MATLAB
for Economics and Econometrics
A Beginners Guide

John C. Frain

TEP Working Paper No. 0414

November 2014

Trinity Economics Papers
Department of Economics
Trinity College Dublin
MATLAB
for Economics and Econometrics
A Beginners Guide

John C. Frain
Economics Department
Trinity College Dublin 1

17th November 2014

1Comments are welcome. My email address is frainj@tcd.ie
This beginners’ guide to MATLAB for economics and econometrics is an updated and extended version of [Frain (2010)](https://doi.org/10.1017/CBO9781107705025). The examples and illustrations here are based on Matlab version 8.3 (R2014a).

It describes the new MATLAB Desktop, contains an introductory MATLAB session showing elementary MATLAB operations, gives details of data input/output, decision and loop structures, elementary plots, describes the LeSage econometrics toolbox and shows how to do maximum likelihood estimation. Various worked examples of the use of MATLAB in economics and econometrics are also given. I see MATLAB not only as a tool for doing economics/econometrics but as an aid to learning economics/econometrics and understanding the use of linear algebra there. This document can also be seen as an introduction to the MATLAB on-line help, manuals and various specialist MATLAB books.
## Contents

1 Introduction .......................................................... 1
   1.1 Preliminaries ..................................................... 1
   1.2 The MATLAB Desktop ............................................ 4
   1.3 Desktop Windows .................................................. 7
       1.3.1 The Command Window ...................................... 7
       1.3.2 The Command History Window .............................. 9
       1.3.3 Current Folder Window .................................... 9
       1.3.4 The Editor Window ......................................... 9
       1.3.5 Graphics Windows .......................................... 11
       1.3.6 The Workspace Browser .................................... 11
       1.3.7 The Path Browser ........................................... 12
       1.3.8 The Help System ........................................... 13
       1.3.9 Miscellaneous Commands ................................... 27

2 Basic Matlab and Some introductory Examples .................... 29
   2.1 Sample MATLAB session in the Command Window ............... 29
       2.1.1 Entering Matrices ........................................ 30
       2.1.2 Basic Matrix operations ................................... 31
       2.1.3 Kronecker Product ......................................... 34
       2.1.4 Examples of number formats ............................. 35
       2.1.5 fprintf function .......................................... 36
CHAPTER 1

Introduction

1.1 Preliminaries

These notes are a guide for students of economics/econometrics who wish to learn MATLAB. Throughout there is an emphasis on MATLAB as used in MS Windows. Apart from interaction with the operating system what is set out here transfers to MATLAB running under Linux. I have not used an Apple PC but I would presume that a similar statement holds.

To get the best benefit from these notes you should read them sitting in front of a computer entering the various MATLAB instructions in the examples and running them as you read the notes. The material in the first three chapters is elementary and will be required by all economists starting with MATLAB. The remaining sections contain some more advanced material and should be read as required.

In these notes I have used a mono-spaced font for MATLAB instructions and computer input/output. Often this material is set in boxes similar to those, for example, on page 30. Here the boxes are divided with the upper part containing MATLAB code and the lower the output arising from that code. Descriptive material, explanations and commentary
on the computer input/output is given in the current font.

While the first aim of these notes is to get the reader started in the use of MATLAB for econometrics it should be pointed out that MATLAB has many uses in economics. In recent years it has been used widely in what is known as computational economics/finance. This has applications in macroeconomics, determination of optimal policies and in finance. Recent references include Cerrato (2012), Kienitz and Wetterau (2012), Anitța et al. (2011), Huynh et al. (2008), Lim and McNelis (2008), Kendrick et al. (2006), Ljungqvist and Sargent (2004), Miranda and Fackler (2002) and Marimon and Scott (1999).

I do not know of any book on MATLAB written specifically for economics. Creel (2014) is a set of lecture notes on econometrics which can be downloaded from the web. This contains examples of econometric analysis using GNU Octave which has a syntax similar to MATLAB (see section [10.1]). LeSage (1999) is a free econometrics toolbox available for download from http://www.spatial-econometrics.com/. This site also contains links to several other MATLAB resources useful in econometrics. A free econometrics for finance toolbox is available at http://www.kevinsheppard.com/HFE_Toolbox.

MathWorks, the composers of MATLAB have a list of books using MATLAB for Economics/Finance (http://www.mathworks.co.uk/support/books/index_by_categorytitle.html?category=4). They have also issued a new econometrics toolbox (see http://www.mathworks.com/products/econometrics/). The MathWorks overview of this toolbox indicates that is is targeted at econometric time series in finance.

For advanced applications in applied probability Paolella (2006, 2007) are comprehensive accounts of computational aspects of probability theory using MATLAB. Higham and Higham (2005) is a good book on MATLAB intended for all users of MATLAB. Pratap (2006) is a good general “getting started” book. There are also many excellent books covering MATLAB for Engineers and/or Scientists which you might find useful if you need to use MATLAB in greater depth. The file exchange section of the MathWorks website contains contributed toolboxes, functions and other files of interest to economists.

These notes can not give a comprehensive account of MATLAB. Your copy of MATLAB comes with one of the best on-line help systems available. Full versions of the manuals are available in portable document format on the web at http://www.mathworks.com.
MATLAB started life, in the late 70’s, as a computer program for handling matrix operations. Over the years it has been extended and the basic version of MATLAB now contains more than 1000 functions. Various “toolboxes” have also been written to add specialist functions to MATLAB. Anyone can extend MATLAB by adding their own functions and/or toolboxes. Any glance at an econometrics textbook shows that econometrics involves much matrix manipulation and MATLAB provides an excellent platform for implementing the various textbook procedures and other state of the art estimators. Before you use MATLAB to implement procedures from your textbook you must understand the matrix manipulations that are involved in the procedure. When you implement them you will understand the procedure better. Using a black box package may, in some cases, be easier but how often do you know exactly what the black box is producing. Using MATLAB for econometrics may appear to involve a lot of extra work but many students have found that it helps their understanding of both matrix theory and econometrics. They then are better equipped to make use of black box based approaches.

In MATLAB as it all other packages it makes life much easier if you organize your work properly. The procedure That I use is some variation of the following –

1. Set up a new directory for each project (e.g. s:\MATLAB\project1)

2. Set up a short-cut for each project. The short-cut should specify that the program start in the data directory for the project. If all your work is on the same PC the short-cut is best stored on the desktop. If you are working on a PC in a computer lab you will not be able to use the desktop properly and the short-cut may be stored in the directory that you have set up for the project. If you have several projects in hand you should set up separate short-cuts and directories for each of them. Each short-cut should be renamed so that you can associate it with the relevant project.

3. Before starting MATLAB you are strongly advised to amend the options in Windows explorer so that full file names (including any file extensions allocated to programs) appear in Windows Explorer and any other Windows file access menus.
1.2 The MATLAB Desktop

The current MATLAB Graphical User Interface (GUI) follows the style of the tabs and ribbon interface introduced in Microsoft Office 2007 and developed in later versions of that program. If you are familiar with the modern Microsoft Office interface you will find the MATLAB one easier to use.

When you start MATLAB you will be presented with the MATLAB desktop. The current default start-up will be similar to that displayed in figure 1.1. In this default 5 windows are displayed:

1. **The Command Window** — This is where you can enter and execute MATLAB commands and display any output.
2. **The Editor Window** — This is where you edit MATLAB files. These files may be script files containing a sequence of Matlab instructions for later execution, definitions of user functions or other text files.
3. **The Command History Window** — This contains a list of commands issued from the command window.
4. **The Workspace or Variables Window** — Here the objects created during the current session are listed. Double clicking on an item in this window opens the item in the Editor where you may examine it or edit it.
5. **The Current or Work Folder Window** is your project directory. At start-up this is the directory that you should have specified in the MATLAB short-cut (see page 3).

A single click on a window makes that window the active window.

In the top left hand corner of each window you will see a ▼ sign in a circle. Right clicking on this brings up a context menu that allows you to do several thing with the window. The full list of actions available depends on the particular window. In particular, you can use this menu to close a window.

In the default desktop the windows are docked to or fixed within the desktop. The windows can also be undocked using this context menu. If you have a smaller screen you may find that you have not got sufficient area to support all 5 screens. In such a case I would like a larger area for the editor and command windows. Undocking the editor

---

1 The new interface was introduced in R2012b. There is an account of the previous MATLAB 7 interface in the earlier edition of these notes [Frain 2010].
window removes it from the desktop and allows it to float on your screen. The editor window then takes up the space that was occupied by the editor window. The keyboard short cut \texttt{Ctrl}+\texttt{Shift} + \texttt{U} undocks the active window while the sequence \texttt{Ctrl} + \texttt{Shift} + \texttt{D} docks it again.

There are 6 tabs across the top of the MATLAB desktop in figure 1.1:

1. HOME
2. PLOTS
3. APPS
4. EDITOR
5. PUBLISH
6. VIEW

If you are editing a variable in the editor window the EDITOR and PUBLISH tabs are replaced by a VARIABLE tab. If no file is open in the EDITOR and no data is being edited only the first three tabs are shown.

Immediately beneath the Tabs is the ribbon. The contents of the ribbon depend on which Tab is active. The contents are divided into groups. For example the HOME tab is divided into 6 groups -

1. FILE — Here you will find the resources necessary to manage your files
2. VARIABLE — This group contains the facilities to import/save/edit data
3. CODE
4. SIMULINK
5. ENVIRONMENT — Here you can set the MATLAB search path
6. RESOURCES

The APPS tab provides a menu access to MATLAB Apps. When you have entered your options on the menu(s) it generates the required MATLAB script and runs it. It can also generate a file containing the MATLAB script that it has generated. It is essential that you save this script. There are many options available in the APPS menus and it may be difficult to replicate your work if you depend only on your memory of what you did in the GUI.

YOU will make a lot of use of the EDITOR and some use of the PUBLISH tab and I will cover these in greater detail later. Note that the EDITOR tab also has a FILE group that duplicates some of the functions of the HOME tab. Thus there is no need
1.2 The MATLAB Desktop

Figure 1.1: Basic Matlab GUI at start-up

to switch tabs when opening and saving files.

To the right of the tabs there is a set of icons giving quick access to some functions and HELP. to the right of this there is a search field for the help documentation.

While you can navigate the MATLAB desktop with a mouse, you can also easily navigate it from the keyboard. When you hold down the Alt key a series of letters/numbers appear across the tab bar as in figure 1.2. To select the tab or other item continue to hold the Alt key and press the key on the keyboard corresponding to the required item.

At this stage a further series of letters/numbers appear on the ribbon. Figure 1.3 shows the top left hand corner of the desktop when the HOME tab has been selected. Each item on the ribbon has been labelled with a letter. Just press that letter on the keyboard to access the relevant item.
1.3 Desktop Windows

1.3.1 The Command Window

The simplest use of the command window is as a calculator. With a little practice it may be as easy, if not easier, to use than a spreadsheet. Most calculations are entered almost exactly as one would write them.

```
>> 2+2
ans = 4

>> 3*2
ans = 6
```
The object \texttt{ans} contains the result of the last calculation of this kind. You may also create an object \texttt{a} which can hold the result of your calculation.

\begin{verbatim}
>> a=3^30
a = 27
>> a
a = 27
>> b=4^2+1
b = 17
>> b=4^2+1;

% continuation lines
>> 3+3 ...
+3
ans = 9
\end{verbatim}

Type each instruction in the command window, press enter and watch the answer.

Note

- The arithmetic symbols \(+\), \(-\), \(*\), \(/\) and \(^\) have their usual meanings
- The assignment operator =
- The MATLAB command prompt >>
- A ; at the end of a command suppresses output but any assignment is made or the command is completed
- If a statement will not fit on one line and you wish to continue it to a second type an ellipsis (\ldots) at the end of the line to be continued.

Individual instructions can be gathered together in an m-file and may be run together from that file (or script). An example of a simple m-file is given in the description of the Edit Debug window below. You may extend MATLAB by composing new MATLAB instructions using existing instructions gathered together in a m-file (or function file).

You may use the up down arrow keys to recall previous commands (from the current or
earlier sessions) to the Command Window. You may then edit the recalled command before running it. Further access to previous commands is available through the command window.

### 1.3.2 The Command History Window

If you now look at the Command History Window you will see that as each command was entered it was copied to the Command History Window. This contains all commands previously issued unless they are specifically deleted. To execute any command in the command history double click it with the left mouse button. To delete a commands from the history select them, right click the selection and select delete from the drop down menu.

At the prompt in the **Command Window** you may also access the **Command History** using the \[ \text{[} \text{]} \text{ and } \text{[} \text{]} \text{ keys. This places the commands one by one at the MATLAB prompt. When you have located the prompt you can use the } \text{[} \text{]} \text{, } \text{[} \text{]} \text{ keys or the mouse to position the cursor and edit the command. If you type the start of a command the } \text{[} \text{]} \text{ and } \text{[} \text{]} \text{ keys will only bring up previous commands that start with the fragment that you have entered.} \]

### 1.3.3 Current Folder Window

This window displays the contents of the working or project directory. The name of this directory is given in the row below the ribbon. You can change the default by clicking on the part of the part of the displayed path that corresponds to the start of the new path and then negotiating to the new path in the Current Folder directory.

You can open m-files in the editor by double-clicking on the file name in the list.

### 1.3.4 The Editor Window

Clearly MATLAB would not be of much use if, every time you used it, one you had to re-enter or retrieve your commands one by one in the Command Window. You can save your commands in an m-file and run the entire set or a selection of the commands in the file. The MATLAB editor has facilities editing and saving your m-file, for deleting
commands or adding new commands to the file before re-running it. Set up and run the simple example below. We shall be using more elaborate examples later.

You can set up a new m-file by selecting **new** and **script** from the **HOME** or **EDITOR** tab. Enter the following in the file

```matlab
% vol_sphere.m
% John C Frain revised 12 November 2006
% This is a comment line
% This M-file calculates the volume of a sphere
echo off
r=2
volume = (4/3) * pi * r^3;
string= ['The volume of a sphere of radius ' ...
         num2str(r) ' is ' num2str(volume)];
disp(string)
% change the value of r and run again
```

In the **EDITOR** tab select **save** and **save as** and name the file as **vol_sphere**. (This will be saved in your default directory if you have set up things properly. Check that this is working properly).

Now return to the Command Window and enter **vol_sphere**. If you have followed the instructions properly MATLAB will process this as if it were a MATLAB instruction. You can change the value of the radius of the sphere in the editor and re-run the file.

The **EDITOR Window** is a programming text editor with various features colour coded. Comments are in green, variables and numbers in black, incomplete character strings in red and language key-words in blue. This colour coding helps to identify errors in a program.

The **EDITOR Window** also provides debug features for use in finding errors and verifying programs. In particular you may set a break point in the file and then run the commands in the file one by one. For example in if you have the **vol_sphere.m** open in

---

2If you are reading this on a computer your pdf reader may allow you to copy and paste material from my boxes to the MATLAB editor. If some symbols do not copy and paste properly from the you may need to edit the file. Most of the examples in this book can be cut and pasted to the editor to save typing.
1. Notice that there is a — next to each line that contains an executable MATLAB command.
2. Left-click on the — next to `echo off` and the — is replaced by a small red circle. This marks the breakpoint.
3. Now run the file from the ribbon. The script runs as far as the break point.
4. five new items have appeared on the ribbon
   - **Continue** Continue running from breakpoint
   - **Step** Run next line
   - **Step in** Run next line and step into function.
   - **Step out** Run until current function returns
   - **Run to Cursor** Run to line containing cursor.
5. As you step through the m-file watch the output in the **COMMAND WINDOW** and the variables in the **WORKSPACE WINDOW**. You can enter various commands in the COMMAND WINDOW if you need to check that everything is going as expected.

You should return to the description of the **EDITOR WINDOW** when you start editing files.

### 1.3.5 Graphics Windows

This is used to display graphics generated in MATLAB. Details will be given later when we are dealing with graphics (chapter [5 on page 89](#)).

### 1.3.6 The Workspace Browser

This is an option in the lower left hand corner of the desktop. Compare this with the material in the command window. Note that it contains a list of the variables already defined. Double clicking on an item in the workspace browser allows one to view and edit it.

The contents of the workspace can also be listed by the `whos` command
1.3 Desktop Windows

1.3.7 The Path Browser

MatLab comes with a large number of functions defined in m-files in various directories. You may also create your own functions and variables. MATLAB has various rules to find these functions, m-files and variables and if two of them have the same name to determine which has precedence.

1. When MATLAB encounters a name it looks first to see if it is a variable name. If it is a variable name the variable takes precedence and any function or m-file with the same name is blocked.

2. It then searches for the name as an m-file in the current directory. (This is one of the reasons to ensure that the program starts in the current directory). In this way you can redefine a MATLAB function and replace it with your own function.

3. The MATLAB search path is a list of directories that MATLAB searches sequentially for any other m-files or functions required. Starting at the first directory in the search path it uses the first such m-file or function found and ignores any in a later directory.

Thus, if one of your variables has the same name as an m-file or a MATLAB function you will not be able to access that m-file or MATLAB function. This is a common cause of problems. If, for example, you have name a variable inv or one of your own functions, in the current session, you will be unable to use the MATLAB inv() function. One way of checking that, for example, inv() is a MATLAB function would be to enter help inv on the command line. This will produce summary help file for inv if inv() is a MATLAB function. This is a common cause of problems in MATLAB but is easily fixed.

The MATLAB search path can be added to or changed at any stage by selecting set path in the ENVIRONMENT section or the HOME tab. Here you can make the following changes to the MATLAB path

add Folder Adds a directory to the MATLAB search path

Add with subfolders Adds a directory and its subdirectories to the MATLAB search path.

Remove folder Removes a directory from the MATLAB search path.

Change the order of directories in the path If there are two versions of a com-
mand in two different directories MATLAB will find the one closest to the top of the path, will use this and ignore the other.

If you need the changes in the current session only click on Close. If you want to make the changes permanent click on Save. The MATLAB command addpath can effect these changes from the COMMAND WINDOW.

The command cd changes the current working directory

1.3.8 The Help System

The help system in MATLAB is very good. It can be accessed in many ways. Perhaps the most obvious access to the help system is the Search Documentation invitation in the box on the extreme right hand side of the tab bar. For example suppose we want to find the inverse of a matrix and have forgotten the command. If you type inverse there you will be presented with the box displayed in figure 1.4. On this occasion the first item in the box shows that the inv() function calculates the inverse of a matrix.

To get additional details about the inv() function click on the first item on the list and you will be presented with the help window displayed in figure 1.5.

Alternatively you can click the ? key next to the Search Documentation box (or press the F1 key) to access the product documentation.

The ? key is repeated on the ribbon for the HOME tab. Below this there is a drop-down menu giving access to a variety of introductory examples and videos.
Figure 1.4: Using *Search Documentation* on Tab Bar
Figure 1.5: Help for \texttt{inv()} function
One can also type `help` at the command prompt to get a list of available help topics for MATLAB and installed MATLAB toolboxes. For example the econometrics tool box is installed on this PC when I type `help` at the command prompt I get (with many lines on other topics deleted)

<table>
<thead>
<tr>
<th>help on Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;&gt; help</td>
</tr>
<tr>
<td>HELP topics:</td>
</tr>
<tr>
<td>Documents\MATLAB</td>
</tr>
<tr>
<td>matlab\testframework</td>
</tr>
<tr>
<td>matlab\demos</td>
</tr>
<tr>
<td>matlab\graph2d</td>
</tr>
<tr>
<td>matlab\graph3d</td>
</tr>
<tr>
<td>matlab\graphics</td>
</tr>
<tr>
<td>econ\econ</td>
</tr>
<tr>
<td>econ\econdemos</td>
</tr>
<tr>
<td>finance\finance</td>
</tr>
<tr>
<td>finance\calendar</td>
</tr>
<tr>
<td>finance\finsupport</td>
</tr>
<tr>
<td>finance\ftseries</td>
</tr>
</tbody>
</table>

The help for the econometrics package described here is perhaps a little complicated. A simpler example would be to look at the help for the `inv()` function as in the box in this footnote. You can return to the help files for the econometrics package when you are reading section 8.3

```matlab
>> help inv
inv    Matrix inverse.
inv(X) is the inverse of the square matrix X.
A warning message is printed if X is badly scaled or nearly singular.
```

**Reference page in Help browser**

```matlab
doc inv
```
help on Command Line (cont.)

<table>
<thead>
<tr>
<th>Series Functions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>finance\findemos</td>
</tr>
</tbody>
</table>

***************lines deleted**************

| optim\optim       | - Optimization Toolbox |
| optim\optimdemos  | - Demonstrations.     |

***************lines deleted**************

| stats\stats       | - Statistics Toolbox  |
| stats\classreg    | - (No table of contents file) |
| stats\clustering  | - (No table of contents file) |
| stats\statsdemos  | - Statistics Toolbox --- Demos |

You can now click on econ\econ to get a list of help items in that toolbox.

help on econ\econ

Econometrics Toolbox
Version 3.0 (R2014a) 30-Dec-2013

== Model Specification & Testing ==

adftest   - Augmented Dickey-Fuller test for a unit root
aicbic     - Akaike and Bayesian information criteria
archtest   - Engle test for residual heteroscedasticity
autocorr   - Sample autocorrelation
collintest - Belsley collinearity diagnostics
corrplot   - Plot variable correlations
crosscorr  - Sample cross-correlation
egcitest    - Engle-Granger cointegration test
hac        - Heteroscedasticity and autocorrelation consistent
covariance estimators
i10test    - Paired integration/stationarity tests
jcitest    - Johansen cointegration test
jcontest   - Johansen constraint test
kpsptest   - KPSS test for stationarity
lbqtest    - Ljung-Box Q-test for residual autocorrelation
### 1.3 Desktop Windows

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lmtest</code></td>
<td>Lagrange multiplier test of model specification</td>
</tr>
<tr>
<td><code>lmctest</code></td>
<td>Leybourne-McCabe test for stationarity</td>
</tr>
<tr>
<td><code>lratiotest</code></td>
<td>Likelihood ratio test of model specification</td>
</tr>
<tr>
<td><code>parcorr</code></td>
<td>Sample partial autocorrelation</td>
</tr>
<tr>
<td><code>pptest</code></td>
<td>Phillips-Perron test for a unit root</td>
</tr>
<tr>
<td><code>vratiotest</code></td>
<td>Variance ratio test for a random walk</td>
</tr>
<tr>
<td><code>waldtest</code></td>
<td>Wald test of model specification</td>
</tr>
</tbody>
</table>

== Univariate Time Series Analysis ==

**Data Filtering**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hpfilter</code></td>
<td>Hodrick-Prescott filter for trend and cyclical components</td>
</tr>
</tbody>
</table>

**ARIMAX/ARMAX/GARCH Specification**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arima</code></td>
<td>Create an ARIMA model</td>
</tr>
<tr>
<td><code>egarch</code></td>
<td>Create an EGARCH conditional variance model</td>
</tr>
<tr>
<td><code>garch</code></td>
<td>Create a GARCH conditional variance model</td>
</tr>
<tr>
<td><code>gjr</code></td>
<td>Create a GJR conditional variance model</td>
</tr>
<tr>
<td><code>regARIMA</code></td>
<td>Create a regression model with ARIMA time series errors</td>
</tr>
</tbody>
</table>

**ARIMAX/ARMAX/GARCH Modeling**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arima/estimate</code></td>
<td>Estimate ARIMA model parameters</td>
</tr>
<tr>
<td><code>egarch/estimate</code></td>
<td>Estimate EGARCH model parameters</td>
</tr>
<tr>
<td><code>garch/estimate</code></td>
<td>Estimate GARCH model parameters</td>
</tr>
<tr>
<td><code>gjr/estimate</code></td>
<td>Estimate GJR model parameters</td>
</tr>
<tr>
<td><code>regARIMA/estimate</code></td>
<td>Estimate parameters of a regression model with ARIMA errors</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arima/filter</code></td>
<td>Filter disturbances through an ARIMA model</td>
</tr>
<tr>
<td><code>egarch/filter</code></td>
<td>Filter disturbances through an EGARCH(P,Q) model</td>
</tr>
<tr>
<td><code>garch/filter</code></td>
<td>Filter disturbances through a GARCH(P,Q) model</td>
</tr>
<tr>
<td><code>gjr/filter</code></td>
<td>Filter disturbances through a GJR(P,Q) model</td>
</tr>
<tr>
<td><code>regARIMA/filter</code></td>
<td>Filter disturbances through a regression model with ARIMA errors</td>
</tr>
</tbody>
</table>
### help on econ (cont.)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arima/forecast</code></td>
<td>Forecast ARIMA model responses and conditional variances</td>
</tr>
<tr>
<td><code>egarch/forecast</code></td>
<td>Forecast EGARCH model conditional variances</td>
</tr>
<tr>
<td><code>garch/forecast</code></td>
<td>Forecast GARCH model conditional variances</td>
</tr>
<tr>
<td><code>gjr/forecast</code></td>
<td>Forecast GJR model conditional variances</td>
</tr>
<tr>
<td><code>regARIMA/forecast</code></td>
<td>Forecast responses of a regression model with ARIMA errors</td>
</tr>
<tr>
<td><code>arima/infer</code></td>
<td>Infer ARIMA model innovations and conditional variances</td>
</tr>
<tr>
<td><code>egarch/infer</code></td>
<td>Infer EGARCH model conditional variances</td>
</tr>
<tr>
<td><code>garch/infer</code></td>
<td>Infer GARCH model conditional variances</td>
</tr>
<tr>
<td><code>gjr/infer</code></td>
<td>Infer GJR model conditional variances</td>
</tr>
<tr>
<td><code>regARIMA/infer</code></td>
<td>Infer innovations of a regression model with ARIMA time series errors</td>
</tr>
<tr>
<td><code>arima/simulate</code></td>
<td>Simulate ARIMA model responses and conditional variances</td>
</tr>
<tr>
<td><code>egarch/simulate</code></td>
<td>Simulate EGARCH model conditional variances</td>
</tr>
<tr>
<td><code>garch/simulate</code></td>
<td>Simulate GARCH model conditional variances</td>
</tr>
<tr>
<td><code>gjr/simulate</code></td>
<td>Simulate GJR model conditional variances</td>
</tr>
<tr>
<td><code>regARIMA/simulate</code></td>
<td>Simulate a regression model with ARIMA time series errors</td>
</tr>
<tr>
<td><code>arima/impulse</code></td>
<td>Impulse response (dynamic multipliers) of an ARIMA model</td>
</tr>
<tr>
<td><code>regARIMA/impulse</code></td>
<td>Impulse response (dynamic multipliers) of regression with ARIMA errors</td>
</tr>
<tr>
<td><code>regARIMA/arima</code></td>
<td>Convert a regression model with ARIMA errors to an ARIMAX model</td>
</tr>
</tbody>
</table>

### Utilities

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>garchar</code></td>
<td>Convert ARMA model to AR model</td>
</tr>
</tbody>
</table>
help on econ\econ (cont.)
garchma - Convert ARMA model to MA model
lagmatrix - Create matrix of lagged time series
price2ret - Convert prices to returns
recessionplot - Add recession bands to time series plot
ret2price - Convert returns to prices

== Multivariate Time Series Analysis ==

VARMAX Specification
vgxget - Get VARMAX model specification parameters
vgxset - Set VARMAX model specification parameters

VARMAX Modeling
vgxinfer - Infer VARMAX model innovations
vgxplot - Plot VARMAX model responses
vgxpred - Forecast VARMAX model responses
vgxproc - Generate VARMAX model responses from innovations
vgxsim - Simulate VARMAX model responses
vgxvarx - Estimate VARX model parameters

VARMAX Utilities
vartovec - Vector autoregression (VAR) to vector error-correction (VEC)
vectovar - Vector error-correction (VEC) to vector autoregression (VAR)
vgxar - Convert VARMA model to VAR model
vgxcount - Count VARMAX model parameters
vgxdisp - Display VARMAX model parameters and statistics
vgxloglik - VARMAX model loglikelihoods
vgxma - Convert VARMA model to VMA model
vgxqual - Test VARMAX model for stability/invertibility

SSM Specification
ssm - Create a state-space model
### SSM Modeling

- `ssm/disp` - Display summary information of state-space models
- `ssm/estimate` - Maximum likelihood parameter estimation of state-space models
- `ssm/filter` - Forward recursion of state-space models
- `ssm/forecast` - Forecast states and observations of state-space models
- `ssm/refine` - Refine initial parameters to aid estimation of state-space models
- `ssm/simulate` - Simulate observations and states of state-space models
- `ssm/smooth` - Backward recursion of state-space models

### Lag Operator Polynomials

- `LagOp` - Create a lag operator polynomial (LagOp) object
- `LagOp/filter` - Apply a lag operator polynomial to filter a time series
- `LagOp/isEqLagOp` - Determine if two LagOp objects are the same mathematical polynomial
- `LagOp/isNonZero` - Find lags associated with non-zero coefficients of LagOp objects
- `LagOp/isStable` - Determine the stability a lag operator polynomial
- `LagOp/minus` - Lag operator polynomial subtraction
- `LagOp/mldivide` - Lag operator polynomial left division
- `LagOp/mrdivide` - Lag operator polynomial right division
- `LagOp/mtimes` - Lag operator polynomial multiplication
- `LagOp/plus` - Lag operator polynomial addition
- `LagOp/reflect` - Reflect lag operator polynomial coefficients around lag zero
- `LagOp/toCellArray` - Convert a lag operator polynomial object to a cell array

>>
estimate Estimate ARIMA model parameters
Syntax:
[EstMdl,EstParamCov,logL,info] = estimate(Mdl,Y)
[EstMdl,EstParamCov,logL,info] = estimate(Mdl,Y,param1,val1,...)

