Week 10+ Class Activities

File:  week-19-02nov07-MACRO.doc

Directory (hp):
C:\baileraj\Classes\Fall 2007\sta402\handouts
Directory:  \\Muserver2\USERS\B\baileraj\Classes\sta402\handouts

* run the programs/macros to see the plots

MACRO LANGUAGE
* Why use macros?
* Macro variables – system-defined and user-defined
* Macros
* Macro parameters
* Macro functions
* Conditional execution and DO loops
* CALL SYMPUT

References:


Other refs …

Look at SAS Trainer kits for macro

Acknowledgments:
Thanks to Jim Deddens (UC) for sharing handouts that served as the starting place for a number of the illustrations contained herein (Deddens Handouts #17-20)

Macro variables

* reference anywhere in SAS program other than data lines
* use either inside or outside macro programs
* can be created by SAS (AUTOMATIC MACRO VARIABLES) or by program (USER-DEFINED MACRO VARIABLES)

* stored in GLOBAL or LOCAL symbol table

* macros are TEXT

* macro variables are valid SAS names

* double quotes (" ") to resolve macro variable ref - single quotes will not (‘ ’)

“options” for macro programming

* symbolgen = show values of macro variables
  = trace resolution of indirect macro references

* mprint = enable printing or processing messages in macros

* mlogic = traces execution of macro programs (debugging)

Creating/Assigning/Displaying macro variable values

\%LET macro-var-name = value;

E.g.  \%LET nobs = 10;

Referencing macro variable in program

   &macro-var-name

Text before macro variable reference

   text&macro-var-name

Text after macro variable reference

   &macro-var-name.text

Displaying the values of macro variables

\%PUT &macro-var-name;
\%PUT _automatic_;  
\%PUT _user_;      
\%PUT _all_;
E.g.

```sas
/* Set up macro variables for later use;
   - all variable are GLOBAL ;

   %LET myopts = nocenter nodate symbolgen mprint mlogic;
   %LET nobs = 10;
   %LET seed = 0;
   %LET mytitle = Random Exponential Data;
   %LET data_name = ran_exp_;
   %LET varname = Ran_var;

options &myopts;

%PUT Requested options = &myopts;

data &data_name.&nobs;
   do ii = 1 to &nobs;
      &varname = ranexp(&seed);
      output;
   end;
   drop ii;

proc print;
title "&mytitle";
   run;
```

From SASLOG file . . .

```
160   %LET myopts = nocenter nodate symbolgen mprint mlogic;
161   %LET nobs = 10;
162   %LET seed = 0;
163   %LET mytitle = Random Exponential Data;
164   %LET data_name = ran_exp_;
165   %LET varname = Ran_var;
166
167   options &myopts;
168
169   %PUT Requested options = &myopts;
SYMBOLGEN: Macro variable MYOPTS resolves to nocenter nodate symbolgen mprint mlogic
Requested options = nocenter nodate symbolgen mprint mlogic
170
SYMBOLGEN: Macro variable DATA_NAME resolves to ran_exp_
SYMBOLGEN: Macro variable NOBS resolves to 10
171   data &data_name.&nobs;
172   do ii = 1 to &nobs;
```
&varname = ranexp(&seed);
SYMBOLGEN: Macro variable VARNAME resolves to Ran_var
SYMBOLGEN: Macro variable SEED resolves to 0
output;
end;
drop ii;

NOTE: The data set WORK.RAN_EXP_10 has 10 observations and 1 variables.
NOTE: DATA statement used:
   real time         0.02 seconds
   cpu time          0.02 seconds

proc print;
SYMBOLGEN: Macro variable MYTITLE resolves to Random Exponential Data
title "&mytitle";
run;

NOTE: There were 10 observations read from the data set WORK.RAN_EXP_10.
NOTE: PROCEDURE PRINT used:
   real time         0.01 seconds
   cpu time          0.01 seconds

From OUTPUT file . . .

Random Exponential Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>Ran_var</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.37907</td>
</tr>
<tr>
<td>2</td>
<td>0.92264</td>
</tr>
<tr>
<td>3</td>
<td>0.94668</td>
</tr>
<tr>
<td>4</td>
<td>0.72873</td>
</tr>
<tr>
<td>5</td>
<td>4.42539</td>
</tr>
<tr>
<td>6</td>
<td>1.15740</td>
</tr>
<tr>
<td>7</td>
<td>3.44957</td>
</tr>
<tr>
<td>8</td>
<td>2.16389</td>
</tr>
<tr>
<td>9</td>
<td>0.11252</td>
</tr>
<tr>
<td>10</td>
<td>0.14274</td>
</tr>
</tbody>
</table>

/* 3 simple MACRO motivating examples from M. Hughes */

/* EXAMPLE 1: Using LET to define macro variables */

options symbolgen;

data ds1;
input y x @@;
cards;
20 15 23 19 22 19 28 25 ;
%let dataname=ds1;
%let response=y;
%let datadescript=a completely PHONY data set;

proc reg data=&dataname;
model &response=x;
title "Regression analysis of &datadescript";
run;

/* EXAMPLE 2:  A simple MACRO PROGRAM  */

options symbolgen;

data ds2;
input y1 x1 @@;
cards;
 20 11  23 19  22 13  32 25
;
data ds3;
input y2 x2 @@;
cards;
 21 12  29 16  22 15  22 25
;
*definition of the macro program;
%macro myregression(dataname=,response=,predictor=,datadescript=);
  proc reg data=&dataname;
  model &response=&predictor;
  title "Regression analysis of dataset &dataname: &datadescript";
  run;
%mend myregression;

*two invocations of macro code;
%myregression(dataname=ds2,response=y1,predictor=x1,datadescript=a completely PHONY data set)
%myregression(dataname=ds3,response=y2,predictor=x2,datadescript=yet ANOTHER completely phony data set)

quit;

*-----------------------------------------------------------------------------*;

*MORE EFFICIENT CODE;

/* EXAMPLE 3:  Processing multiple datasets with macro looping  */

options symbolgen;

data ds1;
input y x @@;
cards;
20 15  23 19  22 19  28 25
;data ds2;
input y x @@;
cards;
data ds3;
input y x @@;
cards;
21 12 29 16 22 15 22 25
;
*definition of the macro program;
%macro myregression(num_datasets=);
  %do i=1 %to &num_datasets;
    proc reg data=ds&i;
    model y=x;
    title "Regression analysis of dataset ds&i";
    run;
  %end;
%mend myregression;

*invocation of macro code;
%myregression(num_datasets=3)
quit;

Macro variable comments . . .

* resolved LEFT to RIGHT

* && resolved to &

* multiple leading ‘s cause macro processor to rescan reference until no more ‘s can be resolved

* either GLOBAL (defined by SAS or in open code or
  %global . . .) or %LOCAL (macro variable defined
  in macro program – deleted when macro program
  associated with this variable ends)

%LET n=1;
%LET var = class;
%LET class1 = statistics_class;
options symbolgen;

data;
  input statistics_class $ nstudents;
datalines;
regression 14
regression 21
sampling 5
sampling 4
;
The following code is executed:

```sas
proc sort; by &&&var&n;
proc print; by &&&var&n;
   var nstudents;
run;
```

From SASLOG file . . .

```
231  %LET n=1;
232  %LET var = class;
233  %LET class1 = statistics_class;
234  options symbolgen;
235
236  data;
237    input statistics_class $ nstudents;
238    datalines;

NOTE: The data set WORK.DATA4 has 4 observations and 2 variables.
NOTE: DATA statement used:
   real time           0.01 seconds
   cpu time            0.01 seconds

243  ;
244  proc sort;
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable VAR resolves to class
SYMBOLGEN:  Macro variable N resolves to 1
SYMBOLGEN:  Macro variable CLASS1 resolves to statistics_class
244!      by &&&var&n;

NOTE: There were 4 observations read from the data set WORK.DATA4.
NOTE: The data set WORK.DATA4 has 4 observations and 2 variables.
NOTE: PROCEDURE SORT used:
   real time           0.15 seconds
   cpu time            0.06 seconds

245  proc print;
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable VAR resolves to class
SYMBOLGEN:  Macro variable N resolves to 1
SYMBOLGEN:  Macro variable CLASS1 resolves to statistics_class
245!     by &&&var&n;
246    var nstudents;
247  run;

NOTE: There were 4 observations read from the data set WORK.DATA4.
NOTE: PROCEDURE PRINT used:
   real time           0.04 seconds
   cpu time            0.04 seconds
```
### Macro Programs

%macro prog-name;
   <text>
%mend prog-name;

* execute by submitting reference to macro program

%prog-name

* no semi-colon needed - NOT a SAS statement

Passing parameters to macro programs

1. Positional

%macro prog-name(var-pos-1, var-pos-2, . . . , var-pos-k)
   < text >
%mend prog-name;

