

NOTE #9: MACRO

SAS Macros are powerful tool for your SAS program being more flexible and efficient. This chapter discusses the basic aspect of the SAS Macro via examples.

```
/* Example 9-1 */
OPTIONS MPRINT MACROGEN SYMBOLGEN SPOOL;

%MACRO CIF(P=,r=,n=);
data AA;
  Do I=1 to 20;
    Aval=&P*(1+&r/&n)**(&n*I); output;
  end;
run;

title "20 year balance with initial depoist of $ &P ";
title2 "at &n month compounding rate of &r";

Proc print label noobs;
format Aval dollar12.2;
label I = 'year' Aval = 'Balance';
run;

%MEND;
```

Note that the Macro starts with `%Macro name(input arguments)`; and ends with `%Mend;`. To invoke the macro, you may add the following at the end of the macro.

```
%CIF(P=3000,r=0.05,n=12);
```

However, you may want to save the macro program as a file in a folder and call whenever you need it. To do that, assuming the macro program is saved as `E:\mymacro\cif.sas` in a new program editor

```
%INCLUDE 'E:\mymacro\cif.sas';
%CIF(P=3000,r=0.05,n=12);

/* Example 9-2 */
OPTIONS MPRINT MACROGEN SYMBOLGEN;
Title;

%MACRO SIMUL(par=, n=, var=, out=);
*****
* This macro simulate random sample from Normal distribution
* Macro variable:
*   par : mean sd, for example par = 5 3
*   n : sample size
*   var : variable name for the random sample
*   out: the output data containing the random sample
*****;
```

```

data &out;
%LET mu=%SCAN(&par,1,' ');
%LET sigma=%SCAN(&par,2,' ');

  Do I=1 to &n;
    &var =&mu+ rannor(0)*&sigma; output;
  end;
run;
PROC univariate normal;
Histogram &var;

run;
%mend;

/* the macro is saved as 'E:\mymacro\simul.sas' */

%INCLUDE 'E:\mymacro\simul.sas';
%SIMUL(par=5 2 , n=100, var=X, out=sim);

/* Example 9-3 */

%MACRO TRIMMEAN(IN=,OUT=,var=, percent=, TM=);
/*****
* This Macro to calculate p% trimmed-mean
* MACRO Variable
* IN: Input dataset name
* OUT: Output dataset containing Trimmed Mean
* percent: Percent of data to be trimmed
* TM: Calculated Trimmed Mean in output data
*****/

%LET p1=%EVAL(&percent/2);
%LET p2=%EVAL(100-&p1);

PROC UNIVARIATE DATA=&IN NOPRINT; VAR &var;
  OUTPUT OUT=ONE PCTLPTS=&p1 &p2 PCTLPRE=P_;
RUN;

DATA TWO; SET &IN; IF _N_=1 THEN SET ONE;
  IF P_&p1<&var<P_&p2 THEN Y=&var;
PROC MEANS NOPRINT DATA=TWO; VAR Y;
  OUTPUT OUT=THREE MEAN=&TM;

DATA &OUT; SET THREE;
  VARIABLE="&var";
  DROP _TYPE_ _FREQ_;
RUN;
%MEND TRIMMEAN;

%INCLUDE 'E:\mymacro\simul.sas';
%SIMUL(par=5 2 , n=100, var=X, out=sim);

%TRIMMEAN (IN=sim, OUT=out, var=X, percent=10, TM=TRMean);
PROC PRINT DATA=out;
RUN;

```

Macro Functions *(extracted from SAS Macro Programming Made Easy, SAS Press)*

Macro character functions

%INDEX(*source*, *string*) returns the position in *source* of the first character of *string*.

%LENGTH(*string/text expression*) returns the length of *string* or the length of the results of the resolution of *text expression*.

%SCAN(*argument*, *n* <,*delimiters*>) returns the *n*th word in *argument* where the words in *argument* are separated by *delimiters*.

%SUBSTR(*argument*, *position* <,*length*>) extracts a substring of *length* characters from *argument* starting at *position*.

%UPCASE(*string/text expression*) converts *character string* or *text expression* to uppercase.

Macro Evaluation functions

%EVAL(*arithmetic expression/logical expression*) evaluates expressions using integer arithmetic.

%SYSEVALF(*arithmetic expression/logical expression* <,*conversion-type*>) evaluates expressions using floating point arithmetic

%SYSFUNC(*function(argument(s))* <,*format*>) executes SAS language *function* or user-written *function* and returns the results to the macro facility.

