# Contents



**ES5512 Semester 2, 2005/2005**

# 1 Batch programming in EVIEWS

What does this mean? In EVIEWS you have encountered single line commands which make EVIEWS execute a particular command. For instance, if you type

```
ls y c x1 x1
```

into the command line, EVIEWS will run a regression of the dependent variable $y$ on a constant and the two explanatory variables $x1$ and $x2$. A batch program collects a number of commands and then executes them in one go.

Advantages of writing batch programs:

1. Repeating essentially the same analysis with another dataset is very easy
2. For some analysis there are some very repetetive things you got to do. In some cases, writing a batch program will eliminate the need to do essentially the same thing many times. Well, better, you instruct the software to do the repetitive work for you.
3. By writing a batch program you essentially also create a record of what you did. This is extremely useful when it comes to understanding previous work and to identify mistakes. This is possibly the most important advantage of batch programs.
4. You can use a number of features which are not obviously available to the ordinary EVIEWS user, such as loops and conditional branching (if statements).

If you did any programming in any other programming language, such as Visual Basic, C, C++, Matlab, GAUSS, etc. you will recognise a few common patterns. EVIEWS provides a very good help manual. Check out the Programming Reference in the EVIEWS help menu!

# 2 First program

Without any further ado we will go to our first little EVIEWS program. For simplicity we will assume that we already have an EVIEWS workfile in which we have saved our basic data. This file is here the file called IBM.wf1. It contains 11080 observations for the S$P500 share index and the IBM share price respectively.

We will start doing a few things you already know how to do in the interactive EVIEWS mode. But later you will surely realise the value of the batch program. In EVIEWS go to `FILE - NEW` and open a new program.

The first thing you should know is, that in that file you can write one of two things. First, commands which are understood by EVIEWS and comments which are preceded by a single `" ' "`. In fact, I should stress that it is rather important to include comments. Programs can get rather unwieldy and after returning from a week vacation, such comments will greatly facilitate your understanding of your own work, not to mention others who may want to understand what you did. So, let's start:

```
==============================
' This is our first program
' This command loads the EVIEWS workfile.wf1
' adjust the path to the location where you saved the EVIEWS file
load c:/teaching/es5512/ibm.wf1
==============================
```

This is your first program. Now press the `RUN` button and click `OK` in the next screen. EVIEWS will open the IBM.wf1 workfile. This is of course not very imaginative. before we continue, close the workfile again and return to your program. For the next go we will open the file, calculate the returns for the S&P500 share index and the IBM share price and then close the file under a new name IBM2.wf1.

```
==============================
' This command loads the EVIEWS workfile.wf1
' adjust the path to the location where you saved the EVIEWS file
load c:/teaching/es5512/ibm.wf1
' calculate the log returns
series sp500_ret = log(sp500) - log(sp500(-1))
series ibm_ret = log(ibm) - log(ibm(-1))
save c:/teaching/es5512/ibm2.wf1
==============================
```

Note, that instead of the command `series` we could have also used `genr`, here, however, we will stick to `series`. It is usually a good idea , as we have done here to leave your original datafile unchanged (here ibm.wf1) and save your results in a new file (here ibm2.wf1).

# 3 Some basics

So far this is all pretty easy. Before we continue we got to address a few rather basic issues to better understand how EVIEWS works. In general EVIEWS can do certain things to what it calls objects. When we uploaded ibm.wf1, it already contained 4 objects. Three data series (ibm, sp500 and resid; as you know, the latter is always included in an EVIEWS workfile) and an coefficient vector called c (whenever you run regressions, parameter estimates will be saved in here). Hence we already know two type of objects, series (`series`) and coefficient vectors (`coef`). Here are a few more objects which will come in handy (this list is not exhaustive:

- `coef`
- `series`
- `equation`
- `graph`

- `matrix`
- `model`
- `sample`
- `scalar`
- `vector`

In general one need to declare an object before it is used. For the series we created earlier, such a declaration, followed by the definition, would be

```
series sp500_ret
sp500_ret = log(sp500) - log(sp500(-1))
```

Conveniently, for series EVIEWS accepts the declaration and definition to appear in the same line

```
series sp500_ret = log(sp500) - log(sp500(-1))
```

However, you should be aware that essentially this line does two things.
There are a number of common things you may want to do with any object. You want to delete it, say

```
delete sp500_ret
```

or you may want to copy it

```
copy sp500_ret sp500_ret_new
```

or alternatively you want to rename an object

```
rename sp500_ret returns_sp500.
```

## 3.1 Error messages

As in any programming language even small typos will stop the program from running. Should there be a mistake in your code, EVIEWS will execute the program until it hits the program line that cannot be executed and will then generate an error message. To test how this works you could type `series` instead of `series` and attempt to execute the program. You will then find a little error message pop up which will tell you in which line you made a mistake. Very helpful. Thanks.

