Using PROC IML

Many statistical procedures can be simply described using matrix algebra. As new procedures are developed, the lag time between the methods, and implementation of the methods in a computer package is often long (as much as 5 years). Additionally, statistical procedures in programs are designed to be menu driven with option specified by the user. There is a direct tradeoff in between the number of options available and the complication of use. This results in some options not being available unless special programming is made. Principally for these reasons, a matrix algebra language is desirable.

The SAS matrix algebra procedure (Interactive Matrix Language, PROC IML) permits simple matrix algebra programming for statistics and data management. The procedure has graphics capability, and can perform most standard matrix algebra statistical functions. It is simple to use, and flexible. We illustrate several examples of the procedure as an introduction to the procedure in this section. Extensive examples and details are contained in the SAS PROC IML manual.

1. Starting and ending the procedure.

The IML procedure can be run interactively. The simplest way to access the procedure, and run it interactively is with the following commands:

```
SAS -NODMS                        /* Begins the SAS session with no display manager    */
PROC IML;                      /* Starts the IML procedure                          */
RESET PRINT;                   /* Requests that each matrix or value be printed afer
                                 specification */
< proc iml statements>
QUIT;                          /* Ends the IML procedure                           */
BYE;                           /* Ends the SAS session                             */
```

When running IML interactively, the result of each command will be displayed immediately after the command is issued. This is a good way to test new commands and gain familiarity with the procedure. Once you are familiar with the procedure, IML commands can also be entered in a batch program, as we have entered other SAS programs. We use this batch mode in illustrating procedures for IML in this section.

Output from the IML procedure is produced in the LOG window, not in the OUTPUT window. The output will be intermixed with the program if the RESET PRINT command is given. Many statements used in DATA steps can be used in connection with proc IML. Also, MACROS can be used in connection with proc IML to create entire procedures. The examples we illustrate show how common statistical procedures can be implemented in PROC IML, and illustrate some of the features and functions of the procedure.

2. An Example of a Simple Linear Regression in PROC IML.

The first example fits a simple linear regression model. Input to the program is a vector
of dependent variables, and a vector of independent variables. These data are entered directly in the IML procedure as vectors.

```sas
options linesize=120 pagesize=60 nocenter nodate nonumber obs=10000;
***********************************************************************************************;
***    PROGRAM:                                                                                
*P*   IML1.SAS on \PH69TXT\ #1      Programmer: Ed Stanek   Date: 5/9/90                       
*P*          Description:  Do simple linear regression in IML                                  
***********************************************************************************************;
libname old '\sph';
   first fit a simple linear regression;
proc iml;
    yt={23 28 19 31 32 37 52 60 };  /* yt is the transposed vector of dependent variables */
    x0t=j(1,8,1);            /* x0t is a row vector of ones      */
    x1t={26 39 41 44 43 62 73 81};/* x1t is a row  vector of the first independent variable */
    xt=x0t//x1t;                        /* xt is the transposed design matrix for a simple linear regression */
    y=yt`;                              /*  y is the vector of dependent variables        */
    x=xt`;                              /*  x is the design matrix for a simple linear regression*/
    n=nrow(y);                          /*  n is the number of observations               */
    p=ncol(x);                          /*  p is the number of parameters that are estimated */
    beta=inv(x`*x)*x`*y;                /*  beta is a vector of estimated regression coefficients*/
    r=y-x*beta;                         /*  r is a vector of residuals            */
    df=n-p;                             /*  df is the degrees of freedom for the error           */
    ssq=r`*r/df;                        /*  ssq is an estimate of the error MSE            */
    beta_v=inv(x`*x)*ssq;               /*  variance matrix for beta           */
    beta_sd=sqrt(vecdiag(beta_v));      /*  standard error of beta            */
    t=beta/beta_sd;             /*  t-statistics for tests of significant regression coefficients*/
    p_value=1-probf(t#t,1,df);          /*  p_value is a vector of p-values for the two tailed test */
    results= beta`/beta_sd`// t`/p_value`; /* matrix of results         */
    print y x n p beta;
    print r ssq nrow(y) nrow(beta) df;
    beta_v=inv(x`*x)*ssq;               /* variance matrix for beta           */
    beta_sd=sqrt(vecdiag(beta_v));      /* standard error of beta            */
    t=beta/beta_sd;             /*  t-statistics for tests of significant regression coefficients*/
    p_value=1-probf(t#t,1,df);          /*  p_value is a vector of p-values for the two tailed test */
    cname={ "Intercept" "X1"};        /*  column header                  */
    rname={ "Beta", "SD", "t-test", "p-value"};
    results= beta`/beta_sd`// t`/p_value`; /* matrix of results         */
    print results[r=rname c=cname];
quit;
run;
```