DATA step interface tools

SYMGET(*argument*) SAS language function that obtains the value of a macro variable specified as *argument* and returns this as a *character* value during DATA step execution.

SYMGETN(*argument*) SAS language function that obtains the value of a macro variable specified as *argument* and returns this as a *numeric* value.

CALL SYMPUT(*macro-variable*, *value*); SAS language routine that assigns *value* produced in a DATA step to a *macro-variable*. This routine does not trim leading and trailing blanks.

CALL SYMPUTX(*macro-variable*, *value* <,*symboltable*>); SAS language routine that assigns *value* produced in a DATA step to a *macro-variable*. This routine removes both leading and trailing blanks. Optionally, this routine can direct the macro processor to store the macro variable in a specific symbol table.

CALL EXECUTE(*argument*); SAS language routine that executes the resolved value of *argument*. Arguments that resolve to a macro facility reference execute immediately. Any SAS language statements resulting from the resolution are executed at the end of the step.

RESOLVE(argument) SAS language function that resolves *argument* during DATA step execution where *argument* is a text expression. Text expressions include macro variables and macro program calls.

```

/* some example */
OPTION MACROGEN SYMBOLGEN;

title "Report for %sysfunc(date(),monname.) %sysfunc(date(),year.)";
Data mprac;
  %LET date = Today is &sysdate &sysday;
  %LET time= The time is &stime;
  CALL SYMPUT('Todaym', SYMGET('date') || "-" || SYMGET('time'));
*Todaym is a Macro variable;
  Todays=SYMGET('Todaym'); *Todays is a SAS variable;
  %put &todaym; *print the macro variable in LOG widow;
run;

%LET pr=%STR(PROC PRINT; RUN;);
&pr;

%LET a = 123456789 ;
%LET b = %SUBSTR(&a,2,3);
%LET c=%EVAL(&a-&b);
%LET d=%SYSEVALF(&a/&b);
%put &a &b &c &d;

*****.

/* Example 9-4 */
/* More examples */
%macro sales (IN=, group=, var= );
OPTIONS LINESIZE=75 PAGESIZE=54 NODATE PAGENO=1;
DM "output;clear;log;clear";
ODS RTF File="J:\STA475\ex9_4.rtf"; *this will make a pdf output file;
ODS Listing Close;

*** some summary stat ***;
PROC TABULATE DATA = &IN;
CLASS &group;
var &var;
table (&group ALL), &var*(sum mean)*f=10.2;
Keylabel ALL = 'Overall';
run;

PROC GCHART DATA = &IN;
  Title "Daily Sales Total";
  PIE &group / Coutline= black Percent = outside
          SUMVAR=&var;
RUN;

**** to make indices for the class variable ****;
PROC SORT data=&IN; by &group;

DATA Data1; set &IN; by &group;
ind+first.&group;
run;

**** to calculate the number of levels of the class variable ***;
PROC MEANS MAX DATA=DATA1 NOPRINT;

```

```

VAR ind;
OUTPUT out = out1 MAX=max;
DATA _NILL_; SET out1;
  CALL SYMPUT ('no_ind',max);
RUN;

**** this will generate separate data sets for each store;
DATA %DO i=1 %TO &no_ind;
  store&i
  %END;
  ;
  SET DATA1;
  %DO i=1 %TO &no_ind;
    IF ind=&i then output store&i;
  %END;
run;

%DO i=1 %TO &no_ind;
  PROC MEANS DATA=store&i;
    *PROC MEANS DATA=DATA1 (WHERE= ( ind = &i) ); *this will do the same;
  VAR &var; RUN;
%END;

ODS Listing;
ODS RTF Close;
%mend sales;

DATA sales (DROP = type p);
INPUT @1 type $ @;
IF type="@" THEN DO;
INPUT @3 Store $10.;
Delete;
END;
RETAIN Store;
ELSE IF type ne "@" THEN
INPUT p $1-3 d 4-6 unit 7-11 @12 price Dollar7.2;
SELECT (p);
WHEN ("011") prod = "CDR50";
WHEN ("012") prod = "DVDR-";
WHEN ("014") prod = "DVDDL";
WHEN ("017") prod = "CDR100";
WHEN ("020") prod = "USB2G";
WHEN ("021") prod = "USB8G";
OTHERWISE;
END;
sales=unit*price/d;
FORMAT price Dollar7.2 sales Dollar8.2;
LABEL prod="Product Name" d="Survey Duration" unit="Unit sold"
price="Unit price" sales="Daily Sales";
DATALINES;
@ Kenwood
0110300023601200
0120600065203650
0140300102504190
0170600150702160
0201200056203650
0210900023410100
@ Westside
0110300017801290
0120600025603470
0140300087204090
0170600180701960