2. Keyword

%macro prog-name(var1=default-value1, var2=default-value3, . . . vark=default-valuek)
   < text >
%mend prog-name;
3. Mix of Positional and Keyword

options macrogen mprint mlogic;

EXAMPLE: one_sample_ci_sim MACRO

%MACRO one_sample_ci_sim(Nsims=4000, Nobs=15,
   Distn=Normal, Seed=0, OUT=);
   %* one_sample_ci_sim;
   %* date: 25 oct 03;
   %* based on “SIM1” macro (written by J. Deddens);
   %*
   %* Determine coverage of one-sample t-based CI when;
   %* Sampling from Normal, Exponential or Uniform;
   %* Distributions;

   %global gnobs;
   %let gnobs=&Nobs;

   %global gnsims;
   %let gnsims = &Nsims;

data simulate;
   do i=1 to &Nsims;
      do k=1 to &Nobs;
         %if &Distn=Normal %then %do;
            X = RANNOR(&Seed);
            %let mu = 0;
         %end;
         %else %if &Distn=Exponential %then %do;
            X = RANEXP(&Seed);
            %let mu = 1;
         %end;
         %else %do;
            X = RANUNI(&Seed);
            %let mu = 0.5;
         %end;
         output;
      end;
   end;

proc sort data=simulate; by i;
proc means data=simulate noprint; by i; var x;
   output out=temp mean=xbar var=ssquare n=num;
data &out; set temp;
lower_m = xbar - tinv(.975, num-1) * sqrt(ssquare/num);
upper_m = xbar + tinv(.975, num-1) * sqrt(ssquare/num);

icover = (lower_m < &mu < upper_m);

%MEND one_sample_ci_sim;

/*
so the macro is now defined . . .
let’s try it out
*/

%one_sample_ci_sim(out=norm_sim)
%one_sample_ci_sim(Distn=Exponential, out=exp_sim)
%one_sample_ci_sim(Distn=Uniform, out=unif_sim)

options formdlim = "-";

proc means data=norm_sim mean;
title "T-based CI for one-sample";
title2 "Normal populations -- # observations = &gnobs";
title3 "(based on &gnsims simulation conditions)";
  var icover;
run;

proc means data=exp_sim mean;
title "T-based CI for one-sample";
title2 "Exponential populations -- # observations = &gnobs";
title3 "(based on &gnsims simulation conditions)";
  var icover;
run;

proc means data=unif_sim mean;
title "T-based CI for one-sample";
title2 "Uniform populations -- # observations = &gnobs";
title3 "(based on &gnsims simulation conditions)";
  var icover;
run;

SAS OUTPUT

--------------------------------------------------------------------------------------------
T-based CI for one-sample
Normal populations -- # observations = 15
(based on 4000 simulation conditions)

The MEANS Procedure

Analysis Variable : icover

Mean
0.9472500

T-based CI for one-sample
Exponential populations -- # observations = 15
(based on 4000 simulation conditions)

The MEANS Procedure

Analysis Variable : icover

Mean
0.9157500

T-based CI for one-sample
Uniform populations -- # observations = 15
(based on 4000 simulation conditions)

The MEANS Procedure

Analysis Variable : icover

Mean
0.9440000

SASLOG

624  %one_sample_ci_sim(out=norm_sim)
MLOGIC(ONE_SAMPLE_CI_SIM):  Beginning execution.
MLOGIC(ONE_SAMPLE_CI_SIM):  Parameter OUT has value norm_sim
MLOGIC(ONE_SAMPLE_CI_SIM): Parameter NSIMS has value 4000
MLOGIC(ONE_SAMPLE_CI_SIM): Parameter NOBS has value 15
MLOGIC(ONE_SAMPLE_CI_SIM): Parameter DISTN has value Normal
MLOGIC(ONE_SAMPLE_CI_SIM): Parameter SEED has value 0
MLOGIC(ONE_SAMPLE_CI_SIM): %GLOBAL GNOBS
MLOGIC(ONE_SAMPLE_CI_SIM): %LET (variable name is GNOBS)
SYMBOLGEN: Macro variable NOBS resolves to 15
MLOGIC(ONE_SAMPLE_CI_SIM): %GLOBAL GNSIMS
MLOGIC(ONE_SAMPLE_CI_SIM): %LET (variable name is GNSIMS)
SYMBOLGEN: Macro variable NSIMS resolves to 4000
MPRINT(ONE_SAMPLE_CI_SIM): data simulate;
SYMBOLGEN: Macro variable NSIMS resolves to 4000
MPRINT(ONE_SAMPLE_CI_SIM): do i=1 to 4000;
SYMBOLGEN: Macro variable NOBS resolves to 15
MPRINT(ONE_SAMPLE_CI_SIM): do k=1 to 15;
SYMBOLGEN: Macro variable DISTN resolves to Normal
MLOGIC(ONE_SAMPLE_CI_SIM): %IF condition &Distn=Normal is TRUE
SYMBOLGEN: Macro variable SEED resolves to 0
MPRINT(ONE_SAMPLE_CI_SIM): X = RANNOR(0);
MLOGIC(ONE_SAMPLE_CI_SIM): %LET (variable name is MU)
MPRINT(ONE_SAMPLE_CI_SIM): output;
MPRINT(ONE_SAMPLE_CI_SIM): end;
MPRINT(ONE_SAMPLE_CI_SIM): end;

NOTE: The data set WORK.SIMULATE has 60000 observations and 3 variables.
NOTE: DATA statement used:
    real time 0.10 seconds
    cpu time 0.10 seconds

MPRINT(ONE_SAMPLE_CI_SIM): proc sort data=simulate;
MPRINT(ONE_SAMPLE_CI_SIM): by i;

NOTE: There were 60000 observations read from the data set WORK.SIMULATE.
NOTE: The data set WORK.SIMULATE has 60000 observations and 3 variables.
NOTE: PROCEDURE SORT used:
    real time 0.17 seconds
    cpu time 0.17 seconds

MPRINT(ONE_SAMPLE_CI_SIM): proc means data=simulate noprint;
MPRINT(ONE_SAMPLE_CI_SIM): by i;
MPRINT(ONE_SAMPLE_CI_SIM): var x;
MPRINT(ONE_SAMPLE_CI_SIM): output out=temp mean=xbar var=ssquare n=num;
SYMBOLGEN: Macro variable OUT resolves to norm_sim
NOTE: There were 60000 observations read from the data set WORK.SIMULATE.
NOTE: The data set WORK.TEMP has 4000 observations and 6 variables.
NOTE: PROCEDURE MEANS used:
   real time           0.10 seconds
   cpu time            0.10 seconds

MPRINT(ONE_SAMPLE_CI_SIM):   data norm_sim;
MPRINT(ONE_SAMPLE_CI_SIM):   set temp;
MPRINT(ONE_SAMPLE_CI_SIM):   lower_m=xbar-tinv(.975,num-1)*sqrt(ssquare/num);
MPRINT(ONE_SAMPLE_CI_SIM):   upper_m=xbar+tinv(.975,num-1)*sqrt(ssquare/num);
SYMBOLGEN:  Macro variable MU resolves to 0
MPRINT(ONE_SAMPLE_CI_SIM):   icover = (lower_m<0<upper_m);
MLOGIC(ONE_SAMPLE_CI_SIM):  Ending execution.
625  %one_sample_ci_sim(Distn=Exponential,out=exp_sim)
MLOGIC(ONE_SAMPLE_CI_SIM):  Beginning execution.
MLOGIC(ONE_SAMPLE_CI_SIM):  Parameter DISTN has value Exponential
MLOGIC(ONE_SAMPLE_CI_SIM):  Parameter OUT has value exp_sim
MLOGIC(ONE_SAMPLE_CI_SIM):  Parameter NSIMS has value 4000
MLOGIC(ONE_SAMPLE_CI_SIM):  Parameter NOBS has value 15
MLOGIC(ONE_SAMPLE_CI_SIM):  Parameter SEED has value 0
MLOGIC(ONE_SAMPLE_CI_SIM):  %GLOBAL  GNOBS
SYMBOLGEN:  Macro variable NOBS resolves to 15
MLOGIC(ONE_SAMPLE_CI_SIM):  %GLOBAL  GNSIMS
MLOGIC(ONE_SAMPLE_CI_SIM):  %LET (variable name is GNOBS)
SYMBOLGEN:  Macro variable NOBS resolves to 15
MLOGIC(ONE_SAMPLE_CI_SIM):  %GLOBAL  GNSIMS
MLOGIC(ONE_SAMPLE_CI_SIM):  %LET (variable name is GNSIMS)
SYMBOLGEN:  Macro variable NSIMS resolves to 4000

NOTE: There were 4000 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.NORM_SIM has 4000 observations and 9 variables.
NOTE: DATA statement used:
   real time           0.24 seconds
   cpu time            0.24 seconds

