ECONOMICS 452

TIME SERIES WITH STATA

0.1 Introduction

This manual is intended for the first half of the Economics 452 course and introduces some of the time series capabilities in Stata 8. I will be writing programs and fixing others throughout the term so this is really just a manual to get started. Other details will follow.

Stata has an excellent online help facility and there are manuals available. The emphasis in this outline is on time series in Stata (note: earlier versions of Stata did not do time series analysis). If you have never used Stata before, it would be a good idea to get the tutorial manual for Economics 351 which goes over the basics for using Stata (http://www.econ.queensu.ca/faculty/abbott/econ351/)

Keep in mind that the principal difference between cross-section and time series data is that order matters in the latter. We cannot shuffle the data as we did, say in a cross-section, using perhaps a *sort* command. The calendar time gives us the order of the variables and it makes no sense to mix this up. We often wish to create leads or lags of certain variables and Stata needs to know what variable in our set it should identify with the time (much more on this below). Once the time variable is identified, Stata can construct leads or lags in a logical manner.

There are many sources for time series data (for example you probably have downloaded some from the CANSIM databank in one of your courses).

0.2 Starting Stata

Double click on the Stata icon. The first screen you will see contains four Stata windows:

The **Stata Command window**, in which you type all Stata commands.

The **Stata Results window**, which displays the results of each Stata command as you enter it.

The **Review window**, which displays the past commands you have issued during the current Stata session.

The **Variables window**, which lists the variables in the currently-loaded data file.

- There are a number of programs (called ado files) that I have written for Stata for the Econ 452 (time series) course. To ensure you have the latest of these type: *Qprofile* and then check Econ452 dialog box that appears. This will

  - automatically download the most recent ado files (creating directories where they do not exist) onto the hardrive and make them accesible to you
  - put you in the appropriate directory for using my demo programs (referred to in the chapters and in theis manual)

- To get out of the QED session, you will need to type: *BREAK* (all capitals)

0.3 Log File

- As you work in Stata, be sure to make a record of your session by creating a 'log' file
- In a Stata do file (explained later) type: *log using a:\logfile,replace text* (the replace writes over any existing file so if you do not want that change the name and the text command writes the file in an ordinary and readable text file)
- Alternatively, open the log file by clicking on the Log button which is the fourth from the top left corner (chose .log option and not .smcl) . Name the file and change directory to *a:* .(if using a diskette)
- You should always open a log file **before** you type any commands in Stata, and close it at the end of your session.. Always save files to your diskette, usb datakey or the temporary directory on *c:\temp* which you can e-mail to yourself at the end of the session.
- Files left on public sites tend to disappear.
- At the end of your session type : *log close* or click on the same button to close the log file.

0.4 Help in Stata

There are three useful ways to get online help.

Suppose you want to use the generate command but forget its syntax.

1. You can type: *help generate*
2. Another neat way to find out stuff is to use the search command. For instance I might want to know about all the different appearances of "*generate*", so we could type: *search generate.* This will generate all kinds of information that you could follow up on.
3. In Stata 9, there are drop down dialogue boxes for all the commands.

0.5 Reading in Data

There are several types of data sets that Stata can use:

1. Stata data sets usually identified with a .dta suffix
2. ASCI data sets or flat files (this is what most of the CANSIM and *Journal of Applied Econometric* data sets are in) and usually has an extension .raw
3. Others kinds (like Excel spreadsheets) can be read in but typically it is easier to paste data in this format directly into the Stata editor and save!

Eventually, you will want to convert any ASCI datafile to a Stata datafile, since this can be read in faster to Stata than flat files and will save any labels or other information that we have created. We will show how to do this later in the tutorial. One other advantage of a Stata datafile is the you can simply click on the datafile and it will open directly into Stata.

Suppose you have just downloaded a Journal of Applied Econometrics dataset and want to read it into Stata. Since it is not a Stata datafile you will have to let Stata know how to read it. To keep things simple, let us assume that you have 200 observations on a variable temperature (called *temp*) stored on your diskette with the name: **bntemp.raw.** This is an ASCI datafile.

