EXTENDING THE SAS® SYSTEM ON MICROCOMPUTERS WITH SAS/IML™ SOFTWARE, THE INTERACTIVE MATRIX LANGUAGE

Linda P. Atkinson
Economic Research Service
U.S. Department of Agriculture

Introduction

The SAS® System contains powerful software for data analysis. However, it cannot be all things to all people. With SAS/IML™ software, the IML language (Interactive Matrix Language) can be used to extend the SAS System to include tailor-made features or personalized procedures to suit individual data processing needs.

IML is a programming language based on matrix algebra notation. A formula for a statistical method expressed in matrix notation can be easily programmed with IML commands. Thus, a statistical technique which has not yet been incorporated into SAS (or which is part of a SAS product not yet available at a particular site) can be coded without great difficulty in IML.

SAS/IML as it is implemented for personal computers also contains display features allowing the creation of windows for full-screen data entry or menuing. Data processing commands can query and extract information from SAS data sets. User-friendly applications can be developed which extend the SAS System similarly to the way SAS/AF and SAS/FSP do.

Syntax

The basic entity in IML is a two-dimensional matrix, where the first dimension represents the row and the second the column. A matrix can be character or numeric but not both. IML takes care of the dimensioning automatically if a variable is redefined. IML commands apply to entire matrices, which reduces the amount of looping that must be programmed.

An IML session is begun by submitting the command:

```
proc iml;
```

The message

```
IML ready
```

appears in the log window and interactive mode is entered (each statement is executed as it is submitted). It is also possible to define modules, or groups of statements to be saved for later execution (with START and FINISH statements), and to delay execution of a set of statements by enclosing the statements between DO; and END;

A matrix can be defined by assigning it elements enclosed in braces. For example, defining z to be a scalar equal to 1:

```
z={1};
```

This can be viewed with a PRINT command:

```
print z;
```

The same variable can be redefined as, for example, a row vector:

```
z={1 2 3};
print z;
```

If you would like to see all results printed automatically, use the RESET command:

```
reset print;
```

Rows of a matrix are separated by commas:

```
A={1 2 3, 4 5 6};
```

Character values can be entered directly or enclosed in quotes if they include spaces:

```
alpha={'a b c d e f g'};
agencies={'Economic Research Service',
          'USDA/NASS',
          'U.S. Postal Service'};
```

A particular element of a matrix is referenced with brackets:

```
A[2,3]=8;
```

A subscript can be left out to refer to an entire row or column:

```
b=A[2,];
```

A matrix can be defined as a submatrix of another one:

```
rows={1 2};
columns={2 3};
C=A[rows,columns];
```

Or more directly:

```
C=A[1:2,2:3];
```

A subscript reduction operator works on an entire dimension. To get column sums:

```
s=A[:,1];
```

These operators can work across both dimensions:

```
grand_s=A[:];
```

There are many operators and built-in functions of IML. These include:

**Addition**

```
A2=A+S;
A3=A+A2;
```

**Subtraction**

```
D=A3-A2;
```

**Matrix multiplication**

```
B=2*A;
F=A*E;
AF=AIIF;
```

**Transpose**

```
AFT=A';
```

**Inverse**

```
CINV=inv(C);
```

**Number of rows or columns in a matrix**

```
numrow(A);
numcol(A);
```
Display Features

IML allows you to extend the SAS System for user applications by creating windows which can be used for full-screen data entry or menus. The WINDOW statement defines a window and the DISPLAY statement displays it and accepts data. You can have any number of windows, of different sizes and positions, and scroll through them. Only one window is active at a time. The fields in a window are part of field groups, which are given names.

Each display area begins with a command line for entering commands and a message line for displaying messages. The rest of the window can be designed to include protected and/or unprotected fields.