Description:
Given an observed univariate time series, estimate the parameters of an ARIMA model. The estimation process infers the residuals of the underlying response series and then fits the model to the response data via maximum likelihood.

Input Arguments:
Mdl - ARIMA model specification object, as produced by the ARIMA constructor or arima/estimate method.

Y - Response data whose residuals and conditional variances are inferred and to which the model Mdl is fit. Y is a column vector, and therefore a single path of the underlying series. The last observation of Y is the most recent.

Optional Input Parameter Name/Value Pairs:
'Y0' Presample response data, providing initial values for the model. Y0 is a column vector, and may have any number of rows, provided at least Mdl.P observations exist to initialize the model. If the number of rows exceeds Mdl.P, then only the most recent Mdl.P observations are used. If Y0 is unspecified, any necessary observations are backcasted (i.e., backward forecasted). The last row contains the most recent observation.

'E0' Mean-zero pre sample innovations, providing initial values for the model. E0 is a column vector, and may have any number of rows, provided sufficient observations exist to initialize the ARIMA model as well as any conditional variance model (the number of observations required is at least Mdl.Q, but may be more if a conditional variance model is included). If the number of rows exceeds the number necessary, then only the most recent observations
help on arima\estimate (cont.)

are used. If E0 is unspecified, any necessary observations are set to zero. The last row contains the most recent observation.

'V0' Positive pre sample conditional variances, providing initial values for any conditional variance model; if the variance of the model is constant, then V0 is unnecessary. V0 is a column vector, and may have any number of rows, provided sufficient observations exist to initialize the variance model. If the number of rows exceeds the number necessary, then only the most recent observations are used. If V0 is unspecified, any necessary observations are set to the average squared value of the inferred residuals. The last row contains the most recent observation.

'X' Matrix of predictor data used to include a regression component in the conditional mean. Each column of X is a separate time series, and the last row of each contains the most recent observation of each series. When pre sample responses Y0 are specified, the number of observations in X must equal or exceed the number of observations in Y; in the absence of pre sample responses, the number of observations in X must equal or exceed the number of observations in Y plus Mdl.P. When the number of observations in X exceeds the number necessary, only the most recent observations are used. If missing, the conditional mean will have no regression component regardless of the presence of any regression coefficients found in the model.

'Options' Optimization options created with OPTIMOPTIONS (or OPTIMSET). If specified, default optimization parameters are replaced by those in options. The default is an OPTIMOPTIONS object designed for the optimization function FMINCON, with 'Algorithm' = 'sqp' and 'TolCon' = 1e-7. See documentation for OPTIMOPTIONS (or OPTIMSET) and FMINCON for details.

'Constant0' Scalar initial estimate of the constant of the model. If missing, an initial estimate is derived from standard time series techniques.

'AR0' Vector of initial estimates of non-seasonal autoregressive coefficients. The number of coefficients in AR0 must equal the number of non-zero coefficients associated with the AR polynomial (excluding lag zero). If missing, initial estimates are derived from standard time series techniques.

'SAR0' Vector of initial estimates of seasonal autoregressive coefficients. The
number of coefficients in SAR0 must equal the number of non-zero coefficients associated with the SAR polynomial (excluding lag zero). If missing, initial estimates are derived from standard time series techniques.

'MA0' Vector of initial estimates of non-seasonal moving average coefficients. The number of coefficients in MA0 must equal the number of non-zero coefficients associated with the MA polynomial (excluding lag zero). If missing, initial estimates are derived from standard time series techniques.

'SMA0' Vector of initial estimates of seasonal moving average coefficients. The number of coefficients in SMA0 must equal the number of non-zero coefficients associated with the SMA polynomial (excluding lag zero). If missing, initial estimates are derived from standard time series techniques.

'Beta0' Vector of initial estimates of the regression coefficients. The number of coefficients in Beta0 must equal the number of columns in the predictor data matrix X (see above). If missing, initial estimates are derived from standard time series techniques.

'Dof0 Scalar initial estimate of the degrees-of-freedom parameter (used for t distributions only, and must exceed 2). If missing, the initial estimate is 10.

'Variance0 A positive scalar initial variance estimate associated with a constant-variance model, or a cell vector of parameter name-value pairs of initial estimates associated with a conditional variance model. As a cell vector, the parameter names must be valid coefficients recognized by the variance model. If missing, initial estimates are derived from standard time series techniques.

'Display' String or cell vector of strings indicating what information to display in the command window. Values are:

<table>
<thead>
<tr>
<th>VALUE</th>
<th>DISPLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>'off'</td>
<td>No display to the command window.</td>
</tr>
<tr>
<td>'params'</td>
<td>Display maximum likelihood parameter estimates, standard errors, and t statistics.</td>
</tr>
<tr>
<td></td>
<td>This is the default.</td>
</tr>
<tr>
<td>'iter'</td>
<td>Display iterative optimization information.</td>
</tr>
<tr>
<td>'diagnostics'</td>
<td>Display optimization diagnostics.</td>
</tr>
</tbody>
</table>
help on arima\estimate (cont.)

'full' Display 'params', 'iter', and 'diagnostics'.

Output Arguments:

**EstMdl** - An updated ARIMA model specification object containing the parameter estimates.

**EstParamCov** - Variance-covariance matrix associated with model parameters known to the optimizer. The rows and columns associated with any parameters estimated by maximum likelihood contain the covariances of the estimation errors; the standard errors of the parameter estimates are the square root of the entries along the main diagonal. The rows and columns associated with any parameters held fixed as equality constraints contain zeros. The covariance matrix is computed by the outer product of gradients (OPG) method.

**logL** - Optimized loglikelihood objective function value.

**info** - Data structure of summary information with the following fields:

- `exitflag` - Optimization exit flag (see FMINCON)
- `options` - Optimization options (see OPTIMOPTIONS)
- `X` - Vector of final parameter/coefficient estimates
- `X0` - Vector of initial parameter/coefficient estimates

Notes:

- Unspecified initial coefficient estimates are indicated by NaNs, which are derived from standard time series techniques.
- Missing values, indicated by NaNs, are removed from Y and X by listwise deletion (i.e., Y and X are merged into a composite series, and any row of the combined series with at least one NaN is removed), reducing the effective sample size. Similarly, missing values in the pre sample data Y0, E0, and V0 are also removed by listwise deletion (Y0, E0, and V0 are merged into a composite series, and any row of the combined series with at least one NaN is removed). The Y and X series, as well as the pre sample data, are also
synchronized such that the last (most recent) observation of each component series occurs at the same time.

- The parameters known to the optimizer and included in EstParamCov are ordered as follows:
  - Constant
  - Non-zero AR coefficients at positive lags
  - Non-zero SAR coefficients at positive lags
  - Non-zero MA coefficients at positive lags
  - Non-zero SMA coefficients at positive lags
  - Regression coefficients (models with regression components only)
  - Variance parameters (scalar for constant-variance models, vector of additional parameters otherwise)
  - Degrees-of-freedom (t distributions only)

- When 'Display' is specified, it takes precedence over the 'Diagnostics' and 'Display' selections found in the optimization 'Options' input. However, when 'Display' is unspecified, all selections related to the display of optimization information found in 'Options' are honoured

References:


See also arima, forecast, infer, simulate.

Reference page in Help browser
doc arima/estimate

When you are in the COMMAND WINDOW you have command completion and an
Say, for example we are interested in finding the eigenvalues and/or eigenvectors of a matrix. At the command line type `eig` and press the ` Enter ` key. A box appears showing a list of commands that start with `eig`. This is illustrated in figure 1.6. Double click on `eig` in that box and the command is completed in the COMMAND WINDOW. Now that the full version of the command is in the COMMAND WINDOW, type `( ` after the function and you will be presented with a set of hints on the completion of the function. This is illustrated in figure 1.7.

left click on the command

The more help at the end of the hints brings up the full help for the `eig` function.

Left-clicking on the `eig` function brings up a context menu which contains provides access to the HELP system.

The help facilities described above are also available in the EDITOR WINDOW.

1.3.9 Miscellaneous Commands

The following MATLAB commands will help in managing the MATLAB desktop

`clear` — Clears the MATLAB workspace.
1.3 Desktop Windows

clc — Clears the contents of the Command Window

clf — Clears the contents of the Figure Window

If MATLAB appears to be caught in a loop and is taking too long to finish a command it may be aborted by Ctrl + C (Hold down the Ctrl key and press C). MATLAB will then return to the command prompt.

diary filename After this command all input and most output is echoed to the specified file. The commands diary off and diary on will suspend and resume input to the diary (log) file.
CHAPTER 2

Basic Matlab and Some introductory Examples

The basic variable in MATLAB is an Array. (The numbers entered earlier can be regarded as $(1 \times 1)$ arrays, Column vectors as $(n \times 1)$ arrays and matrices as $(n \times m)$ arrays. MATLAB can also work with multidimensional arrays.

2.1 Sample MATLAB session in the Command Window

It is recommended that you work through the following sitting at a PC with MATLAB running and enter the commands in the Command window. Most of the calculations involved are simple and they can be checked with a little mental arithmetic. The upper section of each box is MATLAB input, the lower MATLAB output. In some cases the box is not divided and reports input and output in a command window. This session only covers a small proportion of the functionality of MATLAB.

The ideas of this chapter can also help you with learning and understanding other aspects of linear algebra and their implementation in MATLAB. Simply compose a simple example and implement it both on paper and in MATLAB.
## 2.1.1 Entering Matrices

The matrix is entered row by row. The elements in a row are separated by spaces or commas. The rows are separated by semi-colons. The `;`, `<` is the MATLAB prompt. Anything on a line after a `%` sign is regarded as a comment. These are intended for you to read and you need not type these comments. When you are writing your own MATLAB programs you should use comments to explain what you are doing and why you are doing it. As already mentioned if you add a `;` at the end of a command (before the comment sign `%`) the output will be suppressed.

<table>
<thead>
<tr>
<th>MATLAB Session</th>
<th>Description</th>
<th>Output</th>
</tr>
</thead>
</table>
| `>> x=[1 2 3 4] % assigning values to a (1 by 4) matrix (row vector)` | Assigning values to a row vector | `x =
``
``1 2 3 4`
| `>> x=[1; 2; 3; 0] % A (4 by 1) (column) vector` | Assigning values to a column vector | `x =
```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````
2.1 Sample MATLAB session in the Command Window

2.1.2 Basic Matrix operations

The following examples are simple. Check the various operations and make sure that you understand them. This will also help you revise some matrix algebra which you will need for your econometric theory.

\[
\begin{align*}
\text{>> } & \text{x=[1 2;3 4]} \\
& x = \\
& 1 \quad 2 \\
& 3 \quad 4 \\
\text{>> } & \text{y=[3 7;5 4]} \\
& y = \\
& 3 \quad 7 \\
& 5 \quad 4 \\
\text{>> } & \text{x+y \%addition of two matrices - same dimensions} \\
& \text{ans =} \\
& 4 \quad 9 \\
& 8 \quad 8 \\
\text{>> y-x \%matrix subtraction} \\
& \text{ans =} \\
& 2 \quad 5 \\
& 2 \quad 0 \\
\text{>> x*y \% matrix multiplication} \\
& \text{ans =} \\
& 13 \quad 15 \\
& 29 \quad 37 
\end{align*}
\]

Note that when matrices are multiplied their dimensions must conform. The number of columns in the first matrix must equal the number of rows in the second otherwise
MATLAB returns an error. Try the following example. When adding matrices a similar error will be reported if the dimensions do not match.

```
>> x=[[1 2;3 4]]
x =
    1     2
    3     4

>> z=[1 2]
z =
    1     2

>> x*z
??? Error using ==> mtimes
Inner matrix dimensions must agree.

>> inv(x) % find inverse of a matrix
ans =
-2.0000   1.0000
1.5000  -0.5000

>> x*inv(x) % verify inverse
ans =
1.0000   0.0000
0.0000   1.0000
```
\[ x = A \backslash B \] (x=A\backslash B) solves \[ Ax = B \] Using Gaussian elimination. It is more robust numerically and more efficient than first calculating the inverse as in \( x = \text{inv}(A) \times B \). The alternative expression should be used in calculating regression coefficients.
2.1.3 Kronecker Product

\[
A \otimes B = \begin{pmatrix}
a_{11}B & a_{12}B & \ldots & a_{1m}B \\
a_{21}B & a_{22}B & \ldots & a_{2m}B \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1}B & a_{n2}B & \ldots & a_{nm}B
\end{pmatrix}
\]

x>> x=[1 2;3 4]

x =
1  2
3  4

>> I=eye(2,2) % Alternatively I = I(2) produces the same result.

I =
1  0
0  1

>> kron(x,I)

ans =
1  0  2  0
0  1  0  2
3  0  4  0
0  3  0  4
2.1.4 Examples of number formats

This subsection gives examples of some of the ways in which the number formats in output can be changed. The command

```
format
```

sets output to the default. The `format` command only changes the displayed output and does not effect how results are stored internally. Several examples follow of this command followed by a specifier.

```
>> x=12.345678901234567;
>> format loose % includes blank lines to space output
>> x
x =
    12.3457
```

```
>> format compact % Suppress blank lines
>> x
x =
    12.3457
```
2.1 Sample MATLAB session in the Command Window

```matlab
>> format long % 15 digits after decimal for double
>> x

x =
12.345678901234567

>> format short e % exponential or scientific format
>> x

x =
1.2346e+001

>> format long e
>> x

x =
1.234567890123457e+001

>> format short g % decimal or exponential
>> x

x =
12.346

>> format long g
>> x

x =
12.3456789012346

>> format bank % currency format (2 decimals)
>> x

x =
12.35
```

2.1.5 fprintf function
2.1 Sample MATLAB session in the Command Window

```matlab
>> fprintf('%.2f
', x )
12.35
>> fprintf('%.3f
', x )
12.346
>> fprintf('The number is %.4f
', x )
The number is 12.3457
```

Here `fprintf` prints to the command window according to the format specification '6.4f\n'. In this format specification the % indicates the start of a format specification. There will be at least 6 digits displayed of which 4 will be decimals in floating point (f). The \n indicates that the cursor will then move to the next line. For more details see page 75.

### 2.1.6 element by element operations

The +, -, * and / operators do standard matrix addition, subtraction etc, respectively. The dot operators .* and ./ provide element by element operations as in the following examples.

```matlab
% .operations
>> x=[1 2;3 4];
>> y=[3 7;5 4];
>> x .* y %element by element multiplication

ans =

3 14
15 16

>> y ./ x %element by element division

ans =

0.3333 0.2857
0.6000 1.0000
```
2.1 Sample MATLAB session in the Command Window

```matlab
>> z=[3 7;0 4];
>> x./z

Warning: Divide by zero.
ans =
    0.3333    0.2857
        Inf     1.0000
```

2.1.7 Mixed Scalar and Matrix Operations

In MATLAB adding a scalar to or multiplying a scalar by a matrix does that operation on each element of the matrix.

```matlab
>> a=2;
>> x+a

ans =
    3     4
      5     6

>> x-a

ans =
   -1     0
     1     2

>> x*2

ans =
    2     4
    6     8

>> x/2

ans =
    0.5000    1.0000
    1.5000    2.0000
```
2.1.8 Exponents

In MATLAB it is possible to (i) raise a matrix to some power, (ii) raise each element of a matrix to a power and (iii) raise the elements to specified powers (possibly different for each element).

\% x^a is x^2 or x*x i.e.
>> x^a

\begin{verbatim}
ans =
 7 10
15 22
\end{verbatim}

\% element by element exponent
\% element by element to the power
>> x .^ a

\begin{verbatim}
ans =
1 4
9 16
\end{verbatim}
2.1.9 Miscellaneous Functions

Some functions operate on each element of a matrix.

```matlab
>> x = [ 1 2 ; 3 4];
>> exp(x)
```

```
2.7183  7.3891
20.0855 54.5982
```

```matlab
>> log(x)
```

```
an =
0  0.6931
1.0986  1.3863
```

```matlab
>> sqrt(x)
```

```
an =
1.0000  1.4142
1.7321  2.0000
```

Using negative numbers in the argument of logs and square-roots produces an error in many econometric/statistical packages. MATLAB returns complex numbers. Take care!! This is mathematically correct but may not be what you want.

```matlab
>> z=[1 -2]
z =
1  -2
>> log(z)
an =
0.0000 + 0.0000i  0.6931 + 3.1416i
```

```matlab
>> sqrt(z)
an =
1.0000 + 0.0000i  0.0000 + 1.4142i
```
the function $\text{det}(A)$ calculates the determinant of a matrix $A$.

\begin{verbatim}
>> det(x)
ans =
   -2
\end{verbatim}

The function $\text{trace}(A)$ calculates the trace of a matrix $A$.

\begin{verbatim}
>> trace(x)
ans =
    5
\end{verbatim}

The function $\text{diag}(X)$ where $X$ is a square matrix puts the diagonal of $X$ in a matrix. The function $\text{diag}(Z)$ where $Z$ is a column vector outputs a matrix with diagonal $Z$ and zeros elsewhere

\begin{verbatim}
>> z=diag(x)
z =
    1
    4

>> u=diag(z)
u =
    1   0
    0   4
\end{verbatim}

The function $\text{rank}(Z)$ estimates the rank of a matrix.

\begin{verbatim}
>> a=[2 4 6 9
    3 2 5 4
    2 1 7 8]
a =
    2   4   6   9
    3   2   5   4
\end{verbatim}
2.1 Sample MATLAB session in the Command Window

```
2  1  7  8
```

```
>> rank(a)
an =
   3
```

`sum(A)` returns sums along different dimensions of an array. If \( A \) is a row or column vector, \( \text{sum}(A) \) returns the sum of the elements. If \( A \) is a matrix, \( \text{sum}(A) \) treats the columns of \( A \) as vectors, returning a row vector of the sums of each column.

```
>> x=[1 2 3 4]
x =
   1  2  3  4
```

```
>> sum(x)
an =
   10
```

```
>> sum(x')
an =
   10
```

```
>> x=[1 2;3 4]
x =
   1  2
   3  4
```

```
>> sum(x)
an =
   4  6
```

The function `reshape(A,m,n)` returns the \( m \times n \) matrix \( B \) whose elements are taken
column-wise from $A$. An error results if $A$ does not have exactly $mn$ elements

$$
\begin{array}{c}
\text{>> } x=[1 & 2 & 3 ; 4 & 5 & 6] \\
\end{array}
$$

\begin{verbatim}
x =
1 2 3
4 5 6
\end{verbatim}

$$
\begin{array}{c}
\text{>> reshape(x,3,2)} \\
\end{array}
$$

\begin{verbatim}
ans =
1 5
4 3
2 6
\end{verbatim}

$\text{blkdiag}(A, B, C)$ constructs a block diagonal matrix from the matrices $A$, $B$, $C$ etc.

$$
\begin{array}{c}
\text{a =} \\
1 2 \\
3 4 \\
\text{>> b=5} \\
b = \\
5 \\
\text{>> c=[6 7 8;9 10 11;12 13 14]} \\
c = \\
6 7 8 \\
9 10 11 \\
12 13 14 \\
\text{>> blkdiag(a,b,c)} \\
\text{ans =} \\
1 2 0 0 0 0 0 \\
3 4 0 0 0 0 0 \\
0 0 5 0 0 0 0 \\
0 0 0 6 7 8 \\
0 0 0 9 10 11 \\
0 0 0 12 13 14
\end{array}
$$
2.1 Sample MATLAB session in the Command Window

The function

2.1.10 Eigenvalues and Eigenvectors

The function `eig()` calculates eigenvectors and eigenvalues. The following are typical examples of the use of these functions. It is possible to specify outputs in greater detail or specify the algorithm to be used in the calculation — see the manual.

```
>> A=[54 45 68
45 50 67
68 67 95]
>> eig(A) % Vector with eigenvalues
ans =
0.4109
7.1045
191.4846
>> [V,D]=eig(A) % eigen vectors are columns of V
% eiden values are on diagonal of D
V =
0.3970 0.7631 0.5100
0.5898 -0.6378 0.4953
-0.7032 -0.1042 0.7033
D =
0.4109 0 0
0 7.1045 0
0 0 191.4846
>> Test=A*V
Test =
0.1631 5.4214 97.6503
0.2424 -4.5315 94.8336
-0.2890 -0.7401 134.6750
```
2.1 Sample MATLAB session in the Command Window

\[
\begin{align*}
\text{>> Test ./ V} \\
\text{ans =} \\
0.4109 & 7.1045 & 191.4846 \\
0.4109 & 7.1045 & 191.4846 \\
0.4109 & 7.1045 & 191.4846 \\
\end{align*}
\]

2.1.11 Transpose of a matrix

The transpose of a matrix is denoted by a quote mark. i.e. \( A' \) is the transpose of \( A \).

\[
\begin{align*}
\text{>> A = [1 2 3; 4 5 6]} \\
\text{A =} \\
1 & 2 & 3 \\
4 & 5 & 6 \\
\end{align*}
\]

\[
\begin{align*}
\text{>> A'} \\
\text{ans =} \\
1 & 4 \\
2 & 5 \\
3 & 6 \\
\end{align*}
\]

2.1.12 Sequences

\[ \text{[first:increment:last]} \] is a row vector whose elements are a sequence with first element \( \text{first} \), second element \( \text{first} + \text{increment} \) and continues while the new entry is less than \( \text{last} \).

\[
\begin{align*}
\text{>> [1:2:9]} \\
\text{ans =} \\
1 & 3 & 5 & 7 & 9 \\
\end{align*}
\]
or

```
>> [2:2:9]
ans =
  2  4  6  8
```

If only two numbers are specified it is assumed that the increment is 1.

```
>> [1:4]
ans =
  1  2  3  4
```

To get a sequence in a column vector use the transpose operator defined in the previous subsection.

```
>> [1:4]'
ans =
  1
  2
  3
  4
```

### 2.1.13 Creating Special Matrices

The `eye(n)` creates an $n \times n$ identity matrix

```
>> x=eye(4)
x =
   1  0  0  0
   0  1  0  0
   0  0  1  0
   0  0  0  1
```
The `ones(n,m)` creates an $n \times m$ matrix with all its elements equal to one. If the required matrix is square $n \times n$ one has the option of using $n$.

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]

The `zeros()` function is similar to the `ones()` function except that it creates matrices of zeros rather that ones.

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

The `size` function returns the the number of elements in each dimension of a matrix. e.g. if $(A)$ is $n \times m$ then `size(A)` returns $n \times m$. The function `numel(A)` returns the
2.1 Sample MATLAB session in the Command Window

number of elements in \((A)\) \((nm)\).

```matlab
>> size(x)
2 3
```

```matlab
>> numel(x)
ans =
6
```

2.1.14 Random number generators

In MATLAB the basic random number generator is `rand()` which generates pseudo uniform random numbers in the range \([0, 1]\).

```matlab
>> x=rand(5) % generate a \(5 \times 5\) square matrix of random numbers.
x =
0.8155 0.55386 0.78573 0.05959 0.61341
0.5847 0.92263 0.78381 0.80441 0.2093
0.70495 0.89406 0.1167 0.45933 0.05613
0.17658 0.44634 0.64003 0.07634 0.14224
0.98926 0.90159 0.52867 0.93413 0.74421
```

```matlab
>> x=rand(5,1) % generate a \(5 \times 1\) matrix of random numbers.
x =
0.21558
0.62703
0.04805
0.20085
0.67641
```
To generate a matrix of uniform random numbers on the interval \([a, b]\) use the expression

\[a + (b - a)x\]

where \(x\) is a matrix of uniform random numbers on the interval \([0, 1]\).

An economist is probably more interested in generating normal random numbers. The MATLAB function `randn()` generates normal random numbers with zero mean and unit variance. The syntax of this command is similar to that of the `rand` function. If you require normal random numbers with mean \(\mu\) and variance \(\sigma^2\) use something like

\[x = \mu + \sigma z\]

where \(z\) is an array of simulated standardized normal variates.
Every time MATLAB is started it initializes the random number generator to the same state. Unlike many statistical packages every time MATLAB is started and the same instructions are given to generate a set or sets of random numbers, the same random numbers are generated.

At any stage in a MATLAB session you can initialize the random number generator to its initial state. use as In the middle box of the last three boxes above the command `rng('default')` achieves this.

It is also possible to ”seed” the random generator with the command

```
>> rng(seed)
```

where `seed` is an integer between 0 and $2^{32}$. This enables you to produce replicable simulations using random numbers other than the default.

The random number generators in earlier versions of MATLAB used different algorithms to generate random number sequences. If you do need to replicate an older analysis which used these older algorithms there are instructions in the documentation for the random number generators. My recommendation would be to stick with the new generators.

### 2.1.15 Extracting parts of a matrix, Joining matrices together to get a new larger matrix

Extract a single element from a row vector. command

```
>> arr1=[2 4 6 8 10];
>> arr1(3)
an1 = 6
```

Extract a single element from a matrix. The first coordinate refers to the row number, the second to the column.
The : operator generates a sequence of coordinates to be extracted from a matrix (See subsection 2.1.12). Thus the first example extracts columns 2 to 3 of row 2 of the matrix.