## 3.2 Running Regressions

From now on I will mostly only show you little parts of programs and only occasionally show an entire program code. But you should be able to easily build these little code extracts into a full program. Next we shall learn how to run a regression. Regressions are again objects and we start by means of a declaration, letting EVIEWS know that we plan to run a regression and we shall call it `eq1`.

```
equation eq1
```

Next we want to actually run a regression using the IBM returns as dependent and the S&P500 returns as the explanatory variable. Those of you who do Finance will recognise this as the Market Model.

```
eq1.ls ibm_ret c sp500_ret
```

If you run a program which includes this line you will find a little equation object labeled eq1 in your workfile. In case you want to delete an object, say this equation, this is easily achieved by means of the following command:

```
delete eq1
```

But let's not do that here. If you double click on the equation object you will find the usual regression output.

You will recall from your first encounters with EVIEWS that most of the interesting analysis is accessible when you click on the `VIEW` button in the equation window. There you can access such things like, plots of residuals and specification tests. All these things, in EVIEWS language, are called object views. Each object type will have different type of views available and you will have to refer to the programming manual to find a complete list. In this section we will mainly be concerned with the analysis of equations and hence the equation object type is of most interest here. But there will be views associated to series, coefficients, matrices etc.

There are way too many views such that this document cannot give an exhaustive list, not even a list of all useful views, but I will attempt to explain how to use them, such that in conjunction with the EVIEWS help manual you should be able to explore this wondrous new world.

Once you ran a regression you may be interested in, say, a Chow test for structural break. If you regression window is open you can access this via `VIEWS - STABILITY TESTS - CHOW BREAKPOINT TEST`. If you do this, you will then be asked to propose a date for the structural break. If you want to test for stability in the batch program you can do so by means of the following command:

```
eq1.chow 5000
```

This will test whether there was a structural break at observation 5000 and when you run your program you will find the equation window opened with the requested Chow test being displayed.

This command illustrates the basic architecture required to apply one of the EVIEWS views:

```
objectname.view(view_options) list_of_arguments
```

In this case the objectname was the name of our previously estimated equation `eq1` and the view requested was a `chow` test. We did not need any options for this Chow test. Some views have options, other have none; you will again have to refer to the programming manual. Many views need some input to be executed. For the Chow test we needed to indicate at what date we want to have stability tested, here at observation 5000. All calls of object views will have this structure.

Besides different views, objects also allow to call upon certain data members (this is again EVIEWS language, more intuitively we could call them equation outputs). For the case of a regression in form of equation `eq1`, this can be the residual sum of squares, t-statistics, Durbin-Watson test statistics etc. In short these are a number of values which one might be interested in. Again I will only illustrate the use of these data members by means of one example. Let's retrieve the adjusted $R^2$ and save it in a scalar called `r2bar`.

```
scalar r2bar
r2bar = eq1.@rbar2
```

The data members are called by means of referring to the objectname (here `eq1`) and then calling the required information by means of `@datamembername`. Another very useful datamember of the equation object group is the coefficient estimate, `eq1.@coefs(i)`. The value `i` indicates which coefficient you are interested in (1 = the first, 2 = the second etc.). Equally we can access the t-statistic for the individual parameters via `eq1.@tstats(i)` and the standard errors for the parameter estimates from `eq1.@stderrs(i)`. The programming manual provides a list of the available data members.

It is often required to estimate a regression for only a subsample of your data set. You may recall that in the bottom left corner of a regression window you can change the sample for which you want to estimate a regression. Of course, you can also achieve this easily in the batch programming mode. Say, you want to calculate the above regression only for the first 1,000 observations. You should then enter, preceding the regression command, the following line

```
smpl 1 1000
```

This achieves the required sample restriction and if you then run the regression using the above command and check your results you will find the sample suitably restricted. You should, however be careful and

recognise that, after using this command, not only the regression will be restricted to this sample, but virtually everything you do with the data series (e.g. graphs). This may be your intention, but it is useful to know that the command

```
smpl @all
```

reverts the sample to all available data. In fact there is an object class sample in EVIEWS, but I will not discuss this further here.