The following code defines a window called "greeting". The display command which initially displays the window defines one field called "yourname" which has been initialized to blank. A message directs the user to enter a name and type "submit" on the command line. The window is then displayed again with a greeting and instructions for entering "goodbye" to quit the window. At that time the window is "closed" (its display is removed from the screen).

```
   yourname = ;
   window greeting rows=10 columns=50
   cmdline =
   cmddline = cmd color='yellow';
   msgline = 'Enter your name and type submit; on command line.';
   display greeting ('Name: ' yourname);
   if cmd = 'submit;' then do;
      cmnd = 'goodbye';
      msgline = 'Type goodbye on command line.';
   end;
   window close = greeting;
```

Examples

The preceding sections have presented a few of the features available in SAS/IML. Additional commands and options will be demonstrated in the following examples. For complete documentation, consult SAS/IML Guide for Personal Computers and Changes and Enhancements to SAS/IML Software for Personal Computers.

Example 1. Spreadsheet Analysis

Spreadsheet programs are very popular on microcomputers. They are used to store, manipulate, and display data. They are often used to conduct if-then types of analyses, because it is easy to change a portion of a spreadsheet and have the result carried through the rest of the spreadsheet. This example will demonstrate how the SAS System on microcomputers can be extended to include these features through the use of SAS/IML software.

Appendix A contains a listing of the program used in this example. After the proc iml; statement, an IML module called spread is defined as the code between the start spread; statement and the finish spread; statement. The statements between these are not immediately executed. Following the definition of the module, which will be performing calculations and producing tables, a data matrix called belgium is set up. This matrix consists of economic indicators for Belgium, with emphasis on the food sector, for the years 1970-1972 (each column represents a year). The second row will be calculated as the result of other rows in the matrix and is initialized with the SAS missing value indicator of a period (.)

Example

Examples
The loop operates from 1 to 3, that is, for each column (representing year) in the data matrices. First each element in the $i^{th}$ column of the *belgium* matrix is divided by the $i^{th}$ element in the third row of *stats* (that is, the exchange rate for the corresponding year). This calculates expenditures in U.S. dollars. Then the expenditures are divided by the corresponding population values (second row of *stats*) to create the per capita expenditures matrix. Finally, the food consumption matrix is divided by the second row of *belgium* (disposable personal income) and multiplied by 100 to determine the percent of disposable personal income spent on each of the food categories.

After the loop calculating the new matrices is another print command to display the results. Note also the use of formats to specify how the elements of the matrices are to be shown. The slash (/) between the second matrix and the third heading indicates that the results are to be printed with one command. Note also the use of the reset storage command. The reset storage command states which matrices and modules in the file are to be retrieved. If this were a new SAS session, as well as the display command, the module is closed, as are both windows, and the module is ended. If neither a blank nor exit are submitted for the name, then the name which is submitted is searched for in the SAS data set. If exit is typed in as a name then the program will end. Otherwise the program will search through the SAS data set for that name and display observations with that name in the *ed* window. The user can make changes to those observations and type submit; on the command line of the *ed* window to have the changes made to the SAS data set. If no observations matching the name entered are found, blank fields are displayed for entering a new user. If you did not want to enter a new user (for example, you had just misspelled the name), type cancel on the command line of the *ed* window. You are then returned to the *find* window to enter a new name.

Finally, proc *iml* is exited and reentered to show how you could retrieve results at a later time. The *libname* command is repeated as if this were a new SAS session, as well as the *reset storage* command. The show storage; statement requests *iml* to indicate what items are available for retrieval from the specified file. The load command brings specified items into memory. We have requested that all matrices and modules in the file be retrieved. One matrix, exchange, is printed, and one of its values is changed (for example, if an error had been made in data entry or one wanted to see the results of a different scenario). A request to run the *spread* module again results in all calculations and tables being produced with the new data.
to blank. If there were matches, the observations corresponding to the matched names are read in with point used as the range specification of the read command. The ed window is then displayed, either blank fields if there were no matches or with sets of filled-in fields of data for each match. If there was more than one match this window will contain more information than can be displayed on the screen in the allotted space. The user can use the Page Up and Page Down keys to scroll through this window, and make changes to the data fields or add information on blank fields.

