Regression Analysis of Time Series (RATS)

A Brief Introduction to the Computer Program

Introduction

The aim of this note is to introduce students to the powerful econometrics computer program called RATS written by Thomas Doan. For more details on various aspects of the program users are directed towards the manual "RATS" (by Thomas Doan publisher Estima) or the Estima homepage which has a number of prewritten procedures as well as a frequently asked questions (faq) section. The homepage is HTTP:\\WWW.ESTIMA.COM.

While econometrics programs like Microfit and PC-Give are good packages for single equation estimation, producing many diagnostic tests at the touch of a button, they do have limitations. If attention focuses on the more complicated aspects of econometrics, for example, any form of system estimation or Monte Carlo analysis packages such as RATS, Shazam or TSP may well prove more useful.

RATS is available on the FROST server. This package is not menu driven, but entails the user writing his or her own program using RATS' own language. This can be done interactively entering lines one at a time or, more usually, in batch mode. Batch model involves creating a file with all of the instructions to be undertaken by RATS and then this is sent to RATS to process. We will introduce some of the most commonly used commands and provide some example programs for reference. To execute the program in batch model on the server you need to type

rats < input program name > output filename

Input program name and output filename are supplied by you. If you do not type ">" and an output filename the output will go to the screen.

Now we will outline a number of commands that you can use to write a RATS program.

1. Reading in Data

To run RATS it is best to have two files. The first file contains the data and will be referred to as the DATA FILE from now on, the second file contains the commands you want RATS to execute on the data, this file will be referred to as the PROGRAM. It is best if the data file contains just the observations for each of the variables of interest, i.e. RATS does not expect the data file to contain variable names, variable descriptions or sample statements. If the data file does contain information other than the raw data, you can tell RATS to ignore this

information by putting a "*" at the start of each line which contains this extra information. The data in the data file can be organised by observation or by variable as long as you tell RATS which you are using.

An example of data organised by observation, where there are 9 observations on each of 3 variables which we will call consumption (cons), income (inc) and inflation (infl) is (the data file called data.obs)

```
 99.2 106.3 10.6
 99.5 107.2  9.4
100.2 108.3  8.5
100.3 108.7  9.8
101.2 110.6 10.2
102.0 112.3  8.6
103.2 114.1  6.5
104.1 115.2  7.0
104.0 115.3  8.0
```

The same data organised by variable would appear as (this data file is now called data.var)

```
99.2 99.5 100.2 100.3
101.2 102.0 103.2 104.1
104.0
106.2 107.2 108.3 108.7
110.6 112.3 114.1 115.2
115.3
10.6 9.4 8.5 9.8
10.2 8.6 6.5 7.0
8.0
```

To read either of these data files into RATS one would have at the top of your PROGRAM one of the following set of instructions

```
allocate 9                                    allocate 9
open data data.obs                      or    open data data.var
data(format=free,org=obs) / cons inc infl     data(format=free,org=var) / cons inc infl
```

In either case the cons variable would take on the numbers from 99.2 to 104.0, income would take the numbers 106.2 to 115.3 and inflation 10.6 to 8.0. Having read in the data one can then undertake further analysis on this data. Throughout this note we will use the variables cons, inc and infl to demonstrate the things RATS can do. Variable names are limited to seven characters long.

If your data is time series (RATS is predominantly concerned with time series data) and you know the start and finish dates you might want to use dates rather than just the numbers 1 - 9. In which case you will need to use the *calendar* command (as earlier), which is specified as:

cal x1 x2 x3

where *x1*=the starting year, *x2*=starting season and *x3*=periodicity, so assuming quarterly data from 1974.1 to 1976.1. The command *allocate* then specifies the end observation. In this case the data reading command lines for reading the 9 observations, organised by observation, would be

cal 1974 1 4
allocate 1976:1
open data data.obs
data(format=free org=obs)/ cons inc infl

For monthly data the opening line would have been

cal 1974 1 12

and for annual data

cal 1974 1 1

Files which have been created in a lotus spreadsheet (*.wk1) can be read directly into RATS using the commands below. The first row in the spreadsheet contains the name of the variables. The data are in columns, beginning in the second row. There is no need to label observations with dates.