```
>> arr2(2,2:3)
ans =
   5   6
```

using : on its own extracts the entire column or row;

```
>> arr2(2,:)
ans =
   4   5   6
```

If we are not sure how many elements are in a row or column we may use end to signify the last element in the matrix.

```
>> arr2(3,2:end)
ans =
   8   9
```
2.1 Sample MATLAB session in the Command Window

2.1.16 Using sub-matrices on left hand side of assignment

The examples in this subsection show how to assign values to certain sub-matrices of a matrix.

```matlab
>> arr4=[1 2 3 4; 5 6 7 8; 9 10 11 12]
arr4 =
    1  2  3  4
    5  6  7  8
    9 10 11 12

>> arr4(1:2,[1,4])=[20,21;22 23]
arr4 =
    20  2  3  21
    22  6  7  23
    9 10 11 12

>> arr4=[20,21;22 23]
arr4 =
    20   21
    22   23

>> arr4(1:2,1:2)=1
arr4 =
    1   1
    1   1
```
2.1 Sample MATLAB session in the Command Window

>> arr4=[1 2 3 4;5 6 7 8;9 10 11 12]
arr4 =
    1 2 3 4
    5 6 7 8
    9 10 11 12

>> arr4(1:2,1:2)=1
arr4 =
    1 1 3 4
    1 1 7 8
    9 10 11 12

2.1.17 Stacking Matrices

>> x=[1 2;3 4]
x =
    1 2
    3 4

>> y=[5 6;7 8]
y =
    5 6
    7 8

>> z=[x,y,(15:16)'] \% join matrices side by side
z =
    1 2 5 6 15
    3 4 7 8 16
2.1 Sample MATLAB session in the Command Window

```
>> z=[x',y',(15:16)']' % Stack matrices vertically
z =
    1     2
    3     4
    5     6
    7     8
   15    16
```

See also the help files for the MATLAB commands `cat`, `horzcat` and `vertcat`.

### 2.1.18 Special Values

```
>> pi % value of π
pi = 3.1416

>> exp(1) % value of e
ans =
    2.7183

>> clock % Extract date and time
ans =
    1.0e+03 *
    2.0140  0.0110  0.0100  0.0220  0.0070  0.0475
     YEAR    Month    Day    Hours    Minutes    Seconds
```

The command `tic` starts a stopwatch. Subsequent `toc` commands measure the time since the last `tic` command.
2.2 Examples of Use of Command Window

Work through the following examples using MATLAB.

1. let $A = \begin{pmatrix} 3 & 0 \\ 5 & 2 \end{pmatrix}$ and $B = \begin{pmatrix} 1 & 4 \\ 4 & 7 \end{pmatrix}$ Use MATLAB to calculate.
   
   (a) $A + B$
   (b) $A - B$
   (c) $AB$
   (d) $AB^{-1}$
   (e) $A/B$
   (f) $A\backslash B$
   (g) $A.* B$
   (h) $A./ B$
   (i) $A \otimes B$
   (j) $B \otimes A$

   Use pen, paper and arithmetic to verify that your results are correct.

2. Let $A = \begin{pmatrix} 1 & 4 & 3 & 7 \\ 2 & 6 & 8 & 3 \\ 1 & 3 & 4 & 5 \\ 4 & 13 & 15 & 15 \end{pmatrix}$

   Use the relevant MATLAB function to show that the rank of A is three. Why is it not four?

3. Solve $Ax = a$ for $x$ where $A = \begin{pmatrix} 1 & 4 & 3 & 7 \\ 2 & 6 & 8 & 3 \\ 1 & 3 & 4 & 5 \\ 2 & 1 & 7 & 6 \end{pmatrix}$ and $a = \begin{pmatrix} 14 \\ 8 \\ 10 \\ 18 \end{pmatrix}$

4. Generate $A$ which is a $4 \times 4$ matrix of uniform random numbers. Calculate the trace and determinant of A. Use MATLAB to verify that
(a) The product of the eigenvalues of $A$ is equal to the determinant of $A$.

(b) The sum of the eigenvalues of $A$ is equal to the trace of $A$. (You might find the MATLAB functions `sum()` and `prod()` helpful - please see the relevant help files). Do these results hold for an arbitrary matrix $A$.

5. Let $A$ and $B$ be two $4 \times 4$ matrices of independent $N(0,1)$ random numbers. If $\text{tr}(A)$ is the trace of $A$. Show that

(a) $\text{tr}(A + B) = \text{tr}(A) + \text{tr}(B)$

(b) $\text{tr}(4A) = 4\text{tr}(A)$

(c) $\text{tr}(A') = \text{tr}(A)$

(d) $\text{tr}(BA) = \text{tr}(AB)$

Which of these results hold for arbitrary matrices? Under what conditions would they hold for non-square matrices?

### 2.3 Regression Example

In this Example I shall use the instructions you have already learned to simulate a set of observations from a linear equation and use the simulated observations to estimate the coefficients in the equation. In the equation $y_t$ is related to $x_{2t}$ and $x_{3t}$ according to the following linear relationship.

$$y_t = \beta_1 + \beta_2 x_{2t} + \beta_3 x_{3t} + \varepsilon_t, \quad t = 1, 2, \ldots, N$$

or in matrix notation

$$y = X\beta$$

where

- $x_2$ is a trend variable which takes the values $(1, 2, \ldots, 50)$
- $x_3$ is a random variable with uniform distribution on $[3, 5]$
- $\varepsilon_t$ are independent identically distributed normal random variables with zero mean and constant variance $\sigma^2$. 
2.3 Regression Example

- $\beta_1 = 5, \beta_2 = 1$ and $\beta_3 = 0.1$ and $\varepsilon_t$ are iidn$(0,0.04)$ ($\sigma^2 = 0.04$)

1. Verify that the model may be estimated by OLS.

2. Use MATLAB to simulate 50 observations of each of $x_3$ and $\varepsilon_t$ and thus of $x_t$.

3. Using the simulated values find OLS estimates of $\beta$.

4. Estimate the covariance matrix of $\beta$ and thus the t-statistics for a test that the coefficients of $\beta$ are zero.

5. Estimate the standard error of the estimate of $y$.

6. Calculate the F-statistic for the significance of the regression.

7. Export the data to an econometric package of your choice and verify your result.

8. See page 48 for details on replication of simulation exercises.

9. Any answers submitted should be concise and short and should contain
   (a) A copy of the m-file used in the analysis. This should contain comments to explain what is being done
   (b) A short document giving the results of one simulation and any comments on the results. You might also include the regression table from your econometric package. This document should be less than one page in length.

A sample answer follows. First the program, then the output and finally some explanatory notes. The program is a good example of a MATLAB script. If Required you should be able to cut and past it into the MATLAB EDITOR WINDOW.

Regression Example Using Simulated Data

```matlab
% example1.m
% Regression Example Using Simulated Data
% John C Frain
% 16 August 2014
% values for simulation
clear;
 rng(1234);
 nsimul=50;
 beta=[5,1,.1]';
```
Regression Example(cont.)

```
% Step 1 Prepare and process data for X and y matrices/vectors
%
x1=ones(nsimul,1);  %constant
x2=1:nsimul';       %trend
x3=rand(nsimul,1)*2 +3;  % Uniform(3,5)
X=[x1,x2,x3];
e=randn(nsimul,1)*.2;  % N(0,.04)
y= X * beta + e;       % 5*x1 + x2 + .1*x3 + e;
%
[nobs,nvar]=size(X);
%
% Estimate Model
%
Note that I have named my estimated variables ols.betahat, ols.yhat, ols.resid etc. The use of the ols. in front of the variable name has two uses. First if I want to do two different estimate I will call the estimates ols1. and ols2. or IV etc. and I can easily put the in a summary table. Secondly this structure has a meaning that is useful in a more advanced use of MATLAB.
%
ols.betahat=(X'*X)
ols.yhat = X * ols.betahat;  % beta(1)*x1-beta(2)*x2-beta(3)*x;
ols.resid = y - ols.yhat;  % residuals
ols.ssr = ols.resid'*ols.resid;  % Sum of Squared Residuals
ols.sigmasq = ols.ssr/(nobs-nvar);  % Estimate of variance
ols.covbeta=ols.sigmasq*inv(X'*X);  % Covariance of beta
ols.sdbeta=sqrt(diag(ols.covbeta));  % st. dev of beta
ols.tbeta = ols.betahat ./ ols.sdbeta;  % t-statistics of beta
ym = y - mean(y);
ols.rsqr1 = ols.ssr;
ols.rsqr2 = ym'*ym;
ols.rsqr = 1.0 - ols.rsqr1/ols.rsqr2;  % r-squared
ols.rsqr1 = ols.rsqr1/(nobs-nvar);
```
2.3 Regression Example

Regression Example (cont.)

```matlab
ols.rsqr2 = ols.rsqr2/(nobs-1.0);
if ols.rsqr2 ~= 0;
    ols.rbar = 1 - (ols.rsqr1/ols.rsqr2); % rbar-squared
else
    ols.rbar = ols.rsqr;
end;
ols.ediff = ols.resid(2:nobs) - ols.resid(1:nobs-1);
ols.dw = (ols.ediff'*ols.ediff)/ols.ssr; % durbin-watson
fprintf('R-squared = %9.4f 
', ols.rsqr);
fprintf('Rbar-squared = %9.4f 
', ols.rbar);
fprintf('sigma^2 = %9.4f 
', ols.sigmasq);
fprintf('S.E of estimate= %9.4f 
', sqrt(ols.sigmasq));
fprintf('Durbin-Watson = %9.4f 
', ols.dw);
fprintf('Nobs, Nvars = %6d,%6d 
', nobs,nvar);
fprintf('****************************************************

% now print coefficient estimates, SE, t-statistics and probabilities
% tout = tdis_prb(tbeta,nobs-nvar); % find t-stat probabilities - no
% tdis_prb in basic MATLAB - requires leSage toolbox
% tmp = [beta sdbeta tbeta tout]; % matrix to be printed
% tmp = [ols.betahat ols.sdbeta ols.tbeta]; % matrix to be printed
% column labels for printing results
namestr = ' Variable';
bstring = ' Coef.';
sdstring= 'Std. Err.';
tstring = ' t-stat.';
cnames = strvcat(namestr,bstring,sdstring, tstring);
vname = ['Constant','Trend' 'Variable2'];
%
% The fprintf is used to produce formatted output. See subsection 3.6
%printf('%12s %12s %12s %12s 
',namestr, ... bstring,sdstring,tstring)
fprintf('%12s %12.6f %12.6f %12.6f 
',... ' Const',...)
```
Regression Example (cont.)

```matlab
ols.betahat(1), ols.sdbeta(1), ols.tbeta(1))
fprintf('%12s %12.6f %12.6f %12.6f 
', 'Trend', ...)
ols.betahat(2), ols.sdbeta(2), ols.tbeta(2))
fprintf('%12s %12.6f %12.6f %12.6f 
', 'Var2', ...)
ols.betahat(3), ols.sdbeta(3), ols.tbeta(3))
```

The output of this program should look like

- \( R^2 = 0.9999 \)
- \( R_{bar}^2 = 0.9999 \)
- \( \sigma^2 = 0.0314 \)
- S.E of estimate = 0.1773
- Durbin-Watson = 1.9492
- Nobs, Nvars = 50, 3

```
\%

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coef.</th>
<th>Std. Err.</th>
<th>t-stat.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Const</td>
<td>4.911817</td>
<td>0.205177</td>
<td>23.939427</td>
</tr>
<tr>
<td>Trend</td>
<td>1.001412</td>
<td>0.001760</td>
<td>568.997306</td>
</tr>
<tr>
<td>Var2</td>
<td>0.119433</td>
<td>0.047365</td>
<td>2.521525</td>
</tr>
</tbody>
</table>
```

If you are using a recent version of MATLAB you should be able to replicate these results. If you are using an old version you may not generate the same series of random numbers and your answers may differ slightly.

Explanatory Notes

Most of your MATLAB scripts or programs will consist of three parts

1. **Get and Process data.** Read in your data and prepare vectors or matrices of
2.4 Simulation – Sample Size and OLS Estimates

your left hand side \((y)\), Right hand side \((X)\) and Instrumental Variables \((Z)\)

2. **Estimation** Some form of calculation(s) like \(\hat{\beta} = (X'X^{-1})X'y\) implemented by a MATLAB instruction like

\[
\text{betahat} = (X'\cdot X)\cdot X\cdot y
\]

(where \(X\) and \(y\) have been set up in the previous step) and estimate of required variances, covariances, standard errors etc.


4. Run the program with a smaller number of replications (say 25) and see how the t-statistic on \(y_3\) falls. Rerun it with a larger number of replications and see how it rises. Experiment to find how many observations are required to get a significant coefficient for \(y_3\) often. Suggest a use of this kind of analysis.

### 2.4 Simulation – Sample Size and OLS Estimates

This exercise is a study of the effect of sample size on the estimates of the coefficient in an OLS regression. The x values for the regression have been generated as uniform random numbers on the interval \([0,100)\). The residuals are simulated standardized normal random variables. The process is repeated for sample sizes of 20, 100, 500 and 2500. Each simulation is repeated 10,000 times.

```matlab
Simulation of Effect of Sample Size on Regression Estimate

\[
\% \text{ example2.m}
\%
\% MATLAB Simulation Example
\%
\% John C Frain
\%
\% September 2014
\%
\%
\%
\%
\%
The data files x20.csv, x100.csv, x500.csv and x2500.csv were generated using the code below
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
\%
Sample Size Example (cont.)

```matlab
x20 = 100*rand(20,1);
save('x20.csv','x20','-ASCII','-double')
x100 = 100*rand(100,1);
save('x100.csv','x100','-ASCII','-double')
x500 = 100*rand(500,1);
save('x500.csv','x500','-ASCII','-double')
x2500 = 100*rand(2500,1);
save('x2500.csv','x2500','-ASCII','-double')
%
clear
nsimul=10000;
BETA20=zeros(nsimul,1); % vector - results of simulations with 20 obs.
x=load('-ascii','x20.csv'); % load xdata
X=[ones(size(x,1),1),x]; % X matrix note upper case X
beta = [10;2]; % true values of coefficients

for ii = 1 : nsimul;
    eps = 20.0*randn(size(X,1),1); % simulated error term
    y = X * beta + eps; % y values
    betahat = (X'*X)
    BETA20(ii,1)=betahat(2);
end
fprintf('Mean and st. dev of 20 obs simulation %6.3f
',mean(BETA20),std(BETA20))

BETA100=zeros(nsimul,1);
x=load('-ascii','x100.csv'); % load xdata
X=[ones(size(x,1),1),x]; % X matrix note upper case X
beta = [10;2]; % true values of coefficients

for ii = 1 : nsimul;
    eps = 20.0*randn(size(X,1),1); % simulated error term
```
2.4 Simulation – Sample Size and OLS Estimates

Sample Size Example (cont.)

\[ y = X \ast \beta + \varepsilon; \text{ \% y values} \]
\[ \beta = inv(X'\ast X)\ast X'\ast y; \text{ \% estimate of } \beta \]
\[ BETA100(ii,1)=betahat(2); \]
end
\[ fprintf('Mean and st. dev of 100 obs simulation \(6.3f\) \(6.3f\)\n', ... \]
\[ mean(BETA100),std(BETA100)) \]

\[ BETA500=zeros(nsimul,1); \]
\[ x=load('-ascii', 'x500.csv'); \% load xdata \]
\[ X=[ones(size(x,1),1),x]; \% X matrix note upper case X \]
\[ beta = [ 10;2]; \% true values of coefficients \]
\% for ii = 1 : nsimul;
\[ \varepsilon = 20.0\ast\text{randn(size(X,1),1)}; \% simulated error term \]
\[ y = X \ast \beta + \varepsilon; \% y values \]
\[ \beta = inv(X'\ast X)\ast X'\ast y; \% estimate of } \beta \]
\[ BETA500(ii,1)=betahat(2); \]
end
\[ fprintf('Mean and st. dev of 500 obs simulation \(6.3f\) \(6.3f\)\n', ... \]
\[ mean(BETA500),std(BETA500)) \]

\[ BETA2500=zeros(nsimul,1); \]
\[ x=load('-ascii', 'x2500.csv'); \% load xdata note use of lower case x as vector \]
\[ X=[ones(size(x,1),1),x]; \% X matrix note upper case X \]
\[ beta = [ 10;2]; \% true values of coefficients \]
\% for ii = 1 : nsimul;
\[ \varepsilon = 20.0\ast\text{randn(size(X,1),1)}; \% simulated error term \]
\[ y = X \ast \beta + \varepsilon; \% y values \]
\[ \beta = inv(X'\ast X)\ast X'\ast y; \% estimate of } \beta \]
\[ BETA2500(ii,1)=betahat(2); \]
end
\[ fprintf('Mean and st. dev of 2500 obs simulation \(6.3f\) \(6.3f\)\n', ... \]
\[ mean(BETA2500),std(BETA2500)) \]
n = hist([BETA20, BETA100, BETA500, BETA2500], 1.4:0.01:2.6);
plot((1.4:0.01:2.6)', n/nsimul, 'LineWidth', 2);
h = legend('Sample 20', 'Sample 100', 'Sample 500', 'Sample 2500');
print -dpdf 'example2_3.pdf'

The output of this program will look like this. On your screen the graph will display coloured lines.

Mean and st. dev of 20 obs simulation  2.000  0.130
Mean and st. dev of 100 obs simulation  2.000  0.066
Mean and st. dev of 500 obs simulation  2.001  0.032
Mean and st. dev of 2500 obs simulation  2.000  0.014
2.5 Example – Macroeconomic Simulation with MATLAB

Simulation of Macroeconomic Model

Problem

This example is based on the macroeconomic system in Example 10.3 of Shone (2002). There are 10 equations in this economic model. The equations of the system are as follows

\[
\begin{align*}
  c_t &= 110 + 0.75yd_t \\
  yd_t &= y_t - tax_t \\
  tax_t &= -80 + 0.2y_t \\
  i_t &= -4r_t \\
  g_t &= 330 \\
  e_t &= c_t + i_t + g_t \\
  y_t &= e_{t-1} \\
  md_t &= 20 + 0.25y_t - 10r_t \\
  ms_t &= 470 \\
  md_t &= ms_t
\end{align*}
\]

While the aim in Shone (2002) is to examine the system algebraically, here we examine it numerically. Often this may be the only way to solve the system and MATLAB is a suitable tool for this work. The model is too simple to be of any particular use in macroeconomics but it does allow one to illustrate the facilities offered by MATLAB for this kind of work.

Initialise and Describe Variables

\[
\begin{align*}
  N &= 15; \text{ Number of periods for simulation} \\
  c &= NaN \times \text{zeros}(N,1); \text{real consumption} \\
  tax &= NaN \times \text{zeros}(N,1); \text{real tax} \\
  yd &= NaN \times \text{zeros}(N,1); \text{real disposable income}
\end{align*}
\]
Macroeconomic Model (cont.)

\begin{verbatim}
\texttt{i = NaN * zeros(N,1); \% real investment}
\texttt{g = NaN * zeros(N,1); \% real government expenditure}
\texttt{e = NaN * zeros(N,1); \% real expenditure}
\texttt{y = NaN * zeros(N,1); \% real income}
\texttt{md = NaN * zeros(N,1); \% real money demand}
\texttt{ms = NaN * zeros(N,1); \% real money supply}
\texttt{r = NaN * zeros(N,1); \% interest rate}
\end{verbatim}

\textbf{Simulate}

\texttt{g} and \texttt{ms} are the policy variables.

\begin{verbatim}
\texttt{t=(1:N)'; \% time variable}
\texttt{g = 330 * ones(N,1);}
\texttt{ms = 470 * ones(N,1);}
\texttt{y(1) = 2000;}
\end{verbatim}

\% The next step is to simulate the model over the required period. Here this is
\% achieved by a simple reordering of the equations and inverting the money demand
\% equation to give an interest rate equation. In the general case we might need a
\% routine to solve the set of non linear equations or some routine to maximize a
\% utility function. Note that the loop stops one short of the full period and then
\% does the calculations for the final period (excluding the income calculation for the
\% period beyond the end of the sample under consideration).

\begin{verbatim}
\texttt{for ii = 1:(N-1)}
\texttt{\quad tax(ii) = -80 + 0.2 * y(ii);}\%
\texttt{\quad yd(ii) = y(ii) - tax(ii);}\%
\texttt{\quad c(ii) = 110 + 0.75 * yd(ii);}\%
\texttt{\quad md(ii) = ms(ii);}\%
\texttt{\quad r(ii) = (20 + 0.25* y(ii) -md(ii))/10; \% inverting money demand} \%
\texttt{\quad i(ii) = 320 -4 * r(ii);}\%
\texttt{\quad e(ii) = c(ii) + i(ii) + g(ii);}\%
\end{verbatim}
Macroeconomic Model (cont.)

\[ y(i+1) = e(i); \]

\[
\text{tax}(N) = -80 + 0.2 \times y(N); \\
\text{yd}(N) = y(N) - \text{tax}(N); \\
c(N) = 110 + 0.75 \times \text{yd}(N); \\
\text{md}(N) = \text{ms}(N); \\
r(N) = (20 + 0.25 \times y(N) - \text{md}(N))/10; \\
i(N) = 320 - 4 \times r(N); \\
e(N) = c(N) + i(N) + g(N); \\
\]

Now output results and save y for later use. Note that the system is in equilibrium.

Note that in printing we use the transpose of base

\[ \text{base} = [t, y, \text{yd}, c, \text{g-tax}, i, r]; \]

\[ \text{fprintf('} \]
\[ \text{t y yd c g-tax i r} \]
\[ \text{fprintf('%7.0f%7.0f%7.0f%7.0f%7.0f%7.0f%7.0f\n', base')} \]

\[ \text{ybase} = y; \]

<table>
<thead>
<tr>
<th>t</th>
<th>y</th>
<th>yd</th>
<th>c</th>
<th>g-tax</th>
<th>i</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2000</td>
<td>1680</td>
<td>1370</td>
<td>30</td>
<td>300</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2020</td>
<td>1696</td>
<td>1382</td>
<td>26</td>
<td>298</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2030</td>
<td>1704</td>
<td>1388</td>
<td>24</td>
<td>297</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2035</td>
<td>1708</td>
<td>1391</td>
<td>23</td>
<td>297</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>2038</td>
<td>1710</td>
<td>1393</td>
<td>23</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>2039</td>
<td>1711</td>
<td>1393</td>
<td>22</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>2039</td>
<td>1712</td>
<td>1394</td>
<td>22</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>2040</td>
<td>1712</td>
<td>1394</td>
<td>22</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>2040</td>
<td>1712</td>
<td>1394</td>
<td>22</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>2040</td>
<td>1712</td>
<td>1394</td>
<td>22</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>2040</td>
<td>1712</td>
<td>1394</td>
<td>22</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>2040</td>
<td>1712</td>
<td>1394</td>
<td>22</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>2040</td>
<td>1712</td>
<td>1394</td>
<td>22</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>2040</td>
<td>1712</td>
<td>1394</td>
<td>22</td>
<td>296</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>2040</td>
<td>1712</td>
<td>1394</td>
<td>22</td>
<td>296</td>
<td>6</td>
</tr>
</tbody>
</table>

Now we compare results in a table and a graph. Note that income converges to a new
Comparison of Simulations

```matlab
t ybase ypolicy
fprintf('%7.0f%7.0f%7.0f\n', [t, ybase, ypolicy])
plot(t, [ybase, ypolicy])
title('Equilibrium and Shocked Macro-system')
xlabel('Period')
ylabel('Income')
axis([1 15 1995 2045])
```

t ybase ypolicy
1 2000 2000
2 2000 2020
3 2000 2030
4 2000 2035
5 2000 2038
6 2000 2039
7 2000 2039
8 2000 2040
9 2000 2040
10 2000 2040
11 2000 2040
12 2000 2040
13 2000 2040
14 2000 2040
15 2000 2040
Comparison of Simulations (cont.)

Equilibrium and Shocked Macro-system
In this chapter I set out

1. How to import data from and export to an excel file.
2. How to import data from and export data to a text or comma separated value (csv) file.
3. Using native MATLAB format.
4. Formatting output
5. Producing material for inclusion in a paper

3.1 Importing from Excel format files

Figure 3.1 is the upper left-hand corner of the data file This file g10xrate.xls contains daily observations on the US Dollar exchange rates of the G10 countries. The ten

\[\text{This data file is embedded in this document. To extract it, please go to Appendix B. That appendix contains links to three embedded sample data files. The first is in EXCEL format, the second uses comma separated values (csv) and the third native MATLAB format.}\]

\[\text{There are 11 G10 countries - Belgium, Canada, France, Germany, Italy, Japan, the Netherlands, Sweden, Switzerland, the United Kingdom and the United States. When the eleventh country joined}\]
3.1 Importing from Excel format files

<table>
<thead>
<tr>
<th>USXJPN</th>
<th>USXFRA</th>
<th>USXSUI</th>
<th>USXNLD</th>
<th>USXUK</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.331796</td>
<td>19.52248</td>
<td>26.51535</td>
<td>30.95017</td>
<td>234.72</td>
</tr>
<tr>
<td>0.331752</td>
<td>19.49508</td>
<td>26.57031</td>
<td>30.96455</td>
<td>234.87</td>
</tr>
<tr>
<td>0.332005</td>
<td>19.50991</td>
<td>26.54491</td>
<td>30.96263</td>
<td>235.03</td>
</tr>
<tr>
<td>0.331895</td>
<td>19.51981</td>
<td>26.55972</td>
<td>30.96263</td>
<td>235.17</td>
</tr>
<tr>
<td>0.330896</td>
<td>19.61015</td>
<td>26.53012</td>
<td>30.94251</td>
<td>235.1</td>
</tr>
<tr>
<td>0.331049</td>
<td>19.65486</td>
<td>26.57384</td>
<td>30.96455</td>
<td>235.14</td>
</tr>
<tr>
<td>0.331301</td>
<td>19.70249</td>
<td>26.60636</td>
<td>31.02988</td>
<td>235.25</td>
</tr>
<tr>
<td>0.331697</td>
<td>19.65602</td>
<td>26.58726</td>
<td>31.02218</td>
<td>235.13</td>
</tr>
<tr>
<td>0.331499</td>
<td>19.65988</td>
<td>26.58231</td>
<td>31.01352</td>
<td>235.18</td>
</tr>
<tr>
<td>0.331499</td>
<td>19.63017</td>
<td>26.56466</td>
<td>30.9703</td>
<td>235.09</td>
</tr>
<tr>
<td>0.331203</td>
<td>19.62246</td>
<td>26.57242</td>
<td>30.98277</td>
<td>235.09</td>
</tr>
<tr>
<td>0.331005</td>
<td>19.65757</td>
<td>26.59504</td>
<td>30.99526</td>
<td>235.27</td>
</tr>
<tr>
<td>0.330896</td>
<td>19.65486</td>
<td>26.59504</td>
<td>30.98757</td>
<td>235.31</td>
</tr>
<tr>
<td>0.330502</td>
<td>19.64984</td>
<td>26.60282</td>
<td>30.9751</td>
<td>235.25</td>
</tr>
<tr>
<td>0.3306</td>
<td>19.72503</td>
<td>26.88028</td>
<td>31.13519</td>
<td>235.35</td>
</tr>
<tr>
<td>0.331499</td>
<td>19.64984</td>
<td>26.80031</td>
<td>31.11</td>
<td>235.53</td>
</tr>
<tr>
<td>0.331005</td>
<td>19.69628</td>
<td>27.0636</td>
<td>31.17985</td>
<td>235.95</td>
</tr>
</tbody>
</table>

Table 3.1: Top right-hand corner of excel file of g10 exchange rates

Exchange rate series are in the ten columns columns. Each row then gives one observation of each series. There are 6237 observations of each exchange rate. The name of each series is given in the first row at the head of each series. This construction is typical of many such files that are encountered in econometrics.