MPRINT(ONE_SAMPLE_CI_SIM):   data simulate;
SYMBOLGEN:  Macro variable NSIMS resolves to 4000
MPRINT(ONE_SAMPLE_CI_SIM):   do i=1 to 4000;
SYMBOLGEN:  Macro variable NOBS resolves to 15
MPRINT(ONE_SAMPLE_CI_SIM):   do k=1 to 15;
SYMBOLGEN:  Macro variable DISTN resolves to Exponential
MLOGIC(ONE_SAMPLE_CI_SIM):  %IF condition &Distn=Normal is FALSE
SYMBOLGEN:  Macro variable DISTN resolves to Exponential
MLOGIC(ONE_SAMPLE_CI_SIM):  %IF condition &Distn=Exponential is TRUE
SYMBOLGEN:  Macro variable SEED resolves to 0
X = RANEXP(0);

NOTE: The data set WORK.SIMULATE has 60000 observations and 3 variables.
NOTE: DATA statement used:
   real time           0.68 seconds
   cpu time            0.10 seconds

proc sort data=simulate;
by i;

NOTE: There were 60000 observations read from the data set WORK.SIMULATE.
NOTE: The data set WORK.SIMULATE has 60000 observations and 3 variables.
NOTE: PROCEDURE SORT used:
   real time           0.17 seconds
   cpu time            0.17 seconds

proc means data=simulate noprint;
by i;
var x;
output out=temp mean=xbar var=ssquare n=num;

proc sort data=simulate;
by i;
var x;
output out=temp mean=xbar var=ssquare n=num;

NOTE: There were 60000 observations read from the data set WORK.SIMULATE.
NOTE: The data set WORK.TEMP has 4000 observations and 6 variables.
NOTE: PROCEDURE MEANS used:
   real time           0.09 seconds
   cpu time            0.09 seconds

data exp_sim;
set temp;
lower_m=xbar-tinv(.975,num-1)*sqrt(ssquare/num);
upper_m=xbar+tinv(.975,num-1)*sqrt(ssquare/num);

NOTE: The data set WORK.TEMP has 4000 observations and 6 variables.
NOTE: PROCEDURE MEANS used:
   real time           0.09 seconds
   cpu time            0.09 seconds

data exp_sim;
set temp;
lower_m=xbar-tinv(.975,num-1)*sqrt(ssquare/num);
upper_m=xbar+tinv(.975,num-1)*sqrt(ssquare/num);

NOTE: The data set WORK.TEMP has 4000 observations and 6 variables.
NOTE: PROCEDURE MEANS used:
   real time           0.09 seconds
   cpu time            0.09 seconds
MLOGIC(ONE_SAMPLE_CI_SIM): Parameter NOBS has value 15
MLOGIC(ONE_SAMPLE_CI_SIM): Parameter SEED has value 0
MLOGIC(ONE_SAMPLE_CI_SIM): %GLOBAL  GNOBS
MLOGIC(ONE_SAMPLE_CI_SIM): %LET (variable name is GNOBS)
SYMBOLGEN: Macro variable NOBS resolves to 15
MLOGIC(ONE_SAMPLE_CI_SIM): %GLOBAL  GNSIMS
MLOGIC(ONE_SAMPLE_CI_SIM): %LET (variable name is GNSIMS)
SYMBOLGEN: Macro variable NSIMS resolves to 4000

NOTE: There were 4000 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.EXP_SIM has 4000 observations and 9 variables.
NOTE: DATA statement used:
   real time           0.25 seconds
   cpu time            0.25 seconds

MPRINT(ONE_SAMPLE_CI_SIM):   data simulate;
SYMBOLGEN: Macro variable NSIMS resolves to 4000
MPRINT(ONE_SAMPLE_CI_SIM):   do i=1 to 4000;
SYMBOLGEN: Macro variable NOBS resolves to 15
MPRINT(ONE_SAMPLE_CI_SIM):   do k=1 to 15;
SYMBOLGEN: Macro variable DISTN resolves to Uniform
MLOGIC(ONE_SAMPLE_CI_SIM): %IF condition &Distn=Normal is FALSE
SYMBOLGEN: Macro variable DISTN resolves to Uniform
MLOGIC(ONE_SAMPLE_CI_SIM): %IF condition &Distn=Exponential is FALSE
SYMBOLGEN: Macro variable SEED resolves to 0
MPRINT(ONE_SAMPLE_CI_SIM):   X = RANUNI(0);
MPRINT(ONE_SAMPLE_CI_SIM):   %LET (variable name is MU)
MPRINT(ONE_SAMPLE_CI_SIM):   output;
MPRINT(ONE_SAMPLE_CI_SIM):   end;
MPRINT(ONE_SAMPLE_CI_SIM):   end;

NOTE: The data set WORK.SIMULATE has 60000 observations and 3 variables.
NOTE: DATA statement used:
   real time           0.09 seconds
   cpu time            0.09 seconds

MPRINT(ONE_SAMPLE_CI_SIM):   proc sort data=simulate;
MPRINT(ONE_SAMPLE_CI_SIM):     by i;

NOTE: There were 60000 observations read from the data set WORK.SIMULATE.
NOTE: The data set WORK.SIMULATE has 60000 observations and 3 variables.
NOTE: PROCEDURE SORT used:
   real time           0.18 seconds
MPRINT(ONE_SAMPLE_CI_SIM): proc means data=simulate noprint;
MPRINT(ONE_SAMPLE_CI_SIM): by i;
MPRINT(ONE_SAMPLE_CI_SIM): var x;
MPRINT(ONE_SAMPLE_CI_SIM): output out=temp mean=xbar var=ssquare n=num;
SYMBOLGEN: Macro variable OUT resolves to unif_sim
NOTE: There were 60000 observations read from the data set WORK.SIMULATE.
NOTE: The data set WORK.TEMP has 4000 observations and 6 variables.
NOTE: PROCEDURE MEANS used:
   real time 0.09 seconds
   cpu time 0.09 seconds

MPRINT(ONE_SAMPLE_CI_SIM): data unif_sim;
MPRINT(ONE_SAMPLE_CI_SIM): set temp;
MPRINT(ONE_SAMPLE_CI_SIM): lower_m=xbar-tinv(.975,num-1)*sqrt(ssquare/num);
MPRINT(ONE_SAMPLE_CI_SIM): upper_m=xbar+tinv(.975,num-1)*sqrt(ssquare/num);
SYMBOLGEN: Macro variable MU resolves to 0.5
MPRINT(ONE_SAMPLE_CI_SIM): icover = (lower_m<0.5<upper_m);
MLOGIC(ONE_SAMPLE_CI_SIM): Ending execution.

NOTE: There were 4000 observations read from the data set WORK.TEMP.
NOTE: The data set WORK.UNIF_SIM has 4000 observations and 9 variables.
NOTE: DATA statement used:
   real time 0.23 seconds
   cpu time 0.23 seconds

629 proc means data=norm_sim mean;
630 title "T-based CI for one-sample";
SYMBOLGEN: Macro variable GNOBS resolves to 15
631 title2 "Normal populations -- # observations = &gnobs";
SYMBOLGEN: Macro variable GNSIMS resolves to 4000
632 title3 "(based on &gnsims simulation conditions)"
633 var icover;
634 run;

NOTE: There were 4000 observations read from the data set WORK.NORM_SIM.
NOTE: PROCEDURE MEANS used:
   real time 0.01 seconds
   cpu time 0.01 seconds
635 proc means data=exp_sim mean;
636 title "T-based CI for one-sample";
SYMBOLGEN: Macro variable GNOBS resolves to 15
638 title2 "Exponential populations -- # observations = &gnobs";
SYMBOLGEN: Macro variable GNSIMS resolves to 4000
639 title3 "(based on &gnsims simulation conditions)";
640 var icover;
641 run;

NOTE: There were 4000 observations read from the data set WORK.EXP_SIM.
NOTE: PROCEDURE MEANS used:
   real time          0.01 seconds
   cpu time           0.01 seconds

642 proc means data=unif_sim mean;
643 title "T-based CI for one-sample";
SYMBOLGEN: Macro variable GNOBS resolves to 15
645 title2 "Uniform populations -- # observations = &gnobs";
SYMBOLGEN: Macro variable GNSIMS resolves to 4000
646 title3 "(based on &gnsims simulation conditions)";
647 var icover;
648 run;

NOTE: There were 4000 observations read from the data set WORK.UNIF_SIM.
NOTE: PROCEDURE MEANS used:
   real time          0.01 seconds
   cpu time           0.01 seconds

649 options formdlim = "-";
650 proc means data=norm_sim mean;
651 title "T-based CI for one-sample";
SYMBOLGEN: Macro variable GNOBS resolves to 15
653 title2 "Normal populations -- # observations = &gnobs";
SYMBOLGEN: Macro variable GNSIMS resolves to 4000
654 title3 "(based on &gnsims simulation conditions)";
655 var icover;
656 run;