If submit; is entered on the command line, a message is defined that the SAS data set is updated and a new name (or exit) can be entered. If this is a new observation to be added (there had not already been a matching name in the data set, i.e. nrow(p)=0) then the information in the data fields is appended to the SAS data set as a new observation. Otherwise the corresponding observations in the data set are replaced by the new data. The goto command directs the program to display the find window for entry of another name.

If cancel or, actually, anything other than submit; is entered on the command line of the ed window, then no changes are made to the SAS data set. A message to that effect is defined, which is displayed as the program returns to the display of the find window.

The finish; command concludes the definition of the findedit module. The rtdi command which executes the module causes the windows of the system to be displayed and the system is begun. When exit is entered as a name in the find window, windows are closed and IML is exited. A temporary data set is set up in this program for use in a proc tabulate step which concludes the data entry and editing system.

Example 3. Statistical Analysis

Much of our use of IML is in expressing in matrix language the steps and computations to perform a statistical technique not yet available in SAS. This is particularly true on the PC, since the SAS/ETS (Econometric and Time Series Library) software is not yet available for microcomputers. The following example implements an econometric estimation technique known as the Cochrane-Orcutt procedure for estimating a regression with correction for serial correlation.

It is often the case in dealing with time-series data that the error in a particular time period will be correlated with the error in the next period. This is referred to as first-order autocorrelation. The model can be represented as

\[ Y_t = \beta_1 + \beta_2 X_{2t} + \cdots + \beta_k X_{kt} + \epsilon_t \]  

(1)

where \( \epsilon_t = \rho \epsilon_{t-1} + \nu_t \)  

(2)

and \( \nu_t \) is independent of \( \nu_s \) for \( s \neq t \), and \( \epsilon_t \) is distributed as \( N(0, \sigma^2) \) but is not serially independent.

If equation (1) is written for time period \( t-1 \), one obtains

\[ Y_{t-1} = \beta_1 + \beta_2 X_{2t-1} + \cdots + \beta_k X_{kt-1} + \epsilon_{t-1} \]  

(3)

Multiplying (3) by \( \rho \) and subtracting from (1) yields

\[ Y_t^* = \beta_1 (1-\rho) + \beta_2 X_{2t} + \cdots + \beta_k X_{kt} + \nu_t \]  

(4)

where

\[ Y_t^* = Y_t - \rho Y_{t-1} \]

\[ X_{2t} = X_{2t} - \rho X_{2t-1} \]

\[ \vdots \]

\[ X_{kt} = X_{kt} - \rho X_{kt-1} \]

\[ \nu_t = \epsilon_t - \rho \epsilon_{t-1} \]  

(5)

This is referred to as generalized differencing. The transformed model (4) has an error process which is independently distributed with mean 0 and constant variance, by (2). Therefore ordinary least squares regression can be used to obtain efficient estimates of the parameters of the original model, with the intercept for equation (1) calculated from the estimated intercept associated with (4).

The Cochrane-Orcutt procedure involves a series of iterations. First ordinary least squares is used to estimate the original equation (1). The residuals are used to estimate \( \rho \) as the correlation coefficient associated with errors of adjacent time periods. That is, the regression is run

\[ \hat{\epsilon}_t = \rho \hat{\epsilon}_{t-1} + \nu_t \]  

(6)

The estimated value of \( \rho \) is used to transform the data as in (5) and ordinary least squares is performed on (4). From this estimation new residuals are obtained which can be used in (6) to get a new estimate of \( \rho \). The procedure is continued until some convergence criterion is met (typically, that succeeding estimates of \( \rho \) differ by less than some amount).

To use the estimated model for forecasting, it can be shown that the best linear unbiased predictor (BLUP) for \( Y \) at time period \( t+T \) would be

\[ \hat{Y}_{t+T} = \hat{\beta}_1 + \hat{\beta}_2 X_{2t+T} + \cdots + \hat{\beta}_k X_{kT} + \rho \hat{\epsilon}_T \]

(7)

That is, only the last residual in the sample contains information about the future disturbances, and since \( \rho \) is less than one the
effect of the information about the disturbance declines as you predict further into the future.