The MATLAB GUI provides the simplest way to import this data set. Use [Home | Import Data], Navigate to file and open it. If the file is large MATLAB may take some time to load it. You will then be presented with a data import window similar to that in figure 3.1. Note that on this window you can

1. specify the range of data to be imported,
2. specify which row contains the data names,
3. import the data as column vectors or a single matrix
4. specify how to deal with unimportable cells

When you have specified these items press the [Import Selection] button and the data are imported.

the original name was retained.
3.1 Importing from Excel format files

The dropdown menu on the [Import Selection] button allows one to display and save the actual MATLAB instructions that MATLAB is using to import the data. These are displayed in the box below. One could, of course, simply enter these commands in the command window or include them in a MATLAB program and produce the same result. In the interest of replication I would recommend that, if you use the GUI to import data, you also generate and save the MATLAB program file. The generated script in the current case is repeated in the box below. (Some of the commands are too long to fit on my printed line in the box and I have used continuation marks (…) to break the command and continue it on the next line.)

```
%% Import data from spreadsheet
% Script for importing data from the following spreadsheet:
% Workbook: C:\TCD\document preparation\MatlabNew\G10XRATE.XLS
% Worksheet: g10xrate
```

Figure 3.1: Matlab Excel Data Import Window
3.1 Importing from Excel format files

MATLAB Instructions to Import data file (cont.)

```matlab
%% Import the data
[~, ~, raw] = xlsread('C:\TCD\document preparation\MatlabNew\...
G10XRATE.XLS','g10xrate','A2:J6238');
raw(cellfun(@(x) ~isempty(x) && isnumeric(x) && isnan(x),raw)) = {''};

%% Replace non-numeric cells with NaN
R = cellfun(@(x) ~isnumeric(x) && ~islogical(x),raw);
raw(R) = {NaN}; % Replace non-numeric cells

%% Create output variable
data = reshape([raw{:}],size(raw));

%% Allocate imported array to column variable names
USXJPN = data(:,1);
USXFRA = data(:,2);
USXSUI = data(:,3);
USXNLD = data(:,4);
USXUK = data(:,5);
USXBEL = data(:,6);
USXGER = data(:,7);
USXSWE = data(:,8);
USXCAN = data(:,9);
USXITA = data(:,10);

%% Clear temporary variables
clearvars data raw R;
```
The script uses the `xlsread` function to read the data, find non-numeric cells replace these with `NaN`, and extract and name the columns.

### 3.2 Reading data from text files

If your data are in comma separated value (csv) format the same procedure allows you to import the data. There are additional options available that allow you to specify the delimiter, combine delimiters and specify decimal point. You can also generate the script file that MATLAB used to import the data.

The MATLAB function `textscan` can read data files in various formats and allows a greater degree of flexibility at the cost of greater complexity.

### 3.3 Native MATLAB data files

The instruction `save('filename.mat')` saves the contents of the workspace in the file `filename.mat`. `save` used in the default manner saves the data in a binary format specific to MATLAB. The instruction `save('filename.mat', 'var1', 'var2')` saves `var1` and `var2` in the file `filename.mat`. Similarly the commands `load('filename.mat')` and `load('filename.mat', 'var1', 'var2')` load the contents of `filename.mat` or the specified variables from the file into the workspace. In general `.mat` files are not easily readable in most other software. They are ideal for use within MATLAB and for exchange between MATLAB users. (note that there may be some incompatibilities between different versions of MATLAB). These `.mat` files are binary and can not be examined in a text editor.

`.mat` is the default extension for a MATLAB data file. If you use another extension, say `.ext` you must specify the extension when using save or load functions. It is possible to use `save` and `load` to save and load text files but these instructions are very limited. If your data are in EXCEL or csv format the methods described above are better.
3.4 Exporting data to EXCEL and econometric/statistical packages

The command `xlswrite('filename', M)` writes the matrix `M` to an Excel format file `filename` in the current working directory. If `M` is `n x m` the numeric values in the matrix are written to the first `n` row and `m` columns in the first sheet in the spreadsheet. The command `csvwrite('filename', M)` writes the matrix `M` to the file `filename` in the current working directory.

Data in Excel or csv format can be read in most econometric/statistical packages.

The package `R.matlab` allows the R statistical system (R Core Team (2014)) to read native MATLAB files. R is capable of outputting data in formats native to or consistent with many statistical/econometric packages.

3.5 Stat/Transfer

Another alternative is to use the Stat/Transfer package which allows the transfer of data files between a large number of statistical packages.

3.6 Formatted Output

The MATLAB function `fprintf()` may be used to produce formatted output on screen.

The following MATLAB program gives an example of the use of the `fprintf()` function.

```matlab
Sample MATLAB program demonstrating Formatted Output

clear
degrees_c = 10:10:100;
degrees_f = (degrees_c * 9/5) + 32;
```

`fprintf()` is only one of a large number of C-style input/output functions in C. These allow considerable flexibility in sending formatted material to the screen or to a file. The MATLAB help files give details of the facilities available. If further information is required one might consult a standard book on the C programming language.
Sample MATLAB program demonstrating Formatted Output (cont.)

```matlab
fprintf('\n\n Conversion from degrees Celsius \n');
fprintf(' to degrees Fahrenheit\n');
fprintf(' Celsius Fahrenheit\n');
for ii = 1:10;
    fprintf('%12.2f%12.2f\n',degrees_c(ii),degrees_f(ii));
end
% 
fprintf(...
'\n\n%5.2f degrees Celsius is equivalent of %5.3f degrees Fahrenheit\n', ... 
    degrees_c(1),degrees_f(1))
```

---

Output of Sample MATLAB program demonstrating Formatted Output

Conversion from degrees Celsius
to degrees Fahrenheit

<table>
<thead>
<tr>
<th>Celsius</th>
<th>Fahrenheit</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.00</td>
<td>50.00</td>
</tr>
<tr>
<td>20.00</td>
<td>68.00</td>
</tr>
<tr>
<td>30.00</td>
<td>86.00</td>
</tr>
<tr>
<td>40.00</td>
<td>104.00</td>
</tr>
<tr>
<td>50.00</td>
<td>122.00</td>
</tr>
<tr>
<td>60.00</td>
<td>140.00</td>
</tr>
<tr>
<td>70.00</td>
<td>158.00</td>
</tr>
<tr>
<td>80.00</td>
<td>176.00</td>
</tr>
<tr>
<td>90.00</td>
<td>194.00</td>
</tr>
<tr>
<td>100.00</td>
<td>212.00</td>
</tr>
</tbody>
</table>

10.00 degrees Celsius is equivalent of 50.000 degrees Fahrenheit

Note the following

- The first argument of the `fprintf()` function is a kind of format statement included within `’` marks.
The remaining arguments are a list of variables or items to be printed separated by commas.

Within the format string there is text which is produced exactly as set down. There are also statements like %m.nf which produces a decimal number which is allowed m columns of output and has n places of decimals. These are applied in turn to the items in the list to be printed.

This f format is used to output floating point numbers there are a considerable number or other specifiers to output characters, strings, and number in formats other than floating point.

If the list to be printed is too long the formats are recycled.

Note the use of \n which means skip to the next line. This is essential.

I have already made some use of the fprintf() function to format the output of several examples in Chapter 2.

A small amendment to this program allows one to print formatted output to a file.

```
Sample MATLAB program demonstrating Formatted Output

clear
fid = fopen('test01.txt','w');
degrees_c =10:10:100;
degrees_f = (degrees_c * 9 /5) + 32;
fprintf(fid,'\n\n Conversion from degrees Celsius \n\n to degrees Fahrenheit
\n\n Celsius Fahrenheit
\n\n');
for ii = 1:10;
    fprintf(fid,'%12.2f%12.2f
',degrees_c(ii),degrees_f(ii));
end
%
fprintf(fid,...
'\n\n%5.2f degrees Celsius is equivalent of %5.3f degrees Fahrenheit\n',...
degrees_c(1),degrees_f(1));
fclose(fid);
```

There changes in this program are
1. The instruction `fid = fopen('test01.txt','w');` opens the file `test01.txt` for writing. The variable `fid` specifies the output file. If the file exists it will be overwritten. Substitute ‘a’ for ‘w’ if you wish to open a file and append output.

2. The variable `fid` has been added as a first argument to each call of the `fprintf` function.

3. The instruction `fclose(fid);` closes the file.

4. The output is the same as the previous case except that it is written to file rather than the command window,

### 3.7 Producing material for inclusion in a paper

In the MATLAB editor there is a tab labelled **PUBLISH**. The facilities under this tab allow you to produces output in WORD, Powerpoint, \LaTeX, HTML etc. for inclusion in papers, presentations etc. This facility was used to produce some of the material in this set of notes. The facilities are described in the help files and may vary from version to version of MATLAB.

Section 2.5, including its graph, was subject to very minor editing before being added to this document. I have also used the facility to produce transparencies for lectures on the use of MATLAB in economics/econometrics.

The next box contains a sample MATLAB script which has been set up to prepare material for publication in \LaTeX. Note the following

1. The script has been divided into sections by inserting two percent signs (%%) at the beginning of each section.
2. Next to each of these %% markers there is a section title
3. Add MATLAB comments to the script. These will provide a commentary in your finished document.
4. When writing your commentary there are facilities for
   
   (a) Bold, italic and mono-spaced text

---

*In older versions of MATLAB to access these facilities you had to first turn them on in the MATLAB Editor. This was done by [Cell|Enable Cell Mode] in the editor menu. Cell mode enabled you to divide your m-file into cells or sections. Cell mode in the editor was not to be confused with cell data structures in MATLAB. Enabling cell mode added a new set of buttons to the menu bar and enabled a set of items in the cell menu item. Many of the facilities described here were available but their implementation was different.*
3.7 Producing material for inclusion in a paper

(b) Hyperlinks
(c) Inline \LaTeX
(d) Bulleted/Numbered Lists
(e) Images
(f) Pre formatted text
(g) code
(h) Display \LaTeX

Sample Publish routine in MATLAB

```matlab
%% An example of publishing from matlab

%% The first section
% very simple
x=randn(15);
xbar = mean(x);
xsd = std(x);
% x has mean
xbar
% and standard deviation
xsd

%% An Equation
% $$e^{\pi i} + 1 = 0$$
%

%% A bulleted list
% * First bullet
% * second bullet
%```
The next box displays the \LaTeX code produced by this MATLAB script.

\begin{verbatim}
\documentclass{article}
\usepackage{graphicx}
\usepackage{color}
\sloppy
\definecolor{lightgray}{gray}{0.5}
\setlength{\parindent}{0pt}
\begin{document}
\section*{An example of publishing from matlab}
\subsection*{Contents}
\begin{itemize}
\setlength{\itemsep}{-1ex}
\item The first section
\item An Equation
\item A bulleted list
\end{itemize}
\subsection*{The first section}
\begin{verbatim}
very simple
\end{verbatim}
\vspace{1em}
\end{document}
\end{verbatim}
\begin{verbatim}
x = randn(15);
xbar = mean(x);
xsd = std(x);
\end{verbatim}
\color{lightgray} \begin{verbatim}
xbar =

Columns 1 through 7
0.6203  0.4835 -0.3327  0.1027  0.0897  0.0213 -0.1768

Columns 8 through 14
-0.0929  0.0546 -0.4624  0.1530 -0.3130 -0.0270  0.2205

Column 15
-0.0829

xsd =

Columns 1 through 7
1.6560  0.8653  1.1796  0.8674  1.0939  0.9352  1.1211

Columns 8 through 14
The actual output produced by this \LaTeX code is reproduced in figures 3.2 and 3.3. I have chosen \LaTeX for this example as I am using \LaTeX to type-set it. Other options can be configured by clicking on the publish icon under the PUBLISH tab and selecting Edit Publishing Options. There is another example of the use of these facilities in the MATLAB help files — search for "Publishing MATLAB Code".
### An example of publishing from matlab

**Contents**
- The first section
- An Equation
- A bulleted list

**The first section**

very simple

```matlab
x=randn(15);
xbar = mean(x);
xsd = std(x);
% x has mean
xbar
% and standard deviation
xsd

xbar =

<table>
<thead>
<tr>
<th>Columns 1 through 7</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6203</td>
<td>0.4835</td>
<td>-0.3327</td>
<td>0.1027</td>
<td>0.0897</td>
<td>0.0213</td>
<td>-0.1768</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Columns 8 through 14</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0929</td>
<td>0.0546</td>
<td>-0.4624</td>
<td>0.1530</td>
<td>-0.3130</td>
<td>-0.0270</td>
<td>0.2205</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column 15</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0829</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

xsd =

<table>
<thead>
<tr>
<th>Columns 1 through 7</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6560</td>
<td>0.8653</td>
<td>1.1796</td>
<td>0.8674</td>
<td>1.0939</td>
<td>0.9352</td>
<td>1.1211</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Columns 8 through 14</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2128</td>
<td>1.2542</td>
<td>0.9056</td>
<td>1.0532</td>
<td>0.9412</td>
<td>0.7429</td>
<td>0.6715</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column 15</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

| 1                    |         |         |         |         |         |         |
```

Figure 3.2: First page of \LaTeX{} output from Matlab generated script
1.0677

An Equation

\[ e^{\pi i} + 1 = 0 \]

A bulleted list

• First bullet
• second bullet

Figure 3.3: Second page of \LaTeX{} output from Matlab generated script
In composing MATLAB scripts, programs and functions you will often need to run a command or group of commands if certain conditions hold. On other occasions you may need to run the same command or group of commands a number of times.

In MATLAB there are four basic control (Decision or Loop Structures) available

**if statements** The basic form of the if statement is

```matlab
if conditions
    statements
end
```

The statements are only processed if the conditions are true. The conditions can include the following operators
==    equal
~==   not equal
<    less than
>    greater than
<=   less than or equal to
>=   greater than or equal to
&    logical and
&&   logical and (for scalars) short-circuiting
|    logical or
||   logical or (for scalars) short-circuiting
xor  logical exclusive or
all  true if all elements of vector are nonzero
any  true if any element of vector is nonzero

The if statement may be extended

```plaintext
if conditions
    statements1
else
    statements2
end
```

in which case statements1 are used if conditions are true and statements2 if false.

This if statement may be extended again

```plaintext
if conditions1
    statements1
elseif conditions2
    statements2
else
    statements3
end
```

with an obvious meaning (I hope).
for  The basic form of the for group is

```
for variable = expression
    statements
end
```

Here `expression` is probably a vector. `statements` is processed for each of the values in `expression`. The following example shows the use of a loop within a loop:

```
>> for ii = 1:3
    for jj=1:3
      total=ii+jj;
      fprintf(’%d + %d = %d \n’,ii,jj,total)
    end
  end

1 + 1 = 2
1 + 2 = 3
1 + 3 = 4
2 + 1 = 3
2 + 2 = 4
2 + 3 = 5
3 + 1 = 4
3 + 2 = 5
3 + 3 = 6
```

while  The format of the while statement is

```
while conditions
    statements
end
```

The while statement has the same basic functionality as the for statement. The for statement will be used when one knows precisely when and how many times an operation will be repeated. The statements are repeated so long as conditions are true.
**switch**  An example of the use of the **switch** statement follows

```plaintext
switch p
    case 1
        x = 24
    case 2
        x = 19
    case 3
        x = 15
    otherwise
        error('p must be 1, 2 or 3')
end
```

Use matrix statements in preference to loops. Not only are they more efficient but they are generally easier to use. That said there are occasions where one can not use a matrix statement.

If you wish to fill the elements of a vector or matrix using a loop it is good practice to initialise the vector or matrix first. For example if you wish to fill a $100 \times 20$ matrix, $X$, using a series of loops one could initialise the matrix using one of the following commands:

```plaintext
X = ones(100,20)
X = zeros(100,20)
X = ones(100,20)\times NaN
X = NaN(100,20)
```

and then evaluate the elements of the matrix within the loop or other control structure.
Simple graphs can be produced easily in MATLAB. The following sequence

**Script to graph regression residuals**

```matlab
% values for simulation
rng(1234); % same random numbers each time script is run
nsimul=50;
beta=[5,1,.1]';
%
x1=ones(nsimul,1); % constant
x2=[1:nsimul]'; % trend
x3=rand(nsimul,1)*2 +3; % Uniform(3,5)
x=[x1,x2,x3];
e=randn(nsimul,1)*.2; % N(0,.04)
y= x * beta +e; % 5*x1 + x2 + .1*x3 + e;
%
[nobs,nvar]=size(x);
betahat=inv(x'*x)*x'*y; %
```
Script to graph regression residuals (cont.)

\[ yhat = x \times betahat; \quad \% \; \text{beta(1)*x1-beta(2)*x2-beta(3)*x}; \]
\[ \text{resid} = y - yhat; \]
\[ \text{plot}(x2, \text{resid}) \]
\[ \text{title('Graph Title')} \]
\[ \text{xlabel('Time')} \]
\[ \text{ylabel('Residual')} \]
\[ \text{print -dpdf 'test.pdf'} \]

Figure 5.1: Residuals from a simulated regression

repeats the earlier OLS simulation (see page 57), opens a graph window, draws a graph of the residuals against the trend in the ols-simulation exercise, puts a title on the graph and labels the x and y axes. The vectors x2 and resid must have the same dimensions. The graph is then saved in pdf format and then imported into this document.

The following addition shows how to follow the previous plot of residuals by a plot of actual versus fitted values. the first command \texttt{figure}; opens a new graphics window. This is followed by two \texttt{plot} commands — one each for the actual and fitted values.
The **PLOTS** tab provides an easy way to generate a variety of Graphs. The following box illustrates the use of this tab to generate a semi log graph of Irish GDP over time. The file [NationalIncome.csv](www.cso.ie) contains two columns, year and NI (National Income at constant Prices) downloaded from [www.cso.ie](www.cso.ie) on 13 October 2014.

1. I use the Import Data to generate a program to import the data. (see Chapter 3).
2. Now select the data in year and NI. Select the **PLOTS** tab and click on the `semilogy` item in the ribbon. The graph will be generated in a new window.
3. You can now embellish the graph with various features using the `insert` item on the menu of the graphics box.
4. You can now save the graph in a variety of formats using the `File|Save` item from the menu bar on the graph window.
5. You may also use the `File|Generate Code...` menu to generate a function which produces this graph. Save the function in the working directory. The generated file is displayed in the box below. You may need to refer to Chapter 7 for details of user generated functions in MATLAB.

```matlab
function createfigure(X1, Y1)
%CREATEFIGURE(X1, Y1)
% X1: vector of x data
% Y1: vector of y data

% Auto-generated by MATLAB on 13-Oct-2014 21:53:10

% Create figure
figure1 = figure;

% Create axes
```

1 This file is embedded in this document. To extract it, please go to appendix B
The next box contains a script which brings all these details together —

1. Load the data
2. Draw the Graph
3. Save the Graph

```matlab
% Import data from text file.
% Script for importing data from the following text file:
% C:\TCD\document
% preparation\MatlabNew\TeX\Chapter05\NationalIncomeReal.csv
% To extend the code to different selected data or a different text
% file, generate a function instead of a script.

% Auto-generated by MATLAB on 2014/10/13 21:47:37
```
%% Initialize variables.
filename = 'C:\TCD\document preparation...\MatlabNew\TeX\Chapter05\NationalIncomeReal.csv';
delimiter = ',';
startRow = 2;

%% Format string for each line of text:
% column1: double (%f)
% column2: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%f%f[%\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to format string.
% This call is based on the structure of the file used to generate this code. If an error occurs for a different file, try regenerating the code from the Import Tool.
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter, ...
                  'HeaderLines', startRow-1, 'ReturnOnError', false);

%% Close the text file.
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so no post processing code is included. To generate code which works for unimportable data, select unimportable cells in a file and regenerate the script.

%% Allocate imported array to column variable names
year1 = dataArray{:, 1};
The graph produced by this script is reproduced in figure 5.2.

This chapter does not do justice to the graphics facilities available in MATLAB. The MATLAB graphics manual is currently just short of 600 pages. Many other graphs can be drawn using the PLOT tab in the GUI. Several of the additional MATLAB toolboxes that an economist might use also contain additional specialized graphics facilities.
6.1 Using MATLAB to estimate systems of regression equations

This section contains two examples of the estimation of systems of equations. The first is an examination of the classic Grunfeld investment data set. Many textbooks use this dataset to illustrate various features of system estimation. [Green (2000)] is the source of the data used here. Later editions of this book also examine these data but in less detail.

The MATLAB output also includes corresponding analysis using the le Sage econometrics package which is covered in section 8.2 of this note. As an exercise the user might extend the analysis to include various Likelihood Ratio tests of the restrictions imposed by the various estimation procedures.
Analysis of Grunfeld Investment data

Introduction

The basic system model that we are looking at here is

\[ y_{ti} = X_{ti}\beta_i + \varepsilon_{ti} \]

where \( 1 \leq i \leq M \) represents the individual agent or country or item for which we are estimating some equation and \( 1 \leq t \leq T \) represents the \( t^{th} \) measurement on the \( i^{th} \) unit. We assume that the variance of \( \varepsilon_{ti} \), \( \sigma_i^2 \) is constant for \( 1 \leq t \leq T \). Each \( X_i \) is \( T \times k_i \). We may write these equations

\[
\begin{align*}
    y_1 &= X_1\beta_1 + \varepsilon_1 \\
    y_2 &= X_2\beta_2 + \varepsilon_2 \\
    & \ldots \\
    y_M &= X_M\beta_M + \varepsilon_M 
\end{align*}
\]

In this section we will assume that \( X_i \) is exogenous for all \( 1 \leq i \leq M \). By imposing various cross-equation restrictions on the \( \beta_i \) and the covariances of the \( \varepsilon_{ti} \) we obtain a variety of estimators (e.g. Pooled OLS, Equation by Equation OLS, SUR).

The variables included in the Grunfeld analysis are

- **FIRM**: There are 10 firms
- **YEAR**: Data are from 1935 to 1954 (20 years)
- **I**: Gross Investment
- **F**: Value of Firm
- **C**: Stock of Plant and Equipment

For more details see [Green (2000)](2012) or the original references listed there.

- To reduce the amount of detail we shall restrict analysis to 5 firms
- **Firm no 1**: GM - General Motors
6.1 Using MATLAB to estimate systems of regression equations

- Firm no 4: GE - General Electric
- Firm no 3: CH - Chrysler
- Firm no 8: WE - Westinghouse
- Firm no 2: US - US Steel

The file Grunfeld.xlsx was generated from the Green data and transformed to xlsx format by importing the original csv data file into OpenOffice CALC and exporting it in the required format. While the script shows the command to import the data from the excel file the user may prefer to import it using the GUI.

The data set contains five columns. Row one contains variable names. Data for each firm is then given in 20 10 blocks each of 20 columns. The variable names on the data set are not imported as I wish to define the variables myself. The data is imported as a matrix which I have named data.