calendar 1972 1 4
allocate 1987:2
open data data.wk1
data(format=wks,org=obs) / cons inc infl

2. Summary Statistics

To obtain summary statistics on our three variables you type

cal 1974 1 4
allocate 1976:1
open data data.obs
data(format=free org=obs) / cons inc infl
smpl 1974:1 1976:1
statistics cons 1974:1 1976:3

Of course one can calculate summary statistics for any subset of the full sample of observations by simply changing the *sample* in the *statistics* command. It is possible to retrieve a number of variables from this command as:

%mean
%variance
%cdstat
%signif
%skewness
%kurtosis

and these can be used in the program later using the commands

compute m1=%mean this assigns the variable m1 to the mean of the first series
display m1 this reports the mean

You can choose to calculate summary statistics on any set of the variables using the command

table 1974:1 1976:3 cons inc infl

3. Generating Variables

Generating a series of numbers as a transformation of another series of numbers is fairly simple within RATS and uses usual mathematical operators after the *set* command, for example

set save 1974:1 1990:4 = inc-cons - Subtracts consumption from income.
set apc 1974:1 1990:4 = cons/inc - Divides consumption by income.
*set inc2 1974:1 1990:4 = inc**2* - Squares income.
set d1 1974:1 1990:4 = apc>=0.8 - D1=1 if APC > 0.8 and 0 otherwise.
set w=%if(apc<0.9,apc,0) - $w = \begin{cases} apc & \text{if } apc < 0.9 \\ 0 & \text{otherwise} \end{cases}$
set lapc 1974:1 1990:4 = log(apc) - Logarithm of APC.
set trend = t - Time trend.
set cons1 1974:1 1990:4 =cons{1} - Lags consumption by 1, to lead consumption use *{-1}*.
set inc2 1974:1 1990:4 = inc{2} - Lags income by 2.
set dcons 1974:1 1990:4 = cons-cons{1} - First difference of consumption.

Alternatively, to apply a unit difference to the series
diff cons / dcons
Or to difference and seasonal difference the series use:
diff(sdiffs=1,diffs=1) cons / ddscons

compute k=%mean - Generates k as the mean of the series specified in statistics.
compute c11=cov(1,1) - c11 is the (1,1) element from the matrix cov.

to generate seasonal dummies for quarterly data, (the first number after / is the periodicity , the second number the start period)

seasonal seas1 / 4 1
seasonal seas2 / 4 2
seasonal seas3 / 4 3
seasonal seas4 / 4 4

To filter a series use the filter command, such that to obtain $y_t = \frac{x_t + x_{t-1} + x_{t-2} + x_{t-3}}{4}$, the command is:

filter x 1 100 y
 # 0 1 2 3
 # 0.25 0.25 0.25 0.25

where 1 is the start date and 100 the end date, the second row is the list of leads (lags) in the filter and the third row is the weights on these leads (lags).

3.1 Probability Values

It is possible to calculate the probability (significance level or critical values) associated with a test statistic, *x1*, for a variety of distributions. The format of the command is

cdf(option) distrib x1 dg1 dg2

The options are:

distrib	- Specifies the type of distribution you wish to use the choices available include: *ftest, ttest, chisquared, normal*
x1	- The value of the statistic
dg1	-Specifies the degrees of freedom of for t and χ^2
dg1, dg2	-For F dg1 (dg2) are the numerator (denominator) degrees of freedom.

3.2 Printing data

To print data out using the following command, which prints the data over the whole sample period. Using a *smpl* before this command enables the user to print out data for any sub-period.

print / cons infl inc	Prints the series cons inc and infl
display / k c11	Prints scalars and expressions
write / @cov	Prints matrices and vectors

4. OLS

Having transformed your variables you may wish to run an OLS regression. The form of this command is simple

linreg(options) cons 1980:1 1994:2 / resid bs
 # constant inc{0 1} infl cons{1 1}

This runs an OLS regression in which cons is the dependent variable and constant inc, inc(-1), infl, and cons(-1) are the explanatory variables. The commands which appear after the "/" enable you to save the residuals (*resid*) and the coefficient estimates (*bs*).[1] The options are:

vcv	- Prints the covariance matrix of the parameter estimates.
instruments	- Lists the instruments for IV estimation.
robusterrors	- Computes White's heteroscedastic consistent standard errors.