Lastly it should be mentioned that OLS (`.ls`) is not the only way to estimate an equation and EVIEWS is able to perform a number of alternative estimation procedures (in EVIEWS language they are called different methods), such as Generalised Method of Moments (`.gmm`) or two stage least squares (`.tsls`). For the time being, however, we will stick to OLS estimation .

## 3.3 Output into a file

I hope that in the meantime you can imagine that writing batch programs can be useful. You might have to apply a number of different estimations and tests to a certain dataset. If you only have to do them once you might as well do them in the interactive mode. If all these things, however, have to be repeated, or you realise later that you had a typo in your datafile, then you will be extremely grateful that you wrote all these commands down ion a batch program and everything will be redone on the push of one button (`RUN`).

To make full use of this working mode, however, it would be useful to have all your results printed to a text document. In this section I will illustrate how to achieve this. Somewhere at the start of your program, before you perform the first piece of analysis you want to have printed, you should include the following line

```
output(t) c:/teaching/es5512/ibmout.txt
```

This line will tell EVIEWS that whenever you tell the software to print something that rather than sending the output to your default printer it should send the output to a file, here called `ibmout.txt`. If you leave this line out, all requested outputs will go straight to your printer.

Next you should realise that for most commands there are two ways to have the result printed. Let's consider the Chow test. Any of the two following lines will achieve exactly the same

```
print eq1.chow 5000
```

```
eq1.chow(p) 5000
```

You can either explicitly use the print command (first line) or you can make use of an option built into the chow view. The option `(p)` will for most commands indicate to EVIEWS that you want the output printed. Printing the regression output itself can be achieved in the same manner by one of the following two lines:.

```
print eq1.ls ibm_ret c sp500_ret
eq1.ls(p) ibm_ret c sp500_ret
```

At the end of your batch program you should include the line .

```
output off
```

as otherwise EVIEWS will continue to send output to this file the next time you use EVIEWS.
Before we continue I shall show a complete code which incorporates a number of the things we've done so far. See whether you can predict what this program does (adjust the path names to your computer). There is also a mistake in this code. See whether you can spot and fix it.

```
==============================
' load the data
load c:\teaching\ES5512\eVIEWS\ibm
output(t) c:\teaching\ES5512\eVIEWS\ibmtest.txt'
' calculate the log returns
```

```
series sp500_ret = log(sp500) - log(sp500(-1))
series ibm_ret = log(ibm) - log(ibm(-1))
ibm_ret.correl(24,p)
equation eq1
eq1.ls(p) ibm_ret c sp500_ret
print eq1.correl(10)
ssr = eq1.@ssr
save c:/teaching/es5512/eVIEWS/ibm2.wf1
output off
==============================
```

This way of saving output relates to printed output graphs will still be printed to the default printer.

### 3.4 Creating Graphs

If you want to create a graph of a number of series it is convenient to first define a group of series which comprises all these series (a group is another object in EVIEWS language and has a number of very useful views associated with it).

```
group gp1 sp500_ret ibm_ret
```

Then we can apply the command to create, say a line graph.

```
graph gr1.line gp1
```

The first part of the command declares a graph called `gr1` and then we indicate that we want to apply a line graph to all series in group `gp1`. After this command you will find a graph object in your workfile. A graph object can be manipulated by means of a number of graph procedures You can manipulate the scale, the legends and so forth. Importantly, you can save the graph in a file which you can later upload to a word document. The command to achieve this is.

```
gr1.metafile c:\teaching\es5512\eVIEWS\ibmgraph
```

This will create a file ibmgraph.wmf in the indicated directory.

## 4 Vectors and Matrices

You are familiar with vectors and matrices and you know that some special calculation rules apply to them. In the context of EVIEWS programming they can be very useful for a number of purposes but most likely to safe results of say a rolling regression. How to make efficient use of vectors and matrices will be shown in the next Section exploring loops.

As usual you got to declare a matrix or vector before you use it. The command is as follows:

```
matrix(4,2) mata
```

This declares a matrix called `mata` with 4 rows and 2 columns. At this stage all elements of `mata` will be 0. You can now do things to individual elements in that matrix. Let's assume you want to ensure that the element in the 3rs row and the 2nd column takes the value$\sqrt{2}$. You can address this element by mata(3,2) and assign the required value as follows:

```
mata(3,2) = @sqrt(2)
```

where `@sqrt(x)` is the command to calculate the square root of `x`. If you knew what values you wanted all the 8 element in `mata` to take you could either repeat such a command 8 times or use the more economical `fill` command:

```
mata.fill 1, 0.7, 3, 4, 1, 8, @sqrt(2), 0.3
```

which will create the following matrix:

$$\texttt{mata} = \begin{pmatrix} 1 & 1 \\ 0.7 & 8 \\ 3 & \sqrt{2} \\ 4 & 0.3 \end{pmatrix}.$$

Clearly, the `fill` command fills the matrix columnwise, i.e. only once the first column is completely filled values will be assigned to cells in the 2nd column.