```matlab
load data; %
```

to reload and process the data

```
<table>
<thead>
<tr>
<th>Load and Generate Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>data = xlsread('Grunfeld.xlsx','A2:E201');</td>
</tr>
<tr>
<td>Y_GM = data(1:20, 3); % I</td>
</tr>
<tr>
<td>X_GM = [ones(20,1),data(1:20,[4 5])]; % constant F C</td>
</tr>
<tr>
<td>Y_GE = data(61:80, 3); % I</td>
</tr>
<tr>
<td>X_GE = [ones(20,1),data(61:80,[4 5])]; % constant F C</td>
</tr>
<tr>
<td>Y_CH = data(41:60, 3); % I</td>
</tr>
<tr>
<td>X_CH = [ones(20,1),data(41:60,[4 5])]; % constant F C</td>
</tr>
<tr>
<td>Y_WE = data(141:160, 3); % I</td>
</tr>
<tr>
<td>X_WE = [ones(20,1),data(141:160,[4 5])]; % constant F C</td>
</tr>
<tr>
<td>Y_US = data(21:40, 3); % I</td>
</tr>
<tr>
<td>X_US = [ones(20,1),data(21:40,[4 5])]; % constant F C</td>
</tr>
</tbody>
</table>
```

We now estimate the coefficients imposing various restrictions. Each estimation involves the following steps

1. Set up the required stacked $y$ and $X$ matrices.
2. Estimate the required coefficients.
3. Estimate standard errors, t-statistics etc.


6.1.1 Pooled OLS

The restrictions imposed by Pooled OLS are that corresponding coefficients are the same across equations. We also assume that the variance of the disturbances is constant across equations. Thus, in this case $k_i = k$, for all $i$. We can therefore assume that each observation on each unit is one more observation from the same single equation system. We may write the system as follows

$$
\begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_M
\end{pmatrix} =
\begin{pmatrix}
X_1 \\
X_2 \\
\vdots \\
X_M
\end{pmatrix} \beta +
\begin{pmatrix}
\varepsilon_1 \\
\varepsilon_2 \\
\vdots \\
\varepsilon_M
\end{pmatrix}
\quad (MT \times 1) \quad (MT \times k) \quad (k \times 1) \quad (MT \times 1)
$$

or, more compactly, using the obvious notation

$$
y = X\beta + \varepsilon
$$

and $\beta$ may be estimated by $\hat{\beta} = (X'X)^{-1}X'y$ etc. This is implemented in MATLAB as follows –

<table>
<thead>
<tr>
<th>Pooled OLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y = [Y_GM', Y_GE', Y_CH', Y_WE', Y_US']';</td>
</tr>
<tr>
<td>% leave out; for testing if</td>
</tr>
<tr>
<td>% necessary delete during run or output will be unreadable</td>
</tr>
<tr>
<td>X = [X_GM', X_GE', X_CH', X_WE', X_US']';</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>pols.beta = (X'*X)/X'*Y;</td>
</tr>
<tr>
<td>pols.uhat = Y - X*pols.beta;</td>
</tr>
<tr>
<td>pols.sigsq = (pols.uhat'*pols.uhat)/(size(X,1)-size(X,2));%T-k</td>
</tr>
<tr>
<td>pols.sdbeta = sqrt(diag(inv(X'*X))*pols.sigsq);</td>
</tr>
<tr>
<td>pols.tbeta = pols.beta ./ pols.sdbeta;</td>
</tr>
</tbody>
</table>

---

1 We could of course relax this condition and estimate Heteroskedastic Consistent Standard Errors
6.1 Using MATLAB to estimate systems of regression equations

Pooled OLS (cont.)

pols.se = sqrt(pols.sigsq);
label = ['Constant '; 'F '; 'C '];
disp('OLS Results using stacked matrices')
disp(' coef   sd   t-stat')
for ii=1:size(X,2)
    fprintf('%s%10.4f%10.4f%10.4f
',label(ii,:),pols.beta(ii),pols.sdbeta(ii), pols.tbeta(ii))
end
fprintf('Estimated Standard Error %10.4f

',pols.se)

OLS Results using stacked matrices

    coef   sd   t-stat
Constant  -47.0237 21.6292 -2.1741
F          0.1059  0.0114  9.2497
C          0.3014  0.0437  6.8915

Estimated Standard Error  128.1429

%  
% Verification using Lesage package
%  
pooled = ols(Y, X);
vnames= ['I ';
    'Constant ';
    'F ';
    'C '];
prt(pooled,vnames)

Ordinary Least-squares Estimates
Dependent Variable = I
R-squared = 0.7762
Rbar-squared = 0.7716
sigma^-2 = 16420.6075
Durbin-Watson = 0.3533
Nobs, Nvars = 100, 3

*****************************************************************************
6.1 Using MATLAB to estimate systems of regression equations

### Pooled OLS (cont.)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>t-statistic</th>
<th>t-probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>-47.023691</td>
<td>-2.174080</td>
<td>0.032132</td>
</tr>
<tr>
<td>F</td>
<td>0.105885</td>
<td>9.249659</td>
<td>0.000000</td>
</tr>
<tr>
<td>C</td>
<td>0.301385</td>
<td>6.891475</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

6.1.2 Equation by equation OLS

This section assumes that the coefficients vary across units. (In panel data estimation we assume that only the constant terms vary across units). We also assume that there is no contemporaneous correlation between the disturbances in the system. We may write the system of equations as

\[
\begin{pmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_M
\end{pmatrix} = 
\begin{pmatrix}
  X_1 & 0 & \cdots & 0 \\
  0 & X_2 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & X_M
\end{pmatrix} \beta + 
\begin{pmatrix}
  \varepsilon_1 \\
  \varepsilon_2 \\
  \vdots \\
  \varepsilon_M
\end{pmatrix}
\]

or more compactly using the obvious substitutions

\[y = X\beta + \varepsilon\]

where \(y\) and \(\varepsilon\) are \(TM \times 1\), \(X\) is \(TM \times kM\) and \(\beta\) is \(kM \times 1\). \(y\), \(\varepsilon\), and \(\beta\) are stacked versions of \(y_i\), \(\varepsilon_i\), and \(\beta_i\). The variance of \(\varepsilon\) is given by

\[
\Omega = E[\varepsilon \varepsilon'] =
\begin{pmatrix}
  \sigma_1^2 I_T & 0 & \cdots & 0 \\
  0 & \sigma_2^2 I_T & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & \sigma_M^2 I_T
\end{pmatrix}
\otimes I_T
\]
The coding of this example should be relatively clear. Perhaps the most difficult part is the estimation of the variances. The procedure here is very similar to that used in the first step of the SUR estimation procedure except that the contemporaneous correlation is used to improve the estimates. It should be noted that, in this case different variables are likely to be used in different equations. The only change required is in the calculation of the $X$ matrix.

<table>
<thead>
<tr>
<th>Equation by Equation OLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Y as before</td>
</tr>
<tr>
<td>$X = \text{blkdiag}(X_{\text{GM}}, X_{\text{GE}}, X_{\text{CH}}, X_{\text{WE}}, X_{\text{US}})$;</td>
</tr>
<tr>
<td>$\text{eqols.beta} = (X'X)^{-1}X'Y$;</td>
</tr>
<tr>
<td>$\text{eqols.uhat} = Y - X \times \text{eqols.beta}$;</td>
</tr>
<tr>
<td>$\text{eqols.temp} = \text{reshape}(\text{eqols.uhat}, \text{size}(X_{\text{GM}},1),5)$; % residuals for each firm in a column</td>
</tr>
<tr>
<td>$\text{eqols.sigsq1} = \text{eqols.temp}' \times \text{eqols.temp} / (\text{size}(X_{\text{GM}},1)-\text{size}(X_{\text{GM}},2))$;</td>
</tr>
<tr>
<td>$\text{eqols.sigsq} = \text{diag} (\text{diag} (\text{eqols.sigsq1}))$; % Remove non-diagonal elements</td>
</tr>
<tr>
<td>$\text{eqols.sdbeta} = \sqrt{\text{diag} (\text{inv} (X'X) \times X' \times \text{kron} (\text{eye}(\text{size}(X_{\text{GM}},1)), \text{eqols.sigsq}) \times X \times \text{inv} (X'X))}$;</td>
</tr>
<tr>
<td>$\text{eqols.covarbeta} = \text{inv} (X'X) \times \text{kron} (\text{eqols.sigsq}, \text{eye}(3))$;</td>
</tr>
<tr>
<td>$\text{eqols.sdbeta} = \text{diag} (\text{sqrt} (\text{eqols.covarbeta}))$;</td>
</tr>
<tr>
<td>$\text{eqols.tbeta} = \text{eqols.beta} / \text{eqols.sdbeta}$;</td>
</tr>
<tr>
<td>$\text{eqols.se} = \text{sqrt} (\text{diag} (\text{eqols.sigsq}))$;</td>
</tr>
<tr>
<td>%</td>
</tr>
<tr>
<td>% Write results</td>
</tr>
<tr>
<td>%</td>
</tr>
<tr>
<td>disp('OLS equation by equation using stacked matrices')</td>
</tr>
<tr>
<td>disp('OLS estimates GE equation')</td>
</tr>
</tbody>
</table>

```
firm = ['GE';
'GM';
'CH';
'WE';
'US'];
for jj = 1:5 % Loop over firms
fprintf('


');
disp('coef   sd   t-stat')
```
6.1 Using MATLAB to estimate systems of regression equations

for ii=1:3 %Loop over coefficients
    fprintf('%10s%10.4f%10.4f%10.4f\n',label(ii), ... 
        eqols.beta(ii+(jj-1)*3),eqols.sdbeta(ii+(jj-1)*3), ... 
        eqols.tbeta(ii+(jj-1)*3))
end
    fprintf('Standard Error is %10.4f\n',eqols.se(jj));
end

OLS equation by equation using stacked matrices

OLS estimates GE equation

<table>
<thead>
<tr>
<th>coef</th>
<th>sd</th>
<th>t-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-149.7825</td>
<td>105.8421</td>
</tr>
<tr>
<td>F</td>
<td>0.1193</td>
<td>0.0258</td>
</tr>
<tr>
<td>C</td>
<td>0.3714</td>
<td>0.0371</td>
</tr>
</tbody>
</table>

Standard Error is 91.7817

<table>
<thead>
<tr>
<th>coef</th>
<th>sd</th>
<th>t-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-6.1900</td>
<td>13.5065</td>
</tr>
<tr>
<td>F</td>
<td>0.0779</td>
<td>0.0200</td>
</tr>
<tr>
<td>C</td>
<td>0.3157</td>
<td>0.0288</td>
</tr>
</tbody>
</table>

Standard Error is 13.2786

<table>
<thead>
<tr>
<th>coef</th>
<th>sd</th>
<th>t-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-9.9563</td>
<td>31.3742</td>
</tr>
<tr>
<td>F</td>
<td>0.0266</td>
<td>0.0156</td>
</tr>
<tr>
<td>C</td>
<td>0.1517</td>
<td>0.0257</td>
</tr>
</tbody>
</table>

Standard Error is 27.8827
## Equation by Equation OLS (cont.)

<table>
<thead>
<tr>
<th>Coef</th>
<th>SD</th>
<th>T-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-0.5094</td>
<td>8.0153</td>
</tr>
<tr>
<td>F</td>
<td>0.0529</td>
<td>0.0157</td>
</tr>
<tr>
<td>C</td>
<td>0.0924</td>
<td>0.0561</td>
</tr>
</tbody>
</table>

Standard Error is 10.2131

<table>
<thead>
<tr>
<th>Coef</th>
<th>SD</th>
<th>T-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-49.1983</td>
<td>148.0754</td>
</tr>
<tr>
<td>F</td>
<td>0.1749</td>
<td>0.0742</td>
</tr>
<tr>
<td>C</td>
<td>0.3896</td>
<td>0.1424</td>
</tr>
</tbody>
</table>

Standard Error is 96.4345

```matlab
% % Verify using le Sage Toolbox

olsestim=ols(Y_US,X_US);
prt(olsestim, vnames);

Ordinary Least-squares Estimates
Dependent Variable = I
R-squared = 0.4709
Rbar-squared = 0.4086
sigma^2 = 9299.6040
Durbin-Watson = 0.9456
Nobs, Nvars = 20, 3

***************************************************************
Variable Coefficient t-statistic t-probability
Constant -49.198322 -0.332252 0.743761
F 0.174856 2.356612 0.030699
C 0.389642 2.736886 0.014049
```
6.1.3 SUR Estimates

Suppose that we have a random sample of households and we are have time series data on expenditure on holidays \((y_{it})\) and relevant explanatory variables. Suppose that we have sufficient data to estimate a single equation for each person in the sample. We also assume that there is no autocorrelation in each equation (often a rather heroic assumption). During the peak of the business cycle it is likely that many of the persons in the sample spend above what they do at the trough. Thus it is likely that there will be contemporaneous correlation between the errors in the system.

\[
E[\varepsilon_{it}\varepsilon_{sj}] = \begin{cases} 
\sigma_{ij} & \text{if } i = j \\
0 & \text{if } i \neq j
\end{cases}
\]

Thus we may write the contemporaneous covariance matrix \((\Sigma)\) as

\[
\Sigma = \begin{pmatrix}
\sigma_{11} & \sigma_{12} & \cdots & \sigma_{1M} \\
\sigma_{21} & \sigma_{22} & \cdots & \sigma_{2M} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{M1} & \sigma_{M2} & \cdots & \sigma_{MM}
\end{pmatrix}
\]

The total covariance matrix is, in this case, given by

\[
\Omega = \begin{pmatrix}
\Sigma & 0 & \cdots & 0 \\
0 & \Sigma & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \Sigma
\end{pmatrix} = \Sigma \otimes I_T
\]

If \(\Omega\) were known we would use GLS to get optimum estimates of \(\beta\). In this case we can obtain a consistent estimate of \(\Sigma\) from the residuals in the equation by equation OLS estimate that we have just completed. We can then use this consistent estimate in Feasible GLS.
### SUR Estimates

```matlab
Omega = kron(eqols.sigsq1,eye(20,20)); % Page page 256
eqsur.beta = inv(X'*inv(Omega)*X)*X'*inv(Omega)*Y;
eqsur.yhat = X * eqsur.beta;
eqsur.uhat = Y - eqsur.yhat;
eqsur.temp = reshape(eqsur.uhat,20,5);
eqsur.omega = eqsur.temp' * eqsur.temp /size(X_GM,1); %size(X_GM,1)-size(X_GM,2));
eqsur.covar = inv(X'*inv(kron(eqsur.omega, eye(20)))*X);
eqsur.sdbeta = sqrt(diag(eqsur.covar));
eqsur.tbeta = eqsur.beta ./ eqsur.sdbeta;
eqsur.se = sqrt(diag(eqsur.omega));

fprintf('SUR estimates
');
for jj = 1:5 % Loop over firms
    fprintf('n
');
    disp(' coef  sd  t-stat')
    for ii=1:3 %Loop over coefficients
        fprintf(' %10.4f%10.4f%10.4f',label(ii), ... 
            eqsur.beta(ii+(jj-1)*3),eqsur.sdbeta(ii+(jj-1)*3), ... 
            eqsur.tbeta(ii+(jj-1)*3))
    end
    fprintf('Standard Error is %10.4f\n',eqsur.se(jj));
end

SUR estimates

<table>
<thead>
<tr>
<th>coef</th>
<th>sd</th>
<th>t-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>-168.1134</td>
<td>84.9017</td>
<td>-1.9801</td>
</tr>
<tr>
<td>0.1219</td>
<td>0.0204</td>
<td>5.9700</td>
</tr>
<tr>
<td>0.3822</td>
<td>0.0321</td>
<td>11.9109</td>
</tr>
</tbody>
</table>

Standard Error is 84.9836
```
### SUR Estimates (cont.)

<table>
<thead>
<tr>
<th></th>
<th>coef</th>
<th>sd</th>
<th>t-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0.9980</td>
<td>11.5661</td>
<td>0.0863</td>
</tr>
<tr>
<td>F</td>
<td>0.0689</td>
<td>0.0170</td>
<td>4.0473</td>
</tr>
<tr>
<td>C</td>
<td>0.3084</td>
<td>0.0260</td>
<td>11.8766</td>
</tr>
</tbody>
</table>

Standard Error is 12.3789

<table>
<thead>
<tr>
<th></th>
<th>coef</th>
<th>sd</th>
<th>t-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>-21.1374</td>
<td>24.3479</td>
<td>-0.8681</td>
</tr>
<tr>
<td>F</td>
<td>0.0371</td>
<td>0.0115</td>
<td>3.2327</td>
</tr>
<tr>
<td>C</td>
<td>0.1287</td>
<td>0.0212</td>
<td>6.0728</td>
</tr>
</tbody>
</table>

Standard Error is 26.5467

<table>
<thead>
<tr>
<th></th>
<th>coef</th>
<th>sd</th>
<th>t-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1.4075</td>
<td>5.9944</td>
<td>0.2348</td>
</tr>
<tr>
<td>F</td>
<td>0.0564</td>
<td>0.0106</td>
<td>5.3193</td>
</tr>
<tr>
<td>C</td>
<td>0.0429</td>
<td>0.0382</td>
<td>1.1233</td>
</tr>
</tbody>
</table>

Standard Error is 9.7420

<table>
<thead>
<tr>
<th></th>
<th>coef</th>
<th>sd</th>
<th>t-stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>62.2563</td>
<td>93.0441</td>
<td>0.6691</td>
</tr>
<tr>
<td>F</td>
<td>0.1214</td>
<td>0.0451</td>
<td>2.6948</td>
</tr>
<tr>
<td>C</td>
<td>0.3691</td>
<td>0.1070</td>
<td>3.4494</td>
</tr>
</tbody>
</table>

Standard Error is 90.4117

%% SUR in LeSage toolbox

```matlab
y(1).eq = Y_GM;
y(2).eq = Y_GE;
```
6.1 Using MATLAB to estimate systems of regression equations

SUR Estimates (cont.)

\[
\begin{align*}
 y(3).eq &= Y_{CH}; \\
 y(4).eq &= Y_{WE}; \\
 y(5).eq &= Y_{US}; \\
 XX(1).eq &= X_{GM}; \\
 XX(2).eq &= X_{GE}; \\
 XX(3).eq &= X_{CH}; \\
 XX(4).eq &= X_{WE}; \\
 XX(5).eq &= X_{US}; \\
 neqs &= 5; \\
 sur_result &= \text{sur(neqs,y,XX)}; \\
 \text{prt(sur_result)}
\end{align*}
\]

Seemingly Unrelated Regression -- Equation 1

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>t-statistic</th>
<th>t-probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable 1</td>
<td>-168.113426</td>
<td>-1.980094</td>
<td>0.064116</td>
</tr>
<tr>
<td>variable 2</td>
<td>0.121906</td>
<td>5.969973</td>
<td>0.000015</td>
</tr>
<tr>
<td>variable 3</td>
<td>0.382167</td>
<td>11.910936</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Seemingly Unrelated Regression -- Equation 2

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>t-statistic</th>
<th>t-probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable 1</td>
<td>-168.113426</td>
<td>-1.980094</td>
<td>0.064116</td>
</tr>
<tr>
<td>variable 2</td>
<td>0.121906</td>
<td>5.969973</td>
<td>0.000015</td>
</tr>
<tr>
<td>variable 3</td>
<td>0.382167</td>
<td>11.910936</td>
<td>0.000000</td>
</tr>
</tbody>
</table>
### SUR Estimates (cont.)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>t-statistic</th>
<th>t-probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable 1</td>
<td>0.997999</td>
<td>0.086286</td>
<td>0.932247</td>
</tr>
<tr>
<td>variable 2</td>
<td>0.068861</td>
<td>4.047270</td>
<td>0.000837</td>
</tr>
<tr>
<td>variable 3</td>
<td>0.308388</td>
<td>11.876603</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Seemingly Unrelated Regression -- Equation 3

- System R-sqr = 0.8694
- R-squared = 0.6857
- Rbar-squared = 0.6488
- sigma^2 = 11785.8684
- Durbin-Watson = 0.0202
- Nobs, Nvars = 20, 3

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>t-statistic</th>
<th>t-probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable 1</td>
<td>-21.137397</td>
<td>-0.868140</td>
<td>0.397408</td>
</tr>
<tr>
<td>variable 2</td>
<td>0.037053</td>
<td>3.232726</td>
<td>0.004891</td>
</tr>
<tr>
<td>variable 3</td>
<td>0.128687</td>
<td>6.072805</td>
<td>0.000012</td>
</tr>
</tbody>
</table>

Seemingly Unrelated Regression -- Equation 4

- System R-sqr = 0.8694
- R-squared = 0.7264
- Rbar-squared = 0.6943
- sigma^2 = 2042.8631
- Durbin-Watson = 0.0323
- Nobs, Nvars = 20, 3

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>t-statistic</th>
<th>t-probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable 1</td>
<td>1.407487</td>
<td>0.234802</td>
<td>0.817168</td>
</tr>
<tr>
<td>variable 2</td>
<td>0.056356</td>
<td>5.319333</td>
<td>0.000056</td>
</tr>
<tr>
<td>variable 3</td>
<td>0.042902</td>
<td>1.123296</td>
<td>0.276925</td>
</tr>
</tbody>
</table>
### SUR Estimates (cont.)

Seemingly Unrelated Regression -- Equation 5

<table>
<thead>
<tr>
<th>System R-sqr</th>
<th>R-squared</th>
<th>Rbar-squared</th>
<th>sigma^2</th>
<th>Durbin-Watson</th>
<th>Nobs, Nvars</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8694</td>
<td>0.4528</td>
<td>0.3884</td>
<td>173504.8346</td>
<td>0.0103</td>
<td>20, 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable Coefficient t-statistic t-probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable 1 62.256312 0.669105 0.512413</td>
</tr>
<tr>
<td>variable 2 0.121402 2.694815 0.015340</td>
</tr>
<tr>
<td>variable 3 0.369111 3.449403 0.003062</td>
</tr>
</tbody>
</table>

Cross-equation sig(i,j) estimates

<table>
<thead>
<tr>
<th>equation</th>
<th>eq 1</th>
<th>eq 2</th>
<th>eq 3</th>
<th>eq 4</th>
<th>eq 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>eq 1</td>
<td>7222.2204</td>
<td>-315.6107</td>
<td>601.6316</td>
<td>129.7644</td>
<td>-2446.3171</td>
</tr>
<tr>
<td>eq 2</td>
<td>-315.6107</td>
<td>153.2369</td>
<td>3.1478</td>
<td>16.6475</td>
<td>414.5298</td>
</tr>
<tr>
<td>eq 3</td>
<td>601.6316</td>
<td>3.1478</td>
<td>704.7290</td>
<td>201.4385</td>
<td>1298.6953</td>
</tr>
<tr>
<td>eq 4</td>
<td>129.7644</td>
<td>16.6475</td>
<td>201.4385</td>
<td>94.9067</td>
<td>613.9925</td>
</tr>
<tr>
<td>eq 5</td>
<td>-2446.3171</td>
<td>414.5298</td>
<td>1298.6953</td>
<td>613.9925</td>
<td>8174.2798</td>
</tr>
</tbody>
</table>

Cross-equation correlations

<table>
<thead>
<tr>
<th>equation</th>
<th>eq 1</th>
<th>eq 2</th>
<th>eq 3</th>
<th>eq 4</th>
<th>eq 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>eq 1</td>
<td>1.0000</td>
<td>-0.3000</td>
<td>0.2667</td>
<td>0.1567</td>
<td>-0.3184</td>
</tr>
<tr>
<td>eq 2</td>
<td>-0.3000</td>
<td>1.0000</td>
<td>0.0096</td>
<td>0.1380</td>
<td>0.3704</td>
</tr>
<tr>
<td>eq 3</td>
<td>0.2667</td>
<td>0.0096</td>
<td>1.0000</td>
<td>0.7789</td>
<td>0.5411</td>
</tr>
<tr>
<td>eq 4</td>
<td>0.1567</td>
<td>0.1380</td>
<td>0.7789</td>
<td>1.0000</td>
<td>0.6971</td>
</tr>
<tr>
<td>eq 5</td>
<td>-0.3184</td>
<td>0.3704</td>
<td>0.5411</td>
<td>0.6971</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
6.2 Exercise – Using MATLAB to estimate a simultaneous equation system

Consider the demand-supply model

\[ q_t = \beta_{11} + \beta_{21}x_{t2} + \beta_{31}x_{t2} + \gamma_{21}p_t + u_{t1} \]  \hspace{1cm} (6.1)

\[ q_t = \beta_{12} + \beta_{42}x_{t4} + \beta_{52}x_{t5} + \gamma_{22}p_t + u_{t2}, \]  \hspace{1cm} (6.2)

where \( q_t \) is the log of quantity, \( p_t \) is the log of price, \( x_{t2} \) is the log of income, \( x_{t3} \) is a dummy variable that accounts for demand shifts \( x_{t4} \) and \( x_{t5} \) are input prices. Thus equations (6.1) and (6.2) are demand and supply functions respectively. 120 observations generated by this model are in the file demand-supply.csv

1. Comment on the identification of the system. Why can the system not be estimated using equation by equation OLS. For each of the estimates below produce estimates, standard errors and t-statistics of each coefficient. Also produce standard errors for each equation.

2. Estimate the system equation by equation using OLS.

3. Estimate the system equation by equation using 2SLS. Compare the results with the OLS estimates.

4. Set up the matrices of included variables, exogenous variables required to do system estimation.

5. Do OLS estimation using the stacked system and compare results with the equation by equation estimates.

6. Do 2SLS estimation using the stacked system and compare results with the equation by equation estimates.

7. Do 3SLS estimation using the stacked system and compare results with the 2SLS estimates.

8. Comment on the identification of the system.

9. How can the method be generalized to estimate other GMM estimators? Estimate the optimum GMM estimator for the system and compare your results with the previous estimators.
User written functions in MATLAB

7.1 Function m-files

One of the most useful facilities in MATLAB is the facility to write one’s own functions and use them in the same way as a native MATLAB functions. We are already familiar with m-files which contain MATLAB instructions. Such files are known as script files and allow us to do repeat an analysis without having to retype all the instructions. The file normdensity.m contains a function that estimates the density function of a normal distribution,

\[
\frac{1}{\sqrt{2\pi} \sigma} \exp - \frac{(x - \mu)^2}{2\sigma^2},
\]

Example of MATLAB Function

function f = normdensity(z, mu, sigma);
% Calculates the Density Function of the Normal Distribution
% with mean mu
% and standard deviation sigma
% at a point z
7.1 Function m-files

Example of MATLAB Function (cont.)

```matlab
% sigma must be a positive non-zero real number
if sigma <= 0
    fprintf('Invalid input\n');
    f = NaN;
else
    f = (1/(sqrt(2*pi)*sigma))*exp(-(z-mu)^2/(2*sigma^2));
end
```

Note the following

1. The file starts with the keyword function. This is to indicate that this m-file is a function definition.

2. In the first line the f indicates that the value of f when the file has been “run” is the value that will be returned.

3. The function is called with normdensity(z, mu, sigma) where z mu and sigma are given values in calling the function.

4. The commented lines immediately after the first line are a help system for the function

5. All variables within a function are local to the function. Thus if there is a variable within a function called x and one in the program with the same name the one in the function is used when the function is in operation. Once the program has been run the value in the function is forgotten and the value outside the program is used.

6. It is possible to have more than one function stored in a function file. Only the first function (the main function) in the file is visible to the calling program. Second and subsequent function in a function file can only be seen by this main function.

7. The file should be saved in the current working directory or in a folder on the current MATLAB search path. You may change the search path using the set path icon in the HOME tab in the GUI.

8. Your function operates in the same way as the official functions distributed with MATLAB.
The use of the function `normdensity` can be demonstrated as follows –

Get help for `normdensity` function.