Temporary variables can be stored associated with the OLS regression those available are

%beta	-Estimated coefficients
%xx	$-(X'X)^{-1}$
%ndf	- Degrees of freedom
%n	- Number of observations
%nreg	- Number of regressors
%rsquared	- R-squared
%rbarsq	- R-bar squared
%trsq	- Number of observations times R^2
%durbin	- Durbin-Watson statistic
%rss	- RSS
%seesq	- Estimated variance of the error
%qstat	- Q-statistic
%qsignif	- Significance of Q-statistic
%rho	- First correlation coefficient
%uzwzu	- u'ZWZ'u for instruments

You could retrieve these values by typing

linreg(options) cons
 # inc{0 1} infl cons{1 1}

compute [symm] cov=%seesq%xx*	- this saves the estimated covariance matrix
write cov	- to display the estimated covariance matrix
compute [vector] beta=%beta	- this saves the estimated coefficients
write beta	- to display the estimated coefficients

[1] To save the coefficients without the residuals use a * in place of resid.

4.1 Hypothesis Testing

This can be done using 3 alternative methods, depending on the testing to be undertaken:

1. Test

This allows the testing of parameter restrictions (although this does not allow linear restrictions across parameter estimates)

linreg(options) cons
 # constant inc{0 1} infl cons{1 1}
test
 # 2 3
 # 1 1
Tests the joint restriction $\beta_2 = \beta_3 = 1$

2. Restrict

This allows the testing of all linear restrictions on the parameter values of the form $R\beta = r$.

linreg(options) cons
 # constant inc{0 1} infl cons{1 1}
restrict(options) 2
 # 2 3
 # 1 1 0
 # 4
 # 1 0
this estimates the model and tests $\beta_2 + \beta_3 = 0$ and $\beta_4 = 0$. Note if you put *create* as the option in the *restrict* command then it actually estimates the restricted equation (it is then possible to save the residuals and coefficient estimates from this restricted model).

3. Exclude

This allows the testing of exclusion restrictions in a linear regression model:

linreg(options) cons
 # constant inc{0 1} infl cons{1 1}
exclude
inc{0 1}

Tests the joint restriction $\beta_2 = \beta_3 = 0$

5. Instrumental Variables

The format for this instruction is similar to that for OLS

```
instruments constant inc{1 2} infl{0 2} cons{1 2}
linreg(instruments) cons 1980:1 1992:4
# constant inc{0 1} infl cons{1 1}
```

The variables in the instrument list are the exogenous variables used to instrument out *inc* in the regression equation above. A constant is included as an instrument as are all the explanatory variables of the original equation. The options available are the same as for OLS and parameter restrictions can be also be used as in the OLS case.

6. Nonlinear Regression

The form of this instruction is different from that of OLS in that you must explicitly right out the equation you want to estimate. In the following example we estimate the following nonlinear equation with 5 coefficients

$$y_t = \alpha_0 + \alpha_1 x_{1,t}^{\alpha_2} + \frac{\alpha_2 x_{2,t} + \alpha_3 x_{3,t}}{1 - \alpha_4} + u_t$$

The RATS code for this equation would be:

```
*The form of this equation has the number of equations (1) after nl and the number of
coefficients (5) with ncoef
nonlin alpha0 alpha1  alpha2 alpha3 alpha4
frml eq1=alpha0+alpha1*(x1**alpha2)+(alpha2*x2+alpha3*x3)/(1-alpha4)
compute alpha0=0
compute alpha1=0.12
compute alpha2=0.5
compute alpha3=0.14
compute alpha4=0.4
nlls(frml=eq1,trace) y
```

The compute lines provide the initial values for the coefficient estimates and the *nlls* command specifies the dependent variable as being, *y* (the trace command ensures the progress of estimation are reported). Again a series of options exist. In addition, to those available for OLS there exist options to control the method of optimisation using the *nlpar* command, which

enables the user to determine the number of iterations and the optimisation method as well as convergence criteria.

