

## Topic 0 – Getting started with Matlab and EViews

5<sup>th</sup> January 2007

### Trivia

I have been working as a research assistant for nine quarters, mainly working with a bunch of large datasets, and I am familiar with the softwares STATA, Matlab and EViews. I went to the University of Hong Kong for my bachelor degree, and then joined the UW econ program for a PhD in 2003. My current primary research interests are monetary economics and macro-econometrics, and I also work on labor economics.

### EViews Basics

1. EViews has three major features: 1) it is object-oriented. You work with objects that contain different types of information (e.g. a variable, or a regression); 2) you can either type in commands, or rely on the graphic user interface; 3) it is easy to use. The third feature is important.
2. Now start EViews, and you see the white rectangle on top. This is the *command window*, in which all the action happens. Also important is the *main menu*, which is the grey bar right above it.
3. You either have data to begin with, or you need to make up some. We will do both.
4. Go to my class website <http://students.washington.edu/byront/teaching.htm> and download the data in xls format. The file includes monthly US unemployment rate and civilian employment-population ratio, from 1948:01 to 2006:11. How to put this xls file into EViews?
5. First we create a *workfile*, which keeps your data and whatever you will do to them, by selecting **File/New/Workfile....** Choose *Dated-regular frequency* for the structure type, and *Monthly* for frequency. Type in the *Start date* 1948:01 and *End date* 2006:11. Click **OK**. Now the workfile pops up. The series object **resid** keeps the residual from your last regression, and the coefficient object **c** keeps the coefficients you get from the last regression.
6. Now import our data by selecting **File/Import/Read Text-Lotus-Excel...**, and find the xls file you have just downloaded. Choose B2 as the *Upper-left data cell* as we do not need the date in the xls file, and type Sheet1 in the sheet name box (you should

type in the exact sheet name). You have to tell EViews the name of your variables in the xls file (in the correct order, and separate the names with a space), so type *unrate emratio* in the *Names for series* box. Click **OK**. You just get yourself a workfile with data.

7. Now you can take a look at your data in various ways. Right click one of our series objects and choose **Open**. In the new window that pops up, click on **View** and you can see a whole bunch of things you can do with this series. For example, by selecting **View/Graph/Line** and you get a plot of your series, and by selecting **View/Descriptive Statistics/ Histogram and Stats** and you get the histogram and summary statistics of your series.
8. You can also play with more than one series. Click on the object *unrate*, hold on to the ctrl button, and click on the other object *emratio*. Right click and do **Open/as Group**. Now you can do some multivariate things with the two series. For example, the same **View/Graph/Line** procedure plots the two series in the same graph.
9. How to run a regression? Let's say we want to fit a model for the unemployment rate:  
unemployment rate = constant +  $\beta$  \*one-month lag of unemployment rate + error  
In words, we explain the employment rate by a constant and its lag. To do this, type in the command window: **ls unrate c unrate(-1)**, and press Enter. As you might guess, "ls" stands for least squares, and "c" stands for a constant. Unlike some packages, in EViews you always have to specify a constant. Putting (-x) after the variable gives you the x-lagged variable. Our command gives us the following result:

Dependent Variable: UNRATE				
Method: Least Squares				
Date: 01/03/07 Time: 11:51				
Sample (adjusted): 1948M02 2006M10				
Included observations: 705 after adjustments				
Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.061435	0.031116	1.974387	0.0487
UNRATE(-1)	0.989236	0.005351	184.8537	0.0000
R-squared	0.979842	Mean dependent var	5.616312	
Adjusted R-squared	0.979813	S.D. dependent var	1.508726	
S.E. of regression	0.214361	Akaike info criterion	-0.239474	
Sum squared resid	32.30341	Schwarz criterion	-0.226543	
Log likelihood	86.41467	F-statistic	34170.88	
Durbin-Watson stat	1.812698	Prob(F-statistic)	0.000000	

For now we do not need to know the meanings of these numbers. You will learn them in this course.