Appendix C contains the listing showing how this algorithm can be implemented in IML. A data set is created which contains variables dealing with meat consumption from 1964 through 1986 (quarterly observations). Variables included in the data set are beef prices, chicken prices, pork prices, beef consumption, pork consumption, chicken consumption, and personal expenditures. As the IML procedure is entered, the worksize is specified to increase the amount of memory allocated to the workspace. The default available can be viewed with the command show space;.

The module core is defined to have two arguments. The first, nabs, indicates how many of the observations in the input data set are to be used for estimation (the remainder will be used for forecasting). The second argument, itprint, will be equal to 1 if it is desired that estimates be printed at each iteration, 0 otherwise. The use command opens the meat data set for reading, and the subsequent read statement reads the selected variables indicated in braces after the var keyword. A subset of the full data matrix is created of the nobs rows to be used for estimation. The nrow function of IML is used to determine how many observations are available in the full data set.

The dependent variable is read from the data set using the point option to only bring in those observations needed for the estimation. The f function is used to create a column of ones for the intercept term in the regression. This is concatenated horizontally (||) with the original x matrix. Next follows the matrix algebra for ordinary least squares regression. The matrix multiplication, transpose of a matrix, inv being the inverse function, addition, subtraction, multiplication, indicating transpose to a power, the ssq function which returns the sum of squares of all the elements of its argument, and / for division.

From the ordinary least squares results, residuals are computed. A lagged vector of residuals is created, and the residuals are regressed against it. This yields an initial estimate of \( \rho \). An iterative loop is defined with the IML command do until which checks that successive estimates of \( \rho \) differ by less than .01. In each iteration, the data are transformed according to the present estimate of \( \rho \) and the equation is reestimated. The estimate of \( \beta \) is obtained from the intercept of the transformed equation by dividing by 1-\( \rho \) (rhatnew).

New residuals are used to estimate \( \rho \) as described above until the convergence criterion is met and the loop ends. Forecasts are then produced for the remainder of the data set not used in the estimation, and output to a SAS data set. The create statement sets up the new data set. As each forecast is produced, it is added to the output data set with append. At the conclusion of the calculation of the forecasts the data set is closed with the close statement.

The core module as defined is run by specifying values for its arguments in parentheses on the run statement. After exiting IML, the output SAS data set containing the forecasts can be printed or used in other SAS procedures.

Conclusion

The preceding examples have shown some of the ways in which SAS/IML software can be used to extend the SAS system on microcomputers. In the process, many of the IML commands, operators, functions, and display and data processing features have been demonstrated. There are many more, however. Besides the published documentation, the interested user is referred to the Sample Library distributed with SAS/IML software for further examples.

Acknowledgments

I would like to thank the facilitators of this workshop (Charlie Hallahan, USDA/ERS; Penni Korb, USDA/ERS; Dale Atkinson, USDA/NASS; and Ron Steele, U.S. Postal Service) for their assistance in preparing this presentation. Also, Penni Korb provided the data for Example 1 and Charlie Hallahan provided the data for Example 3.

References


SAS, SAS/AF, SAS/FSP, and SAS/ETS are registered trademarks of SAS Institute Inc., Cary, NC, USA. SAS/IML is the trademark of SAS Institute Inc., Cary, NC, USA. The use of trade or firm names in this paper is for reader information and does not imply endorsement by the U.S. Department of Agriculture of any product or service.

Author:

Linda P. Atkinson
USDA/Economic Research Service
1301 New York Avenue NW Room 240
Washington, D.C. 20005-4768
(202) 786-1505

699
This program performs spreadsheet-type operations on data matrices. It demonstrates defining matrices, performing mathematical calculations with matrices, printing tables with optional row and column labels and formats, printing character text as table headings, concatenating rows of data to form new matrices, referencing particular rows or elements of matrices, using DO loops, using the RESET command to direct destination of print and storage, defining program modules, and storing and retrieving modules and matrices.