```

```
0201200114803290
0210900040209900
@ SouthHill
0110300030700900
0120600037403750
0140300087404290
0170600099802045
0201200078403880
0210900041509990
```

```
;
```

```
RUN;
```

```
PROC PRINT LABEL U NOOBS DATA= sales;
```

```
RUN;
```

```
%sales (IN=sales, group=Store, var=sales);
```

```
Proc Print; run;
```

```
/* Example 9-5 More complex example (from Dr. J Deddens (U Cincinnati) note)*/
=====;
```

This handout writes a SAS macro to check for linearity in linear regression, by creating dummy CATi variables for the percentile groups of a independent variable (CHECK). Then it performs linear regression using the dependent variable (DEP), and the independent variables (CATi) together with the other independent variables (IND).

```
OPTIONS MACROGEN; *this will generate output helpful in finding errors;
```

```
%macro chklin(data=_last_, check=, n=5, dep=, ind=);
```

```
%*****;
```

```
Macro Parameters:
```

```
DATA - dataset to be used, defaults to most recently created
```

```
CHECK - variable to be checked for linearity
```

```
N - number of groups to break the CHECK variable into, default is 5
(quintiles)
```

```
DEP - the dependent variable
```

```
IND - the independent variables in the model, excluding CHECK
```

```
THANKS TO DAVID WALL OF NIOSH FOR HELPING WRITE THIS MACRO
```

```
*****;
```

```
/* This creates a macro variable that contains a list of all the percentiles;
```

```
%let percents=;
```

```
data _null_;
```

```
do i=0 to &n;
```

```
call symput('percents', symget('percents') ||'
```

```
' || trim(left(round(i*100/&n, .01))));
```

```
end;
```

```
/* Find the needed percentiles;
```

```
proc univariate data=&data noprint;
```

```
var &check;
```

```
output out=_stats pctlpre=p_ pctlpts=&percents;
```

```
*could print _stats;
```

```
proc print data=_stats;
```

```
run;
```

```
/* Transpose the dataset output by PROC UNIVARIATE so we can define the dummy
variables;
```

```
proc transpose data=_stats out=_temp1;
```

```
/* Find the names or the variables containing the percentiles and put them into
a macro variable;
```

```

%let names=;
data _null_; set _temp1;
  call symput('names', symget('names') || " " || trim(_name_));

%* Create the indicator variables;
%let categor=;
%let list=;
data _temp2;
  if _n_=1 then set _stats;
  set &data;
  if &check ne . then do;
    %do i=2 %to &n;
      cat&i = ( %scan(&names, &i, %str( )) < &check <= %scan(&names,
        &i+1, %str( )) );
      label cat&i="%upcase(&check) group &i";
      %let categor=&categor cat&i;
      %let list=&list "cat&i",;
    %end;
  end; keep &categor &dep &ind;

%* Run a model with the indicator variables in place of the
  variable to be checked for linearity;
proc REG data=_temp2 ;
  model &dep = &categor &ind;
  title "Checking %upcase(&check) for linearity";
  title2 "Dep Var = %upcase(&dep)";
  title3 "Covariates: %upcase(&ind)";
%endmacro;
%mend;
*=====;
*the following is a sample application of the macro;

DATA GNP4; SET SASHELP.GNP;

%CHKLIN(DATA=GNP4,CHECK=INVEST,DEP=GNP,IND=EXPORTS GOVT);
*notice I did not specify N= so N=5;
RUN;

*now I will do N=7;
%CHKLIN(DATA=GNP4,CHECK=INVEST,DEP=GNP,IND=EXPORTS GOVT,N=7);
RUN;

/* IN-CLASS 9-1

Redo Assignment 1 problem 2 using SAS Macro. Define the initial salary, annual
interest rate, compounding term, raise rate, contribution percentages as input
arguments.
*/

/* IN-CLASS 9-2
Write a SAS MACRO program to divide a continuous variable into K equally spaced
categories. Your MACRO should
i) read in a data set, a variable, and the number of categories
ii) compute the high and low values of the variable, and the width, for
example, if k=4 and high=60 and low=20 then width=(60-20)/4=10
iii) compute a variable whose value is the category the observations belongs in
iv) print the number of observations in each category
v) make a nice histogram of the resulting counts (with nice title, etc)

SHOW how to apply the MACRO to a random sample of size 200 from a Poisson
distribution with mean 30. Make six equally spaced categories.
*/

```