7. Nonlinear Systems

Here we look at nonlinear system estimation using an example borrowed from Barro. The equation we want to estimate is of the form

$$z_t = z_t^* + u_t = \beta_0 + \beta_1 x1_t + \beta_2 x2_t + u_t$$
$$y_t = \delta_0 + \delta_1(z_t - z_t^*) + \delta_2 z_{t-1}^* + \eta_t$$

The RATS code for this would be:

```
nonlin b0 b1 b2 d0 d1 d2
frml eq1 z=b0+b1*x1+b2*x2
frml eq2 y=d0+d1*(z-b0-b1*x1-b2*x2)+d2*(b0+b1*x1{1}+b2*x2{1})
compute b0=0.04
compute b1=0.1
compute b2=0.3
compute d0=-0.3
compute d1=0.2
compute d2 -0.1
nlsystem(options) /eq1 eq2
```

Options
instruments - If an instrument list is supplied does GMM estimation.

8. ARIMA Modelling

In general this process is split into 2 stages:

i. Identification and Estimation

```
boxjenk(options) x1 / resids
```

The options available include

diffs=n1 - Differences the data n1 times
sdiffs=n2 - Seasonal differences the data n2 times
span=n3 - Specifies the number of periods in the cycle
ar=p - Number of AR parameters 1 through to p. However if type ar=||1,2,4|| only uses these lags
ma=q - Number of MA parameters
sar=sp - Number of seasonal AR parameters
sma=sq - Number of seasonal MA parameters

initial=b - Vector of initial guesses in the following order:
1. Constant
2. AR
3. SAR
4. MA
5. SMA

define=bjeq1 - Define a name this is essential if forecasting is required with this model.

ii <u>Forecasting</u>

boxjenk(define=eq1,diffs=1, ar=||1,4,6|| ma=2) x1 / resids
forecast 1 24 1990:4
 # eq1 x1f

In the forecast line *1* refers to the number of equations, *24* the number of steps ahead and *1990:4* to the start of the forecasts. The second line instructs as to which equation is being forecast, *eq1*, and then supplies a variable name in which forecasts are written, *x1f*.

9. <u>Matrices</u>

To read a matrix into RATS you need the following instructions

declare vector a
declare rect b
dimension a(3) b(3,3)
compute [vect] a=|| 1.0, 2.0, 3.0||
compute b=|| 1.0, 2.0, 3.0| 2.0, 6.1, 1.1| 2.0, 2.0, 1.0||

It is always advisable to *declare* the array first as either (i) *vect*, (1×n) (ii) *rect*, (m×n) or (iii) *symm* (n×n), where A=A', and *dimension* the matrix (number of rows, number of columns). Alternatively you can create a matrix from the different variables you have created in the program, for example

make z start end numobs numvars
#cons inc infl

This copies the observations from start to end on *cons*, *inc* and *infl* into a matrix *z* which has dimension (*numobs* x *numvars*).

Once the matrix exists, it is possible to do a series of operations on the matrix

-	Subtraction
*	Multiplication
+	Addition

In addition to these there exists a series of other operations

Tr(matrix)	- Transpose
inv(matrix)	- Inverts a matrix
%diag(matrix)	- Creates an Nx1 vector of diagonal elements from an NxN matrix. Or other way round
%kroneker(A,B)	- Kroneker product of A and B
%decomp(matrix)	- Computes a Choleski decomposition of a symmetric matrix.
%kronid(A,B)	- Forms (A⊗I)B
%mqform(A,B)	- Computes B'AB
%decomp(matrix)	- Does a Choleski decomposition of matrix
%det(matrix)	- Takes the determinant of matrix
%trace(matrix)	- Takes the trace of matrix
%dot(A,B)	- Forms the dot product of A and B (of same dimension)
%corr(A,B)	- Forms the correlation of A and B.
%maxvalue(matrix)	- Forms the maximum value of the matrix.
%minvalue(matrix)	- Forms the minimum value of the matrix.

With these instructions it is imperative that the matrix has been declared and dimensioned earlier

%ran(x)	- Creates an array of $N(0,x^2)$.
%uniform(x1,x2)	- Creates an array of U(x1,x2).
%const(x)	- Creates an array with the value x.
%identity(n)	- Creates an I (n×n) matrix.
eigen A EVL EVC	- Calculates the eigenvalues of the matrix *A* and write the eigenvalues to *EVL* and the eigenvectors to *EVC.*
*ewise sigma(i,j)=0.9**(i-j)*	- Formulates the matrix sigma on an element by element basis.