Several aspects of the program are worth emphasizing. Vectors and matrices in PROC IML are defined by specifying elements of the matrices, separating column elements by spaces, and rows by commas. Many matrix algebra functions are available in PROC IML.
to make the construction of design matrices and calculations easy. Some of the functions
used in the procedure are described as follows:

<table>
<thead>
<tr>
<th>Procedure/ Function</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>j(A,B,C)</td>
<td>j(1,8,1)</td>
<td>Automatic matrix generator with A rows, B columns, and values of C.</td>
</tr>
<tr>
<td>// x0t//x1t</td>
<td></td>
<td>Row concatenation that results in a matrix</td>
</tr>
<tr>
<td>`</td>
<td>y=y′</td>
<td>Transpose of a vector or matrix, interchanging rows and columns.</td>
</tr>
<tr>
<td>nrow(A)</td>
<td>nrow(y)</td>
<td>Number of rows in the A matrix.</td>
</tr>
<tr>
<td>ncol(A)</td>
<td>ncol(x)</td>
<td>Number of columns in the A matrix.</td>
</tr>
<tr>
<td>A*B</td>
<td>x′*x</td>
<td>Matrix multiplication, multiplying each row of A times each column of B.</td>
</tr>
<tr>
<td>inv(A)</td>
<td>inv(x′*x)</td>
<td>Matrix inverse of A (for full rank matrices)</td>
</tr>
<tr>
<td>vecdiag(A)</td>
<td>vecdiag(beta_v)</td>
<td>Creates a column vector the contains the diagonal elements of a square matrix A.</td>
</tr>
<tr>
<td>sqrt(A)</td>
<td>sqrt(vecdiag(beta_v))</td>
<td>Takes the square root of each element of A.</td>
</tr>
<tr>
<td>A/B</td>
<td>beta/beta_sd</td>
<td>Divides elements of A by corresponding elements of B.</td>
</tr>
<tr>
<td>A#B</td>
<td>t#t</td>
<td>Multiplies elements of A by corresponding elements of B.</td>
</tr>
<tr>
<td>PROBF(A,N,D)</td>
<td>probf(t#t,1,df)</td>
<td>Produces the cumulative percentile for each value of A from an F distribution with N=numerator degrees of freedom and D denominator degrees of freedom.</td>
</tr>
<tr>
<td>A[r=rn c=cn]</td>
<td>results[r=rname c=cname]</td>
<td>Associates a row name (rn) and column name (cn) with the matrix A when printed.</td>
</tr>
</tbody>
</table>

The output from the IML procedure is as follows:

```
600   print yt x0t x1t;
    YT
      23  28  19  31  32  37  52  60

    X0T
      1  1  1  1  1  1  1  1

    X1T
      26  39  41  44  43  62  73  81

607   print y x n p beta;
    Y    X      N    P   BETA
      23  1  26     8  2  -0.556773
      28  1  39     8 2  0.700377
      19  1  41     8 2  0.740186
      31  1  44     8 2  0.740186
      32  1  43     8 2  0.740186
      37  1  62     8 2  0.740186
      52  1  73     8 2  0.740186
      60  1  81     8 2  0.740186

612   print r ssq nrow(y) nrow(beta) df;
    R SSQ  NROW   Y NROW BETA  DF
      5.3469716 28.603066 23  -0.556773  6
      1.2420709  28        28  0.700377
      -9.158683  19        19
      0.740186   31        31
```
2.440563  32
-5.8666  37
1.4292536  52
3.8262377  60

625      print beta beta_sd t p_value ;
          BETA   BETA_SD         T   P_VALUE
          -0.556773  5.7998186  -0.095998  0.926648
          0.700377  0.1072456   6.5305919  0.0006156