NOTE: There were 4000 observations read from the data set WORK.NORM_SIM.
NOTE: PROCEDURE MEANS used:
Step 1: write, test, debug SAS program that is to be built by a macro

Step 2: replace hard-coded constants with macro variables (using %LET and &var-name refs)

Step 3: create macro program wrapper for this code

Step 4: generalize code with Macro programming
(e.g. %IF-%THEN . . . %DO . . . )

**Step 5: store macro for later use

---

**Example – subsampling a data set using a Macro program**

```
%macro sample_ds(dsn=,pct=,outtest=,outver=);
    /*
    * samples a specified percentage (&pct) of a
    * dataset (&dsn) storing the observations selected
    * in &outtest and the observations not selected in &outver
    */
    %global gpct;
    %let gpct = &pct;
    %global gdsn;
    %let gdsn = &dsn;

data temp maxi; set &dsn end=eof;
    unif_index = ranuni(0);
    nobs + 1;
    output temp;
    if eof then do;
        maxnobs = nobs;
        output maxi;
    end;

data maxi; set maxi;
    do i = 1 to maxnobs;
        nobs = i;
        keep nobs maxnobs;
        output;
    end;

proc print data=maxi;
    title "DEBUG: maxnobs calculation check";
    run;

data temp;
    merge temp maxi;
    by nobs;

proc print data=temp;
    title "DEBUG: checking merged data";
    run;

proc sort data=temp; by unif_index;
    run;

data temp;
    set temp;
    cutpt = ceil(maxnobs*&pct/100);

proc print;
```
title "DEBUG: checking cut point calculation";
run;

data &outtest &outver;
  set temp;
  retain nobs;
  put "nobs = " nobs;
  put "cutpt = " cutpt;
  if _n_ <= cutpt then output &outtest;
  else output &outver;
  drop cutpt nobs;
%mend sample_ds;

* ------------------ test sampling macro -------------------------------;

data tester;
  do i = 1 to 150;
    x = i;
    y = 3 + 4.5*x + 10*rannor(0);
    output;
  end;

options symbolgen mprint mlogic;
%sample_ds(dsn=tester,pct=33,outtest=outt,outver=outv)

options nocenter nodate;
proc print data=outt;
  title "Selected data containing &gpct of data set &gdsn";
  run;

proc print data=outv;
  title "NOT Selected data from data set &gdsn";
  run;

proc reg data=outt;
  title "Regression using the test data set";
  model y = x;
  run;

Example – adequacy plots for simple linear regression

%macro linreg_plots(dsn=&syslast, title=, resp_var=, pred_var=);
  proc reg data=&dsn;
  title "&title";
  model &resp_var = &pred_var;
  output out=mout p=yhat r=resid;

  proc gplot data=mout;
}
title2 "Plot of &resp_var vs &pred_var with the fitted regression line superimposed"
  plot &resp_var*&pred_var yhat*&pred_var / overlay;
run;

proc gplot data=mout;
title2 "Plot of residuals vs. &pred_var"
  plot resid*&pred_var / vref=0;
run;

proc gplot data=mout;
title2 "Plot of residuals vs. predicted values"
  plot resid*yhat/ vref=0;
run;

/*
 construct the normal scores - Z[(i-.375)/(n+.25)]
 note: not multiplied by sqrt(mse)
 */
proc rank data=mout normal=blom out=rnew;
  var resid;
  ranks nscore;
proc gplot data=rnew;
title2 "Normal Probability Plot"
  plot resid*nscore;
%mend linreg_plots;

*---------------------- end of macro definition ------------------;

libname mydat 'D:\baileraj\Classes\Fall 2003\sta402\data';

proc contents data=mydat._all_; run;

ODS RTF file='D:\baileraj\Classes\Fall 2003\sta402\handouts\week10-plots.rtf';
%
linreg_plots(ds=dsn=mydat.manatee, title=Manatee deaths by boats registered, resp_var=manatees, pred_var=nboats)

ODS RTF Close;
Manatee deaths by boats registered

The REG Procedure
Model: MODEL1
Dependent Variable: manatees

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>1</td>
<td>1711.97866</td>
<td>1711.97866</td>
<td>93.61</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>12</td>
<td>219.44991</td>
<td>18.28749</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>13</td>
<td>1931.42857</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Root MSE          4.27639  R-Square  0.8864
Dependent Mean    29.42857  Adj R-Sq  0.8769
Coeff Var         14.53141

| Variable      | DF | Parameter Estimate | Standard Error | t Value | Pr > |t| |
|---------------|----|--------------------|----------------|---------|------|---|
| Intercept     | 1  | -41.43044          | 7.41222        | -5.59   | 0.0001|
| nboats        | 1  | 0.12486            | 0.01290        | 9.68    | <.0001|
Saving macros for repeated use

Two ways to store macro programs for later use in SAS

0. `%include <filename>/source2;`
   * not unique to storing macros but any code in an external file
   * inserts text of <filename> into SAS program

1. Autocall facility
   * consists of external files
   * must be compiled the first time it is used in SAS session
   * compiled into WORK.SASMCR (deleted at end of session)
   * TO SAVE - define directory and add macro program to it
   * store as individual file with a file type/ext SAS
   * options mautosource;
   * advantages: don’t have to submit program for compiling before you reference the macro program (macro processor does it if it finds the macro program in the autocall library).
   * disadvantages: macro program must be compiled the first time it is used in a SAS session (takes resources)

Save the following code in the file directory as `one_sample_ci_sim.sas` in directory D:\baileraj\Classes\Fall 2003\sta402\sta402_macros.

```sas
%MACRO one_sample_ci_sim(Nsims=4000, Nobs=15, Distn=Normal, Seed=0, OUT=); %* one_sample_ci_sim.sas stored in ; %* D:\baileraj\Classes\Fall 2003\sta402\sta402_macros ; %* date: 25 oct 03; %* based on “SIM1” macro (written by J. Deddens); %* %* Determine coverage of one-sample t-based CI when; %* Sampling from Normal, Exponential or Uniform; %* Distributions; %global gnobs; %let gnobs=&Nobs; %global gnsims;
```
%let gnsims = &Nsims;

data simulate;
  do i=1 to &Nsims;
    do k=1 to &Nobs;
      %if &Distn=Normal %then %do;
        X = RANNOR(&Seed);
        %let mu = 0;
      %end;
      %else %if &Distn=Exponential %then %do;
        X = RANEXP(&Seed);
        %let mu = 1;
      %end;
      %else %do;
        X = RANUNI(&Seed);
        %let mu = 0.5;
      %end;
      output;
    end;
  end;
proc sort data=simulate; by i;
proc means data=simulate noprint; by i; var x;
  output out=temp mean=xbar var=ssquare n=num;

data &out; set temp;
  lower_m=xbar-tinv(.975,num-1)*sqrt(ssquare/num);
  upper_m=xbar+tinv(.975,num-1)*sqrt(ssquare/num);
  icover = (lower_m<&mu<upper_m);
%MEND one_sample_ci_sim;

Now in a future run of SAS, use the following code.

filename mymacs
  'D:\baileraj\Classes\Fall 2003\sta402\sta402_macros';

* identify location of autocall library
options mautosource sasautos=mymacs;

%one_sample_ci_sim(out=norm_sim)
proc means data=norm_sim mean;
title "T-based CI for one-sample";
title2 "Normal populations";
  var icover;
  run;

* listing names of the autocall libs that are defined in SAS;
proc options option=sasautos;
  run;

FROM SAS OUTPUT File . . .

T-based CI for one-sample
Normal populations

The MEANS Procedure

Analysis Variable : icover

  Mean
    ffffffff
    0.9570000
    ffffffff

FROM SAS LOG File . . .

SAS (r) Proprietary Software Release 8.2  TS2M0
SASAUTOS=mymacs  Search list for autocall macros

2. stored compiled macro

* SAS catalogs will be defined to contain compiled macros
* two options needed to compile and store macro programs
  MSTORED and SASMSTORE.
* advantages: prevents repeated compiling of macro programs
  that are repeatedly used
* disadvantage: cannot be moved to other operating systems
  must be saved and recompiled under new OS
  compiled macro does not have original code
  (need to maintain source in a separate location).

