Key Elements to Develop SAS® Macros and Programs

Wei Cheng

IONIS® PHARMACEUTICALS

Outline of Topics

- Macro Documentation
- Macro Variables and Their Scopes
- Macro Processing and Timing
- Macro Quoting Functions
- Macro Functions

Outline of Topics

- Macro Documentation
- Macro Variables and Their Scopes
- Macro Processing and Timing
- Macro Quoting Functions
- Macro Functions

Outline of Topics

- Macro Documentation
- Macro Variables and Their Scopes
- Macro Processing and Timing
- Macro Quoting Functions
- Macro Functions

Outline of Topics

- Macro Documentation
- Macro Variables and Their Scopes
- Macro Processing and Timing
- Macro Quoting Functions
- Macro Functions

Outline of Topics

- Macro Documentation
- Macro Variables and Their Scopes
- Macro Processing and Timing
- Macro Quoting Functions
- Macro Functions

Outline of Topics

- Macro Storage
- Macro Debugging
- Macro Design
- General Techniques to Develop SAS Macros
- General Techniques to Develop SAS Programs

Outline of Topics

- Macro Storage
- Macro Debugging
- Macro Design
- General Techniques to Develop SAS Macros
- General Techniques to Develop SAS Programs

Outline of Topics

- Macro Storage
- Macro Debugging
- Macro Design
- General Techniques to Develop SAS Macros
- General Techniques to Develop SAS Programs

Outline of Topics

- Macro Storage
- Macro Debugging
- Macro Design
- General Techniques to Develop SAS Macros
- General Techniques to Develop SAS Programs

Outline of Topics

- Macro Storage
- Macro Debugging
- Macro Design
- General Techniques to Develop SAS Macros
- General Techniques to Develop SAS Programs

Macro Documentation

```
%*************************************************************************************;
%*  MACRO NAME:                                    AUTHOR: WEI CHENG    *;
%*                                                                       *;
%*  USAGE:                                         APRIL, 2001  SAS V8.2 *;
%*                                                                       *;
%*  DESCRIPTION:                                                         *;
%*                                                                       *;
%*   E.G.                                                                *;
%*                                                                       *;
%*  INPUT:  (+ ARE REQUIRED PARAMETERS)                                  *;
%*                                                                       *;
%*                                                                       *;
%*  OUTPUT:                                                              *;
%*                                                                       *;
%*  LIMITS:                                                              *;
%*                                                                       *;
%*  NOTES:                                                               *;
%*                                                                       *;
%*  HISTORY:                                                             *;
%*************************************************************************************;
```

Macro Documentation

1. Purpose
2. Macro Parameters
    - 2.1. Required Parameters
    - 2.2. Optional Parameters
3. Special Considerations
4. Restrictions
5. Input Data

……

Macro Documentation

Macro Style Comments

```
%macro macro_style_comments(indata);

%* This is the a macro style comment with a single quote, so it won't work;

data dummy;
        set  &indata;
        %* Another macro style comment;
        count = _n_;
run;

proc print data = dummy;
run;

%mend;

%macro macro_style_comments(indata);
```

Macro Documentation

Asterisk Style Comments

```
%macro asterisk_style_comments(indata);

* This is the an asterisk style comments with %let var = this would not work;

data dummy;
        set  &indata;
        count = _n_;
run;

proc print data = dummy;
run;

%mend;

%astrisk_style_comments(comments_test);
```

Macro Documentation

PL/1 Style Comments

```
%macro PL1_style_comments(indata);

/* This is the a PL/1 style comment with a single quote, so it'll work. */
 …
 %mend;

%PL1_style_comments(comments_test);


%macro PL1_style_comments2(indata);

/* This is the a PL/1 style comment with %let var = this will work*/
 …
 %mend;

%PL1_style_comments2(comments_test);
```

Macro Documentation

Writing Notes to SAS log

```
%macro put_notes_in_log(indata);

%if %bquote(&indata) eq %then %do;
        %put !!! PLEASE SPECIFY AN INPUT DATA !!!;
        %goto exit;
%end;

data dummy;
        set  &indata;          count = _n_;
run;

proc print data = dummy;       run;

%exit:

%mend;

%put_notes_in_log();
```