To load the variable *temp* into Stata, we use the **infile** command.

```
. infile temp using a:\bntemp
```

Stata will reply

```
(200 observations read)
```

Notice that we left off the extension .raw since this is assumed by Stata. If instead our file was called **bntemp.txt,** we would write

```
. infile temp using a:\bntemp.txt
```

Now that we have loaded the data into Stata we might just start to analyze the data, create new variables and so on. Imagine that we have done this and created several new variables and we wish to save the entire dataset as a Stata datafile. This is easily done by typing

```
. save _all using a:\bntemp
```

This will create a Stata datafile (called bntemp.dta) on our *a* directory (we use lower case letters in this manual, since all commands in Stata must be lower case). Notice we left out the .dta extension (it is the default extension).

Whenever we want to use this dataset in the future, we can simply click on it with our mouse and it will open in Stata. If on another occasion we were to create new variables or add data, we can save the new dataset by writing over the old Stata dataset (we could create a new dataset by using another name too) by typing

```
save _all using a:\bntemp,replace
```

Now part of the first project involves calculating descriptive statistics of your data set. This can be accomplished by using the **summarize** command. If you do not specify a particular variable, Stata will summarize the entire dataset.

Again we will assume that we have only one variable called *temp*.

```
. summarize

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+--------------------------------------------------------
    temp |     200      33.161   9.274295       13.8       56.5
```

If you want more descriptive statistics (percentiles, kurtosis, skewness and so on) type

```
. summarize temp, detail
```

To have your data listed type **list**

```
. list
         temp
  1.     22.9
  2.     13.8
  3.     31.4
   .
   .
   .
198.     44.8
199.     46.7
200.     49.4
```

0.6 Creating a time variable

Often when we download data, there will be a variable that identifies the time frequency of the variable. Typically this variable would not be in the appropriate

time format for Stata and will be of no use. Instead, we must construct the time variable. To do this, we need to know what the date is for the first observation and the frequency of the data (daily, weekly, quarterly or annual).

Suppose we know that the first observation is for the first quarter of 1990, the second for the second quarter, and so on, one should type:

generate time = q(1990q1) + _n-1
format time %tq
tsset time

The command **tsset time,** tells Stata that the variable *time* is to be identified as the variable giving the calendar time, all leads and lags are then based on the ordering from this variable. For all of the time series commands, you will need to declare a time variable through the command **tsset.**

For yearly data starting at 1842 type:

generate time = y(1842) + _n-1
format time %ty
tsset time

For half yearly data starting at 1921h2 type:

generate time = h(1921h2) + _n-1
format time %th
tsset time

For monthly data starting at 2004m7 type:

generate time = m(2004m7) + _n-1
format time %tm
tsset time

For weekly data starting at 1994w1 type:

generate time = m(1994w1) + _n-1
format time %tw
tsset time

For daily data starting at 1jan1999 type:

```
.  generate time = d(1jan1999) + _n-1
. format time %td
. tsset time
  time variable:  time, 01jan1999 to 19jul1999
. list
         temp       time

  1.      22.9  01jan1999
  2.      13.8  02jan1999
  3.      31.4  03jan1999
   .
```

```
   .
   .
198.      44.8  17jul1999
199.      46.7  18jul1999
200.      49.4  19jul1999
```

0.7 Differencing, Lags and Leads

Once you have specified a time variable using the **ttset** command, there are various Stata commands available. Useful operators include the Differencing, Lags and Leads operators.

- L.temp=temp_t-1
- L2.temp=temp_t-2
- F.temp=temp_t+1
- D.temp = temp_t - temp_t-1
- D2.temp = temp_t - temp_t-1 - (temp_t-1 - temp_t-2) = temp_t - 2temp_t-1 + temp_t-2
- S.temp = temp_t - temp_t-1
- S2.temp = temp_t - temp_t-2
- S12.temp = temp_t - temp_t-12
- $ld.temp = \Delta temp_{t-1}$

0.8 Graphing

With time series data, it is always a good idea to graph your data to get a feel for the data and how the data fluctuated over time. For instance, does the data cycle (interest rate data) or does it simply trend (like real GNP or population). Most time series papers that you will encounter have graphs. **ALl graph commands have been changed in Stata 8.** (To use the old commands in 7 just put gr7 and proceed with the commands)