options pagesize=55;
proc iml;
start spread;
years={'1970' '1971' '1972'};
items={'Gross Domestic Product',
'Total Personal Income',
'Private Final Consumption Expenditures',
'Food',
'Non-alcoholic Beverages',
'Alcoholic Beverages',
'Tobacco',
'Net Savings'};
belgium[2,]=belgium[3,]+belgium[9,];
reset nolog;
print 'BELGIUM IN CURRENT PRICES';
print '(in millions of Belgian francs)',
belgium[rowname=items colname=years];
stats=pop // pop/1000 // exchange;
statrows={' POPULATION in thousands',
'POPULATION in millions',
'EXCHANGE RATE'};
year=stats;
print 'BELGIUM STATISTICS';
year[rowname=statrows colname=year];
percap=belgium;
food=belgium[4:8,];
fooditem=items[4:8];
do i=1 to 3;

expend[i]=belgium[i]/stats[3,i]*100000;
percap[i]=expend[i]/stats[2,i];
food[i]=food[i]/belgium[2,i]*100;
end;
print 'EXPENDITURES (in millions of U.S. $)',
expend[rowname=items
colname=years format=6.0],
'PER CAPITA EXPENDITURES (in U.S. $)',
percap[rowname=items
colname=years format=6.0],
'PER CENT OF DPI - (%)',
food[rowname=fooditem colname=years
format=5.1];
reset log;
finish;
belgium={1280924 1402401 1568509, 759013 847717 947776, 241700 257472 280276, 185437 195068 212701, 8543 9727 11531, 32339 35483 39463, 15381 17194 19381, 158833 171072 199906};
exchange={5000000 4886970 4401460};
rout spread;
libname matrices '.';
reset storage='matrices.belgium';
store module=spread _all_; quit;
proc iml;
libname matrices '.';
reset storage='matrices.belgium';
show storage;
load _all_module=_all_; print exchange; exchange[2]=4787530;
rout spread;
quit;

700
### BELGIUM IN CURRENT PRICES

**(in millions of Belgian francs)**

<table>
<thead>
<tr>
<th></th>
<th>1970</th>
<th>1971</th>
<th>1972</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Gross Domestic Product</strong></td>
<td>1280924</td>
<td>1402401</td>
<td>1568509</td>
</tr>
<tr>
<td><strong>Disposable Personal Income</strong></td>
<td>927846</td>
<td>1018789</td>
<td>1147686</td>
</tr>
<tr>
<td><strong>Private Final Consumption Expend</strong></td>
<td>769013</td>
<td>847717</td>
<td>947778</td>
</tr>
<tr>
<td><strong>Food, Beverages and Tobacco</strong></td>
<td>241700</td>
<td>257472</td>
<td>282076</td>
</tr>
<tr>
<td><strong>Food</strong></td>
<td>185437</td>
<td>195068</td>
<td>212701</td>
</tr>
<tr>
<td><strong>Non-alcoholic Beverages</strong></td>
<td>8543</td>
<td>9727</td>
<td>11531</td>
</tr>
<tr>
<td><strong>Alcoholic Beverages</strong></td>
<td>32339</td>
<td>35483</td>
<td>39463</td>
</tr>
<tr>
<td><strong>Tobacco</strong></td>
<td>15361</td>
<td>17194</td>
<td>18381</td>
</tr>
<tr>
<td><strong>Net Savings</strong></td>
<td>158833</td>
<td>171072</td>
<td>189508</td>
</tr>
</tbody>
</table>

### BELGIUM STATISTICS

<table>
<thead>
<tr>
<th></th>
<th>1970</th>
<th>1971</th>
<th>1972</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>YEAR</strong></td>
<td>1970</td>
<td>1971</td>
<td>1972</td>
</tr>
<tr>
<td><strong>POPULATION in thousands</strong></td>
<td>9638</td>
<td>9672</td>
<td>9709</td>
</tr>
<tr>
<td><strong>POPULATION in millions</strong></td>
<td>9.638</td>
<td>9.672</td>
<td>9.709</td>
</tr>
<tr>
<td><strong>EXCHANGE RATE</strong></td>
<td>5000000</td>
<td>4886970</td>
<td>4401460</td>
</tr>
</tbody>
</table>