```matlab
help normdensity
```

Calculates the Density Function of the Normal Distribution
with mean mu
and standard deviation sigma
at a point z
sigma must be a positive non-zero real number

Evaluate Standard Normal density function at zero

```matlab
normdensity(0,0,1)
```

ans =
0.3989

Plot standard normal density function

```matlab
fplot('normdensity(x,0,1)',[-3 3])
```
7.2 Anonymous functions

MATLAB Version 7 introduced the idea of an anonymous function. This is illustrated in the box below which shows how to define the normal distribution function in the previous example as an anonymous function.

\[
f1 = @(z,mu,sigma) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right);
\]

\[
f1\left(0,0,1\right)
\]

\[
\text{ans} =
\]

\[
0.3989
\]

The anonymous function is defined in the first line of the box.
1. The \texttt{f1} is a function handle for the anonymous function. This can be passed as an
input argument to another function.
2. The \texttt{@} character constructs the function. The parentheses \texttt{()} which follow contain
the input arguments of the function within brackets.
3. This is then followed by the definition of the function with a single MATLAB
statement.
4. You may use variables that are not in the argument list in the definition of the
function. If the value of these variables is changed after the function is defined the
function is not revised. If you wish to use the revised values of these variables you
must redefine the function.
5. One advantage of anonymous functions is that you do not have to maintain a
separate function file.
6. Perhaps more useful, the file handle can be passed as an input argument to another
function. For example, in estimating maximum likelihood estimates we may need
to pass the likelihood function to be maximised to a general optimisation routine.
8.1 Introduction

If you are accustomed to using one of the many packages that deal specifically with econometrics you may think that MATLAB takes a long time to do simple things. It is also clear that many or the more difficult tasks are often easier in MATLAB than in these packages. MATLAB is less of a “black box” than many of the other programs. One must really learn and understand the algebra before one can use MATLAB for econometrics. One also knows exactly what one is doing when one has written the routines in MATLAB.

The big problem is the lack of elementary econometric facilities in MATLAB. Two toolboxes are available to economists —

1. **The LeSage econometric toolbox.** This covers most of the econometric routines required for an econometrics course which is part of an economics primary or master’s degree.

2. **The MathWorks econometrics toolbox.** To use this toolbox you must also have installed the statistics, optimization and financial toolboxes. The MATLAB
8.2 LeSage Toolbox

**econometrics** toolbox concentrates to a large extent on the type of time series econometrics used in finance and does not cover much of the material that is in the LeSage toolbox.

### 8.2 LeSage Toolbox

2. Unzip the file to this directory. This creates 13 directories.
3. If you have administrative rights create a subdirectory in the toolbox subdirectory of your MATLAB installation. I called this directory `LeSage` as I also had an econ directory holding the MathWorks toolbox. If you do not have permissions to create the `LeSage` directory here create it some where in your user directory.
4. In your working directory create a file named `startup.m` with the following content. You may, if necessary copy theses directories to a subdirectory on a flash drive.

   ```matlab
   addpath(genpath('c:\Program Files\MATLAB\R2014a\toolbox\LeSage\'))
   
   On my PC I copied the LeSage econometrics sub-directories to the directory `c:\Program Files\MATLAB\R2014a\toolbox\LeSage\`. If you copied them to another directory or to a flash drive then you must amend this path. This instruction adds the LeSage directory and all its subdirectories at the front of the MATLAB path during the current session. Thus, if a function or variable in LeSage conflicts with a similar function in MATLAB or in a MATLAB toolbox the LeSage one takes precedence. I do not save the new MATLAB path. In this way if I start MATLAB in a directory which has not got that command in a `startup.m` it will start with the default MATLAB path.

6. All the programs, functions etc. in the LeSage toolbox contain excellent comments. The code is well written and relatively easy to understand. The commentary in the manual is also helpful. It should be relatively easy to extend the functions or to add new functions if the need arises.

---

1When MATLAB finds a `startup.m` file in the working directory at start-up it executes any commands in that file. In this way you can automatically load the data for your project or do any other initializations that you find useful.
The current version of this toolbox is optimized for MATLAB version 7. According to the material on the web-site the manual was last up-dated in September, 1999. It does give a good account of the reasoning behind the functions in the toolbox. The examples distributed with the code for the functions are more up to date and should be used rather than the examples in the manual. All of the examples in the distribution end in \ldots.d.m. Thus the function to estimate a vector autoregression is \texttt{vare.m} and the demonstration of that function is \texttt{vare_d.m}.

In each subdirectory of the toolbox there is a file \texttt{contents.html} which lists the functions, demonstrations etc. in that subdirectory. The material in appendix \ref{sec:appendixA} has been taken from these files.

One can summarize the contents of the subdirectories as follows

1. **Regression/Estimation Functions** \texttt{in table A.1}\; These cover estimation of OLS, HCSE Models, Box-Cox Models, Limited Dependent Variable Models, Simultaneous Equations, Bayesian Models, Panel Models etc.

2. **Diagnostics** \texttt{in table A.2}\; Included are a variety of residual and specification tests e.g. ARCH, Breusch-Pagan, Q-test, cusum, BKW, recursive residuals.

3. **Unit Roots and Cointegration** \texttt{in A.3}\; Covers Dickey-Fuller, Philips-Peron, HEGY seasonal unit roots and Johansen procedures.

4. **Vector Autoregression** \texttt{in table A.4}\; Covers standard VAR estimation routines, impulse response functions, causality tests and various Bayesian procedures.

5. **Markov Chain Monte Carlo** \texttt{in table A.5}\; A Gibbs sampling library of utility functions along with various estimation procedures is included here. functions are described in this chapter.

6. **Time Series Aggregation/Disaggregation** \texttt{in table A.6}\; These functions appear to have been contributed after the toolbox manual was written. This is the best collection of interpolation disaggregation and aggregation routines that I have encountered. The subdirectory contains two pdf files describing the methodologies. The demonstration files in this directory show how to run the functions within MATLAB. (The pdf documentation in the directory is based on calling the MATLAB functions from within EXCEL which I do not recommend.)

7. **Optimization** \texttt{in table A.7}\; The optimization function \texttt{fminsearch} in recent version of MATLAB employed as in Chapter \ref{chap:estimation} can satisfy most of an economist’s requirements.

8. **Plots and Graphs** \texttt{in table A.8}\; This section contains functions that produce
9. **Statistical Functions in table A.9** The functions listed in table A.8 calculate density functions, distribution functions, quantiles and simulate random samples for beta, binomial, chisquared, F, gamma, Hypergeometric, lognormal, logistic, normal, left- and right-truncated normal, multivariate normal, Poison, t and Wishart distributions. This is essential if you do not have access to the statistics toolbox in MATLAB.

10. **Utilities in table A.10** This is a collection of functions that an economist might find useful. There are four kinds of utilities — (i) functions for working with time series and dates, (ii) general functions for printing and plotting matrices as well as producing \texttt{LaTeX} formatted output of matrices for tables, (iii) econometric data transformations and (iv) functions that mimic functions in other software which are not available in MATLAB.

11. **Spatial Econometrics.** The toolbox also contains a collection of functions for spatial econometrics. For details see http://www.spatial-econometrics.com/

12. The **USCD-GARCH** package is now depreciated and has been replaced by the MFE Toolbox (see section 8.4).

You can use the \texttt{help} function in the COMMAND WINDOW to get help on the LeSage functions. This is illustrated in the following box which shows the help on the LeSage \texttt{ols} function.

```
>> help ols

PURPOSE: least-squares regression
---------------------------------------------------
USAGE: results = ols(y,x)
where: y = dependent variable vector (nobs x 1)
x = independent variables matrix (nobs x nvar)
---------------------------------------------------
RETURNS: a structure
results.meth = 'ols'
results.beta = bhat (nvar x 1)
results.tstat = t-stats (nvar x 1)
results.bstd = std deviations for bhat (nvar x 1)
```
8.2 LeSage Toolbox

Help on `ols` function in LeSage toolbox (cont.)

```matlab
results.yhat = yhat  (nobs x 1)
results.resid = residuals (nobs x 1)
results.sige = e'*e/(n-k)  scalar
results.rsqr = rsquared  scalar
results.rbar = rbar-squared scalar
results.dw = Durbin-Watson Statistic
results.nobs = nobs
results.nvar = nvars
results.y = y data vector (nobs x 1)
results.bint = (nvar x 2 ) vector with 95% confidence intervals on beta
```

SEE ALSO: `prt(results)`, `plt(results)`

The next box shows the demonstration program for the `ols()` function.

```matlab
ols demonstration program

```% PURPOSE: An example using ols(),
```
```
```% prt(),
```
```
```% plt(),
```
```% ordinary least-squares estimation
```
```
```% USAGE: ols_d
```
```
```%---------------------------------------------------
```
```rng(12345)  %JCF
nobs = 1000;
nvar = 15;
beta = ones(nvar,1);
```
```
```xmat = randn(nobs,nvar-1);
```
```
```x = [ones(nobs,1) xmat];
evec = randn(nobs,1);
```
```%---------------------------------------------------
```
y = x*beta + vec;

vnames = strvcat('y-vector','constant','x1','x2','x3','x4','x5','x6', ... 'x7','x8','x9','x10','x11','x12','x13','x14');

% do ols regression
result = ols(y,x);

% print the output
prt(result,vnames);
title('title string')
% plot the predicted and residuals
plt(result);
print -dpdf 'ols_d_01.pdf';%JCF
% pause; %JCF

% recover residuals
resid = result.resid;

% print out tstats
fprintf(1,'tstatistics = 
');
result.tstat

% plot actual vs. predicted
(tt=1:nobs;
figure % JCF
plot(tt,result.y,tt,result.yhat,'--');
title('Actual and Predicted')
print -dpdf 'ols_d_02.pdf';%JCF

>> ols_d

Ordinary Least-squares Estimates
8.2 LeSage Toolbox

ols demonstration program (cont.)

Dependent Variable = y-vector
R-squared = 0.9328
Rbar-squared = 0.9319
sigma^2 = 0.9954
Durbin-Watson = 2.0224
Nobs, Nvars = 1000, 15

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>t-statistic</th>
<th>t-probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>0.984796</td>
<td>31.085490</td>
<td>0.000000</td>
</tr>
<tr>
<td>x1</td>
<td>1.048733</td>
<td>33.075864</td>
<td>0.000000</td>
</tr>
<tr>
<td>;x2</td>
<td>0.980482</td>
<td>29.999217</td>
<td>0.000000</td>
</tr>
<tr>
<td>x3</td>
<td>1.052899</td>
<td>32.421795</td>
<td>0.000000</td>
</tr>
<tr>
<td>x4</td>
<td>0.996301</td>
<td>31.395523</td>
<td>0.000000</td>
</tr>
<tr>
<td>x5</td>
<td>0.970616</td>
<td>31.216304</td>
<td>0.000000</td>
</tr>
<tr>
<td>x6</td>
<td>1.016998</td>
<td>33.323262</td>
<td>0.000000</td>
</tr>
<tr>
<td>x7</td>
<td>1.005818</td>
<td>32.351407</td>
<td>0.000000</td>
</tr>
<tr>
<td>x8</td>
<td>0.995431</td>
<td>31.215225</td>
<td>0.000000</td>
</tr>
<tr>
<td>x9</td>
<td>0.990791</td>
<td>31.155604</td>
<td>0.000000</td>
</tr>
<tr>
<td>x10</td>
<td>1.003293</td>
<td>31.389013</td>
<td>0.000000</td>
</tr>
<tr>
<td>x11</td>
<td>0.993757</td>
<td>29.455230</td>
<td>0.000000</td>
</tr>
<tr>
<td>x12</td>
<td>1.014091</td>
<td>31.298235</td>
<td>0.000000</td>
</tr>
<tr>
<td>x13</td>
<td>0.984371</td>
<td>29.961983</td>
<td>0.000000</td>
</tr>
<tr>
<td>x14</td>
<td>1.047248</td>
<td>33.044102</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

tstatistics =

ans =

31.0855
33.0759
29.9992
32.4218
31.3955
31.2163
I have made some small amendments to the distributed `ols_d()` program.
These amendments are indicated by `%JCF` in the box above.

1. The `rng(12345)` command ensures that the same random numbers are generated each time the program is run. If you are working through LeSage demonstrations involving simulated data you may consider including a similar statement to ensure that you can replicate results.

2. Many of the LeSage demonstrations include `pause` statements. I prefer to remove these.

3. The `print -dpdf 'ols_d_01.pdf';%JCF` has been added to save the graph in pdf format for inclusion here. I have also save the second graph in the same way.

4. I have added a `figure` statement before the second graph is drawn.

5. I have added a title to the second graph.
8.3 MATLAB Econometrics package

http://www.mathworks.co.uk/help/econ/product-description.html contains the following overview of the econometrics toolbox. The help files and contents of this toolbox were described in more detail in subsection 1.3.8

<table>
<thead>
<tr>
<th>Model and analyse financial and economic systems using statistical methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Econometrics Toolbox provides functions for modelling economic data.</td>
</tr>
<tr>
<td>You can select and calibrate economic models for simulation and forecasting.</td>
</tr>
<tr>
<td>For time series modelling and analysis, the toolbox includes univariate ARMAX/GARCH composite models with several GARCH variants, multivariate VARMAX models, and cointegration analysis.</td>
</tr>
<tr>
<td>It also provides methods for modelling economic systems using state-space models and for estimating using the Kalman filter. You can use a variety of diagnostic functions for model selection, including hypothesis, unit root, and stationarity tests.</td>
</tr>
</tbody>
</table>

**Key Features**

- Univariate ARMAX/GARCH composite models, including EGARCH, GJR, and other variants
- Multivariate simulation and forecasting of VAR, VEC, and cointegrated models
- State-space models and the Kalman filter for estimation
- Tests for unit root (Dickey-Fuller, Phillips-Perron) and stationarity (Leybourne-McCabe, KPSS)
- Statistical tests, including likelihood ratio, LM, Wald, Engle’s ARCH, and Ljung-Box Q Cointegration tests, including Engle-Granger and Johansen
- Diagnostics and utilities, including AIC/BIC model selection and partial-, auto-, and cross-correlations
- Hodrick-Prescott filter for business-cycle analysis

This toolbox is of interest to those working in the financial econometrics and computational finance. It should be noted that the econometrics toolbox requires that the statistics, optimization and financial toolboxes be also installed.
8.4 Oxford MFE Toolbox

The Oxford MFE can be downloaded from [http://www.kevinsheppard.com/MFE_Toolbox](http://www.kevinsheppard.com/MFE_Toolbox). This toolbox covers many time-series and other econometric methods used in financial econometrics and quantitative finance. It also contains some relevant functions similar to those in the MATLAB statistics and financial toolboxes that can be substituted if the MATLAB versions are not available.

Unless you are sure about what you are doing you should not use the LeSage and MFE toolboxes together. The following details of the functions in the MFE toolbox are taken from its website.

The USCD-GARCH package is now depreciated and has been replaced by the MFE Toolbox. The new toolbox and a manual can be downloaded from [http://www.kevinsheppard.com/MFE_Toolbox](http://www.kevinsheppard.com/MFE_Toolbox). The MFE toolbox covers many time-series procedures used in financial econometrics. I would avoid loading the both the LeSage and MFE toolboxes.

<table>
<thead>
<tr>
<th>Details of MFE toolbox</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High Level List of Functions</strong></td>
</tr>
<tr>
<td>- Regression</td>
</tr>
<tr>
<td>- ARMA Simulation</td>
</tr>
<tr>
<td>- ARMA Estimation</td>
</tr>
<tr>
<td>- Heterogeneous Autoregression</td>
</tr>
<tr>
<td>- Information Criteria</td>
</tr>
<tr>
<td>- ARMA Forecasting</td>
</tr>
<tr>
<td>- Sample autocorrelation and partial autocorrelation</td>
</tr>
<tr>
<td>- Theoretical autocorrelation and partial autocorrelation</td>
</tr>
<tr>
<td>- Testing for serial correlation</td>
</tr>
<tr>
<td>- Ljung-Box Q Statistic</td>
</tr>
<tr>
<td>- LM Serial Correlation Test</td>
</tr>
<tr>
<td>- Filtering</td>
</tr>
<tr>
<td>- Baxter-King Filtering</td>
</tr>
<tr>
<td>- Hodrick-Prescott Filtering</td>
</tr>
<tr>
<td>- Regression with Time Series Data</td>
</tr>
</tbody>
</table>
### Details of MFE toolbox (cont.)

- **Long-run Covariance Estimation**
  - Newey-West covariance estimation
  - Den Hann-Levin covariance estimation

- **Nonstationary Time Series**
  - Unit Root Testing
  - Augmented Dickey-Fuller testing
  - Augmented Dickey-Fuller testing with automated lag selection

- **Vector Autoregressions**
  - Granger Causality Testing: grangercause
  - Impulse Response function calculation

- **Volatility Modeling**
  - ARCH/GARCH/AVARCH/TARCH/ZARCH Simulation
  - EGARCH Simulation
  - APARCH Simulation
  - FIGARCH Simulation
  - GARCH Model Estimation
  - ARCH/GARCH/GJR-GARCH/TARCH/AVGARCH/ZARCH Estimation
  - EGARCH Estimation
  - APARCH Estimation
  - AGARCH and NAGARCH estimation
  - IGARCH estimation
  - FIGARCH estimation
  - HEAVY models

- **Density Estimation**
  - Kernel Density Estimation

- **Distributional Fit Testing**
  - Jarque-Bera Test
  - Kolmogorov-Smirnov Test
  - Berkowitz Test

- **Bootstraps**
  - Block Bootstrap
  - Stationary Bootstrap
8.4 Oxford MFE Toolbox

<table>
<thead>
<tr>
<th>Details of MFE toolbox (cont.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Multiple Hypothesis Tests</td>
</tr>
<tr>
<td>– Reality Check and Test for Superior Predictive Accuracy</td>
</tr>
<tr>
<td>– Model Confidence Set</td>
</tr>
<tr>
<td>• Multivariate GARCH</td>
</tr>
<tr>
<td>– CCC MVGARCH</td>
</tr>
<tr>
<td>– Scalar Variance Targetting VECCH</td>
</tr>
<tr>
<td>– MATRIX GARCH</td>
</tr>
<tr>
<td>– DCC and ADCC</td>
</tr>
<tr>
<td>– OGARCH</td>
</tr>
<tr>
<td>– GOGARCH</td>
</tr>
<tr>
<td>– RARCH</td>
</tr>
<tr>
<td>• Realized Measures</td>
</tr>
<tr>
<td>– Realized Variance</td>
</tr>
<tr>
<td>– Realized Covariance</td>
</tr>
<tr>
<td>– Realized Kernels</td>
</tr>
<tr>
<td>– Multivariate Realized Kernels</td>
</tr>
<tr>
<td>– Realized Quantile Variance</td>
</tr>
<tr>
<td>– Two-scale Realized Variance</td>
</tr>
<tr>
<td>– Multi-scale Realized Variance</td>
</tr>
<tr>
<td>– Realized Range</td>
</tr>
<tr>
<td>– QMLE Realized Variance</td>
</tr>
<tr>
<td>– Min Realized Variance, Median Realized Variance (MinRV, MedRV)</td>
</tr>
<tr>
<td>– Integrated Quarticity Estimation</td>
</tr>
</tbody>
</table>

**Functions Missing available from Previous UCSD GARCH Toolbox**

The following list of function have not been updated and so if needed, you should continue to use the UCSD_GARCH code.

- GARCH in mean
- IDCC MVGARCH
- Shapirowilks
- Shapirofrancia
In many cases of maximum likelihood estimation there is no analytic solution to the optimisation problem and one must use numerical techniques. On some occasions the analytic solution may be very complicated and numerical techniques might be easier to use. The aim of this section is to demonstrate these techniques. The example used is based on the estimation of a tobit process.

This basic maximum likelihood algorithm could be applied in many econometric packages. In most cases, as in theory, one works with the log-likelihood rather than the likelihood. MATLAB, like many other packages contain a minimisation routines rather than maximisation. Thus one uses the equivalent procedure of minimising the negative of the log-likelihood.

The steps involved are as follows –

1. Load and process your data.
2. Write a MATLAB function to estimate the log-likelihood.
3. Calculate an initial estimate of the parameters to be estimated. You may use OLS, previous studies or even guesses. This will serve as starting values for the
optimisation routine.

4. Check the defaults for the optimization routine (e.g., maximum number of iterations, convergence criteria). If your initial attempt does not converge you may have to change these and/or use different starting values.

5. Call the optimisation routine.

6. Check for convergence. After a specified number of iterations the optimisation routine will stop and output results. If the routine has not converged the results, however good they look, are worthless. You must go back and look at your starting values or change the number of iterations or consider whether your model is appropriate. Many economists try to estimate models with many parameters and little data and the likelihood function may be almost flat over a wide region.

7. Estimate standard errors of your coefficients.

8. Print out results.

I shall use the \texttt{fminsearch} MATLAB function to carry out the optimisation. This routine uses what is known as the Nelder-Mead simplex\footnote{This has nothing to do with the simplex method in linear programming} method and does not require the use of derivatives. It is generally regarded as slow but sure. For a description of this and alternative minimisation routines I would recommend \textcite{Press et al. 2007}. Earlier versions of this book are available on-line at \url{www.nr.com}.

The MATLAB optimisation toolbox, \texttt{optim}, provides a wide range of optimisation functions. The Le Sage \texttt{econometrics} package also provides some other optimisation functions.

The data to be used in this example is a random sample drawn from the tobit process

\begin{equation}
Y^*_i = 5 + 2X_{2i} - 3X_{3i} + \varepsilon_i
\end{equation}

\begin{align*}
Y_i & = 0 \quad \text{if} \quad y^*_i \leq 0 \\
Y_i & = 1 \quad \text{if} \quad y^*_i > 0
\end{align*}

(9.1)

where \(\varepsilon_i\) follows a normal distribution with mean 0 and variance 9. The code in the following box generates a sample of 100 from this distribution.
Generation of Tobit Sample

```matlab
rng(2468);
nobs = 100;
X = [ones(nobs,1),10*rand(nobs,2)];
beta = [5,2,-3]';
epsilon = 3*randn(nobs,1);
Y = X * beta + epsilon;
% Truncate
Ytrunc = (Y > 0) .* Y;
```

Using the usual notation the log-likelihood for such a tobit process can then be written as

$$
\sum_{i=1}^{N} \left[ d_i \left(-\frac{1}{2} \ln 2\pi - \frac{1}{2} \ln \sigma^2 - \frac{1}{2\sigma^2}(y_i - x_i\beta)^2\right) + (1 - d_i) \ln \left(1 - \Phi \left(\frac{x_i\beta}{\sigma}\right)\right)\right]
$$

where $d_i$ is a dummy variable where

$$
d_i = \begin{cases} 
1 & \text{if } y_i > 0 \\
0 & \text{if } y_i \leq 0
\end{cases}
$$

The MATLAB function (tobit_like(b,x,y)) programmed in the following box calculates the likelihood of such a tobit process. It is an amended version of the corresponding program from the Le Sage econometrics toolbox. If you have access to the MATLAB statistics toolbox the probability density and distribution functions there can be used to simplify this function.

```matlab
function tobitlike = tobit_like(b,y,x);
% PURPOSE: evaluate tobit log-likelihood
% to demonstrate optimization routines
%-----------------------------------------------------
% USAGE: like = tobit_like(b,y,x)
% where: b = parameter vector (k x 1) to be estimated
%       b contains beta parameters and variance of
```
tobit_like.m (cont.)

% disturbance term (b(k)) - see note below
% y = dependent variable vector (n x 1)
% x = explanatory variables matrix (n x m)

%-----------------------------------------------------
% NOTE: this function returns a scalar equal to the
% negative of the log likelihood function
% or a scalar sum of the vector depending
% on the value of the flag argument
% k ~= m because we may have additional parameters
% in addition to the m bhat's (e.g., sigma)
%-----------------------------------------------------

% error check
if nargin ~= 3, error('wrong # of arguments to to_like1'); end;
[m1 m2] = size(b);
if m1 == 1
b = b';
end;

h = .000001; % avoid sigma = 0
[m junk] = size(b);
beta = b(1:m-1); % pull out bhat
sigma = max([b(m) h]); % pull out sigma
xb = x*beta;
% next two lines amended
llf1 = -0.5*log(2*pi) - 0.5*log(sigma^2) -((y-xb).^2)./(2*sigma^2);
xbs = xb./sigma; cdf = .5*(1+erf(xbs./sqrt(2)));
llf2 = log(h+(1-cdf));
llf = (y > 0).*llf1 + (y <= 0).*llf2;
tobitlike = -sum(llf); % scalar result

The next step is to estimate starting values. In the next box I use OLS to estimate these starting values.
The next box gives the call to the minimisation routine and the values of the output of that routine.

**parm** The values of the parameters as the routine finishes. The values found are close to the values used to simulate the process

**fval** In this case the negative of the maximum likelihood

**exitflag** An exit flag of 1 indicates that the process has converged. (0 indicates that it has not and that you have more work to do)

**output** This gives some details of the minimisation process.

### Minimisation

```matlab
[parm, fval, exitflag, output] = ...
fminsearch(@(parm)tobit_like(parm,Y,X),parm0)
fval
exitflag
output
```

parm =
5.1849  2.0336  -3.1373  3.4256

fval =
145.0262

exitflag =
1

output =

Minimisation (cont.)

iterations: 132
funcCount: 226
algorithm: 'Nelder-Mead simplex direct search'
message: 'Optimization terminated: the current x satisfies the ter...'

The next box shows how to obtain estimates of the standard errors of these estimates. The method used here involves using numerical methods to estimate the \( ij \)th element of the information matrix

\[
I(\theta)_{ij} = -\frac{\partial^2 l(\theta)}{\partial \theta_i \partial \theta_j}
\]

This matrix is then inverted to estimate the variance covariance matrix of the estimators. Their standard errors are given by the square roots of the diagonal elements of the inverted matrix. Numerical differentiation is calculated using the routines for the MATLAB hessian function available on the MATLAB exchange site at [http://www.mathworks.com/matlabcentral/fileexchange/13490-adaptive-robust-numerical-differentiation](http://www.mathworks.com/matlabcentral/fileexchange/13490-adaptive-robust-numerical-differentiation)

The next box summarises the results in a format similar to that produced by an econometric package.

Standard Errors

```matlab
%...tobit_like2 = @(parm)-tobit_like(parm, Y, X);
%...[hess,err] = hessian(tobit_like2,parm');
%...varcov = -inv(hess);
%...separm = sqrt(diag(varcov));
%...tparm = parm' ./ separm;
```

Printing Results

```matlab
fprintf('Summary Estimate of Tobit\n')
fprintf(' Coef. Std Error z\n')
fprintf('%12s%12.4f%12.4f%12.4f\n',...)
```
The next box gives the results of using the Gretl econometric package (Cottrell and Lucchetti, 2014) to estimate this Tobit process. As you can see both results are in close agreement.
Exercise

In Frain (2010) I illustrated this procedure by replicating the tobit analysis of tobacco expenditure on in Table 7.9 on page 237 of Verbeek (2008). Verify the results obtained there using the routines described in this chapter.\footnote{The relevant file, in native MATLAB format is embedded in this document. To extract it, please go to appendix B.}
MATLAB is an excellent program and is widely used in finance science and engineering. The documentation and user interface for MATLAB are excellent. There is also a lot of material on MATLAB available on the internet.

There have been many occasions when I needed access to MATLAB and was forced to find an alternative. For security reasons, your employer may place various restrictions on the programs you can use on a work computer. If you want to use MATLAB you may need local management approval, IT management approval and purchase through a central purchasing unit. Your employer’s MATLAB license may not allow you to use MATLAB on a home computer where you want to do some work. In some cases you may only use MATLAB on an occasional basis and the cost of licenses for MATLAB and a variety of toolboxes may not be justified.

In such cases, you might consider Octave, Scilab or R which are free programs with similar functionality to MATLAB.
10.1 Octave

Octave is free software distributed under the Gnu General Public License. A new version containing a modern GUI is under development. For the moment instructions for downloading and installing an unofficial version of Octave 3.8.2 for windows can be obtained from [http://wiki.octave.org/Octave_for_Windows#Octave_3.8_MXE_Builds](http://wiki.octave.org/Octave_for_Windows#Octave_3.8_MXE_Builds). If you are running Windows 8 be sure that you follow the instructions as you may have problems running the GUI in Windows 8. Several toolboxes are available with this distribution. The file

Octave is largely compatible with MATLAB to the extent that most programs written in base MATLAB will also run in base Octave. In base Octave there are some additional options that are not available in MATLAB. Some MATLAB toolboxes have full or partial implementation as Octave toolboxes. The file [README.html](http://wiki.octave.org/Octave_for_Windows#Octave_3.8_MXE_Builds) in the install directory of this distribution describes how to install these packages and where to get additional packages.

A program for base MATLAB will run in Octave with at most minor changes and, in all likelihood with none. The original drafts of these note were completed at home with Octave as I had no access to MATLAB there at the time. Programs written in Octave may not run in base MATLAB as base Octave contains many functions similar to those in add-on MATLAB toolboxes. These make Octave a better package for econometrics than base MATLAB. [Creel (2014)](http://www.econometrics.com/creel.html) is a set of econometrics notes based with applications in Octave. Examples, data-sets and programs are available on the web with the notes.

Two useful references for those making the transition from MATLAB to octave are

1. [http://wiki.octave.org/FAQ#How_is_Octave_different_from_Matlab.3F](http://wiki.octave.org/FAQ#How_is_Octave_different_from_Matlab.3F)
2. Differences between Octave and MATLAB (Wikibooks.)

Many of the programs and functions in the LeSage package can be made to run in Octave. Many of the LeSage programs will crash and report a missing `fcnchk()` function. Create a file containing the following and save it on the Octave path. I have saved it in the `util` subdirectory of the LeSage toolbox.