To graph the time series we shall use the **graph** command. To graph 1-month Treasury Bill Rate against time with the title "1 Month Treasury Bill Rates" and saving this to the file

.twoway line r1 time, title("1 Month Treasury Bill Rates") legend(label(1 "1-month TB Rate")
graph save int.gph

Suppose we use short rates as predictors of long rates. we could graph the actuals vs. the predicted with the following commands (notice the semi colon for end of line)

regress r11 r1;
predict fit;
twoway (line r11 time) (line fit time), title("Short-Rate as Predictor of Long")
legend(label(1 "10-year TB Rate") label(2 "Predictions"));

To print a graph:

1. For a graph in memory just type: graph print
2. Display the graph. Either
   (a) Draw it using the **graph** command, or
   (b) Redisplay your last graph by pressing the **Graph** button, or
   (c) Retrieve a previously saved graph by typing **graph using dtemp**
3. Pull down **File** and choose **Print Graph**

After you have graphed something you can save it as a Stata graph file or as a bitmap file, which can then be put into most word processors (Word, WordPerfect and so on). You can also pull down **Prefs** and choose **Graph Preferences** . This will allow you to set options specifying how the graph is printed.

0.9 Correlogram and Partial Autocorrelation

The **corrgram** command lists a table of autocorrelations, partial autocorrelations, and Q statistics. It will also list a character-based plot of the autocorrelations and partial autocorrelations.

```
. corrgram  D.temp, lags(20)
```

```
LAG       AC       PAC      Q     Prob>Q
-----------------------------------------
1       -0.1178  -0.1180   2.8038  0.0940
2       -0.1706  -0.1831   8.7163  0.0128
3       -0.0497  -0.0939   9.2196  0.0265
```

```
4       -0.0795  -0.1429   10.515  0.0326
5       -0.0258  -0.0910   10.652  0.0587
6       -0.0375  -0.1081   10.944  0.0901
7       -0.0161  -0.0929   10.998  0.1387
8       -0.0037  -0.0835   11.001  0.2017
9        0.0477  -0.0092    11.48  0.2442
10       0.0310  -0.0128   11.684  0.3067
11       0.0229   0.0112   11.796  0.3792
12       0.0219   0.0327   11.898  0.4539
13      -0.0049   0.0201   11.903  0.5356
14      -0.0392  -0.0124   12.235  0.5874
15      -0.1196  -0.1259   15.344  0.4269
16      -0.0301  -0.0925   15.542  0.4853
17       0.0473  -0.0309   16.034  0.5214
18       0.0445  -0.0087   16.471  0.5597
19       0.0657   0.0368   17.429  0.5608
20      -0.0660  -0.0742   18.402  0.5609
```

The **ac** command produces a correlogram (the autocorrelations) with pointwise confidence intervals obtained from the Q statistic.

```
. ac D.temp, lags(20)
```

We can emphasize the discrete nature of the autocorrelations using the **needle** option.

```
. ac D.temp, lags(20) needle
```

The **pac** command produces a graph of the partial correlogram (the partial autocorrelations) with confidence intervals calculated using a standard error of $\frac{1}{\sqrt{n}}$. The residual variances for each lag are also included on the graph.

```
. pac D.temp, lags(20) needle
```

The command *wntesq* calculates the portmanteau Q test for randomness (see Chapter 6).

```
wntesq D.temp, lags(20)
```

0.10 Arima Estimation

The basic syntax for an ARIMA(p,d,q) model is:

arima **depvar**, **ar**(**1**/#$_{\mathbf{p}}$) **ma**(**1**/#$_{\mathbf{q}}$) **bfgs**

or

arima **depvar**, **arima**(#$_{\mathbf{p}}$, #$_{\mathbf{d}}$, #$_{\mathbf{q}}$) **bfgs**

For the first command if you want to have differenced data (say $\triangle y$) you would have to enter it via the $D.y$. To run a pure $AR(p)$ or $MA(q)$, you need to use the commands:

arima depvar, ar(1/#$_p$) bfgs
arima depvar, ma(1/#$_\mathbf{q}$) bfgs
or
arima depvar, arima(p,d,0) bfgs
arima depvar, arima(0,d,q) bfgs

The **ar**$(\mathbf{1}/\#_p)$ says to use lags 1 through p. If we just wrote **ar**$(\#_p)$, Stata would estimate

$$y_t = \alpha + \phi_p y_{t-p} + \varepsilon_t$$

omiting the lags $1 \ldots p-1$.

