SAS Macro #2
Round #2

JC Wang
Outline

1. More About Macros
   - SAS Macro Processing
   - Some Macro-related System Options
   - Stored Macro Files

2. Macros Statements, revisited
   - Macro Variables

3. PROC SQL Macro Facility, round 1
   - Using PROC SQL Macro Facility
SAS Macro Processing

SAS Program (input stack) → SAS Word Scanner (Tokenization) → non-macro (tokens) → SAS Compiler

SAS Word Scanner (Tokenization) → expanded tokens → Macro Facility

macro triggers (% and &)

expanded tokens
What Macros Can Do with Ease
revisited

- text substitution in double-quoted text strings
- communication across SAS step boundaries
- establishing default values
- conditional execution of SAS steps
- code hiding
What Macros Can Do with Ease
revisited

- text substitution in double-quoted text strings
- communication across SAS step boundaries
- establishing default values
- conditional execution of SAS steps
- code hiding
What Macros Can Do with Ease

- text substitution in double-quoted text strings
- communication across SAS step boundaries
- establishing default values
- conditional execution of SAS steps
- code hiding
What Macros Can Do with Ease

- text substitution in double-quoted text strings
- communication across SAS step boundaries
- establishing default values
- conditional execution of SAS steps
- code hiding
What Macros Can Do with Ease

revisited

- text substitution in double-quoted text strings
- communication across SAS step boundaries
- establishing default values
- conditional execution of SAS steps
- code hiding
What Macros Can Do with Ease

revisited

text substitution in double-quoted text strings
communication across SAS step boundaries
establishing default values
conditional execution of SAS steps
code hiding
SAS Macro Language for Users

to communicate with SAS macro processor

by string (sequence of characters) manipulation

- entire input to the macro language is a string
- usually strings are SAS codes
- macro processor manipulates strings and may send them back for further scanning
SAS Macro Language for Users

to communicate with SAS macro processor

by string (sequence of characters) manipulation

- entire input to the macro language is a string
- usually strings are SAS codes
- macro processor manipulates strings and may send them back for further scanning
SAS Macro Language for Users

to communicate with SAS macro processor

by string (sequence of characters) manipulation

- entire input to the macro language is a string
- usually strings are SAS codes

- macro processor manipulates strings and may send them back for further scanning
SAS Macro Language for Users

to communicate with SAS macro processor

by string (sequence of characters) manipulation

- entire input to the macro language is a string
- usually strings are SAS codes
- macro processor manipulates strings and may send them back for further scanning
SAS Macro Processing Example

SAS Program

%LET dsn = MYDATA;
DATA &dsn;
INFILE raw_file;
INPUT Name $ ht_in;
ht_cm = 2.54 * ht_in;
RUN;
PROC PRINT DATA=&dsn;
RUN;

SAS Compiler

1. Check syntax
2. If seeing % or &
3. Compile program

Macro Processor

symbol       value
DSN           MYDATA

Executable Program

DATA MYDATA;
INFILE raw_file;
INPUT Name $ ht_in;
ht_cm = 2.54 * ht_in;
RUN;
PROC PRINT DATA=MYDATA;
RUN;

joe 72
jane 67
jake 69
josh 75
System Options

- MPRINT shows the actual SAS code being generated
- MLOGIC traces the flow of macro code execution
- SYMBOLGEN used to resolve macro variable references
System Options

- **MPRINT** shows the actual SAS code being generated
- **MLOGIC** traces the flow of macro code execution
- **SYMBOLGEN** used to resolve macro variable references
System Options

- MPRINT shows the actual SAS code being generated
- MLOGIC traces the flow of macro code execution
- SYMBOLGEN used to resolve macro variable references
System Options

- MPRINT shows the actual SAS code being generated
- MLOGIC traces the flow of macro code execution
- SYMBOLGEN used to resolve macro variable references
Using Stored Macro Files

use of %INCLUDE

Create SAS macros in separate files and use them:
Suppose macros %mymacro1, %mymacro2, ... etc. are stored in f:\macros\mymacros.sas. Do the following