```
function f=fcnchk(x, n)
    f = x;
end
```
You will also get “Short-circuit & and | operators” warnings when you are using the LeSage toolbox. To avoid these add

```octave
do_braindead_shortcircuit_evaluation(1)
```

to your `.octavrc` file or otherwise run this instruction before you call any LeSage functions.

### 10.2 Scilab

Scilab is another free matrix manipulation language available from [www.scilab.org](http://www.scilab.org). Scilab has the same basic functionality as MATLAB. Scilab syntax is different to that of MATLAB. Scilab programs will need considerable editing before they could be used in MATLAB. Scilab contains a utility for translating MATLAB programs to Scilab. Campbell et al. (2006) is a good introduction to Scilab and contains a lot of tutorial material. There is also an econometrics toolbox for Scilab called GROCER. While this package is partly derived from the LeSage package it has been updated and includes a lot of additional features.

One may well ask which is the best program. MATLAB is definitely the market leader in the field. It is very much embedded in the scientific/engineering fields with branches in Finance. It has applications in advanced macroeconomics and is a suitable tool for empirical research. Octave is very similar to MATLAB. Octave has better facilities than base MATLAB. The combination of Scilab and GROCER makes a most interesting tool for economics and deserves to be better known. If I was working in a MATLAB environment where good support was available in-house I would not try anything else. If I wanted a program to run my MATLAB programs at home and did not want to go to the expense of acquiring a licence for basic MATLAB and tools I would try Octave first. If I was just interested in some private empirical research Scilab would be worth trying. Scilab and Grocer should be used more widely in economics/econometrics Experience gained in programming Octave or Scilab would transfer easily to MATLAB.
The third alternative that I have used for this kind of work is R \cite{R-Core-Team-2014}. R has been more suitable for much of the work that I have been doing recently.

R is “GNU S”, a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and non-linear modelling, statistical tests, time series analysis, classification, clustering, etc. More information is available from The Comprehensive R Archive Network (CRAN) at http://www.r-project.org/ or at one of its many mirror sites. Not only does R cover all aspects of statistics but it has most of the computational facilities of MATLAB. It is the package in which most academic statistical work is being completed. There is a large amount of free tutorial material available on CRAN and an increasing number of textbooks on R have been published in recent years.

If it cannot be done in basic R then one is almost certain to find a solution in one of the 6000+ “official packages” on CRAN or the “unofficial packages” on other sites. R is regarded as having a steep learning curve but there are several graphical interfaces that facilitate the use of R. Summary information of the use of R in economics and finance can be seen on the Task views on the CRAN web site or on one of its many mirrors. \cite{Kleiber-Zeileis-2008} is a good starting point for econometrics in R.
A.1 Regression

The regression function library is in a subdirectory `regress`. This covers estimation of OLS, HCSE Models, Box-Cox Models, Limited Dependent Variable Models, Simultaneous Equations, Bayesian Models, Panel Models etc.

Table A.1: Regression in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>Programs and Demonstrations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar1_like</td>
<td>evaluate ols model with AR1 errors log-likelihood</td>
</tr>
<tr>
<td>ar_g</td>
<td>MCMC estimates Bayesian heteroscedastic AR(k) model</td>
</tr>
<tr>
<td>ar_gd</td>
<td>An example using ar_g(),</td>
</tr>
<tr>
<td>box_lik</td>
<td>evaluate Box-Cox model likelihood function</td>
</tr>
<tr>
<td>boxc_trans</td>
<td>compute box-cox transformation</td>
</tr>
<tr>
<td>boxcox</td>
<td>box-cox regression using a single scalar transformation</td>
</tr>
<tr>
<td>boxcox_d</td>
<td>An example using box_cox(),</td>
</tr>
<tr>
<td>demo_reg</td>
<td>demo using most all regression functions</td>
</tr>
</tbody>
</table>

*continued on next page*
Regression in LeSage Econometrics Toolbox  continued

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>felogit</td>
<td>computes binomial logistic regression with a one-dimensional fixed effect</td>
</tr>
<tr>
<td>felogit_demo</td>
<td>demonstrate use of felogit.m</td>
</tr>
<tr>
<td>felogit_lik</td>
<td>Compute probabilities and value of log-likelihood</td>
</tr>
<tr>
<td>garch_like</td>
<td>log likelihood for garch model</td>
</tr>
<tr>
<td>garch_sigt</td>
<td>generate garch model sigmas over time</td>
</tr>
<tr>
<td>garch_trans</td>
<td>function to transform garch(1,1) a0,a1,a2 garch parameters</td>
</tr>
<tr>
<td>ham_itrans</td>
<td>inverse transform Hamilton model parameters</td>
</tr>
<tr>
<td>ham_like</td>
<td>log likelihood function for Hamilton’s model</td>
</tr>
<tr>
<td>ham_trans</td>
<td>transform Hamilton model parameters</td>
</tr>
<tr>
<td>hwhite</td>
<td>computes White’s adjusted heteroscedastic</td>
</tr>
<tr>
<td>hwhite_d</td>
<td>An example of hwhite(),</td>
</tr>
<tr>
<td>ksmooth</td>
<td>Kim’s smoothing for Hamilton() model</td>
</tr>
<tr>
<td>lad</td>
<td>least absolute deviations regression</td>
</tr>
<tr>
<td>lad_d</td>
<td>An example using lad(),</td>
</tr>
<tr>
<td>lmtest</td>
<td>computes LM-test for two regressions</td>
</tr>
<tr>
<td>lmtest_d</td>
<td>demo using lmtest()</td>
</tr>
<tr>
<td>lo_like</td>
<td>evaluate logit log-likelihood</td>
</tr>
<tr>
<td>logit</td>
<td>computes Logit Regression</td>
</tr>
<tr>
<td>logit_d</td>
<td>An example of logit(),</td>
</tr>
<tr>
<td>mlogit</td>
<td>multinomial logistic regression</td>
</tr>
<tr>
<td>mlogit_d</td>
<td>An example of mlogit(),</td>
</tr>
<tr>
<td>mlogit_lik</td>
<td>Calculates likelihood for multinomial logit regression model.</td>
</tr>
<tr>
<td>multilogit</td>
<td>implements multinomial logistic regression</td>
</tr>
<tr>
<td>multilogit_demo</td>
<td>demonstrates the use of multilogit.m</td>
</tr>
<tr>
<td>multilogit_lik</td>
<td>Computes value of log likelihood function for multinomial logit regression</td>
</tr>
<tr>
<td>nwest</td>
<td>computes Newey-West adjusted heteroscedastic-serial</td>
</tr>
<tr>
<td>nwest_d</td>
<td>An example using nwest(),</td>
</tr>
<tr>
<td>ols</td>
<td>least-squares regression</td>
</tr>
<tr>
<td>ols_d</td>
<td>An example using ols(),</td>
</tr>
<tr>
<td>ols_g</td>
<td>MCMC estimates for the Bayesian heteroscedastic linear model</td>
</tr>
<tr>
<td>ols_gcbma</td>
<td>MC^3 x-matrix specification for homoscedastic OLS model</td>
</tr>
</tbody>
</table>

continued on next page
<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ols_gcbmad</td>
<td>Demo of ols_gcbma() model comparison function</td>
</tr>
<tr>
<td>ols_gd</td>
<td>demo of ols_g()</td>
</tr>
<tr>
<td>ols_gv</td>
<td>MCMC estimates for the Bayesian heteroscedastic linear model</td>
</tr>
<tr>
<td>ols_gvd</td>
<td>demo of ols_g()</td>
</tr>
<tr>
<td>olsar1</td>
<td>computes maximum likelihood ols regression for AR1 errors</td>
</tr>
<tr>
<td>olsar1_d</td>
<td>demonstrate olsc, olsar1 routines</td>
</tr>
<tr>
<td>olsc</td>
<td>computes Cochrane-Orcutt ols Regression for AR1 errors</td>
</tr>
<tr>
<td>olsc_d</td>
<td>demonstrate ols_corc roc</td>
</tr>
<tr>
<td>olse</td>
<td>OLS regression returning only residual vector</td>
</tr>
<tr>
<td>olsrs</td>
<td>Restricted least-squares estimation</td>
</tr>
<tr>
<td>olsrs_d</td>
<td>An example using olsrs(),</td>
</tr>
<tr>
<td>olst</td>
<td>ols with t-distributed errors</td>
</tr>
<tr>
<td>olst_d</td>
<td>An example using olst(),</td>
</tr>
<tr>
<td>panel_d</td>
<td>Demonstrates use of panel data estimation</td>
</tr>
<tr>
<td>pfixed</td>
<td>performs Fixed Effects Estimation for Panel Data</td>
</tr>
<tr>
<td>phaussman</td>
<td>prints haussman test, use for testing the specification of the fixed or</td>
</tr>
<tr>
<td>plt_eqs</td>
<td>plots regression actual vs predicted and residuals for:</td>
</tr>
<tr>
<td>plt_gibbs</td>
<td>Plots output from Gibbs sampler regression models</td>
</tr>
<tr>
<td>plt_reg</td>
<td>plots regression actual vs predicted and residuals</td>
</tr>
<tr>
<td>plt_tvp</td>
<td>Plots output using tvp regression results structures</td>
</tr>
<tr>
<td>ppooled</td>
<td>performs Pooled Least Squares for Panel Data(for balanced or unbalanced data)</td>
</tr>
<tr>
<td>pr_like</td>
<td>evaluate probit log-likelihood</td>
</tr>
<tr>
<td>prrandom</td>
<td>performs Random Effects Estimation for Panel Data</td>
</tr>
<tr>
<td>probit</td>
<td>computes Probit Regression</td>
</tr>
<tr>
<td>probit_d</td>
<td>demo of probit()</td>
</tr>
<tr>
<td>probit_g</td>
<td>MCMC sampler for the Bayesian heteroscedastic Probit model</td>
</tr>
<tr>
<td>probit_gd</td>
<td>demo of probit_g</td>
</tr>
<tr>
<td>prt_bmao</td>
<td>print results from ols_gcbma function</td>
</tr>
<tr>
<td>prt_eqs</td>
<td>Prints output from multiple equation regressions</td>
</tr>
<tr>
<td>prt_felogit</td>
<td>Prints output from felogit function</td>
</tr>
<tr>
<td>prt_gibbs</td>
<td>Prints output from Gibbs sampler regression models</td>
</tr>
<tr>
<td>prt_multilogit</td>
<td>Prints output from multilogit function</td>
</tr>
</tbody>
</table>

*continued on next page*
Regression in LeSage Econometrics Toolbox  

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prt_panel</td>
<td>Prints Panel models output</td>
</tr>
<tr>
<td>prt_reg</td>
<td>Prints output using regression results structures</td>
</tr>
<tr>
<td>prt_swm</td>
<td>Prints output from Switching regression models</td>
</tr>
<tr>
<td>prt_tvp</td>
<td>Prints output using tvp() regression results structures</td>
</tr>
<tr>
<td>ridge</td>
<td>computes Hoerl-Kennard Ridge Regression</td>
</tr>
<tr>
<td>ridge_d</td>
<td>An example using ridge(), bkw()</td>
</tr>
<tr>
<td>ridge_d2</td>
<td>An example using ridge(), bkw()</td>
</tr>
<tr>
<td>robust</td>
<td>robust regression using iteratively reweighted</td>
</tr>
<tr>
<td>robust_d</td>
<td>An example using robust(),</td>
</tr>
<tr>
<td>rtrace</td>
<td>Plots atheta ridge regression estimates</td>
</tr>
<tr>
<td>sur</td>
<td>computes seemingly unrelated regression estimates</td>
</tr>
<tr>
<td>sur_d</td>
<td>An example using sur(),</td>
</tr>
<tr>
<td>switch_em</td>
<td>Switching Regime regression (EM-estimation)</td>
</tr>
<tr>
<td>switch_emd</td>
<td>Demo of switch_em</td>
</tr>
<tr>
<td>theil</td>
<td>computes Theil-Goldberger mixed estimator</td>
</tr>
<tr>
<td>theil_d</td>
<td>An example using theil(),</td>
</tr>
<tr>
<td>thsls</td>
<td>computes Three-Stage Least-squares Regression</td>
</tr>
<tr>
<td>thsls_d</td>
<td>An example using thsls(),</td>
</tr>
<tr>
<td>to_like</td>
<td>evaluate tobit log-likelihood</td>
</tr>
<tr>
<td>to_rlike</td>
<td>evaluate tobit log-likelihood</td>
</tr>
<tr>
<td>tobit</td>
<td>computes Tobit Regression</td>
</tr>
<tr>
<td>tobit_d</td>
<td>An example using tobit()</td>
</tr>
<tr>
<td>tobit_d2</td>
<td>An example using tobit()</td>
</tr>
<tr>
<td>tobit_g</td>
<td>MCMC sampler for Bayesian Tobit model</td>
</tr>
<tr>
<td>tobit_gd</td>
<td>An example using tobit_g()</td>
</tr>
<tr>
<td>tobit_gd2</td>
<td>An example using tobit_g()</td>
</tr>
<tr>
<td>tsls</td>
<td>computes Two-Stage Least-squares Regression</td>
</tr>
<tr>
<td>tsls_d</td>
<td>An example using tsls(),</td>
</tr>
<tr>
<td>tvp</td>
<td>time-varying parameter maximum likelihood estimation</td>
</tr>
<tr>
<td>tvp_d</td>
<td>An example using tvp(),</td>
</tr>
<tr>
<td>tvp_garch</td>
<td>time-varying parameter estimation with garch(1,1) errors</td>
</tr>
<tr>
<td>tvp_garch_like</td>
<td>log likelihood for tvp_garch model</td>
</tr>
<tr>
<td>tvp_garchd</td>
<td>An example using tvp_garch(),</td>
</tr>
</tbody>
</table>

continued on next page
Regression in LeSage Econometrics Toolbox  

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tvp_like</td>
<td>returns -log likelihood function for tvp model</td>
</tr>
<tr>
<td>tvp_markov</td>
<td>time-varying parameter model with Markov switching error variances</td>
</tr>
<tr>
<td>tvp_markov_lik</td>
<td>log-likelihood for Markov-switching TVP model</td>
</tr>
<tr>
<td>tvp_markovd</td>
<td>An example using tvp_markov(),</td>
</tr>
<tr>
<td>tvp_markovd2</td>
<td>An example using tvp_markov(), and tvp_garch()</td>
</tr>
<tr>
<td>tvp_zglike</td>
<td>returns -log likelihood function for tvp model with Zellner’s g-prior</td>
</tr>
<tr>
<td>waldf</td>
<td>computes Wald F-test for two regressions</td>
</tr>
<tr>
<td>waldf_d</td>
<td>demo using waldf()</td>
</tr>
</tbody>
</table>

A.2 Diagnostics

Regression/residual diagnostics from the diag directory are included in Table ???. Included are the usual ARCH, Breusch-Pagan, Q-test, cusum, BKW, recursive residuals etc.

Table A.2: Regression/Residual Diagnostics in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>Diagnostics and Demonstrations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>arch</td>
<td>computes a test for ARCH(p)</td>
</tr>
<tr>
<td>arch_d</td>
<td>demo of arch() test for ARCH(p)</td>
</tr>
<tr>
<td>bkw</td>
<td>computes and prints BKW collinearity diagnostics</td>
</tr>
<tr>
<td>bkw_d</td>
<td>demo of bkw()</td>
</tr>
<tr>
<td>bpagan</td>
<td>Breusch-Pagan heteroscedasticity test</td>
</tr>
<tr>
<td>bpagan_d</td>
<td>An example of bpagan()</td>
</tr>
<tr>
<td>cusums</td>
<td>computes cusum-squares test</td>
</tr>
<tr>
<td>cusums_d</td>
<td>demo of rec_resid()</td>
</tr>
<tr>
<td>dfbeta</td>
<td>computes BKW (influential observation diagnostics)</td>
</tr>
<tr>
<td>dfbeta_d</td>
<td>demo of dfbeta(), plt_dfb()</td>
</tr>
<tr>
<td>rdiagnose</td>
<td>computes regression diagnostic measures (see RETURNS)</td>
</tr>
<tr>
<td>rdiagnose_d</td>
<td>demo of rdiagnose()</td>
</tr>
<tr>
<td>plt_cus</td>
<td>plots cusum squared tests</td>
</tr>
<tr>
<td>plt_dfb</td>
<td>plots BKW influential observation diagnostics</td>
</tr>
<tr>
<td>plt_dff</td>
<td>plots BKW influential observation diagnostics</td>
</tr>
</tbody>
</table>

continued on next page
### Appendix A.3

Regression in LeSage Econometrics Toolbox *continued*

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qstat2</td>
<td>computes the Ljung-Box Q test for AR(p)</td>
</tr>
<tr>
<td>qstat2_d</td>
<td>demo of qstat2() Ljung-Box Q test</td>
</tr>
<tr>
<td>rdiag</td>
<td>residual analysis plots</td>
</tr>
<tr>
<td>rdiag_d</td>
<td>demo of rdiagn()</td>
</tr>
<tr>
<td>recresid</td>
<td>compute recursive residuals</td>
</tr>
<tr>
<td>recresid_d</td>
<td>demo of recresid()</td>
</tr>
<tr>
<td>studentize</td>
<td>If x is a vector, subtract its mean and divide</td>
</tr>
<tr>
<td>unstudentize</td>
<td>returns reverse studentized vector</td>
</tr>
<tr>
<td>unstudentize_d</td>
<td>demonstrate unstudentize function</td>
</tr>
</tbody>
</table>

### A.3 Unit Roots and Cointegration

The unit root and co-integration tests and estimation procedures include Dickey-Fuller, Philips-Peron, HEGY seasonal unit roots and Johansen.

Table A.3: Unit Roots and Cointegration in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>Program and Demonstrations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acf</td>
<td>Estimate the coefficients of the autocorrelation</td>
</tr>
<tr>
<td>adf</td>
<td>carry out DF tests on a time-series vector</td>
</tr>
<tr>
<td>adf_d</td>
<td>demonstrate the use of adf()</td>
</tr>
<tr>
<td>c_sja</td>
<td>find critical values for Johansen maximum eigenvalue statistic</td>
</tr>
<tr>
<td>c_sjt</td>
<td>find critical values for Johansen trace statistic</td>
</tr>
<tr>
<td>cadf</td>
<td>compute augmented Dickey-Fuller statistic for residuals</td>
</tr>
<tr>
<td>cadf_d</td>
<td>demonstrate the use of cadf()</td>
</tr>
<tr>
<td>cthegy</td>
<td>return critical values for the hegy statistics</td>
</tr>
<tr>
<td>detrend</td>
<td>detrend a matrix y of time-series using regression</td>
</tr>
<tr>
<td>hegy</td>
<td>performs the Hylleberg, Engle, Granger and Yoo(1990)</td>
</tr>
<tr>
<td>hegy_d</td>
<td>demo program for hegy</td>
</tr>
<tr>
<td>johansen</td>
<td>perform Johansen cointegration tests</td>
</tr>
<tr>
<td>johansen_d</td>
<td>demonstrate the use of johansen()</td>
</tr>
<tr>
<td>phillips</td>
<td>compute Phillips-Perron test of the unit-root hypothesis</td>
</tr>
<tr>
<td>phillips_d</td>
<td>demonstrate the use of phillips()</td>
</tr>
</tbody>
</table>

*continued on next page*
### Appendix A.4

Unit Roots and Cointegration LeSage Econometrics Toolbox *continued*

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prt_coint</td>
<td>Prints output from co-integration tests</td>
</tr>
<tr>
<td>ptrend</td>
<td>produce an explanatory variables matrix</td>
</tr>
<tr>
<td>rztcrit</td>
<td>return critical values for the Zt statistic used in cadf()</td>
</tr>
<tr>
<td>ztcrit</td>
<td>return critical values for the Zt statistic used in adf()</td>
</tr>
</tbody>
</table>

#### A.4 Vector Autoregression — Classical/Bayesian

Included are the standard var estimation routines, impulse response functions, causality tests and various Bayesian procedures.

Table A.4: Vector Autoregression in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>Programs and Demonstrations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
</tr>
<tr>
<td>arf</td>
</tr>
<tr>
<td>arf_d</td>
</tr>
<tr>
<td>becm</td>
</tr>
<tr>
<td>becm_d</td>
</tr>
<tr>
<td>becm_g</td>
</tr>
<tr>
<td>becmf</td>
</tr>
<tr>
<td>becmf_d</td>
</tr>
<tr>
<td>becmf_g</td>
</tr>
<tr>
<td>becmf_gd</td>
</tr>
<tr>
<td>bvar</td>
</tr>
<tr>
<td>bvar_d</td>
</tr>
<tr>
<td>bvar_g</td>
</tr>
<tr>
<td>bvar_gd</td>
</tr>
<tr>
<td>bvarf</td>
</tr>
<tr>
<td>bvarf_d</td>
</tr>
<tr>
<td>bvarf_g</td>
</tr>
<tr>
<td>bvarf_gd</td>
</tr>
<tr>
<td>ecm</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecm_d</td>
<td>demonstrate the use of ecm()</td>
</tr>
<tr>
<td>ecmf</td>
<td>estimates an error correction model of order n</td>
</tr>
<tr>
<td>ecmf_d</td>
<td>demonstrate the use of ecmf</td>
</tr>
<tr>
<td>irf</td>
<td>Calculates Impulse Response Function for VAR</td>
</tr>
<tr>
<td>irf_d</td>
<td>An example of using irf</td>
</tr>
<tr>
<td>irf_d2</td>
<td>An example of using irf</td>
</tr>
<tr>
<td>lrratio</td>
<td>performs likelihood ratio test for var model</td>
</tr>
<tr>
<td>lrratio_d</td>
<td>demonstrate the use of lrratio()</td>
</tr>
<tr>
<td>pftest</td>
<td>prints VAR model ftests</td>
</tr>
<tr>
<td>pftest_d</td>
<td>An example of using pftest</td>
</tr>
<tr>
<td>pgranger</td>
<td>prints VAR model Granger-causality results</td>
</tr>
<tr>
<td>plt_var</td>
<td>plots VAR model actual vs predicted and residuals</td>
</tr>
<tr>
<td>plt_varg</td>
<td>Plots Gibbs sampled VAR model results</td>
</tr>
<tr>
<td>prt_var</td>
<td>Prints vector autoregressive models output</td>
</tr>
<tr>
<td>prt_varg</td>
<td>Prints vector autoregression output</td>
</tr>
<tr>
<td>recm</td>
<td>performs Bayesian error correction model estimation</td>
</tr>
<tr>
<td>recm_d</td>
<td>demonstrate the use of recm</td>
</tr>
<tr>
<td>recm_g</td>
<td>Gibbs sampling estimates for Bayesian error correction</td>
</tr>
<tr>
<td>recm_gd</td>
<td>An example of using recm_g function</td>
</tr>
<tr>
<td>recmf</td>
<td>Estimates a Bayesian error correction model of order n</td>
</tr>
<tr>
<td>recmf_d</td>
<td>An example of using recmf()</td>
</tr>
<tr>
<td>recmf_g</td>
<td>Gibbs sampling forecasts for Bayesian error correction</td>
</tr>
<tr>
<td>recmf_gd</td>
<td>An example of using recmf_g function</td>
</tr>
<tr>
<td>rvar</td>
<td>Estimates a Bayesian vector autoregressive model</td>
</tr>
<tr>
<td>rvar_d</td>
<td>An example of using rvar() function</td>
</tr>
<tr>
<td>rvar_g</td>
<td>Gibbs estimates for a Bayesian vector autoregressive</td>
</tr>
<tr>
<td>rvar_gd</td>
<td>An example of using rvar_g function</td>
</tr>
<tr>
<td>rvarf</td>
<td>Estimates a Bayesian autoregressive model of order n</td>
</tr>
<tr>
<td>rvarf_d</td>
<td>An example of using rvarf()</td>
</tr>
<tr>
<td>rvarf_g</td>
<td>Gibbs forecasts for a Bayesian vector autoregressive</td>
</tr>
<tr>
<td>rvarf_gd</td>
<td>An example of using rvarf_g()</td>
</tr>
<tr>
<td>svar</td>
<td>svar verifies the identification conditions for a given structural form</td>
</tr>
</tbody>
</table>

continued on next page
Appendix A.6

Vector Autoregression LeSage Econometrics Toolbox continued

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>svar_d</td>
<td>An example of using svar</td>
</tr>
<tr>
<td>vare</td>
<td>performs vector autoregressive estimation</td>
</tr>
<tr>
<td>var_d</td>
<td>An example of using var, pgranger, prt_var, plt_var</td>
</tr>
<tr>
<td>varf</td>
<td>estimates a vector autoregression of order n</td>
</tr>
<tr>
<td>varf_d</td>
<td>An example of using varf()</td>
</tr>
</tbody>
</table>

A.5 Markov chain Monte Carlo (MCMC)

Table A.5: MCMC in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>Programs and Demonstrations</th>
</tr>
</thead>
<tbody>
<tr>
<td>apm</td>
</tr>
<tr>
<td>apm_d</td>
</tr>
<tr>
<td>coda</td>
</tr>
<tr>
<td>coda_d</td>
</tr>
<tr>
<td>momentg</td>
</tr>
<tr>
<td>momentg_d</td>
</tr>
<tr>
<td>prt_coda</td>
</tr>
<tr>
<td>raftery</td>
</tr>
<tr>
<td>raftery_d</td>
</tr>
</tbody>
</table>

A.6 Time Series Aggregation/Disaggregation

This is the best collection of interpolation disaggregation and aggregation routines that I have encountered.

Table A.6: Unit Roots and Cointegration in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>Programs and Demonstrations</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggreg</td>
</tr>
<tr>
<td>aggreg_test</td>
</tr>
</tbody>
</table>

continued on next page
### Time Series Aggregation/Disaggregation in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggreg_v</td>
<td>Generate a temporal aggregation vector</td>
</tr>
<tr>
<td>bal</td>
<td>Proportional adjustment of y to a given total z</td>
</tr>
<tr>
<td>bal_d</td>