### EXPENDITURES (in millions of U.S. $)

<table>
<thead>
<tr>
<th></th>
<th>1970</th>
<th>1971</th>
<th>1972</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EXPEND</strong></td>
<td>1970</td>
<td>1971</td>
<td>1972</td>
</tr>
<tr>
<td><strong>Gross Domestic Product</strong></td>
<td>25618</td>
<td>28697</td>
<td>35636</td>
</tr>
<tr>
<td><strong>Disposable Personal Income</strong></td>
<td>20557</td>
<td>20847</td>
<td>26075</td>
</tr>
<tr>
<td><strong>Private Final Consumption Expend</strong></td>
<td>15380</td>
<td>17346</td>
<td>21533</td>
</tr>
<tr>
<td><strong>Food, Beverages and Tobacco</strong></td>
<td>4834</td>
<td>5269</td>
<td>6409</td>
</tr>
<tr>
<td><strong>Food</strong></td>
<td>3709</td>
<td>3992</td>
<td>4633</td>
</tr>
<tr>
<td><strong>Non-alcoholic Beverages</strong></td>
<td>171</td>
<td>199</td>
<td>262</td>
</tr>
<tr>
<td><strong>Alcoholic Beverages</strong></td>
<td>647</td>
<td>726</td>
<td>897</td>
</tr>
<tr>
<td><strong>Tobacco</strong></td>
<td>308</td>
<td>352</td>
<td>418</td>
</tr>
<tr>
<td><strong>Net Savings</strong></td>
<td>3177</td>
<td>3501</td>
<td>4542</td>
</tr>
</tbody>
</table>

### PER CAPITA EXPENDITURES (in U.S. $)

<table>
<thead>
<tr>
<th></th>
<th>1970</th>
<th>1971</th>
<th>1972</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PERCAP</strong></td>
<td>1970</td>
<td>1971</td>
<td>1972</td>
</tr>
<tr>
<td><strong>Gross Domestic Product</strong></td>
<td>2659</td>
<td>2967</td>
<td>3670</td>
</tr>
<tr>
<td><strong>Disposable Personal Income</strong></td>
<td>2155</td>
<td>2195</td>
<td>2898</td>
</tr>
<tr>
<td><strong>Private Final Consumption Expend</strong></td>
<td>1596</td>
<td>1793</td>
<td>2218</td>
</tr>
<tr>
<td><strong>Food, Beverages and Tobacco</strong></td>
<td>502</td>
<td>545</td>
<td>660</td>
</tr>
<tr>
<td><strong>Food</strong></td>
<td>385</td>
<td>413</td>
<td>498</td>
</tr>
<tr>
<td><strong>Non-alcoholic Beverages</strong></td>
<td>18</td>
<td>21</td>
<td>27</td>
</tr>
<tr>
<td><strong>Alcoholic Beverages</strong></td>
<td>67</td>
<td>75</td>
<td>92</td>
</tr>
<tr>
<td><strong>Tobacco</strong></td>
<td>32</td>
<td>36</td>
<td>43</td>
</tr>
<tr>
<td><strong>Net Savings</strong></td>
<td>330</td>
<td>362</td>
<td>468</td>
</tr>
</tbody>
</table>

### PER CENT OF DPI - (%)

<table>
<thead>
<tr>
<th></th>
<th>1970</th>
<th>1971</th>
<th>1972</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FOOD</strong></td>
<td>1970</td>
<td>1971</td>
<td>1972</td>
</tr>
<tr>
<td><strong>Food, Beverages and Tobacco</strong></td>
<td>26.0</td>
<td>25.3</td>
<td>24.6</td>
</tr>
<tr>
<td><strong>Food</strong></td>
<td>20.0</td>
<td>19.1</td>
<td>18.5</td>
</tr>
<tr>
<td><strong>Non-alcoholic Beverages</strong></td>
<td>0.9</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td><strong>Alcoholic Beverages</strong></td>
<td>3.5</td>
<td>3.5</td>
<td>3.4</td>
</tr>
<tr>
<td><strong>Tobacco</strong></td>
<td>1.7</td>
<td>1.7</td>
<td>1.6</td>
</tr>
</tbody>
</table>
This program sets up a system for data entry and editing. It demonstrates windowing, data processing with EDIT, FIND, READ, APPEND, REPLACE, and CLOSE commands, and programming statements such as DO WHILE, IF ... THEN ... ELSE, and GO TO. The matrix function NROW is used. An IML module is defined and run.