In SAS run . . .
libname mymaccat
   "D:\baileraj\Classes\Fall 2003\sta402\sta402_macros";
options mstored sasmstore=mymaccat;

%MACRO one_sample_ci_sim(Nsims=4000, Nobs=15,
                          Distn=Normal, Seed=0, OUT=) /
    store des="One Sample t-based CI sim";
%*  one_sample_ci_sim.sas stored in ;
%*  D:\baileraj\Classes\Fall 2003\sta402\sta402_macros ;
%*  date:  25 oct 03;
%*  based on "SIM1" macro (written by J. Deddens);
%*
%*  Determine coverage of one-sample t-based CI when;
%*  Sampling from Normal, Exponential or Uniform;
%*  Distributions;
%
%global gnobs;
%let gnobs=&Nobs;
%
%global gnsims;
%let gnsims = &Nsims;

data simulate;
  do i=1 to &Nsims;
    do k=1 to &Nobs;
      %if &Distn=Normal %then %do;
        X = RANNOR(&Seed);
        %let mu = 0;
      %end;
      %else %if &Distn=Exponential %then %do;
        X = RANEXP(&Seed);
        %let mu = 1;
      %end;
      %else %do;
        X = RANUNI(&Seed);
        %let mu = 0.5;
      %end;
      output;
    end;
  end;
  proc sort data=simulate; by i;
  proc means data=simulate noprint; by i; var x;
    output out=temp mean=xbar var=ssquare n=num;
  data &out; set temp;
lower_m=xbar-tinv(.975,num-1)*sqrt(ssquare/num);
upper_m=xbar+tinv(.975,num-1)*sqrt(ssquare/num);

icover = (lower_m<&mu<upper_m);

%MEND one_sample_ci_sim;

In a later SAS run . . .

libname mymaccat "D:\baileraj\Classes\Fall 2003\sta402\sta402_macros";
proc catalog c=mymaccat.sasmacr;
contents;
run;
quit;

options mstored sasmstore=mymaccat;
%one_sample_CI_sim(out=norm_sim2)

proc means data=norm_sim2 mean;
title "T-based CI for one-sample";
title2 "Normal populations";
var icover;
run;

Contents of Catalog MYMACCAT.SASMACR
# Name               Type           Create Date        Modified Date  Description

T-based CI for one-sample
Normal populations

The MEANS Procedure

Analysis Variable : icover

Mean

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.9417500</td>
</tr>
</tbody>
</table>

Example - MACRO to simulate simple linear regression
%macro lreg_out(b0=0, bl=1, sig=5, nob=10, nsim=10, out=keep);

/* -------------------------------------------------------
"lreg_out"
- linear regression simulation
- generate data from a normal distribution
- fit a linear regression model
- store est. beta0 and beta1
- write parameter estimates to output data set
INPUT VARIABLES:
  b0 = intercept
  bl = slope
  sig = standard deviation
  nob = number of observations sampled per experiment
  nsim = number of simulated experiments
OUTPUT VARIABLE:
  out = output data set with estimated beta0 and beta1
Details:
  X values generated uniformly over [0, nob]
  Y ~ N(b0 + bl*x, sig^2)
------------------------------------------------------- */

%do isim= 1 %to &nsim;

%* generate sample;
data ransamp;
  do iobs = 1 to &nob;
    x = 0 + &nob*ranuni(0);
    y = &b0 + &bl*x + &sig*rannor(0);
    output;
    keep x y;
  end;
%end;

%* extract estimates;
ODS OUTPUT ParameterEstimates=parm_data loadChildren1;)

proc reg;
  model y = x;
  run;

ODS OUTPUT CLOSE;
quit;
run;
  * proc print data=parm_data_&isim;
  * run;
  
proc transpose data=parm_data_&isim out=tran_&isim prefix=beta_est;
  var Estimate;
  run;

%if &isim=1 %then %do;
  data &out; set tran_&isim;
%end;
%else %do;
  data &out; set &out tran_&isim;
%end;

%end;

* proc print data=keep;
* run;

%mend;
%*---------------- end of macro definition ------------;
%
%lreg_out(nsims=1000, out=keep)

proc print data=keep;
    run;

data keep; set keep;
    label beta_est1 = "Beta0";
    label beta_est2 = "Beta1";

ODS RTF file='D:\baileraj\Classes\Fall 2003\sta402\handouts\week10-plots2.rtf';

proc gplot;
title "Plot of SLOPE vs. INTERCEPT estimates from 1000 simulated expts";
    plot beta_est2*beta_est1;
    run;

proc gchart;
title "Histograms of Parameter Estimates";
vbar beta_est1;
vbar beta_est2;
run;

ODS RTF CLOSE;

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>NAME</em></th>
<th><em>LABEL</em></th>
<th>beta_est1</th>
<th>beta_est2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-0.71578</td>
<td>1.62810</td>
</tr>
<tr>
<td>2</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-1.75949</td>
<td>0.75661</td>
</tr>
<tr>
<td>3</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-7.05808</td>
<td>2.37633</td>
</tr>
<tr>
<td>4</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-3.10676</td>
<td>2.22525</td>
</tr>
<tr>
<td>5</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>3.79070</td>
<td>0.60780</td>
</tr>
<tr>
<td>6</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>3.11441</td>
<td>0.71842</td>
</tr>
<tr>
<td>7</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-2.27818</td>
<td>1.47052</td>
</tr>
<tr>
<td>8</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>6.22616</td>
<td>0.23557</td>
</tr>
<tr>
<td>9</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>5.38658</td>
<td>0.01776</td>
</tr>
<tr>
<td>10</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>0.46641</td>
<td>1.02923</td>
</tr>
<tr>
<td></td>
<td>... lines deleted (from Proc PRINT output)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>989</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-0.08216</td>
<td>1.32843</td>
</tr>
<tr>
<td>990</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>5.69996</td>
<td>0.19428</td>
</tr>
<tr>
<td>991</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>0.65316</td>
<td>0.64180</td>
</tr>
<tr>
<td>992</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-0.01910</td>
<td>1.03797</td>
</tr>
<tr>
<td>993</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-0.78082</td>
<td>1.17947</td>
</tr>
<tr>
<td>994</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-4.13196</td>
<td>1.89480</td>
</tr>
<tr>
<td>995</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-3.74987</td>
<td>1.52555</td>
</tr>
<tr>
<td>996</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-0.06003</td>
<td>1.30704</td>
</tr>
<tr>
<td>997</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>0.14970</td>
<td>0.90455</td>
</tr>
<tr>
<td>998</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-1.39439</td>
<td>1.29525</td>
</tr>
<tr>
<td>999</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-8.11269</td>
<td>2.38930</td>
</tr>
<tr>
<td>1000</td>
<td>Estimate</td>
<td>Parameter Estimate</td>
<td>-4.10057</td>
<td>1.42655</td>
</tr>
</tbody>
</table>

/* some testing code */
/*
%let b0=0;  %let b1=1;  %let sig=5; %let nobs=10;  %let nsims=1; %let out=keep; %let isim=1;

data ransamp;
do iobs = 1 to &nobs;
x = 0 + &nobs*ranuni(0);
y = &b0 + &b1*x + &sig*rannor(0);
output;
  keep x y;
end;

proc print;
run;

ods OUTPUT ParameterEstimates=parm_data_&isim;  /* extract quantiles *//*
/*
ods trace on;
proc reg data=ransamp;
  model y = x;
  run;
ods trace off;
MACRO language functions . . .

%index(source, string) – position IN “source” of first “string” occurrence

%length(string/text expression)

%scan(argument,n<,delim.>) – nth word in argument (%qscan)

%substr(argument, position <,length>) – extract substring of “length” characters from “argument” starting at “position” (%qsubstr)

%upcase(string/text expression) (%upcase)

%str = used to surround character strings

Examples
Extract month from date --> %substr(&sysdate,3,3)

%let months = Jan Feb Mar Apr;
%let pickmth = 3;
%scan(&months, &pickmth) --> Mar selected

* to evaluate values

%EVAL(arithmetic expression/logical expression)

%SYSEVALF(arithmetic expr./logical expr.<,conversion-type>)

%SYMPUT = assigns a SAS variable value to a macro variable call symput(macro-var, value)

%SYMGET = assigns a macro variable value to a SAS variable symget(argument)
Example – MACRO variable SYMPUT manipulations highlighted

Libname mydat 'D:\baileraj\Classes\Fall 2003\sta402\data';

/* expand into single line for each brood */
data test; set mydat.nitrofen;
   brood=1; conc=conc; nyoung=brood1; output;
   brood=2; conc=conc; nyoung=brood2; output;
   brood=3; conc=conc; nyoung=brood3; output;
%macro threeregs;
   proc sort data=test out=test;
      by brood;
   run;

data _null_;   
   set test;
   by brood;
   if first.brood then do;
      i+1;
      ii = left(put(i, 2.));
      call symput('mbrood'||ii, trim(left(brood)));
      call symput('total', ii);
   end;
   run;
%do ibrood = 1 %to &total;
   proc gplot data=test;
      where brood=&mbrood&ibrood;
      plot nyoung*conc;
      symbol1 interpol=rq
      value=diamond;
      title "Plot of Number of Young vs. Nitrofen Concentration";
      title2 "for brood &&mbrood&ibrood";
   run;
%end;
%mend;

options mlogic mprint symbolgen;

ODS RTF file='D:\baileraj\Classes\Fall 2003\sta402\handouts\week10-plots3.rtf';

%threeregs()
quit;
ODS RTF CLOSE;
Given the simple regression model

\[ Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \text{ where } \varepsilon_i \sim N(0, \sigma^2) \text{ for } i = 1, \ldots, n, \]

obtain marginal and joint sampling distributions for \( b_0 \) and \( b_1 \) by a simulation study.