    %INCLUDE 'f:\macros\mymacros.sas';

Multiple %INCLUDE allowed.
Use it during development stage.
Using Stored Macro Files

use SAS Autocall Facility

- Store macros in an aggregate storage location containing members in individual files with matching names.
- Assume two aggregate storage locations e:\proj1 and f:\proj2.

```sas
FILENAME a 'e:\proj1';
FILENAME b 'f:\proj2';
OPTIONS MAUTOSOURCE SASAUTOS=a;
"OPTIONS MAUTOSOURCE SASAUTOS=(a b);
```

- Use Base SAS Software Autocall Library by simply specifying MAUTOSOURCE system option.
Using Stored Macro Files

*use SAS Autocall Facility*

- Store macros in an aggregate storage location containing members in individual files with matching names.
- Assume two aggregate storage locations e:\proj1 and f:\proj2.

```sas
FILENAME a 'e:\proj1';
FILENAME b 'f:\proj2';
OPTIONS MAUTOSOURCE SASAUTOS=a;
*OPTIONS MAUTOSOURCE SASAUTOS=(a b);
```

- Use Base SAS Software Autocall Library by simply specifying MAUTOSOURCE system option.
Using Stored Macro Files

use SAS Autocall Facility

- Store macros in an aggregate storage location containing members in individual files with matching names.
- Assume two aggregate storage locations e:\proj1 and f:\proj2.

FILENAME a 'e:\proj1';
FILENAME b 'f:\proj2';
OPTIONS MAUTOSOURCE SASAUTOS=a;
*OPTIONS MAUTOSOURCE SASAUTOS=(a b);

- Use Base SAS Software Autocall Library by simply specifying MAUTOSOURCE system option.
Using Stored Macro Files

use SAS Autocall Facility

- Store macros in an aggregate storage location containing members in individual files with matching names.
- Assume two aggregate storage locations e:\proj1 and f:\proj2.

  `FILENAME a 'e:\proj1';`
  `FILENAME b 'f:\proj2';`
  `OPTIONS MAUTOSOURCE SASAUTOS=a;`
  `*OPTIONS MAUTOSOURCE SASAUTOS=(a b);`

- Use Base SAS Software Autocall Library by simply specifying MAUTOSOURCE system option.
Outline

1. More About Macros
   - SAS Macro Processing
   - Some Macro-related System Options
   - Stored Macro Files

2. Macros Statements, revisited
   - Macro Variables

3. PROC SQL Macro Facility, round 1
   - Using PROC SQL Macro Facility
Global and Local Variables

- Variables inside a macro definition are local by default
- A %global statement makes variable values available outside macro definitions
Global and Local Variables

- Variables inside a macro definition are local by default
- A %global statement makes variable values available outside macro definitions
Global and Local Variables

- Variables inside a macro definition are local by default
- A %global statement makes variable values available outside macro definitions
Selected Automatic Macro Variables

- SYSDATE (SYSDATE9): date macro execution began in date7. (date9.) format
- SYSDAY: current day of the week
- SYSTIME: starting time of job
- SYSDSN/SYSLAST: name and libref of last data set processed, but in different formats
- SYSBUFF: all macro parameters passed
- SYSERR: return code status of SAS procedure
- SYSRC: return code status of system commands sent to the host environment
Selected Automatic Macro Variables

- **SYSDATE (SYSDATE9):** date macro execution began in date7. (date9.) format
- **SYSDAY:** current day of the week
- **SYSTIME:** starting time of job
- **SYSDSN/SYSLAST:** name and libref of last data set processed, but in different formats
- **SYSBUFF:** all macro parameters passed
- **SYSERR:** return code status of SAS procedure
- **SYSRC:** return code status of system commands sent to the host environment
Selected Automatic Macro Variables