I have also included the option for *bfgs optimizer* (see Chapter 4), since I have found it to work the best. The switching algorithm (which is the default) seems to get stuck and often produces rubbish. Estimation can take quite a while, so be judicious in the number of these you consider at one time.

0.10.1 A word on the Intercept in $STATA$

- Stata scales the intercept (this allows a wider class of $ARMA$ models the so-called $ARMAX$ models that include other regressors).
- For example, in an $ARMA(p,0,q)$ model

$$y_t = \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q}$$

Stata would estimate the $\phi's$ and $\theta's$ as above but the constant estimate would be scaled as

$$\frac{\alpha}{(1-\phi_1-\ldots-\phi_p)}$$

By the way, this scaled estimate is also a consistent estimate of the unconditional mean (why ?).

To estimate an arima(0,1,1) model (with a constant) on our temperature data we type:

```
. arima  D.temp, ma(1)bfgs

ARIMA regression
```

The estimates would follow and for brevity we omit them.

To estimate an arima(0,1,2) model on our temperature data we type:

```
. arima  D.temp, ma(1/2)
```

We again omit the results except to note the value of the loglikelihood:

```
Log likelihood = -569.2156          Prob > chi2         =    0.0000
```

0.11 ARIMA: Hypothesis Testing

Sometimes we want to test one model against another: a restricted model against an unrestricted model (see Chapter 5). For example, the unrestricted model may be an $ARMA(0,1,2)$ model and the restricted model may be an $ARMA(0,1,1)$. Strictly speaking we are testing an hypothesis about the first difference of temperature ($\triangle temp_t = temp_t - temp_{t-1}$), since the level of the variable is nonstationary.

We may formally state this hypothesis test as

$$\begin{aligned} H_o &: ARMA(0,1,1) \\ H_1 &: ARMA(0,1,2) \end{aligned}$$

First we estimate the unrestricted model, and denote the maximized value of the log likelihood as $logL(\Omega)$. Next we estimate the restricted model and obtain the restricted log likelihood as $logL(\omega)$. Then asymptotically

$$-2(logL(\omega) - logL(\Omega)) \sim \chi^2_m$$

under the null hypothesis that the restricted model is correct. m is the number of restrictions. In our example $m = 1$ as we have the single restriction that the coefficient on the $MA(2)$ term is zero. The values for the log likelihoods can be found in the output of the arima estimation.

Let the value of the statistic be LR (using the above likelihoods). Then to calculate the p-value type:

```
display chiprob(m,LR)
```

In our example we would type:

```
. display chiprob(1,(-2*(-576.1117 - -569.2156)))
```

The result would be

```
.00020418
```

Thus we would reject (for any reasonable level of significance), the restricted model of $ARMA(0,1,1)$ for our differenced data, in favour of the $ARMA(0,1,2)$ model. Of course, a quick glance at the $t - statistic$ (a Wald test-see Chapter 6) $ARMA(0,1,2)$ on the second MA term clearly rejects the null hypothesis that this coefficient is 0.

0.12 ARIMA: Prediction

Generating predictions for a time series is often very important. To do this in Stata, one must first *extend the time variable* into the forecast region.

0.12.1 Use *fore.ado* (see Chapter 6)

In the lecture notes (see Chapter 6: Programing Tips there is a discussion on using *fore.ado*) which automaticllay extends the interval

0.12.2 Using the Editor to Extend the Sample

Another way to do this is to edit your time variable using the Stata **editor** (located right by the **browse** button).

Go down to the bottom of your time variable and note at the top what value STATA is using for your date (although your dates show up as dates to you, keep in mind Stata assigns its own consecutive number for each date). You can then add as many dates (using the consecutive numbers that Stata assigns) as you want for forecasting. Check to see as you type in the consecutive numbers that they are correctly translated into consecutive dates. As you add new dates, all of the variables in your dataset will get a '**.**', which is a missing observation for Stata.

