SAS Macro #2
Round #2

JC Wang

SAS Macro Processing

- SAS Program (input stack)
- SAS Word Scanner (tokenization)
- Macro triggers (% and &)
- Macro Facility
- expanded tokens
- non-macro (tokens)
- SAS Compiler
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 Processing Example

- 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
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.

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.
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
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 ***

Using PROC SQL Macro Facility: An Example

Complete data set all_data contains clinical study outcomes from multiple sites

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 */