<td>Demo of bal()</td>
</tr>
<tr>
<td>bfl</td>
<td>Temporal disaggregation using the Boot-Feibes-Lisman method</td>
</tr>
<tr>
<td>bfl_d</td>
<td>demo of bfl()</td>
</tr>
<tr>
<td>calt</td>
<td>Phi-weights of ARIMA model in matrix form</td>
</tr>
<tr>
<td>chowlin</td>
<td>Temporal disaggregation using the Chow-Lin method</td>
</tr>
<tr>
<td>chowlin_d</td>
<td>demo of chowlin()</td>
</tr>
<tr>
<td>chowlin_fix</td>
<td>Temporal disaggregation using the Chow-Lin method</td>
</tr>
<tr>
<td>conta</td>
<td>determine number of non-f elements in polynomial</td>
</tr>
<tr>
<td>denton</td>
<td>Multivariate temporal disaggregation with transversal constraint</td>
</tr>
<tr>
<td>denton_d</td>
<td>Demo of denton()</td>
</tr>
<tr>
<td>denton_uni</td>
<td>Temporal disaggregation using the Denton method</td>
</tr>
<tr>
<td>denton_uni_d</td>
<td>Demo of denton_uni()</td>
</tr>
<tr>
<td>denton_uni_prop</td>
<td>Temporal disaggregation: Denton method, proportional variant</td>
</tr>
<tr>
<td>denton_uni_prop_d</td>
<td>Demo of denton_uni_prop()</td>
</tr>
<tr>
<td>desvec</td>
<td>Creates a matrix unstacking a vector</td>
</tr>
<tr>
<td>dif</td>
<td>Generate the difference operator in matrix form</td>
</tr>
<tr>
<td>difonzo</td>
<td>Multivariate temporal disaggregation with transversal constraint</td>
</tr>
<tr>
<td>difonzo_d</td>
<td>Demo of difonzo()</td>
</tr>
<tr>
<td>fernandez</td>
<td>Temporal disaggregation using the Fernandez method</td>
</tr>
<tr>
<td>fernandez_d</td>
<td>demo of fernandez()</td>
</tr>
<tr>
<td>guerrero</td>
<td>ARIMA-based temporal disaggregation: Guerrero method</td>
</tr>
<tr>
<td>guerrero_d</td>
<td>demo of guerrero()</td>
</tr>
<tr>
<td>inter_xls</td>
<td>Interface via Excel Link for univariate temporal disaggregation</td>
</tr>
<tr>
<td>inter_xls_d</td>
<td>demo of inter_xls()</td>
</tr>
<tr>
<td>litterman</td>
<td>Temporal disaggregation using the Litterman method</td>
</tr>
<tr>
<td>litterman_d</td>
<td>demo of litterman()</td>
</tr>
<tr>
<td>litterman_fix</td>
<td>Temporal disaggregation using the Litterman method</td>
</tr>
<tr>
<td>minter_xls</td>
<td>Interface via Excel Link for multivariate temporal disaggregation</td>
</tr>
<tr>
<td>movingsum</td>
<td>Accumulates h consecutive periods of a vector nx1</td>
</tr>
<tr>
<td>mtd_plot</td>
<td>Graphic output of multivariate temporal disaggregation methods</td>
</tr>
</tbody>
</table>

*continued on next page*
### Time Series Aggregation/Disaggregation in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mtd_print</td>
<td>Save output of multivariate temporal disaggregation methods</td>
</tr>
<tr>
<td>numpar</td>
<td>determine the number of non-zero values of ARIMA model</td>
</tr>
<tr>
<td>rossi</td>
<td>Multivariate temporal disaggregation with transversal constraint</td>
</tr>
<tr>
<td>rossi_d</td>
<td>Demo of rossi()</td>
</tr>
<tr>
<td>ssc</td>
<td>Temporal disaggregation using the dynamic Chow-Lin method</td>
</tr>
<tr>
<td>ssc_d</td>
<td>demo of ssc()</td>
</tr>
<tr>
<td>ssc_fix</td>
<td>Temporal disaggregation using the dynamic Chow-Lin method</td>
</tr>
<tr>
<td>sw</td>
<td>Temporal disaggregation using the Stram-Wei method.</td>
</tr>
<tr>
<td>sw_d</td>
<td>demo of sw()</td>
</tr>
<tr>
<td>tasa</td>
<td>Compute the year-on-year rate of growth</td>
</tr>
<tr>
<td>td_plot</td>
<td>Generate graphic output of temporal disaggregation methods</td>
</tr>
<tr>
<td>td_print</td>
<td>Generate output of temporal disaggregation methods</td>
</tr>
<tr>
<td>td_print_g</td>
<td>Generate output of temporal disaggregation methods.</td>
</tr>
<tr>
<td>tduni_plot</td>
<td>Generate graphic output of the BFL or Denton</td>
</tr>
<tr>
<td>tduni_print</td>
<td>Save output of BFL or Denton temporal disaggregation methods</td>
</tr>
<tr>
<td>temporal_agg</td>
<td>Temporal aggregation of a time series</td>
</tr>
<tr>
<td>vdp</td>
<td>Balancing by means of van der Ploeg method</td>
</tr>
<tr>
<td>vdp_d</td>
<td>Demo of vdp()</td>
</tr>
</tbody>
</table>

### A.7 Optimization Programs

The optimization function `fminsearch` in recent version of MATLAB employed as in Chapter 9.

#### Table A.7: Unit Roots and Cointegration in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>Programs and Demonstrations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization functions</td>
</tr>
<tr>
<td>banana</td>
</tr>
<tr>
<td>banana_d</td>
</tr>
<tr>
<td>dfp_min</td>
</tr>
<tr>
<td>dfp_mind</td>
</tr>
</tbody>
</table>

*continued on next page*
Appendix A.8

Time Series Aggregation/Disaggregation in LeSage Econometrics Toolbox continued

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>frpr_min</td>
<td>Fletcher, Reeves, Polak, Ribiere minimization routine to minimize func</td>
</tr>
<tr>
<td>frpr_mind</td>
<td>Demonstrate frpr_min optimization function</td>
</tr>
<tr>
<td>hessian</td>
<td>Computes finite difference Hessian</td>
</tr>
<tr>
<td>maxlik</td>
<td>minimize a log likelihood function</td>
</tr>
<tr>
<td>optim1_d</td>
<td>An example using dfp_min, frpr_min, pow_min, maxlik</td>
</tr>
<tr>
<td>optim2_d</td>
<td>An example using fmin function</td>
</tr>
<tr>
<td>optim3_d</td>
<td>An example of optimization</td>
</tr>
<tr>
<td>pow_min</td>
<td>Powell minimization routine to minimize func</td>
</tr>
<tr>
<td>pow_mind</td>
<td>Demonstrate pow_min optimization function</td>
</tr>
<tr>
<td>to_like1</td>
<td>evaluate tobit log-likelihood</td>
</tr>
<tr>
<td>to_like2</td>
<td>evaluate tobit log-likelihood</td>
</tr>
<tr>
<td>to_liked</td>
<td>evaluate tobit log-likelihood</td>
</tr>
<tr>
<td>tvp_beta</td>
<td>generate tvp model betas and forecast error variance</td>
</tr>
<tr>
<td>tvp_like1</td>
<td>returns -log likelihood function for tvp model</td>
</tr>
<tr>
<td>tvp_like2</td>
<td>returns log likelihood function for tvp model</td>
</tr>
</tbody>
</table>

A.8 Plots and Graphs

This section contains functions that produce graphs of interest in econometrics.

Table A.8: Plots and Graphs in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>Programs and Demonstrations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>histo</td>
<td>Plot a histogram</td>
</tr>
<tr>
<td>k_pdf</td>
<td>Plots a univariate kernel density</td>
</tr>
<tr>
<td>pairs</td>
<td>Pairwise scatter plots of the columns of x</td>
</tr>
<tr>
<td>pairs_d</td>
<td>demo of pairs (pairwise scatterplots)</td>
</tr>
<tr>
<td>plt_turns</td>
<td>plot turning points in a time series</td>
</tr>
<tr>
<td>plt_turnsd</td>
<td>demo of ftturns()</td>
</tr>
<tr>
<td>plttdens</td>
<td>Draw a nonparametric density estimate.</td>
</tr>
<tr>
<td>plttdens_d</td>
<td>demo of plttdens (non-parameter density plot)</td>
</tr>
<tr>
<td>spyc</td>
<td>colorful version of matlab spy.c function</td>
</tr>
</tbody>
</table>

continued on next page
Plots and Graphs in LeSage Econometrics Toolbox continued

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsplot</td>
<td>time-series plot with dates and labels</td>
</tr>
<tr>
<td>tsplot_d</td>
<td>demo of tsplot (time-series plot)</td>
</tr>
</tbody>
</table>

A.9 Statistical Functions

This functions listed in table A.8 calculate density functions, distribution functions, quantiles and simulate random samples for beta, binomial, chisquared, F, gamma, Hypergeometric, lognormal, logistic, normal, left- and right-truncated normal, multivariate normal, Poison, t and Wishart distributions. This is essential material if you do not have access to the statistics toolbox in MATLAB.

Table A.9: Statistical Functions in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>Programs and Demonstrations</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta_cdf</td>
</tr>
<tr>
<td>beta_d</td>
</tr>
<tr>
<td>beta_inv</td>
</tr>
<tr>
<td>beta_pdf</td>
</tr>
<tr>
<td>beta_rnd</td>
</tr>
<tr>
<td>bincoef</td>
</tr>
<tr>
<td>bingen</td>
</tr>
<tr>
<td>bino_cdf</td>
</tr>
<tr>
<td>bino_d</td>
</tr>
<tr>
<td>bino_pdf</td>
</tr>
<tr>
<td>bino_rnd</td>
</tr>
<tr>
<td>chis_cdf</td>
</tr>
<tr>
<td>chis_d</td>
</tr>
<tr>
<td>chis_inv</td>
</tr>
<tr>
<td>chis_pdf</td>
</tr>
<tr>
<td>chis_prb</td>
</tr>
<tr>
<td>chis_rnd</td>
</tr>
<tr>
<td>com_size</td>
</tr>
<tr>
<td>demo_distr</td>
</tr>
</tbody>
</table>

continued on next page
Statistical Functions in LeSage Econometrics Toolbox *continued*

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fdis_cdf</td>
<td>returns cdf at x of the F(a,b) distribution</td>
</tr>
<tr>
<td>fdis_d</td>
<td>demo of F-distribution functions</td>
</tr>
<tr>
<td>fdis_inv</td>
<td>returns inverse (quantile) at x of the F(a,b) distribution</td>
</tr>
<tr>
<td>fdis_pdf</td>
<td>returns pdf at x of the F(a,b) distribution</td>
</tr>
<tr>
<td>fdis_prb</td>
<td>computes f-distribution probabilities</td>
</tr>
<tr>
<td>fdis_rnd</td>
<td>returns random draws from the F(a,b) distribution</td>
</tr>
<tr>
<td>gamm_cdf</td>
<td>returns the cdf at x of the gamma(a) distribution</td>
</tr>
<tr>
<td>gamm_d</td>
<td>demo of gamma distribution functions</td>
</tr>
<tr>
<td>gamm_inv</td>
<td>returns the inverse of the cdf at p of the gamma(a) distribution</td>
</tr>
<tr>
<td>gamm_pdf</td>
<td>returns the pdf at x of the gamma(a) distribution</td>
</tr>
<tr>
<td>gamm_rnd</td>
<td>a matrix of random draws from the gamma distribution</td>
</tr>
<tr>
<td>hypg_cdf</td>
<td>hypergeometric cdf function</td>
</tr>
<tr>
<td>hypg_d</td>
<td>demo of Hypergeometric distribution functions</td>
</tr>
<tr>
<td>hypg_inv</td>
<td>hypergeometric inverse (quantile) function</td>
</tr>
<tr>
<td>hypg_pdf</td>
<td>hypergeometric pdf function</td>
</tr>
<tr>
<td>hypg_rnd</td>
<td>hypergeometric random draws</td>
</tr>
<tr>
<td>is_scalar</td>
<td>determines if argument x is scalar</td>
</tr>
<tr>
<td>logn_cdf</td>
<td>cdf of the lognormal distribution</td>
</tr>
<tr>
<td>logn_d</td>
<td>demo of log-normal distribution functions</td>
</tr>
<tr>
<td>logn_inv</td>
<td>inverse cdf (quantile) of the lognormal distribution</td>
</tr>
<tr>
<td>logn_pdf</td>
<td>pdf of the lognormal distribution</td>
</tr>
<tr>
<td>logn_rnd</td>
<td>random draws from the lognormal distribution</td>
</tr>
<tr>
<td>logt_cdf</td>
<td>cdf of the logistic distribution</td>
</tr>
<tr>
<td>logt_d</td>
<td>demo of logistic distribution functions</td>
</tr>
<tr>
<td>logt_inv</td>
<td>inv of the logistic distribution</td>
</tr>
<tr>
<td>logt_pdf</td>
<td>pdf of the logistic distribution at x</td>
</tr>
<tr>
<td>logt_rnd</td>
<td>random draws from the logistic distribution</td>
</tr>
<tr>
<td>norm_cdf</td>
<td>computes the cumulative normal distribution</td>
</tr>
<tr>
<td>norm_crnd</td>
<td>random numbers from a contaminated normal distribution</td>
</tr>
<tr>
<td>norm_inv</td>
<td>computes the quantile (inverse of the CDF)</td>
</tr>
<tr>
<td>norm_pdf</td>
<td>computes the normal probability density function</td>
</tr>
<tr>
<td>norm_prb</td>
<td>computes normal probability given</td>
</tr>
<tr>
<td>norm_prbd</td>
<td>demo of norm_prb function</td>
</tr>
</tbody>
</table>

*continued on next page*
### Statistical Functions in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>norm_rnd</code></td>
<td>random multivariate random vector based on</td>
</tr>
<tr>
<td><code>norme_d</code></td>
<td>demo of contaminated normal random numbers</td>
</tr>
<tr>
<td><code>normlt_d</code></td>
<td>demo of left-truncated normal random numbers</td>
</tr>
<tr>
<td><code>normlt_inv</code></td>
<td>compute inverse of cdf for left-truncated normal</td>
</tr>
<tr>
<td><code>normlt_rnd</code></td>
<td>compute random draws from a left-truncated normal</td>
</tr>
<tr>
<td><code>normrt_d</code></td>
<td>demo of right-truncated normal random numbers</td>
</tr>
<tr>
<td><code>normrt_inv</code></td>
<td>compute inverse of cdf for right-truncated normal</td>
</tr>
<tr>
<td><code>normrt_rnd</code></td>
<td>compute random draws from a right-truncated normal</td>
</tr>
<tr>
<td><code>normt_d</code></td>
<td>demo of truncated normal random numbers</td>
</tr>
<tr>
<td><code>normt_inv</code></td>
<td>compute inverse of cdf for truncated normal</td>
</tr>
<tr>
<td><code>normt_rnd</code></td>
<td>random draws from a normal truncated to (left,right) interval</td>
</tr>
<tr>
<td><code>pois_cdf</code></td>
<td>computes the cumulative distribution function at x</td>
</tr>
<tr>
<td><code>pois_d</code></td>
<td>demo of poisson distribution functions</td>
</tr>
<tr>
<td><code>pois_inv</code></td>
<td>computes the quantile (inverse of the cdf) at x</td>
</tr>
<tr>
<td><code>pois_pdf</code></td>
<td>computes the probability density function at x</td>
</tr>
<tr>
<td><code>pois_rnd</code></td>
<td>generate random draws from the poisson distribution</td>
</tr>
<tr>
<td><code>quantile</code></td>
<td>compute empirical quantile (percentile).</td>
</tr>
<tr>
<td><code>stdn_cdf</code></td>
<td>computes the standard normal cumulative</td>
</tr>
<tr>
<td><code>stdn_d</code></td>
<td>demo of standard normal distribution functions</td>
</tr>
<tr>
<td><code>stdn_inv</code></td>
<td>computes the quantile (inverse of the CDF)</td>
</tr>
<tr>
<td><code>stdn_pdf</code></td>
<td>computes the standard normal probability density</td>
</tr>
<tr>
<td><code>tdis_cdf</code></td>
<td>returns cdf at x of the t(n) distribution</td>
</tr>
<tr>
<td><code>tdis_d</code></td>
<td>demo of Student t-distribution functions</td>
</tr>
<tr>
<td><code>tdis_inv</code></td>
<td>returns the inverse (quantile) at x of the t(n) distribution</td>
</tr>
<tr>
<td><code>tdis_pdf</code></td>
<td>returns the pdf at x of the t(n) distribution</td>
</tr>
<tr>
<td><code>tdis_prb</code></td>
<td>calculates t-probabilities for elements in x-vector</td>
</tr>
<tr>
<td><code>tdis_rnd</code></td>
<td>returns random draws from the t(n) distribution</td>
</tr>
<tr>
<td><code>trunc_d</code></td>
<td>demo of truncated normal draws</td>
</tr>
<tr>
<td><code>unif_d</code></td>
<td>demo of uniform distribution functions</td>
</tr>
<tr>
<td><code>unif_rnd</code></td>
<td>returns a uniform random number between a,b</td>
</tr>
<tr>
<td><code>wish_d</code></td>
<td>demo of random draws from a Wishart distribution</td>
</tr>
<tr>
<td><code>wish_rnd</code></td>
<td>generate random wishart matrix</td>
</tr>
</tbody>
</table>
A.10 Utilities

This is a collection of functions that an economist might find useful.

Table A.10: Utilities in LeSage Econometrics Toolbox

<table>
<thead>
<tr>
<th>Programs and Demonstrations</th>
</tr>
</thead>
<tbody>
<tr>
<td>accumulate</td>
</tr>
<tr>
<td>blockdiag</td>
</tr>
<tr>
<td>cal</td>
</tr>
<tr>
<td>cal_d</td>
</tr>
<tr>
<td>ccorr1</td>
</tr>
<tr>
<td>ccorr2</td>
</tr>
<tr>
<td>cols</td>
</tr>
<tr>
<td>crlag</td>
</tr>
<tr>
<td>csumprodc</td>
</tr>
<tr>
<td>csumsumc</td>
</tr>
<tr>
<td>delif</td>
</tr>
<tr>
<td>diagrv</td>
</tr>
<tr>
<td>dmult</td>
</tr>
<tr>
<td>find_big</td>
</tr>
<tr>
<td>find_bigd</td>
</tr>
<tr>
<td>findnear</td>
</tr>
<tr>
<td>fturns</td>
</tr>
<tr>
<td>fturns_d</td>
</tr>
<tr>
<td>growthr</td>
</tr>
<tr>
<td>ical</td>
</tr>
<tr>
<td>ical_d</td>
</tr>
<tr>
<td>indexcat</td>
</tr>
<tr>
<td>indicator</td>
</tr>
<tr>
<td>invccorr</td>
</tr>
<tr>
<td>invpd</td>
</tr>
<tr>
<td>invpd_d</td>
</tr>
<tr>
<td>kernel_n</td>
</tr>
<tr>
<td>lag</td>
</tr>
<tr>
<td>levels</td>
</tr>
</tbody>
</table>

*continued on next page*
Utilities in LeSage Econometrics Toolbox continued

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lprint</td>
<td>print an (nobs x nvar) matrix in LaTeX table format</td>
</tr>
<tr>
<td>lprint_d</td>
<td>demo of lprint()</td>
</tr>
<tr>
<td>lprintf</td>
<td>Prints a matrix of data with a criteria-based symbol next</td>
</tr>
<tr>
<td>lprintf_d</td>
<td>demo of lprintf()</td>
</tr>
<tr>
<td>make_contents</td>
<td>makes pretty contents.m files for the Econometrics Toolbox</td>
</tr>
<tr>
<td>matadd</td>
<td>performs matrix addition even if matrices</td>
</tr>
<tr>
<td>matdiv</td>
<td>performs matrix division even if matrices</td>
</tr>
<tr>
<td>matmul</td>
<td>performs matrix multiplication even if matrices</td>
</tr>
<tr>
<td>matsub</td>
<td>performs matrix subtraction even if matrices</td>
</tr>
<tr>
<td>mlag</td>
<td>generates a matrix of n lags from a matrix (or vector)</td>
</tr>
<tr>
<td>mprint</td>
<td>print an (nobs x nvar) matrix in formatted form</td>
</tr>
<tr>
<td>mprint3</td>
<td>Pretty-prints a set of matrices together by stacking the</td>
</tr>
<tr>
<td>mprint3_d</td>
<td>An example of using mprint3</td>
</tr>
<tr>
<td>mprint_d</td>
<td>demo of mprintf()</td>
</tr>
<tr>
<td>mth2qtr</td>
<td>converts monthly time-series to quarterly averages</td>
</tr>
<tr>
<td>nclag</td>
<td>Generates a matrix of lags from a matrix containing</td>
</tr>
<tr>
<td>plt</td>
<td>Plots results structures returned by most functions</td>
</tr>
<tr>
<td>prodc</td>
<td>compute product of each column</td>
</tr>
<tr>
<td>prt</td>
<td>Prints results structures returned by most functions</td>
</tr>
<tr>
<td>recserar</td>
<td>computes a vector of autoregressive recursive series</td>
</tr>
<tr>
<td>recsercp</td>
<td>computes a recursive series involving products</td>
</tr>
<tr>
<td>roundoff</td>
<td>Rounds a number(vector) to a specified number of decimal places</td>
</tr>
<tr>
<td>rows</td>
<td>return rows in a matrix x</td>
</tr>
<tr>
<td>sacf</td>
<td>find sample autocorrelation coefficients</td>
</tr>
<tr>
<td>sacf_d</td>
<td>demo of sacf()</td>
</tr>
<tr>
<td>sdiff</td>
<td>generates a vector or matrix of lags</td>
</tr>
<tr>
<td>sdummy</td>
<td>creates a matrix of seasonal dummy variables</td>
</tr>
<tr>
<td>selif</td>
<td>select values of x for which cond is true</td>
</tr>
<tr>
<td>seqa</td>
<td>produce a sequence of values</td>
</tr>
<tr>
<td>seqm</td>
<td>produce a sequence of values</td>
</tr>
<tr>
<td>shist</td>
<td>spline-smoothed plot of a histogram</td>
</tr>
<tr>
<td>spacf</td>
<td>find sample partial autocorrelation coefficients</td>
</tr>
<tr>
<td>spacf_d</td>
<td>demo of spacf()</td>
</tr>
</tbody>
</table>

continued on next page
Utilities in LeSage Econometrics Toolbox *continued*

<table>
<thead>
<tr>
<th>program</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdc</td>
<td>standard deviation of each column</td>
</tr>
<tr>
<td>sumc</td>
<td>compute sum of each column</td>
</tr>
<tr>
<td>tally</td>
<td>calculate frequencies of distinct levels in x</td>
</tr>
<tr>
<td>tdiff</td>
<td>produce matrix differences</td>
</tr>
<tr>
<td>trimc</td>
<td>return a matrix (or vector) x stripped of the specified columns.</td>
</tr>
<tr>
<td>trimr</td>
<td>return a matrix (or vector) x stripped of the specified rows.</td>
</tr>
<tr>
<td>tsdate</td>
<td>produce a time-series date string for an observation</td>
</tr>
<tr>
<td>tsdate_d</td>
<td>demonstrate tsdate functions</td>
</tr>
<tr>
<td>tsprint</td>
<td>print time-series matrix or vector with dates and column labels</td>
</tr>
<tr>
<td>tsprint_d</td>
<td>Examples of using tsprint()</td>
</tr>
<tr>
<td>unsort</td>
<td>takes a sorted vector (or matrix) and sort index as input</td>
</tr>
<tr>
<td>unsort_d</td>
<td>demo of unsort()</td>
</tr>
<tr>
<td>util_d</td>
<td>demonstrate some of the utility functions</td>
</tr>
<tr>
<td>vec</td>
<td>creates a column vector by stacking columns of x</td>
</tr>
<tr>
<td>vech</td>
<td>creates a column vector by stacking columns of x</td>
</tr>
<tr>
<td>vecr</td>
<td>creates a column vector by stacking rows of x</td>
</tr>
<tr>
<td>vprob</td>
<td>returns ( \text{val} = (1/\sqrt{2\pi he}) \exp(-0.5<em>ev</em>ev/he) )</td>
</tr>
<tr>
<td>xdiagonal</td>
<td>spreads an nxk observation matrix x out on</td>
</tr>
<tr>
<td>yvector</td>
<td>repeats an nx1 vector y n times to form</td>
</tr>
</tbody>
</table>
APPENDIX B

Data Sets

Data files used in this book are embedded in the book. To extract a data file when you are reading the book using Adobe Acrobat, right-click on the (red) file name below and choose "Save Embedded File to Disk. . ." (or in older versions of Adobe Acrobat, "Extract File. . ."). You can also double-click on the file to open it immediately. If you're unable to access the attached file, or you observe miscellaneous strange behaviour, your pdf viewer might not be capable of handling file attachments properly. You should check the file for viruses an other mall-ware before using the file.

See Section 5 of (Pakin, 2011) for details of some pdf viewer problems that may occur.

1. g10xrate.xls. This file g10xrate.xls contains daily observations on the US Dollar exchange rates of the G10 countries.¹ The ten exchange rate series are in the ten columns. Each row then gives one observation of each series. There are 6237 observations of each exchange rate. The name of each series is given in the first row at the head of each series. This construction is typical of many such files.

¹There are 11 G10 countries - Belgium, Canada, France, Germany, Italy, Japan, the Netherlands, Sweden, Switzerland, the United Kingdom and the United States. When the eleventh country joined the original name was retained.
that are encountered in economics.

2. **NationalIncomeReal.csv.** This file Quarterly Irish National Income Data and is used in chapter 5. It contains two columns, year and NI (National Income at constant Prices) downloaded from [www.cso.ie](http://www.cso.ie) on 13 October 2014.

3. **tobacco.mat.** This dataset is in native MATLAB format. It is analysed in [Verbeek (2008, 2012)](http://example.com). The data set contains the following variables:

- **bluecol:** dummy, 1 if head is blue collar worker (1)
- **whitecol:** dummy, 1 if head is white collar worker (1)
- **flanders:** dummy, 1 if living in Flanders (2)
- **walloon:** dummy, 1 if living in Wallonie (2)
- **nkids:** number of children > 2 years old
- **nkids2:** number of children <= 2 years old
- **nadults:** number of adults in household
- **lnx:** log total expenditures
- **share2:** budgetshare tobacco
- **share1:** budgetshare alcohol
- **nadlnx** nadults × lnx
- **agelnx:** age × lnx
- **age:** age in brackets (0—4)
- **d1:** dummy, 1 if share1>0
- **d2:** dummy, 1 if share2>0
- **w1:** budgetshare alcohol ,if >0, missing otherwise
- **w2:** budgetshare tobacco ,if >0, missing otherwise
- **lnx2:** lnx squared
- **age2:** age squared