10. Sometimes you need to make variables up. Say you want to draw for each date a number from the standard normal distribution. Type in the command window **genr e = nrnd**, and press Enter, and you will get a new series of random numbers called “e”. We will use this **genr** command a lot.
11. What if you want to create a noisy unemployment rate by adding the true unemployment rate and the noise e? As you may have guessed, you need the command **genr unrate2 = unrate + e**. The noisy unemployment rate is called unrate2.
12. Now plot unrate and unrate2 together in the same graph and see how they look like. Remember, it is helpful to plot your data and “eyeball” them. Pictures inspire new thoughts.
13. Finally, when you are done or sick with your work, whichever comes earlier, you need to save your workfile by clicking **Save** in your workfile window.

## Matlab Basics

1. Now we are done with EViews, and we move on to something more difficult. As suggested by its name, Matlab is a programming language dealing with matrices. Since graduate-level econometrics is usually one damn matrix after another, we need Matlab.
2. Start Matlab, and you will probably see three windows. The *Workspace* at the top left shows you the data (created or imported) currently in use, the *Command History* at the bottom left shows what you have done to Matlab, and the *Command Window* on the right is where you let your creativity run wild.
3. Let's try with some easy things in the Command Window.

### I. Creating Vectors and Matrices

Here is how to create a **column vector**:

```
>> A=[1;2;3]
```

A =

```
1
2
3
```

To create a **row vector**, just omit the semicolons:

```
>> B=[1 2 3]
```

B =

```
1  2  3
```

To create a matrix you just need to put the row and column vectors together.

Using a 3 by 3 matrix as an example:

```
>> C=[1 2 3; 4 5 6 ; 7 8 9]
```

C =

```
1  2  3
4  5  6
7  8  9
```

When you want to obtain a particular element in a vector or a matrix, just tell Matlab where to look for it. To get the second element of a column vector:

```
>> A(2)
```

```
ans =
```

```
2
```

Similarly for a row vector:

```
>> B(2)
```

```
ans =
```

```
2
```

To get the second column of a matrix:

```
>> C(:,2)
```

```
ans =
```

```
2
```

```
5
```

```
8
```

Likewise, to get the second row of a matrix:

```
>> C(2,:)
```

```
ans =
```

```
4 5 6
```

As you may expect, to get the element at the second row and second column of a matrix:

```
>> C(2,2)
```

```
ans =
```

```
5
```

What if do not want part of the thing you have created? For example, if you want to delete the third column of our matrix:

```
>> C(:,3)=[]
```

```
C =
```

```
1 2
```

```
4 5
```

```
7 8
```

You do not need to create an identity matrix by the above commands. To create a 3 by 3 identity matrix, just do:

```
>> eye(3,3)
```

```
ans =
```

```
1 0 0
0 1 0
0 0 1
```

To create a 3 by 3 matrix of zeros (the same holds for a vector), you can do:

```
>> zeros(3,3)
```

```
ans =
```

```
0 0 0
0 0 0
0 0 0
```

Finally, to create a 3 by 3 matrix of ones (the same holds for a vector), you can do:

```
>> ones(3,3)
```

```
ans =
```

```
1 1 1
1 1 1
1 1 1
```

## II. Matrix Manipulations

Use the operator “+” to add two matrices up, “-“ to subtract one matrix from another, and “\*” to multiply two matrices. For example, to multiply a 3 by 3 matrix of ones by our matrix C above:

```
>> C*ones(3,3)
```

```
ans =
```

```
6 6 6
15 15 15
24 24 24
```

Of course the dimensions of matrices should agree for the multiplication to work.