626      print results[r=rname c=cname];
          RESULTS Intercept       X1
          Beta   -0.556773  0.700377
          SD     5.7998186  0.1072456
          t-test  -0.095998  6.5305919
          p-value  0.926648  0.0006156

2. A Revised Regression program using a IML Module

Once a set of IML statements have been specified, it is often useful to save these
statements in a module. The module can then be accessed using different input from
different but similar problems. The program listed below illustrates use of a module for the
simple linear regression.

options linesize=120 pagesize=60 nocenter nodate nonumber obs=10000;

*** PROGRAM: ;
*P* IML2.SAS on \PH69TXT\ #1 Programmer: Ed Stanek Date: 5/7/90 ;
*P* Description: Do simple linear regression, multiple regression, and ANOVA, and ;
*P* MANOVA in IML ;
* Input: none ;

libname old '\sph';

* develop module to fit a simple linear regression;
*
proc iml;
  *;
  * read in data;
  *
   yt={23 28 19 31 32 37 52 60 }; /* yt is the transposed vector of dependent variables
     */
   x1t={26 39 41 44 43 62 73 81}; /* x1t is a row vector of the first independent variable
     */

start reg;
  n=ncol(yt);             /* n is the number of observations
                         */
  x0t=j(1,n,1);          /* x0t is a row vector of ones
                         */
xt=x0t//x1t; /* xt is the transposed design matrix for a simple linear regression */

y=yt`; /* y is the vector of dependent variables */
x=xt`; /* x is the design matrix for a simple linear regression */
p=ncol(x); /* p is the number of parameters that are estimated */
beta=inv(x`*x)*x`*y; /* beta is a vector of estimated regression coefficients */

r=y-x*beta; /* r is a vector of residuals */
df=n-p; /* df is the degrees of freedom for the error */
ssq=r`*r/df; /* ssq is an estimate of the error MSE */

beta_v=inv(x`*x)*ssq; /* variance matrix for beta */

beta_sd=sqrt(vecdiag(beta_v)); /* standard error of beta */
t=beta/beta_sd; /* t-statistics for tests of significant regression coefficients */
p_value=1-probf(t#t,1,df); /* p_value is a vector of p-values for the two tailed test */
cname={ "Intercept" "X1"}; /* column header */
rname={ "Beta", "SD", "t-test", "p-value"};

results= beta` beta_sd` t` p_value`; /* matrix of results */

print results[r=rname c=cname]; finish;

run reg; quit; run;

The module created has been named "REG". The module is begun with the statement:

START REG;

and ended with the statement:

FINISH;

To run the module, we simply type:

RUN REG;
The results from the program are simplified to represent simply the estimates of the parameters, and their significance. These results are the same as in the previous program, and are given as follows:

```
728    run reg;
RESULTS Intercept        X1
Beta   -0.556773  0.700377
SD      5.7998186 0.1072456
 t-test  -0.095998 6.5305919
 p-value  0.926648 0.0006156
```

3. A Multiple Regression program with Input from a SAS data set.

Most often it will be more convenient to read data from an external data set (such as a SAS data set), rather than entering data directly into PROC IML. To do this, we create another module that will read in SAS data. To enable subsequent executions of the program on different data be readily made, we locate all of the "user defined" steps in the program in the data input module. An example is given as follows:

```sas
options linesize=120 pagesize=60 nodate nonumber obs=10000;
* *********************************************************************************************** *
*** PROGRAM: 
*P* IML4.SAS on \PH69TXT\ #1 Programmer: Ed Stanek Date: 5/7/90 
*P* Description: Do multiple regression, and ANOVA, 
* Input: none 
* *********************************************************************************************** *
libname old '\sph';
*;
* develop module to fit a simple linear regression;
*;
data d;
  input health knowldg attitud  age race educ;
cards;
23 26 18  2 6  7
28 39 10  3 9  12
19 28 17  4 1  8
31 40 19  4 4  13
32 27 16  4 3  0 16
37 38 19  6 2  1 8
52 35 17  7 3  1 12
60 37 14  8 1  2 11
;
proc iml;
  reset spaces=5 nocenter;
  start datain; /* Module to read in data */
  /* Requires input of XVAR: design matrix */
  /* Y: dependent variable */
  /* ID: subject variable */
```
titlea="Multiple Regression. Dependent variable: Heath";
cvars={age educ};     /* Column labels for output */
use d;                       /* Data set is opened */
read all var {age educ} into x;   /* Independent variables */
read all var {health} into y;     /* Dependent variables */
close d;                          /* Data set is closed */
finish;