What do we need?
1. A set of values for the independent variable,
2. Values for \( \beta_0, \beta_1, \sigma \),
3. Number of times to perform simulation

```plaintext
options ls=78 formdlim='\-';
/* n = 10, x-bar = 4.9, Sxx = 100.9 */
data independent;
  input x @@;
cards;
1 2 3 3 4 4 4 7 10 11;
data model;
  set independent;
y = 100 - 2\*x + 5\*rannor(0);
run;
proc reg data=model outest=parms;
  model y = x;
run;
proc print data=parms;
run;
quit;

\[ \text{var}(b_0) = \sigma^2 \left( \frac{1}{n} + \frac{\bar{x}^2}{Sxx} \right) = 5^2 \left( \frac{1}{10} + \frac{4.9^2}{100.9} \right) = 8.449 \]

\[ \text{var}(b_1) = \frac{\sigma^2}{Sxx} = \frac{5^2}{100.9} = 0.248 \]

\[ \text{cov}(b_0, b_1) = \frac{-\bar{x}\sigma^2}{Sxx} = \frac{-4.9(5^2)}{100.9} = -1.214 \]
options ls=78 formdlmm='-' nodate pageno=1;
/* n = 10, x-bar = 4.9, Sxx = 100.9 */
data independent;
        input x @@;
cards;
        1 2 3 3 4 4 4 7 10 11;
%
%macro sim_reg(datain=,beta0=,beta1=,sigma=,its=);
  /*----------------------------------------------+
  |  macro parameters                            |
  |    datain = data set containing "x" values   |
  |    beta0 = intercept                         |
  |    beta1 = slope                             |
  |    sigma = error standard deviation          |
  |    its = number of iterations in simulation  |
  +----------------------------------------------*/
  %do i = 1 %to &its;
    data model;
        set &datain;
        y = &beta0 + &beta1 * x + &sigma * rannor(0);
        run;
    proc reg data=model outest=parms noprint;
        model y = x;
        run;
    %if &i = 1 %then %let code = parms;
    %else %let code = history parms;
    data history;
        set &code;
        run;
  %end;
%
proc corr data=history cov;
    var x intercept;
    run;
    title 'Sampling distribution of the slope';
    proc gchart data=history;
        vbar x;
        run;
    title 'Sampling distribution of the intercept';
    proc gchart data=history;
        vbar intercept;
        run;
    title 'Estimated Slope versus Intercept';
    axis1 label=('Intercept');
    axis2 label=('Slope');
    proc gplot data=history;
        plot x*intercept / haxis=axis1 vaxis=axis2;
        run;
%
%mend;
%
%sim_reg(datain=independent,beta0=100,beta1=-2,sigma=5,its=1000);
quit;
Sampling distribution of the intercept

The CORR Procedure

2 Variables: x Intercept

Covariance Matrix, DF = 999

\[
\begin{array}{ccc}
x & \text{Intercept} \\
0.255494029 & -1.265581012 \\
-1.265581012 & 8.880333355 \\
\end{array}
\]

Simple Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1000</td>
<td>-2.00872</td>
<td>0.50546</td>
<td>-2009</td>
</tr>
<tr>
<td>Intercept</td>
<td>1000</td>
<td>100.00919</td>
<td>2.97999</td>
<td>100009</td>
</tr>
</tbody>
</table>

Simple Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>-3.88086</td>
<td>-0.19809</td>
<td>Intercept</td>
</tr>
<tr>
<td>Intercept</td>
<td>90.18441</td>
<td>109.86399</td>
<td>Intercept</td>
</tr>
</tbody>
</table>

Pearson Correlation Coefficients, N = 1000

<table>
<thead>
<tr>
<th>x</th>
<th>Intercept</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1.00000</td>
</tr>
<tr>
<td></td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Intercept</td>
<td>-0.84020</td>
</tr>
<tr>
<td>Intercept</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>
Sampling distribution of the slope

Sampling distribution of the slope
Saving macros for repeated use

0. `%include `<filename>`
   
   - not unique to storing macros but any code in an external file
   - inserts text of `<filename>` into SAS program

```sas
options ls=78 formdlim='-' nodate pageno=1;
filename mymacs 'E:\STA502\SLR simulation.sas';
%include mymacs;

/* n = 10, x-bar = 4.9, Sxx = 100.9 */
data independent;
   input x @@;
cards;
1 2 3 3 4 4 4 7 10 11
;
%sim_reg(datain=independent,beta0=100,beta1=-2,sigma=5,its=10);
quit;
```
1. autocall facility

- consists of external files
- must be compiled the first time it is used in SAS session
- compiled into WORK.SASMCR (deleted at end of session)
- To save – define directory and add macro program to it as individual file type/ext SAS
- options mautosource;
- advantages: don’t have to submit program for compiling before you reference the macro program (macro processor does it if it finds the macro program in the autocall library)
- disadvantages: macro program must be compiled the first time it is used in a SAS session (takes resources)

```sas
filename mymacs 'E:\STA502\SLR simulation.sas';
options ls=78 formdlim='-' nodate pageno=1 mautosource sasautos=mymacs;
/* n = 10, x-bar = 4.9, Sxx = 100.9 */
data independent;
input x @@;
cards;
1 2 3 3 4 4 4 7 10 11;
%
sim_reg(datain=independent,beta0=100,beta1=-2,sigma=5,its=10);
quit;
```

2. stored compiled macro

- SAS catalogs will be defined to contain compiled macros
- Two options needed to compile and store macro programs MSTORED and SASMSTORE
- Advantages: prevents repeated compiling and store of macro programs that are repeatedly used
- Disadvantages: cannot be moved to other operating systems. Must be saved and recompiled under new OS. Compiled macro does not have original code (need to maintain source in separate location).

