashelp.cars), OUT =CLASS, OPT =NODUPRECS, WHR =%str(where ORIGIN="Asia"), VAR = MAKE MODEL TYPE)
```

```
CALL 2: %sort(OUT =CLASS, WHR =%str(where ORIGIN="Asia"), VAR = MAKE MODEL TYPE);
```

```
CALL 3: %sort(OUT =CLASS, VAR = MAKE MODEL TYPE);
```

**** Above 3 macros calls execute without any errors.**

Why is needed ? – Contd.

Query 2,

If not mandatory, then how a programmer knows

- I. List of parameters ?
- II. Order of parameters ?
- III. Default values if any ?

Query 3,

How to deal this in case of multiple macros ?

- ✓ Extract macro parameters and save to a SAS Dataset.
- ✓ This way it will be easy for a programmer to reference them at one place within SAS.
- ✓ This will give an edge to automatically create a macro call.

Cytel

How to program ?

1. Read macro program files to SAS
2. Extract macro name
3. Determine macro start & End points
4. Handle Nested macros
5. Handle macro programs with no parameters
6. Retain macro parameters position
7. Bring out Default values
8. Generate Macro call



How to program ? - Contd.

1. Read macro programs into SAS

- Use INFILE statement to read program files into SAS.

Sample Code to read the programs:

```
filename m_param "&path./macros/*.sas";
```

```
data m_param;
```

```
  infile m_param lrecl = 32760 truncover;
```

```
  input description $32760. ;
```

```
run;
```

*Make sure you read entire program codes.



How to program ? - Contd.

2. Extract Macro name

- Program header plays an important role in identifying Macro name.

```
1 *-----*;
2 * Protocol: Macro_Testing
3 * Study: Testing
4 *
5 * Program name: QC_COMPARE.sas
6 * Programmer Name: Eswara G
7 *
8 * SAS Version: 9.4
9 *-----*;
```

- GPP(Good programming practices) plays key role to code/program in identical way across the projects and skill levels.
- * Make sure program name and the macro name are same.

Cytel

How to program ? - Contd.

3. Determine Macro start & End points:

- Search for %MACRO to identify initiation point.
- Macro name provided after %MACRO
- Parameters provided in '()'

Important points to look for :

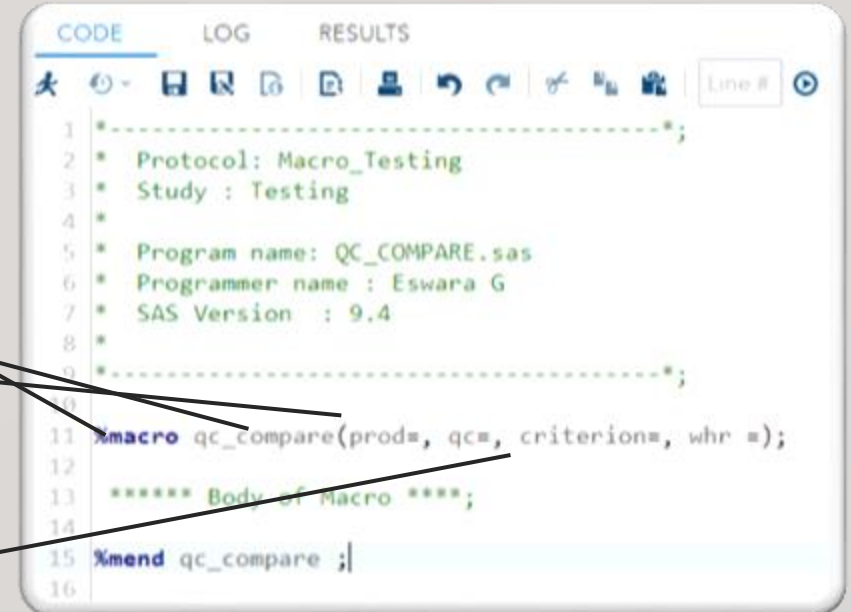
Are there any parameters defined ?

If defined, then

Are the parameters provided in
single line side by side/separate lines ?

Any commented text in between parameters

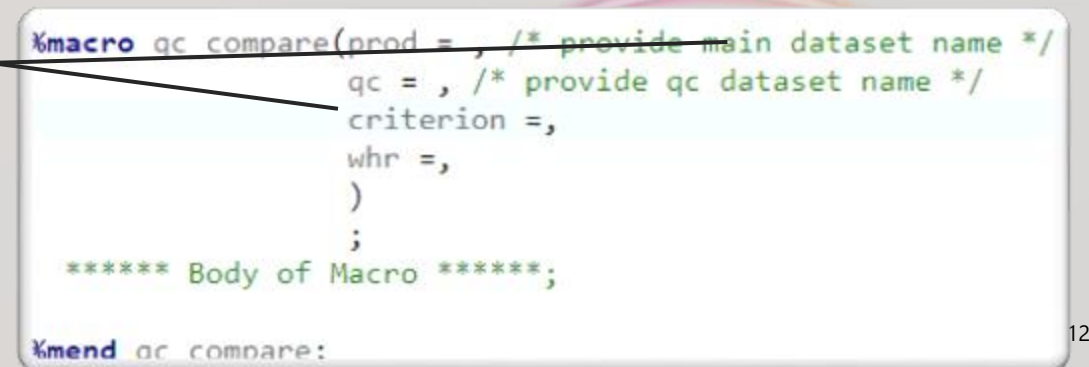
Identify end statement with key special characters
'); ' which by syntax ends the statement.