As you recall, vectors are special cases of matrices where either the number of rows or of columns is equal to one. EVIEWS has a special object type for row vectors (only one row), called `rowvector` and another object for columnvectors (only one column) called `vector`. As they are only special cases of matrices you can do almost everything you wish to do by means of matrices.

A matrix can be either printed into the output file using the command

```
print mata
```

or alternatively you could display the columns of a matrix as a line graph using

```
mata.line
```

This will print a line for each of the two columns of the matrix. If you only want to show the graph for one of the two columns you first ought to extract the particular column in question and then use the line command in the same manner as above.

```
vector mata2 = @columnextract(mata,2)
mata2.line
```

The first line declares a new vector, called `mata2`, and defines it as the second column of matrix `mata`. The second line then works just as above.

The next section will illuminate the value of matrices.

# 5 Loops

This is one of the most powerful tools available to you only when you write batch programming. Just imagine you want to do the following. In our example data set we have 11,080 data stretching across more than 40 years. When running the Market Model regression

$$ret\_ibm_t = \alpha + \beta\ ret\_sp500_t + \varepsilon_t$$

as we did above, the parameter estimate for $\beta$, $\widehat{\beta}$, delivers an estimate for the systematic (market) risk of the IBM share. There is a sizeable literature which argues that this systematic risk of a share might very well be varying through time. We hence might be interested in estimates for $\beta$ at different points in time. It might be argued that at any point in time only the past 500 observations (app. equivalent to two calender years) should be used. We will write some code which will achieve this.

Let us briefly reflect on how many regressions we should expect to run. We started with 11,080 observations for the IBM share price. After calculating the returns we were left with 11,079 return observations (in EVIEWS these are observations 2 to 11,080). If we need to use the last 500 observations for one estimation at the time then we can run the first estimation of $\beta$ for observation 501 (using observations 2 to 501), the second for observation 502 (using observations 3 to 502) and the last for observation 11,080 (using observations 11,581 to 11,080). That leaves us with 10,580 regressions, quite clearly too many to achieve manually.

A loop will do the repetitive work for us. To find out what a loop does and how it works, however we will shelf the rolling regression problem for later. Let us first define a matrix with 100 rows and 2 columns. Then we will run through a loop and fill every cell in the first column of this matrix with a counter, e.g. 1

in the first row, 2 in the second etc. and every cell in the second column with a random number drawn from a standard normal distribution.

```
matrix test(100,1)
series rand = nrnd
for !i = 1 to 100
 test(!i,1)= !i
 test(!i,2) = rand(!i)
next
```

This code requires some explanation. As said before, we first declare the matrix `test` which is to be filled. We then define a new series which contains random numbers ($\sim N(0,1)$) in each observation. The command `nrnd` is the EVIEWS command which generates such random numbers. Then we start the loop. Every loop has the following structure

```
for !counter = start to end
 ' some commands which are to repeated
 ' these commands usually involve the
 ' counter variable !counter
next
```

Here we called the counter variable `!i`. If you do not use an exclamation mark as the first letter for the counter variable you will have to declare the scalar counter variable beforehand. Once the commands inside the loop are executed the counter variable will click up by one and the commands inside the loop will be executed again. The last execution inside the loop will occur for `!counter = end`. There may be occasions where you do not want the counter to count up by one unit at a time but rather by some other increment (e.g. 2 or -1). to achieve that your first line of the loop may look like.

```
for !i = 1 to 100 step 2
```

Here only every second line will be filled.

It should now already be pretty obvious how, in principle, this loop structure can be used to obtain the rolling regression estimates of $\beta$, discussed earlier. Before I show how to do this, you should try yourself to write down the structure of this code. The code will have to include a matrix or vector declaration to store the results, changing sample definition, running regressions and the extraction of the parameter estimate.