```sas
libname mymaccat "E:\STA502";
options mstored sasmstore=mymaccat;
%macro sim_reg(datain=,beta0=,beta1=,sigma=,its=) / store des="SLR simulation";
/*----------------------------------------------+
| macro parameters                            |
|    datain = data set containing "x" values   |
|    beta0 = intercept                         |
|    beta1 = slope                             |
|    sigma = error standard deviation          |
|    its = number of iterations in simulation  |
+----------------------------------------------*/
%do i = 1 %to &its;
    data model;
    set &datain;
    y = &beta0 + &beta1 * x + &sigma * rannor(0);
```

run;
proc reg data=model outest=parms noprint;
  model y = x;
run;
%if &i = 1 %then %let code = parms;
%else %let code = history parms;
data history;
  set &code;
run;%end;
proc corr data=history cov;
  var x intercept;
run;
title 'Sampling distribution of the slope';
proc gchart data=history;
  vbar x;
run;
title 'Sampling distribution of the intercept';
proc gchart data=history;
  vbar intercept;
run;
title 'Estimated Slope versus Intercept';
axis1 label=('Intercept');
axis2 label=('Slope');
proc gplot data=history;
  plot x*intercept / haxis=axis1 vaxis=axis2;
run;%mend;

Then run at some later time

libname mymacs 'E:\STA502';
options ls=78 formdlim='-' nodate pageno=1 mstored sasmstore=mymacs;
/* n = 10, x-bar = 4.9, Sxx = 100.9 */
data independent;
  input x @@;
cards;
1 2 3 3 4 4 4 7 10 11
;run;
%sim_reg(datain=independent,beta0=100,betal=-2,sigma=5,its=10);
proc catalog c=mymacs.sasmacr;
  contents;
run;
quit;

THREE ADDITIONAL EXAMPLES – from M. Hughes
/* Calculate Area under the curve using a trapezoidal rule */

%MACRO AUCTRAP(DATA=_LAST_,OUT=_AUC_,VAR=,X=,BY=);
%**********************************************************************;
%*                                                                    *
%* AUCTRAP Computes the area under a curve by means of the trapezium  *
%* rule.                                                             *
%*                                                                    *
%* Author: Luc Wouters (Life Sciences 2744)                          *
%*                                                                    *
%* References:                                                        *
%* - Yeh, K.C, Kwan, K.C.: A comparison of numerical integration     *
%* Algorithms by trapezoidal, Lagrange, and spline approximation.     *
%* J Pharmacokin & Biopharmac 6(1):79-98, 1978                       *
%*                                                                    *
%* Arguments:                                                         *
%* -DATA= Data set to be analyzed. This data set may not contain the  *
%* variable names AUC1 AUCn.                                         *
%* -OUT= Output data set that contains the original data with the    *
%* variables specified replaced by their integral value.             *
%* -VAR= List of variables to be integrated.                         *
%* -BY= List of BY-variables. A separate analysis is obtained for    *
%* each BY-variable. The input data set needs not to be sorted in    *
%* their sequence.                                                    *
%* Data Sets Used:                                                    *
%* _W_, _AUC_                                                        *
%*                                                                    *
%* Procedures Used:                                                   *
%* SORTT                                                               *
%* Reserved variables in input data set:                             *
%* _AUC1-_AUCn, _OLD1-_OLDn, _OLDX_, _H_                             *
%**********************************************************************;
%* Macro Variables Used:                                              *
%LOCAL X;
%LOCAL Y;
%LOCAL I;
%LOCAL LASTBY;
%LOCAL V;
%* Process BY variables *

%IF &BY EQ %THEN %DO;
   %LET I=0;PROC SORTT DATA=&DATA OUT=_W_;BY &X;
   %LET FIRST=%STR(_N_=1);%LET LAST=LAST;
   %END;
%ELSE %DO;
   %LET I=1;
   %LET V=%SCAN(&BY,1);
   %DO %UNTIL (&V EQ);
      %LET I=%EVAL(&I+1);
      %LET V=%SCAN(&BY,&I);
   %END;
   %LET LASTBY=%SCAN(&BY,%EVAL(&I-1));
   %LET FIRST=FIRST.&LASTBY;
   %LET LAST=LAST.&LASTBY;
   PROC SORTT DATA=&DATA OUT=_W_;BY &BY &X;
   %END;
%LET I=1;
%LET V=%SCAN(&VAR,1);
%DO %UNTIL(&V EQ);
   %LET I=%EVAL(&I+1);
   %LET V=%SCAN(&VAR,&I);
%END;
%LET NRVAR=%EVAL(&I-1);

DATA &OUT;
   LENGTH
      %DO I=1 %TO &NRVAR;_AUC&I _OLD&I %END; 8;
   RETAIN _OLDX_ _AUC1-_AUC&NRVAR _OLD1-_OLD&NRVAR 0;
   SET _W_(KEEP=&X &VAR &BY) END=LAST;
   ARRAY AUC _AUC1-_AUC&NRVAR;
   ARRAY OLD _OLD1-_OLD&NRVAR;
   ARRAY YVAR &VAR;
   BY &BY;
   IF &FIRST THEN DO;
      _OLDX_=&X;
      DO OVER OLD;OLD=YVAR;AUC=0;END;
   END;
ELSE DO;
_H_=&X-_OLDX_;  
DO OVER AUC;AUC=AUC+0.5*_H_* (YVAR+OLD); OLD=YVAR; END;  
_OLDX_=&X;  
END;  
IF &LAST THEN DO;  
DO OVER AUC; YVAR=AUC; END;  
OUTPUT;  
END;  
DROP _OLD1__OLD&NRVAR _AUC1__AUC&NRVAR _OLDX__ _H_ &X;  
RUN;  
%MEND AUCTRAP;  

************************** test data set **************************;  
data a;  
do x = 100 to 1000 by 10;  
  fx = 20000/(x*x*x);  
  output;  
end;  
proc print data=a;  
run;  
proc gplot data=a; plot fx*x;  
run;  
************************** macro invocation **************************;  
%auctrap(data=a,out=auc,var=fx,x=x,by=);  
proc print data=auc;  
run;  

// * Calculate proportion of species present */  

***************************************************************************;  
* SAS macro program to calculate the proportion of n species present in a given *  
* combination of canyon segments.                                         *;  
* *;  
* The macro variables include the total number of canyons, the number of segments *;  
* per canyon, and the total number of species under consideration. The test data  *;  
***************************************************************************;
Program written by Michael Hughes, Oct 2002

---

data a;
*THE NUMBER OF SPECIES IN THE 'INPUT' STATEMENT MUST BE SET TO THE CORRECT NUMBER OF SPECIES IN THE INPUT FILE;
input canyon segment canyonsegment $ sp1-sp6;
cards;
  1 1 BA01 0 1 0 0 0 1 0
  1 2 BA02 1 0 0 1 0 0
  1 3 BA03 0 1 0 1 0 0
  2 1 DU01 1 0 1 1 0 0
  2 2 DU02 0 0 0 0 0 0
  2 3 DU03 1 0 1 0 0 1
  2 4 DU04 1 0 1 1 0 0
  3 1 FF01 0 0 1 0 0 0
  3 2 FF02 1 0 1 0 0 1
  4 1 QQ01 1 1 1 0 0 0
  4 2 QQ02 0 1 0 0 0 1;
%

%macro assess(animaltype=,n_canyons=,n_can1segs=,n_can2segs=,n_can3segs=,n_can4segs=,n_species=);
*THE NUMBER OF CONDITIONS IN THE 'IF' STATEMENT BELOW MUST MATCH THE NUMBER OF CANYONS;
  %do canseg=1 %to &n_canyons;
    %do can1seg=1 %to &n_can1segs;
      %do can2seg=1 %to &n_can2segs;
        %do can3seg=1 %to &n_can3segs;
          %do can4seg=1 %to &n_can4segs;
            *THE NUMBER OF DELIMITERS IN THE 'DATA' STATEMENT BELOW MUST MATCH THE NUMBER OF CANYONS;
              data ds&&can1seg.&can2seg.&can3seg.&can4seg;
              set a;
              *THE NUMBER OF CONDITIONS IN THE 'IF' STATEMENT BELOW MUST MATCH THE NUMBER OF CANYONS;
if (canyon=1 and segment=&can1seg) or
  (canyon=2 and segment=&can2seg) or
  (canyon=3 and segment=&can3seg) or
  (canyon=4 and segment=&can4seg);
   %do c=1 %to &n_canyons;
      canyon&c.seg=&can&c.seg;
   %end;
run;

*THE NUMBER OF DELIMITERS IN THE 'DATA' STATEMENT BELOW
MUST MATCH THE NUMBER OF CANYONS;
  data ds&&can1seg.&can2seg.&can3seg.&can4seg
    (drop=canyon segment canyonsegment sp1-sp&n_species
     status_sp1-status_sp&n_species);
  *THE NUMBER OF DELIMITERS IN THE 'SET' STATEMENT BELOW
MUST MATCH THE NUMBER OF CANYONS;
  set ds&&can1seg.&can2seg.&can3seg.&can4seg
    end=lastobs;
   %do t=1 %to &n_species;
      status_sp&t + sp&t;
   %end;
   if lastobs;
      array allspecies status_sp1-status_sp&n_species;
   do over allspecies;
      allspecies = (allspecies gt 0);
   end;
   n_species = n(of status_sp1-status_sp&n_species);
   p_conserve = sum(of status_sp1-status_sp&n_species)/&n_species;
run;

*THE NUMBER OF DELIMITERS IN THE 'NEW=' OPTION BELOW MUST MATCH
THE NUMBER OF CANYONS;
  proc append base=null
     new=ds&&can1seg.&can2seg.&can3seg.&can4seg;
  run;
   %end;
%end;
%end;
proc print data=null noobs;
title1 "Species Conservation Analysis: &animaltype";
title2 'canyon1=BA canyon2=DU canyon3=FF canyon4=QQ';
var %do c=1 %to &n_canyons;
canyon&c.seg
%end;
n_species p_conserve;
run;

%mend assess;

*PRIOR TO EXECUTION, SET THE COUNT INFORMATION AS MACRO PARAMENTERS BELOW.;
%assess(animaltype=BIRDS,n_canyons=4,n_can1segs=3,n_can2segs=4,n_can3segs=2,n_can4segs=2,n_species=6)

/* Processing 1000s of spreadsheets in a single macro */

filename junk dummy;
proc printto log=junk; run;

**** MACRO CODE FOR LOOPING THROUGH DATA INDICES 00000-00009 **************;
%macro dataloop1(start,finish);
   %do i = &start %to &finish;
      data dateinfo;
         length day $ 3. month $ 3.;
         infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.0000&i..Master.scope"
            firstobs=3 obs=3;
         input @7 day $ month $ date $ year $ time $;
            if month='Jan' then n_month=1;
            else if month='Jan' then n_month=1;
            else if month='Feb' then n_month=2;
            else if month='Mar' then n_month=3;
            else if month='Apr' then n_month=4;
   %end;
%mend dataloop1;
else if month='May' then n_month=5;
else if month='Jun' then n_month=6;
else if month='Jul' then n_month=7;
else if month='Aug' then n_month=8;
else if month='Sep' then n_month=9;
else if month='Oct' then n_month=10;
else if month='Nov' then n_month=11;
else if month='Dec' then n_month=12;
date=input(compress(date,','),?? 2.);
year=input(compress(year,','),?? 4.);
hour=input(substr(time,1,2),?? 2.);
min=input(substr(time,4,2),?? 2.);
sec=input(substr(time,7,2),?? 2.);
datetime_stamp=dhms(mdy(n_month,date,year),hour,min,sec);
drop time n_month;
run;

data oobdata;
infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.0000&i..Master.scope";
dlm='09'x dsd missover firstobs=643 obs=855;
input wavelength intensity;
run;
proc means data=oobdata noprint;
var intensity;
output out=oob_out sum=OOB;
run;

data ooadata;
infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.0000&i..Master.scope"
    dlm='09'x dsd missover firstobs=856 obs=1360;
input wavelength intensity;
run;
proc means data=ooadata noprint;
var intensity;
output out=ooa_out sum=OOA;
run;

data oovisdata;
infil "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.0000&i..Slavel.scope"
  dlm='09'x dsd misover firstobs=144 obs=1035;
  input wavelength intensity;
run;
proc means data=oovisdata noprint;
  var intensity;
  output out=oovis_out sum=OOVIS;
run;

  data oob_ooa_oovis; merge dateinfo oob_out(keep=OOB) ooa_out(keep=OOA) oovis_out(keep=OOVIS);
  run;

  proc append base=all_of_it data=oob_ooa_oovis force; run;
  proc datasets; delete oob_ooa_oovis dateinfo oobdata ooadata ooovisdata out_oob out_ooa out_oovis; quit; run;
%end;
%mend dataloop1;

**** MACRO CODE FOR LOOPING THROUGH DATA INDICES 00010-00099 *************;
%macro dataloop2(start,finish);

  %do i = &start %to &finish;

  data dateinfo;
    length day $ 3. month $ 3.;
    infil "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.000&i..Master.scope"
      firstobs=3 obs=3;
    input @7 day $ month $ date $ year $ time $;
      if month='Jan' then n_month=1;
      else if month='Jan' then n_month=1;
      else if month='Feb' then n_month=2;
      else if month='Mar' then n_month=3;
      else if month='Apr' then n_month=4;
      else if month='May' then n_month=5;
  %end;
else if month='Jun' then n_month=6;
else if month='Jul' then n_month=7;
else if month='Aug' then n_month=8;
else if month='Sep' then n_month=9;
else if month='Oct' then n_month=10;
else if month='Nov' then n_month=11;
else if month='Dec' then n_month=12;
date=input(compress(date,','),?? 2.);
year=input(compress(year,','),?? 4.);
hour=input(substr(time,1,2),?? 2.);
min=input(substr(time,4,2),?? 2.);
sec=input(substr(time,7,2),?? 2.);
datetime_stamp=dhms(mdy(n_month,date,year),hour,min,sec);
drop time n_month;
run;

data oobdata;
  infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.000&..Master.scope"
    dlm='09'x dsd missover firstobs=643 obs=855;
  input wavelength intensity;
run;
proc means data=oobdata noprint;
  var intensity;
  output out=oob_out sum=OOB;
run;

data ooadata;
  infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.000&..Master.scope"
    dlm='09'x dsd missover firstobs=856 obs=1360;
  input wavelength intensity;
run;
proc means data=ooadata noprint;
  var intensity;
  output out=ooa_out sum=OOA;
run;

data oovisdata;
  infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.000&..Master.scope"
    dlm='09'x dsd missover firstobs=856 obs=1360;
  input wavelength intensity;
run;
proc means data=oovisdata noprint;
  var intensity;
  output out=oovis_out sum=OOA;
run;
data oob_ooa_oovis; merge dateinfo oob_out(keep=OOB) ooa_out(keep=OOA) oovis_out(keep=OOVIS);
  run;

proc append base=all_of_it data=oob_ooa_oovis force; run;
proc datasets; delete oob_ooa_oovis dateinfo oobdata ooadata oovisdata out_oob out_ooa out_oovis; quit; run;

%end;
%mend dataloop2;

**** MACRO CODE FOR LOOPING THROUGH DATA INDICES 00100-00999 **************;
%macro dataloop3(start,finish);

%do i = &start %to &finish;

data dateinfo;
  length day $ 3. month $ 3.;
  infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.00&i..Master.scope"
    firstobs=3 obs=3;
  input @7 day $ month $ date $ year $ time $;
  if month='Jan' then n_month=1;
  else if month='Feb' then n_month=2;
  else if month='Mar' then n_month=3;
  else if month='Apr' then n_month=4;
else if month='May' then n_month=5;
else if month='Jun' then n_month=6;
else if month='Jul' then n_month=7;
else if month='Aug' then n_month=8;
else if month='Sep' then n_month=9;
else if month='Oct' then n_month=10;
else if month='Nov' then n_month=11;
else if month='Dec' then n_month=12;
date=input(compress(date,''),2.);
year=input(compress(year,''),4.);
hour=input(substr(time,1,2),2.);
min=input(substr(time,4,2),2.);
sec=input(substr(time,7,2),2.);
datetime_stamp=dhms(mdy(n_month,date,year),hour,min,sec);
drop time n_month;
run;

data oobdata;
  infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.00&i..Master.scope"
    dlm='09'x dsd missover firstobs=643 obs=855;
  input wavelength intensity;
run;
proc means data=oobdata noprint;
  var intensity;
  output out=oob_out sum=OOB;
run;

data ooadata;
  infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.00&i..Master.scope"
    dlm='09'x dsd missover firstobs=856 obs=1360;
  input wavelength intensity;
run;
proc means data=ooadata noprint;
  var intensity;
  output out=ooa_out sum=OOA;
run;

data oovisdata;
 infile "c:\Mike\'s Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.00&i..Slavel.scope"
      dlm='09'x dsd missover firstobs=144 obs=1035;
      input wavelength intensity;
    run;
    proc means data=oovisdata noprint;
      var intensity;
      output out=oovis_out sum=OOVIS;
    run;

    data oob_ooa_oovis; merge dateinfo oob_out(keep=OOB) ooa_out(keep=OOA) oovis_out(keep=OOVIS);
    run;

    proc append base=all_of_it data=oob_ooa_oovis force; run;
    proc datasets; delete oob_ooa_oovis dateinfo oobdata ooadata oovisdata out_oob out_ooa out_oovis; quit; run;
%end;
%mend dataloop3;

**** MACRO CODE FOR LOOPING THROUGH DATA INDICES 01000-02517 ***************;
%macro dataloop4(start,finish);

%do i = &start %to &finish;

  data dateinfo;
    length day $ 3. month $ 3.;
    infile "c:\Mike\'s Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.0&i..Master.scope"
         firstobs=3 obs=3;
    input @7 day $ month $ date $ year $ time $;
      if month='Jan' then n_month=1;
      else if month='Jan' then n_month=1;
      else if month='Feb' then n_month=2;
      else if month='Mar' then n_month=3;
      else if month='Apr' then n_month=4;

else if month='May' then n_month=5;
else if month='Jun' then n_month=6;
else if month='Jul' then n_month=7;
else if month='Aug' then n_month=8;
else if month='Sep' then n_month=9;
else if month='Oct' then n_month=10;
else if month='Nov' then n_month=11;
else if month='Dec' then n_month=12;
date=input(compress(date,','),?? 2.);
year=input(compress(year,','),?? 4.);
hour=input(substr(time,1,2),?? 2.);
min=input(substr(time,4,2),?? 2.);
sec=input(substr(time,7,2),?? 2.);
datetime_stamp=dhms(mdy(n_month,date,year),hour,min,sec);
drop time n_month;
run;

data oobdata;
  infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.0&i..Master.scope"
    dlm='09'x dsd missover firstobs=643 obs=855;
  input wavelength intensity;
run;
proc means data=oobdata noprint;
  var intensity;
  output out=oob_out sum=OOB;
run;

data ooadata;
  infile "c:\Mike's Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.0&i..Master.scope"
    dlm='09'x dsd missover firstobs=856 obs=1360;
  input wavelength intensity;
run;
proc means data=ooadata noprint;
  var intensity;
  output out=ooa_out sum=OOA;
run;

data oovisdata;
infile "c:\Mike\'s Documents\SCC Client files\Oris, Jim\Shipboard\01May-June\PWS04.time.0&i..Slave1.scope"
    dlm='09'x dsd missover firstobs=144 obs=1035;
    input wavelength intensity;
run;

proc means data=oovisdata noprint;
   var intensity;
   output out=oovis_out sum=OOVIS;
run;

data oob_ooa_oovis; merge dateinfo oob_out(keep=OOB) ooa_out(keep=OOA) oovis_out(keep=OOVIS);
   run;
   proc append base=all_of_it data=oob_ooa_oovis force; run;
proc datasets; delete oob_ooa_oovis dateinfo oobdata ooadata oovisdata out_oob out_ooa out_oovis; quit; run;
%end;
%mend dataloop4;

%dataloop1(0,9);
%dataloop2(10,99);
%dataloop3(100,999);
%dataloop4(1000,2517);

data 'c:\Mike\'s Documents\SCC Client files\Oris, Jim\summaries0506';
   set all_of_it;
run;

proc print data=all_of_it;
   format datetime_stamp datetime.;
run;

proc printto; run;
quitar;