libname lpa '.';
options pagesize=55;
proc iml;

/*---INITIALIZE THE VARIABLES---*/
start initvars;
name=" ";
division=" ";
room=" ";
phone=" ";
finish;

start findedit;

/*---DEFINE THE WINDOWS---*/
window ed row=12 column=50 icolumn=1 cmdline=icolor="green";
window find row=5 column=70 icolumn=1 msg1 line=msg color="green".

/*---BRING IN DATASET FOR EDITING---*/
edit lpa.userlist;

/*---EDIT OR ADD NAMES UNTIL EXIT IS ENTERED AS A NAME---*/
do while(1);
    msg="
    name="
    display find ("Name: " color="black" name color="red" )
    if name=" " then do;
        msg="Enter exit as a name and submit to end"
        goto again;
        if name="exit" then goto x;
        find all where(name/name into p;
        if nrow(p)=0 then do;
            division=" ";
            room=" ";
            phone=" ";
            end;
            if nrow(p) ^= 0 then read point p;
            display ed ("NAME: " color="yellow" name color="red"
                division color="red"
                room color="red"
                phone color="red"
                phone color="red") repeat;
            if c='submit;' then do;
                msg="Data set updated, enter request"
                goto again;
            end;
            else replace point p;
            goto again;
            end;
            else do;
                msg="Data set not updated, enter request"
                goto again;
                end;
            end;
        x,
    display find ("Closing data set and exiting")
    close lpa.userlist;
    window close=ed;
    window close=find;
    finish;
    run findedit;
quilt;

/*---CREATE TEMPORARY DATA SET FOR TABULATING LIST---*/
data new; set lpa.userlist;
    rm=1; ph=1;
    rm=room; ph=phone;
    run;

/*---SUMMARIZE AND LIST USERS BY DIVISION---*/
proc tabulate missing;
title 'PC SAS USERS';
class division name;
var ra ph;
label rm='ROOM' ph='PHONE';
table division all, n*f=6.0;
table division, name, (rm ph)max*f=6.0
    n*f=6.0 / misstext=' '; keylabel max = ' n='copies';
run;
Appendix C

PC Hands-on Workshop (IML)
Example 3

This program performs the Cochrane-Orcutt procedure for estimating a model with first-order serial correlation. It demonstrates reading selected variables from a SAS data set into IML, defining and running a module that has arguments, and coding a technique expressed in matrix algebra notation in IML. Matrix operators and functions are used as well as program loops, indexing and subscripting.

data meat;
  infile 'meat.dat';
  input beefpr chknpr porkpr beefcons porkcons chkncons persexps;
run;