```
1 *-----*;
2 * Protocol: Macro_Testing
3 * Study : Testing
4 *
5 * Program name: QC_COMPARE.sas
6 * Programmer name : Eswara G
7 * SAS Version : 9.4
8 *
9 *-----*;
10
11 %macro qc_compare(prod=, qc=, criterion=, whr =);
12
13 ***** Body of Macro ****;
14
15 %mend qc_compare ;|
16
```

Arrows from the text '3. Determine Macro start & End points:' point to the following elements in the code:

- Arrow to line 11: `%macro` (initiation point)
- Arrow to line 11: `qc_compare` (macro name)
- Arrow to line 11: `(prod=, qc=, criterion=, whr =)` (parameters)



```
%macro qc compare(prod = , /* provide main dataset name */
                    qc = , /* provide qc dataset name */
                    criterion =,
                    whr =,
                    )
;
***** Body of Macro *****;

%mend qc compare;
```

Arrows from the text 'Important points to look for :' point to the following elements in the code:

- Arrow to line 1: `prod = ,`
- Arrow to line 2: `qc = ,`
- Arrow to line 3: `criterion =,`
- Arrow to line 4: `whr =,`
- Arrow to line 5: `)`

Cytel

How to program ? - Contd.

4. Handle Nested macros

Nested macros are macros written within macros.

In this case, once we identify the macro with %MACRO,

- match the macro name with program name identified from program header.

- This will help us to identify the correct macro naming and delete information related to nested macros.

* Make sure program name in header is correct.



```
-----*  
** Protocol: Macro_Testing  
** Study: Testing  
** Program name: QC_COMPARE.sas  
** Programmer Name: Eswara G  
** SAS Version: 9.4  
-----*  
  
%macro qc_compare(prod =, qc =, criterion =, whr =);  
***** Body of Macro *****;  
    %macro sort(dsn =, out =, opt = NODUPRECS, whr =, var = );  
        proc sort data = &dsn out = &out. &opt.;  
            &whr.;  
            by &var.;  
        run;  
    %mend;  
  
***** Body of Macro *****;  
    %macro print(dsn = %str(SASHELP.CLASS), opt =, var = );  
        proc print data = &dsn &opt.;  
            var &var.;  
        run;  
    %mend;  
%mend qc_compare;
```

How to program ? - Contd.

5. Handle macro programs with no parameters

It's important to identify files with no macro parameters defined and provide a consistent text **'No parameters defined'** for such cases.