start reg;
run datain;                          /* read in data */
n=nrow(y);                          /* n is the number of observations */
x0=j(n,1,1);         /* x0 is a column vector of ones */
x=x0||x;                            /* x is the design matrix for a linear regression */
p=ncol(x);                          /* p is the number of parameters that are estimated */
beta=inv(x`*x)*x`*y;                /* beta is a vector of estimated regression coefficients */
r=y-x*beta;                         /* r is a vector of residuals */
df=n-p;                             /* df is the degrees of freedom for the error */
ssq=r`*r/df;                        /* ssq is an estimate of the error MSE */

beta_v=inv(x`*x)*ssq;               /* variance matrix for beta */
beta_sd=sqrt(vecdiag(beta_v));      /* standard error of beta */
t=beta/beta_sd;                     /* t-statistics for tests of significant regression coefficients */
p_value=1-probf(t#t,1,df);          /* p_value is a vector of p-values for the two tailed test */
cname={ "Intercept"}||cvars;        /* column header */
rname={ "Beta", "SD", "t-test", "p-value"};
results=beta`//beta_sd`//t`//p_value`;    /* matrix of results */
print ,, titlea;
print , results[r=rname c=cname];
finish;
run reg;
quit;
run;

The program has two modules, with the DATAIN module run in the REG module. Data
must be stored in a data set named D. The DATAIN module allows the used to specify a title, and alter the dependent and independent variables. Variable names are created that are used when displaying the output, and are user defined.

The REG module has also been altered, since the data that is read in is automatically in a column format. A vector of "ones" corresponding to the intercept in the multiple regression model is added to the matrix of independent variables (the Design matrix), where the dimension of the vector corresponds to the number of observations in y. The resulting program is simpler. The corresponding output follows:

1341 run reg;

TITLEA
Multiple Regression. Dependent variable: Heath

RESULTS Intercept AGE EDUC
Beta -6.702507 0.686957 0.6282144
SD 8.8561303 0.1094925 0.677911
 t-test -0.756821 6.2740092 0.9266916
p-value 0.4832735 0.0015104 0.3966154

4. A Simple IML Module for a One Way Analysis of Variance

Regression and analysis of variance are two aspects of the same type of modelling, general linear modeling. The models are defined in terms of a linear relationship between a dependent variable, and independent variables. In ANOVA problems, the independent variables correspond to 0/1 variables that are dummy variables representing groups or categories. A simple modification of the IML DATAIN module will allow ANOVA models to be fit. We use a function in IML to generate the design matrix using deviations from means parameterization. The change in the DATAIN module is illustrated below, with a listing of the resulting X matrix, and the program output.

start datain; /* Module to read in data */
/* Requires input of XVAR: design matrix */
/* Y: dependent variable */
/* ID: subject variable */
titlea="Multiple Regression. Dependent variable: Heath");
cvars={race1 race2}; /* Column labels for output */
use d; /* Data set */
read all var {race} into xa; /* Independent variables */
read all var {health} into y; /* Dependent variables */
close d;
x=designf(xa); /* Create deviations from means design matrix */
print x;
finish;
Output:

1413 run reg;

X
1 0
0 1
1 0
-1 -1
1 0
0 1
0 1
-1 -1

TITLEA
Multiple Regression. Dependent variable: Heath
RESULTS Intercept RACE1 RACE2
Beta 36.388889 -11.72222 2.6111111
SD 4.563205 6.2186001 6.2186001
t-test 7.9744147 -1.885026 0.4198873
p-value 0.0005004 0.1181119 0.6920002

The variable RACE1 represent the deviation from mean effect for the "0" race group. The RACE2 variable represent a deviation from mean effect for the "1" race group. The deviation from mean effect for the "3" race group is given by the negative of the sum of the coefficients for the other two race effects.