Macro variables and their scopes

- Automatic macro variables
- User-defined macro variables
- Global macro variables
- Local macro variables
- Symbol tables

Macro Processing and Timing

- Macro Definition and Compilation Time
- Macro Invocation and Execution Time
- SAS Step Parsing Time
- SAS Step Execution Time

Macro Processing and Timing

- Step Boundary
- Macro statement in DATA steps
- CALL SYMPUT(X) timing
- Triple-ampersand macro variables

Macro Processing and Timing

```
%macro main_analysis (indata);

        title "Analysis Result";
        proc freq data = &indata;    tables sex;

        %data_list(&indata)

%mend main_analysis;

%macro data_list(indata);

        title "Data Listing";
        proc print data = &indata;   run;

%mend data_list;

%main_analysis(sashelp.class);
```

Macro Processing and Timing

```
data dummy;
        set sashelp.class;
        if sex = "M" then do;
                %let g = Male;
        end;
        else do;
                %let g = Female;
        end;
        Gender = "&g";
run;
```

Macro Processing and Timing

```
data dummy;
        set sashelp.class;
        if sex = "M" then do;
                call symput ("g", "Male");
        end;
        else do;
                call symput ("g", "Female");
        end;
        Gender = "&g";
run;
```

Macro Processing and Timing

```
data dummy;
        set sashelp.class;
        if sex = "M" then do;
                call symput ("g", "Male");
        end;
        else do;
                call symput ("g", "Female");
        end;
        Gender = symget ("g");
run;
```

Macro Processing and Timing

```
data dummy;
        set sashelp.class end = last ;
        call symput ("name"||left(put(_n_,2.)), name);
        if last then call symputx ("numObs", put(_n_,2.));
run;

%macro show_triple_ampersand;

%do i = 1 %to &numobs;
        %put name&i is &&name&i ;
%end;

%mend;
%show_triple_ampersand;
```

Macro Quoting Functions

- %STR and %NRSTR
- %BQUOTE and %NRBQUOTE
- %SUPERQ and %UNQUOTE

Macro Functions

- %UPCASE and %LOWCASE
- %EVAL and %SYSEVALF
- %SYSFUNC
- …

Macro Storage

- %INCLUDE
- Autocall library
- Compiled stored macro

Macro Debugging

- SYMBOLGEN
- MPRINT
- MLOGIC
- MFILE
- %PUT

Macro Design

- The purpose and the condition of the macro need to be clear
- Only write a macro when it's needed
- If a macro grows too large, it's probably the time to break down it into separate pieces or write a new macro
- Build utility macro library, standard macro library
- The macro should be locally contained and readable, not depend on global macro variables or outside data
- Communicate with outside via parameters
- Don't reinvent the wheel

General Techniques to Develop Macros

- Create macro variable as local variable unless you need to define a global variable
- Check if the temporary data set exists before you create it
- Check if global variable exists before you create it
- Save original system options at the beginning of the macro, and restore them at the end of the macro
- Delete the temporary data sets created inside the macro if they are not needed any more
- Use keyword parameters
- Have a name convention for global macro variables
- Have a name convention for DATA step variables

General Techniques to Develop Macros

- Have a name convention for keyword parameters
- Explicitly capture errors or warnings in SAS log
- Describe result in SAS log when no output is created
- Comment the macro code
- Use a value instead of a reference, if a reference is used as a default for a parameter, convert it to a value immediately
- Build debugging tool and help documentation within the macro
- Avoid nested macros or calling outside macros
- Use extreme values to test parameters including quotes

General Techniques to Develop SAS Programs

- Write simple and elegant programs
- Write readable programs
- Write error-free programs
- Build a habit/pattern to write SAS programs
- Prevent over programming
- Keep improving your program code
- Go to SAS conferences, learn from other people, find your mentor
- Keep learning new skills and new features from new SAS versions

Conclusion

- Understand key elements
- Follow good programming practice
- Become a power SAS developer