```
-----*;  
: Protocol: Macro_Testing  
: Study: Testing  
:  
: Program name: IMPUTE_DATE.sas  
: Programmer Name: Eswara G  
:  
: SAS Version: 9.4  
:-----*;  
  
%macro impute_date();  
    ***** Body of Macro to do impurations *****;  
  
%mend impute_date;  
  
%impute_date;
```

How to program ? - Contd.

6. Retain Macro parameters position

Retaining their original position would support

- to keep them in order
- helps while automating the macro call.

7. Bring out Default values:

Extracting Default values would support programmer to

- have an idea about provided value
- decide whether to include them while invocation or skip

Look out for :

- I. Macro functions %STR, %NRSTR, %BQUOTE, etc
- II. Special characters such as “; &”
- III. Commented text provided
 - * Mask these special characters while extracting



Cytel

How to program ? - Contd.

8. Generate Macro call

- Simple retain statement would gather all parameters together
- Macro variable can support to print the call to log file

```
data autocall;  
  length autocall $32000;  
  set p_macro;  
  by macro name paramn;  
  retain autocall;  
  if param_d_val ne "" then m_parameter = strip(m_parameter)||"="||strip(param_d_val);  
  else m_parameter = strip(m_parameter)||"=";  
  if first.macro_name then autocall = m_parameter;  
  else autocall = strip(autocall)||","||strip(m_parameter);  
  if last.macro_name then autocall = 'macro('||strip(autocall)||")';"  
  if upcase(m_parameter) = "NO PARAMETERS DEFINED" then autocall = m_parameter;  
  if last.macro_name; * and autocall ne "";  
run;
```

```
data _null_;  
  set autocall;  
  by macro name paramn;  
  call symputx(scan(macro_name,1,"."),autocall);  
  *put _all_;  
run;
```

```
%put %bquote(&qc_compare.);
```

```
93  
94          %put %bquote(&qc_compare.);  
macro(prod=,qc=,criterion=0.001,whr=);
```

Cytel

Limitations

1. Default values provided with %LET statement

- %Let can be used to conditionally check and assign a default value
- In this case since no value is provided initially, no default value will be extracted

2. Adding PARMBUFF:

- Using **PARMBUFF** option and SYSPBUFF, you can define a macro that accepts a varying number of parameters at each invocation.
- SYSPBUFF resolves to the text supplied as parameter values in the invocation
- Providing a common text for all the macro files that use PARMBUFF would help.

* Watch out for other special keywords such as /SECURE, /STORE etc.,

```
%macro printz/parmbuff;  
  %let num=1;  
  %let dsname=%scan(&syspbuff,&num);  
  %do %while(&dsname ne);  
    proc print data=&dsname;  
      run;  
    %let num=%eval(&num+1);  
    %let dsname=%scan(&syspbuff,&num);  
  %end;  
%mend printz;
```