Here is now the complete code to estimate the rolling regression and then to print a line graph of the series of $\widehat{\beta}$s. You will also see that this code defines the sample size, here 500, as a variable, which will enable you to easily change this sample size and evaluate whether changing it has an important impact on the results.

```
==============================
' load the data
load c:\teaching\ES5512\eVIEWS\ibm
output(t) c:\teaching\ES5512\eVIEWS\ibmtest.txt

' calculate the log returns
series sp500_ret = log(sp500) - log(sp500(-1))
series ibm_ret = log(ibm) - log(ibm(-1))

scalar nobs = 11080 ' total number of observations
scalar nw = 500 ' estimation window size
matrix(11080,1) results ' save the results in here

' start the loop at nw+1 as we are
' loosing one observation
' due to the return calculation

for !i = nw+1 to nobs
 smpl !i-nw+1 !i            ' this leaves a sample size of nw
 equation eq1
 eq1.ls ibm_ret c sp500_ret
 smpl @all
 results(!i,1) = eq1.@coefs(2)    ' 2nd parameter is beta
next

results.line

save c:/teaching/es5512/eVIEWS/ibm2.wf1

output off
==============================
```

Just sit back for a minute and think about how long this would have taken you to do manually!!! There might be occasions where inside the loop you might want to have another loop. That is not a problem at all, but you got to be careful to use a different counter variable and ensure you save the results correctly.

# 6 Conditional branching

This last section in this introduction to EVIEWS batch programming will introduce you to another element of programming which is not as easily available if you were to do EVIEWS work interactively. It is the use of `if` commands. These are useful in situations where the course of action depends on some previous result. We will use this here for the following little exercise. We will declare another matrix the same size as matrix results and fill it with a one whenever the beta estimate is significantly larger than 1 and 0 otherwise.

An `if` command has the following general structure

```
if conditionstatement then
    ' some commands only to be executed
    ' if conditionstatement is true
else
    ' some commands only to be executed
    ' if conditionstatement is not true
```

```
endif
```

At the core of such an `if` statement is the `conditionstatement`. It is a statement which is either true or not true. A few examples are below

```
4 = 4    ' true
4 < 7    ' true
5 <= 3    ' false
4 <> 4    ' <> means unequal and hence this statement is false
```

such statements can also involve variables which are previously defined and one can even compare string variables (which have not been mentioned here)Outlook

Of course this cannot be a complete course in EVIEWS programming. This lesson had two goals. First it was to introduce you to the basic batch programming structures in EVIEWS. Second, it is also intended that you start to realise that you can achieve things which on first sight looked too complex. Depending on whether the `conditionstatement` is true or false the computer program will execute the code in the `then` branch or in the `else` branch respectively.

To achieve the purpose of our exercise, the `conditionstatement` has to check whether the null hypothesis $H_0 : \beta \leq 1$ can be rejected at the, say, 5% significance level. Hence before we continue we continue with the `if` command we got to ensure we know how to formulate the `conditionstatement`. It was discussed earlier that after running a regression of the type

```
equation eq1
eq1.ls ibm_ret c sp500_ret
```

We can access the t-statistic ($\left(\left(\widehat{\beta} - 1\right) / s_{\widehat{\beta}}\right)$ by means of the following command:

```
scalar ts = (eq1.@stderrs(2)-1)/eq1.@stderrs(2)
```

Hence, all we need to do is to compare ts against the appropriate critical value 1.645 (one sided critical value from the standard normal distribution). Therefore the complete loop, should look like

```
matrix(11080,1) results ' save the results in here
matrix(11080,1) results2 ' save the test indicator here
scalar ts

' start the loop at nw+1 as we are
' loosing one observation
' due to the return calculation

for !i = nw+1 to nobs
 smpl !i-nw+1 !i            ' this leaves a sample size of nw equation eq1
 eq1.ls ibm_ret c sp500_ret
 smpl @all
 results(!i,1) = eq1.@coefs(2)    ' 2nd parameter is beta
 ts = (eq1.@stderrs(2)-1)/eq1.@stderrs(2)
 if ts > 1.645 then
    results2(!i,1) = 1
 else
    results2(!i,1) = 0
 endif
next
```

This achieves the required task. Admittedly, it is not the most elegant way of doing things, but it serves the purpose. As for loops, you can nest one `if` statement in another, meaning that in the `then` command block you might have another `if` command.

# 7 Outlook

This document has two aims. First, it aims to introduce you to the basics of EVIEWS batch programming. More importantly, however, it aims to liberate you in terms of the things you think you can do. You should realise that you can achieve things with EVIEWS which on first sight perhaps appeared too complex.