options pagesize=55;
proc iml worksize=50;
  reset nolog;
  start corc(nobs,ltprint);
  use meat;
  read all var{beefcons porkcons chkncons persexps} into xf;
* x is a subset of xf used for estimation;
  x=xf[1:nobs,];
  nobsf=nrow(xf) ;
  read point(l:nobs) var{beefpr} into y;
* create a column of ones for the intercept;
  unit=J(nobsf,1,1);
  x=unit[l:nobs,]Ix;
  iter=0;
* perform OLS on original equation;
  beta=inv(x'x)*x'y;
  ssy=sum(y)**2/nobs;
  ssreg=(beta'x)'y-(sum(y)**2)/nobs;
  rsq=ssreg/ssy;
  print beta ssy ssreg rsq;
* compute fitted values and residuals;
  yhat=x*beta;
  res=y-yhat;
  nobs=nobs-1;
  reslag=rs[1:nobs];
  resl=res[2:nobs];
* regress the residuals on the residuals lagged one period to get estimate of rho;
  rhatnew=inv(reslag''reslag)*reslag''resl;
  print iter rhatnew;
  do until (abs(rhatnew-rhatold)< 0.01);
    iter=iter+1;
    transform the data (generalized differencing);
    newy=y[2:nobs]-rhatnew*y[1:nobs];
    newx=x[2:nobs,]-rhatnew*x[1:nobs,];
    newx[,1]=unit[1:nobs];
  * re-estimate equation;
    beta=inv(newx''newx)*newx''newy;
    beta[1]=beta[1]/(1.0-rhatnew);
* get new fitted values and residuals;
    yhat=x*beta;
    res=y-yhat;
    reslag=rs[1:nobs];
    resl=res[2:nobs];
    rhatold=rhatnew;
* get new estimate of rho, saving old estimate for comparison;
    rhatnew=inv(reslag''reslag)*reslag''resl;
    if itprint=1 then
      print iter beta rhatnew;
  end;
  x=unit[1:lx];
* produce forecasts, output to SAS data set;
  create fcset var{forecast};
  do t=nobs to (nobs:f-l);
    tn=t-newobs;
    forecast=C-xr[t+l,]*beta+(rhatnew*tn)*res[nobs];
    append;
  end;
close fcset;
finish;
run corc(75,1);
reset log;
quit;
proc print data=fcset;
run;
### SAS 11:02 Thursday, March 17, 1988

#### BETA SSY SSREG RSQ

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BETA</td>
<td>75.670375</td>
<td>4070.9583</td>
<td>586.34517</td>
</tr>
<tr>
<td>-0.599812</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.058246</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.7183641</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ITER RHTANEW**

<table>
<thead>
<tr>
<th>ITER</th>
<th>RHTANEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.62515184</td>
</tr>
</tbody>
</table>

**ITER BETA RHTANEW**

<table>
<thead>
<tr>
<th>ITER</th>
<th>BETA</th>
<th>RHTANEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>149.79968</td>
<td>.89015964</td>
</tr>
<tr>
<td>-2.205477</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-.8837557</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.24228202</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-.001959</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ITER BETA RHTANEW**

<table>
<thead>
<tr>
<th>ITER</th>
<th>BETA</th>
<th>RHTANEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>156.23299</td>
<td>.89439142</td>
</tr>
<tr>
<td>-2.312329</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-.9440734</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.07832139</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-.0029867</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Final Estimates

**ITER BETA RHTANEW**

<table>
<thead>
<tr>
<th>ITER</th>
<th>BETA</th>
<th>RHTANEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>156.23299</td>
<td>.89439142</td>
</tr>
<tr>
<td>-2.312329</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-.9440734</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.07832139</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-.0029867</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### OBS FORECAST

<table>
<thead>
<tr>
<th>OBS</th>
<th>FORECAST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>84.9870</td>
</tr>
<tr>
<td>2</td>
<td>87.3079</td>
</tr>
<tr>
<td>3</td>
<td>85.8826</td>
</tr>
<tr>
<td>4</td>
<td>83.9207</td>
</tr>
<tr>
<td>5</td>
<td>84.1908</td>
</tr>
<tr>
<td>6</td>
<td>87.4707</td>
</tr>
<tr>
<td>7</td>
<td>87.7586</td>
</tr>
<tr>
<td>8</td>
<td>86.8801</td>
</tr>
<tr>
<td>9</td>
<td>85.2166</td>
</tr>
<tr>
<td>10</td>
<td>88.9871</td>
</tr>
<tr>
<td>11</td>
<td>86.5140</td>
</tr>
<tr>
<td>12</td>
<td>84.8362</td>
</tr>
<tr>
<td>13</td>
<td>97.2385</td>
</tr>
<tr>
<td>14</td>
<td>88.9771</td>
</tr>
<tr>
<td>15</td>
<td>86.4163</td>
</tr>
<tr>
<td>16</td>
<td>86.1451</td>
</tr>
<tr>
<td>17</td>
<td>88.3985</td>
</tr>
</tbody>
</table>