In our case we will forecast 7 days so we modify the file to look like this at the end.

```
44.8
46.7
49.4
 .
 .
 .
 .
 .
 .
 .
```

When you quit the editor, Stata will ask you whether to preserve the changes: answer yes (answering no will leave you with the old data set). Now that you have changed the data set, you will want to save it (either use a new name or use the replace command). For instance, we could save this file under another name, say bntempP.raw.

Now we re-estimate our chosen model but specify the dates that we want to use (since we can't use observations that are blank)

```
. arima temp, arima(0,1,2)bfgs, if tin( ,19jul1999)
```

Notice the handy interval command *tin(d1,d2)* where *d1* gives the start date and *d2* gives the end date. If one of the fields is left blank (as the above *d1*) then the first observation is presumed).

Now we are ready to use the **predict** command. However, firstly we need to know the date of the first future date we want to forecast for. If we type **browse** we can scroll down our time series to find this date. In our example it is 20jul1999.

The basic syntax for **predict** is:

predict **newvarname**, **y dynamic**(**timeconstant**)

where *newvarname* is the name of the new variable we are creating that holds the predicted values of temperature. *y* is to specify that we want the predicted values in levels rather than differences. *dynamic* is to specify that we will be generating forecasts from after the date *timeconstant* using past forecasts. The date *timeconstant* is always the first date into the future. i.e. 20jul1999 in our example. The command *predict* will use the estimates of the last estimated model, which is the $ARIMA(0,1,2)$. Thus we type:

```
. predict ptemp , y dynamic(d(20jul1999))
(8 missing values generated)
(1 missing value generated)
. list
          temp       time      ptemp
  1.      22.9  01jan1999          .
  2.      13.8  02jan1999   23.01461
  3.      31.4  03jan1999   15.16839
  4.      30.1  04jan1999   31.21309
   .
   .
   .
196.      40.9  15jul1999   46.59648
197.      41.1  16jul1999   44.80204
198.      44.8  17jul1999   43.92847
199.      46.7  18jul1999   45.91249
200.      49.4  19jul1999   46.34773
201.         .  20jul1999   48.54581
202.         .  21jul1999   47.66978
203.         .  22jul1999   47.78439
204.         .  23jul1999     47.899
205.         .  24jul1999   48.01361
206.         .  25jul1999   48.12821
207.         .  26jul1999   48.24282
```

To graph our predictions we type:

```
. gr ptemp temp time,
c(ll) title(''Predicted Temperature'') saving(pretemp)
```

0.13 Do Files

The discussion in this manual has presumed that you are working *interactively* on Stata. That is, you type a command and Stata executes it and then you issue another command. Often you might want to run a single program which is contained a file that has a number of Stata commands. This is called a *batch file* which Stata has called a *do* file.

Suppose that you have a file called *a:\temp.do* which contained the following:

```
#delimit ;
set more off;
capture drop _all;
infile temp using a:\bntemp;
log using a:\bntemp, replace;
generate time = d(1jan1999) + _n-1;
format time %td;
tsset time;
corrgram  D.temp, lags(20);
arima temp, arima(0,1,2) tin ( ,19jul1999);
predict ptemp , y dynamic(d(20jul1999));
gr ptemp temp time,
c(ll) title(''Predicted Temperature'') saving(a:\pretemp);
log close;
save _all using a:\bntemp,replace;
```

This program contains most of the commands that we did in this manual in a single (executable) file. To run this program in Stata, type in the command line

```
do a:\temp.do
```

You can read more about the *do* command in the Stata manual.

0.14 ADO Files

I have written a bunch of *.ado* files that are discussed in your notes at various stages (see Programming Tips at the end of the chapters). These are programs that for all intents and purposes are just like built-in STATA functions. I have created help files (*.hlp*) that can give you information on how to use the program.

We have these kinds of files that automatically downloaded to you upon clicking into STATA and telling us what course you are in (see above) . We will discuss this more in class.

0.15 Help

For more information Stata has a **Help** menu. It is near the top of the screen.

0.16 Exit

To Exit Stata, first close the log file, then click on **File** and then **Exit**