5. An example of a Multivariate ANCOVA using IML

Multivariate analyses can be performed in a similar manner to univariate analyses using PROC IML. The matrix algebra language makes multivariate analyses particularly simple to specify and compute. Multivariate test statistics can also be computed using special functions in the IML procedure. Although not illustrated in the following example, they can be easily constructed upon consulting a multivariate text and the IML manual. The module illustrated below will do individual t-tests for the multivariate estimates.
options linesize=120 pagesize=60 nocenter nodate nonumber obs=10000;

*** PROGRAM: ;
*P* IML6.SAS on \PH69TXT\ #1  Programmer: Ed Stanek  Date: 5/7/90 ;
*P*   Description:  Do Multivariate regression ;

* Input:  none ;
***********************************************************************************************;
libname old '\sph'; *
* develop module to fit a simple linear regression; *
data d;
   input health knowldg attitud age race educ;
cards;
23 26 18  26 0  7
28 39 10  39 1 12
19 28 17  41 0  8
31 40 19  44 2 13
32 27 16  43 0 16
37 38 19  62 1  8
52 35 17  73 1 12
60 37 14  81 2 11
;
proc iml;
   reset spaces=5 nocenter;
start datain;                       /*  Module to read in data                            */
   /*  Requires input of XVAR:  design matrix            */
   /*                       Y:  dependent variable       */
   /*                      ID:  subject variable         */
titlea="Multiple Regression.  Dependent variable: Heath Knowledge and Attitude";
rvars={age, educ, race1, race2};     /*  Row labels for output                             */
cvars={health knowledge attitude};/*  Column labels for output                        */
use d;       /*  Data set                                          */
read all var {race} into xa;      /*  Read nominal independent variables                */
read all var {age educ} into xb;  /*  Read other continuous independent variables       */
read all var {health knowldg attitud} into y;     /*  Dependent variables */
close d;
   x=xb||designf(xa);                /*  Create deviations from means design matrix for nominal */
variables */
print x;
finish;
start reg;
run datain;                          /*  read in data                                      */
n=nrow(y);                          /*  n is the number of observations                   */
q=ncol(y);                          /*  q is the number of dependent variables            */
x0=j(n,1,1);                /*  x0 is a column vector of ones                     */
proc iML;
x=x0||x;                  /* x is the design matrix for a linear regression */
p=ncol(x);                /* p is the number of parameters that are estimated */
beta=inv(x'x)x'y;        /* beta is a vector of estimated regression coefficients */
r=y-x*beta;              /* r is a vector of residuals */
df=n-p;                  /* df is the degrees of freedom for the error */
ssq=r'^r/df;             /* ssq is an estimate of the error MSE (variance covariance matrix) */
varx=vecdiag(ssq);       /* column vector of diagonal elements of variance matrix */
do i=1 to q;
beta_v=inv(x'x)*varx[i,1]; /* variance matrix for beta */
beta_sd=sqrt(vecdiag(bbeta_v)); /* standard error of beta */
if i=1 then b_sd=beta_sd;
else b_sd=b_sd||beta_sd;
end;
print beta b_sd;        /* matrix of sd for regression parameters */
t=beta/b_sd;             /* t-statistics for tests of significant regression coefficients */
p_value=1-probf(t#t,1,df); /* p_value is a vector of p-values for the two tailed test */
rname={ "Intercept"}//rvars; /* column header */
cname=cvars;

print titlea;
print ,, beta[r=rname c=cname] b_sd[r=rname c=cname];
print , t[r=rname c=cname] p_value[r=rname c=cname];
finish;

run reg;
quit;
run;

The resulting output is as follows:

1725 run reg;

X
26  7  1  0
39 12  0  1
41  8  1  0
44 13 -1 -1
6. A simple Simulation using PROC IML.