What if we want to square each element of our matrix C? Here is how:

```
>> C.^2
```

```
ans =
```

```
1  4  9
16 25 36
49 64 81
```

Do not forget the “.”, or we will get C\*C:

```
>> C^2
```

```
ans =
```

```
30 36 42
66 81 96
102 126 150
```

To keep the diagonal elements of a matrix as a vector, just do:

```
>> diag(C)
```

```
ans =
```

```
1
5
9
```

Concepts like trace, determinant, inverse or eigenvalue are easy:

```
>> trace(C)
```

```
ans =
```

```
15
```

```
>> det(C)
```

```
ans =
```

```
0
```

```
>> inv(C)
```

**Warning: Matrix is close to singular or badly scaled.**

**Results may be inaccurate. RCOND = 1.541976e-018.**

```
ans =
```

```
1.0e+016 *
-0.4504  0.9007 -0.4504
0.9007 -1.8014  0.9007
```

**-0.4504 0.9007 -0.4504**

```
>> eig(C)
```

```
ans =
```

**16.1168**

**-1.1168**

**-0.0000**

What is that warning after taking the inverse? Since our matrix C is singular, Matlab warns us that the inverse calculated might be meaningless.

### III. For-Loop and While-Loop

Usually you want the computer to do something repeatedly while you can have a cup of coffee (decaf). It is easy to do in Matlab, and not so in EViews.

With the matrix C, we want to find the product of the i-th row and the (i,i) element, for i=1,2,3. Instead of doing it one by one, we can do a for-loop:

```
>> for i=1:3
```

```
  C(i,i)*C(:,i)
```

```
end
```

```
ans =
```

**1**

**4**

**7**

```
ans =
```

**10**

**25**

**40**

```
ans =
```

**27**

**54**

**81**

This for-loop will be handy when you need to say do the same thing for a 100 by 100 matrix. The while-loop works in a similar way. If we type:

```
>> i=1
```

```
i =
```

```
1
```

```
>> while i<3
```

```
C(i,i)*C(:,i)
```

```
i=i+1
```

```
end
```

```
ans =
```

```
1
```

```
4
```

```
7
```

```
i =
```

```
2
```

```
ans =
```

```
10
```

```
25
```

```
40
```

```
i =
```

```
3
```

We define a counter  $i$  and set it to be one, ask Matlab to do our operation  $C(i,i)*C(:,i)$ , and after that add 1 to  $i$ . The while-loop stops when  $i$  is no longer less than three. Make sure that you reset the counter after each loop so that the loop eventually stops!

#### IV. Miscellaneous

Two important tricks: First, to avoid Matlab showing every single thing you ask it to do in the command window, add a semicolon “;” after each command. For example, if you type:

```
>> C=[1 2 3; 4 5 6; 7 8 9];
```

Now the matrix C will be in the workplace, but it will not show up in the command window.

Second, Matlab will disregard anything after “%” in that line. For example:

```
>> %Hip Hip Hooray!
```

Matlab will not do anything, though it might find you silly. More on this “%” trick later.

4. Now you know the elementary commands in Matlab. One important reminder: when you are doing simple things with a few commands, typing them one by one in the Command Window is fine; when you need to more complicated stuff (which you will) which involves dozens of commands, you should first create an *m-file* by clicking the white little rectangle *New M-File* at top-left, and type your commands one by one in it. After you are done, click the *Run* button at top-middle and Matlab will execute your commands one by one. Keep your m-files for future reference (or for turning in as an assignment!), and add comments to your script with “%”.
5. Now delete whatever we have done by the **clear** command (so be careful with this command!). Now type create a vector of 1000 standard normal random numbers by typing **e = randn(1000,1);**. To plot this series, type **plot(e)**. Matlab can do much fancier things with graphics, which you can investigate in your spare time.
6. There is one more important topic about Matlab: creating a function m-file. I will talk about it a few weeks later.

#### Final Comments

1. There are more tricks and skills about using these two programs, and knowing them is one of the purposes of the upcoming challenging assignments.
2. Browse through the Help files in Matlab and EViews whenever you are confused or you simply want to learn more.