- **SYSDATE (SYSDATE9)**: date macro execution began in date7. (date9.) format
- **SYSDAY**: current day of the week
- **SYSTIME**: starting time of job
- **SYSDSN/SYSLAST**: name and libref of last data set processed, but in different formats
- **SYSBUFF**: all macro parameters passed
- **SYSErr**: return code status of SAS procedure
- **SYSRC**: return code status of system commands sent to the host environment
Selected Automatic Macro Variables

- **SYSDATE (SYSDATE9):** date macro execution began in date7. (date9.) format
- **SYSDAY:** current day of the week
- **SYSTIME:** starting time of job
- **SYSDSN/SYSLAST:** name and libref of last data set processed, but in different formats
- **SYSBUFF:** all macro parameters passed
- **SYSERR:** return code status of SAS procedure
- **SYSRC:** return code status of system commands sent to the host environment
Selected Automatic Macro Variables

- **SYSDATE (SYSDATE9):** date macro execution began in date7. (date9.) format
- **SYSDAY:** current day of the week
- **SYSTIME:** starting time of job
- **SYNSDN/SYSLAST:** name and libref of last data set processed, but in different formats
- **SYSBUFF:** all macro parameters passed
- **SYSSERR:** return code status of SAS procedure
- **SYSRC:** return code status of system commands sent to the host environment
Selected Automatic Macro Variables

- **SYSDATE** (SYSDATE9): date macro execution began in date7. (date9.) format
- **SYSDAY**: current day of the week
- **SYSTIME**: starting time of job
- **SYSDSN/SYSLAST**: name and libref of last data set processed, but in different formats
- **SYSBUFF**: all macro parameters passed
- **SYSERR**: return code status of SAS procedure
- **SYSRC**: return code status of system commands sent to the host environment
Selected Automatic Macro Variables

- **SYSDATE (SYSDATE9):** date macro execution began in date7. (date9.) format
- **SYSDAY:** current day of the week
- **SYSTIME:** starting time of job
- **SYSDSN/SYSLAST:** name and libref of last data set processed, but in different formats
- **SYSBUFF:** all macro parameters passed
- **SYSERR:** return code status of SAS procedure
- **SYSRC:** return code status of system commands sent to the host environment
Selected Automatic Macro Variables

- **SYSDATE (SYSDATE9):** date macro execution began in date7. (date9.) format
- **SYSDAY:** current day of the week
- **SYSTIME:** starting time of job
- **SYSDSN/SYSLAST:** name and libref of last data set processed, but in different formats
- **SYSBUFF:** all macro parameters passed
- **SYSERR:** return code status of SAS procedure
- **SYSRC:** return code status of system commands sent to the host environment
Displaying Macro Variables

%PUT displays macro variables to SAS log at compile time

Syntax:

- `%PUT text macrovariables;`
- `%PUT _ALL_;`

Example: `%PUT *** &SYSDATE, &SYSDAY ***;`

in SAS log:

```plaintext
*** 05NOV09, Thursday ***
```
Displaying Macro Variables

%PUT displays macro variables to SAS log at compile time

Syntax:
- %PUT text macrovariables;
- %PUT _ALL_;

Example: %PUT *** &SYSDATE, &SYSDAY ***;

in SAS log:

 *** 05NOV09, Thursday ***
Outline

1. More About Macros
   - SAS Macro Processing
   - Some Macro-related System Options
   - Stored Macro Files

2. Macros Statements, revisited
   - Macro Variables

3. PROC SQL Macro Facility, round 1
   - Using PROC SQL Macro Facility
Using PROC SQL Macro Facility: An Example

Complete data set `all_data` contains clinical study outcomes from multiple sites

```sql
PROC SQL;
  SELECT LEFT(PUT(COUNT(DISTINCT site),3.))
    INTO:nsites FROM all_data;
    /* number of unique sites */
  SELECT DISTINCT site /* each site name */
    INTO:site1-:site&nsites FROM all_data;
  SELECT COUNT(*) /* number of observations per site */
    INTO:nobs1-:nobs&nsites FROM all_data
    GROUP BY site;
QUIT;
%PUT *** nsites=&nsites; /* display number of sites */
```