```
%printz(purple,red,blue,teal)
```

Cytel Conclusion

- SASHELP.VCOLUMN holds the information about metadata for libraries, datasets and Variables for the active session.
- Similarly, the resultant dataset from P_MACRO supports programmer to find list of macros within a folder, their complete list of parameters in defined order along with default values at one place within SAS.

SASHELP.VCOLUMN

	libname	memname	name	varnum
1	SASHELP	CLASS	Name	1
2	SASHELP	CLASS	Sex	2
3	SASHELP	CLASS	Age	3
4	SASHELP	CLASS	Height	4
5	SASHELP	CLASS	Weight	5
6	SASHELP	CLASSFIT	Name	1
7	SASHELP	CLASSFIT	Sex	2
8	SASHELP	CLASSFIT	Age	3
9	SASHELP	CLASSFIT	Height	4
10	SASHELP	CLASSFIT	Weight	5
11	SASHELP	CLASSFIT	predict	6

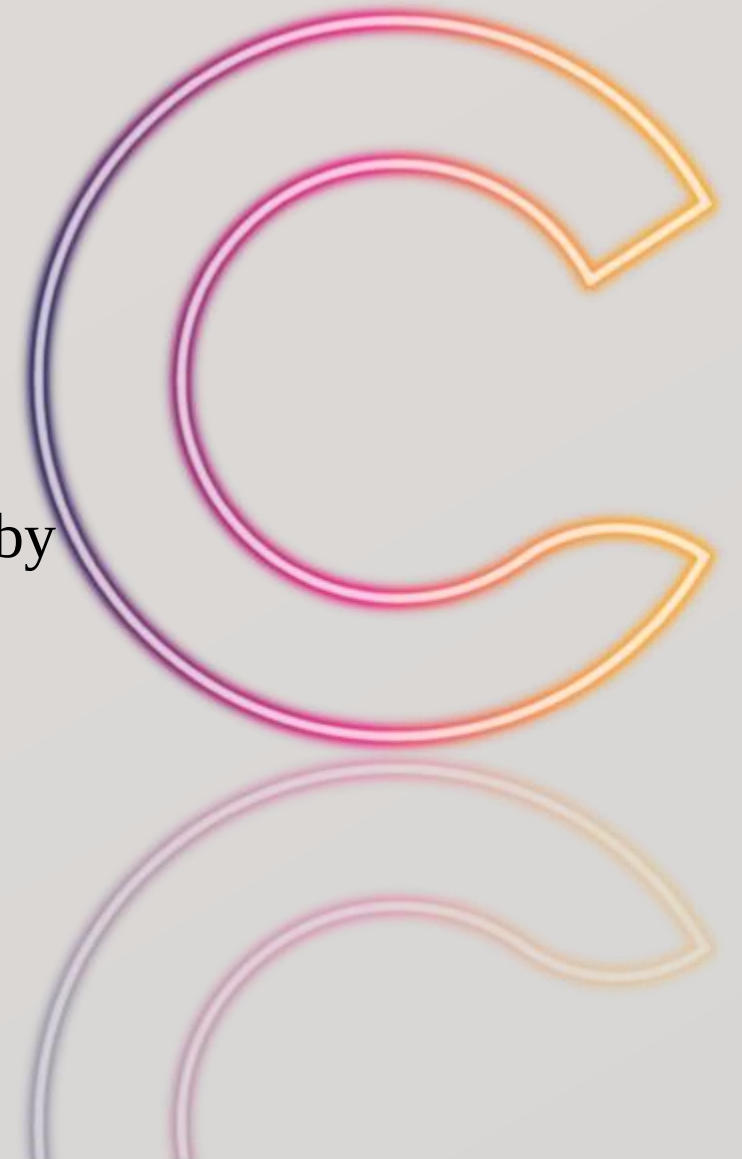
P_MACRO Dataset

Table:	WORK.P_MACRO	View:	Column labels	Filter: (none)
Total rows: 4 Total columns: 4				
	Macro Name	Macro Parameter Name	Macro Parameter Default Value	Parameters Order as defined in Macro
1	QC_COMPARE	prod		1
2	QC_COMPARE	qc		2
3	QC_COMPARE	criterion	%str(0.001)	3
4	QC_COMPARE	whr		4

Generating a macro call using the dataset would support programmer to have entire parameters list handy.

REFERENCES

- Carpenter, A. (2017). Building Intelligent Macros: Using Metadata Functions with the SAS® Macro Language.
- SAS Macro Programming Made Easy, Third Edition, Book by Michele M. Burlew



Any
Question



Cytel



Thank you!

Contact: eswara.gunisetti@cytel.com