The IML procedure can be combined random number generators to perform simulations in a straight forward manner. We illustrate this with an example of a simulation on a problem in linear regression analysis known as the "errors in variables" problem. Ideally, a simple linear regression model consists of a set of dependent variables, and a set of non-random independent variables. Often the set of independent variables are not fixed, but rather are also random variables, and assumed fixed by conditioning. In both settings, standard simple linear regression methods are appropriate. However, if there is measurement error in the independent variables, bias will occur in the regression estimates. For example, suppose we are interested in the relationship between salt intake and DBP. Let :

\[ y_i = \text{DBP for the } i\text{th subject} \]
\[ u_{xi} = \text{the true salt intake for the } i\text{th subject, recorded as the difference in the true salt intake for the} \]
\[ \text{ith subject from the average salt intake in a population.} \]
\[ x_i = \text{the difference in salt intake for the } i\text{th subject from the average salt intake for a population of} \]
\[ \text{subjects, self-reported by diet.} \]
Also, let us assume that the relationship between the true salt intake and DBP is as follows:

\[ y_i = B_0 + B_1 u_{xi} + e_i \]  
(\text{Model 1})

where \( B_0 = 10 \) and \( B_1 = 2 \). Also assume that the errors (\( e_i \)) are independent and identically normally distributed with mean zero and variance 16. We will investigate the performance of a linear regression model when \( u_{xi} \) is recorded, and compare this performance with a linear regression model when only \( x_i \) is recorded, where

\[ x_i = u_{xi} + e_{xi} \]

and the errors \( e_{xi} \) are independent and identically normally distributed with mean 0 and variance 9.

To simulate Model 1, we generate 100 sets of data with \( n=100 \) in each set. We then fit the simple linear regression model using ordinary least square, and save the resulting parameter estimates in a SAS data set. Finally, we calculate descriptive statistics on the parameter estimates and compare them with their true values. A program to perform this simulation follows:

```sas
libname old '\sph';
*
*  generate simulated regression data with normal errors;
*  y is normally distributed with mean 10 and mse 16;
*  x is fixed and uniform in the range from -2 to 2;
*;
data betax;
  input beta0 beta1;
cards;
retain seed 1237491;
data d;
do i=1 to 10000;
  my=10;
  slope=2;
  x=4*(.5-ranuni(seed));                         /* x is uniform (-2, 2)                  */
  y=10+2*x+4*rannor(seed);                       /* Y is normal mean=10 +2*x var(y)=16 */
  output;
end;
proc means data=d;
proc iml;
  reset spaces=5 nocenter;
do j=1 to 100;
  beg=(j-1)*100+1;
  end=j*100;
  use d;                                   /*  Data set */
  read point (beg:end) var {x} into x;        /* Independent variables */
  read point (beg:end) var {y} into y;        /* Dependent variables */
```

```
/*
   close d;
   n=nrow(y);                                      /* n is the number of observations
   */
   x0=j(n,1,1);                   /* x0 is a column vector of ones
   */
   x=x0|x;                                        /* x is the design matrix for a linear
   regression */
   beta=(inv(x`*x)*x`*y)`;                         /* beta is a vector of estimated regression
   coefficients */
   edit betax;
   setout betax;
   append from beta;
   end;
   quit;
proc means data=betax;
title1 'Simulation of simple linear regression with no measurement error';
run;

The simulation creates a single data set of 10000 records, where the 10000 records are
then read into 100 separate data sets with 100 records each. Descriptive statistics of the
simulated data are calculated to verify proper simulation. In order to read sets of 100
records, the case numbers are specified with a POINT statement in the READ command
in IML. After calculating regression coefficients, the coefficients are added to a data set
BETAX. First the data set is opened with the command "EDIT BETAX;". Next, the data
set is specified as the current output data set with the command "SETOUT BETAX;".
Finally, the vector of regression coefficients are output to the data set with the command
"APPEND FROM BETA;". The result of the simulated data and the simulation is given
next.

N Obs Variable N Minimum Maximum Mean Std Dev
-----------------------------------------------------------------------------------------
10000 SEED 10000 1237491.00 1237491.00 1237491.00 0
10000 I 10000 1.0000000 10000.00 5000.50 2886.90
10000 MY 10000 10.0000000 10.0000000 10.0000000 0
10000 SLOPE 10000 2.0000000 2.0000000 2.0000000 0
10000 X 10000 -1.9991354 1.9997314 -0.0037019 1.1463412
10000 Y 10000 -5.2897684 26.3606532 10.0020343 4.6003182
-----------------------------------------------------------------------------------------

Simulation of simple linear regression with no measurement error

N Obs Variable N Minimum Maximum Mean Std Dev
-------------------------------------------------------------------------------------------------
250 BETA0 250 8.7906974 11.1064517 10.0047689 0.4065827
250 BETA1 250 0.8568666 3.0593637 2.0084932 0.3533958
-------------------------------------------------------------------------------------------------

The means and variance of the simulated data are reasonably close to the specified
parameter values. Note that the average of the regression parameter estimates are
nearly identical to the parameters specified in the linear regression.

To see the effect of errors in the variable X, we alter the data generated in the simulation. The value of y is generated using the true value of x (that is not observable), while the values of x used in the regression equation include measurement error. The simulation program, which generates 250 data sets of 100 cases, is as follows:

```sas
libname old '\sph';
*
*   generate simulated regression data with normal errors;
*   y is normally distributed with mean 10 and mse 16;
*   x is fixed and uniform in the range from -2 to 2;
*;
data betax;
  input beta0 beta1;
cards; ;
data d;
  retain seed 1237491;
  do i=1 to 25000;
    my=10;
    slope=2;
    truex=4* (.5-ranuni(seed));
    x=truex+3*rannor(seed);
    y=10+2*truex+4*rannor(seed);
    output;
  end;
  proc means data=d;
```

proc iml;
    reset spaces=5 nocenter;
    do j=1 to 250;
    beg=(j-1)*100+1;
    end=j*100;
    use d;                              /*  Data set */
    read point (beg:end) var {x} into x;          /*  Independent variables */
    */
    read point (beg:end) var {y} into y;          /*  Dependent variables */
    */
    close d;
    n=nrow(y);                          /*  n is the number of observations */
    */
    x0=j(n,1,1);         /* x0 is a column vector of ones */
    */
    x=x0||x;                            /* x is the design matrix for a  linear regression */
    */
    beta=(inv(x`*x)*x`*y)``;                /*  beta is a vector of estimated regression coefficients */
    edit betax;
    setout betax;
    append from beta;
    end;
    quit;
proc means data=betax;
    title1 'Simulation of simple linear regression with measurement error';
    run;

The results of the simulations are given below:

Simulation of simple linear regression with measurement error

<table>
<thead>
<tr>
<th>N Obs</th>
<th>Variable</th>
<th>N</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Mean</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>25000</td>
<td>SEED</td>
<td>25000</td>
<td>1237491.00</td>
<td>1237491.00</td>
<td>1237491.00</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>25000</td>
<td>1.0000000</td>
<td>25000.00</td>
<td>12500.50</td>
<td>7217.02</td>
<td></td>
</tr>
<tr>
<td>MY</td>
<td>25000</td>
<td>10.0000000</td>
<td>10.0000000</td>
<td>10.0000000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SLOPE</td>
<td>25000</td>
<td>2.0000000</td>
<td>2.0000000</td>
<td>2.0000000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TRUEX</td>
<td>25000</td>
<td>-1.9999172</td>
<td>1.9997314</td>
<td>0.0054080</td>
<td>1.1524490</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>25000</td>
<td>-12.5333700</td>
<td>13.0171535</td>
<td>0.0290004</td>
<td>3.2240965</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>25000</td>
<td>-8.7730082</td>
<td>28.1123118</td>
<td>10.0090650</td>
<td>4.6130473</td>
<td></td>
</tr>
</tbody>
</table>

Simulation of simple linear regression with measurement error

<table>
<thead>
<tr>
<th>N Obs</th>
<th>Variable</th>
<th>N</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Mean</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>BETA0</td>
<td>250</td>
<td>8.7908251</td>
<td>11.7286628</td>
<td>10.0001125</td>
<td>0.5061296</td>
</tr>
<tr>
<td>BETA1</td>
<td>250</td>
<td>-0.2370609</td>
<td>0.6593156</td>
<td>0.2681453</td>
<td>0.1445254</td>
<td></td>
</tr>
</tbody>
</table>
Although the mean for y is still unbiased, the slope coefficient is biased. The expected value of the slope coefficient is biased towards zero. This result is typical of "errors in variable" problems.