For example,

```
declare u rect
dimension u(100,2)
* Takes the inverse of the matrix z'z
compute a=inv(z'z)
* Forms a Choleski decomposition of the matrix a
compute c=decomp(a)
* Forms two series with 30 observations of random normals
compute u=%ran(2)
```

Alternatively, the following commands form the variance covariance matrices between two vectors of series.

```
make ut 1 100
# u1 u2
make vt 100
# v1 v2
compute sigmauu=u*tr(ut)
```

```
compute sigmavv=v*tr(vt)
compute sigmauv=u*tr(vt)
compute c=decomp(sigmavv)
compute bigmat=inv(c)*sigmavu*inv(sigmauu)*sigmauv*inv(c')
eigen bigmat eigval
```

Note: Compute *a=20/3* produces 6

whereas *a=20./3* produces 6.666667

10. Do Loops

Do loops are very important for simulations. Here are two examples of the use of *do* loops. The first is a simple count loops. The second uses a *dofor* loop.

Example 1

```
*these set the number of replications and the number of observations
compute ndraws=1000,endobs=100
all endobs
*these create series to store regression output
set beta 1 ndraws = 0.0
set tstat 1 ndraws = 0.0
set counter 1 ndraws = 0.0
set durbin 1 ndraws = 0.0
*this sets the seed for random number generator
seed 123
do draws=1,ndraws
  *these create two independent stationary series y and x, which
  *are n(0,1)
  set y 1 endobs = %ran(1.0)
  set x 1 endobs = %ran(1.0)

  *bivariate regression of y on x; output is omitted
  linreg(noprint) y 1 endobs
  # constant x
  *this computes variance-covariance matrix
  compute std=%xx*(%rss/%ndf)
  *this computes standard error of the slope coefficient
  compute stde=std(2,2)**0.5
  *this stores the slope coefficient in beta
  compute beta(draws)=%beta(2)
  *this calculates and stores the t ratio for ho: b=0 in tstat
  compute tstat(draws)=beta(draws)/stde
  *this defines the counter which is equal to 1 if the absolute
  *value of the t ratio is greater than 2, 0 otherwise
  compute counter(draws)=%if(abs(tstat(draws))>2,1,0)
  *this stores the durbin-watson statistic
```

compute durbin(draws)=%durbin
end do draws

Example 2
dofor rho = 1 0.999 0.99 0.98 0.95 0.9 0.85 0.8 0.7 0.5
* Rest of program
****
end dofor rho

11. VAR Models

The setup of a standard VAR estimation of a three equation system (for cons, inc and infl), with 4 lags for each variable as well as a constant and a time trend takes the form:

system 1 to 3
variables cons inc infl
lags 1 to 4
deterministic constant trend
end(system)

For specification of a system with gaps in the lags, for example if you only want lags 1,2 and 4, you must write

lags 1 2 4

Alternatively, for a near VAR system you must specify:

equation 1 cons
 # constant cons{1 to 4} inc{1 to 4}
equation 2 inc
 # constant cons{1 to 4} inc{1 to 4} infl{1 to 4}
equation 3 infl
 # constant cons{1 to 4} inc{1 to 4} infl{1 to 4}
end(system)

11.1 Estimating VARs

To estimate these VAR (type) models use the *estimate* command.

estimate(outsigma=vcv1, vcv) start end

Estimates the VAR model over the period *start* to *end* of the residuals and then saves this matrix as *vcv1*. The saving of this matrix is important for variance decompositions. To save the residuals for the system requires the use of the *declare* option as

declare vector[series] res(3)
estimate(outsigma=vcv1, vcv) start end/res(1)

To test for the appropriate lag length you can use the *ratio* option, which undertakes a likelihood ratio test between two VAR models

```
system 1 to 3
variables cons inc infl
lags 1 to 4
deterministic constant trend
end(system)
estimate(outsigma=vcv1, vcv) start end/1
system 1 to 3
variables cons inc infl
lags 1 to 2
deterministic constant trend
end(system)
estimate(outsigma=vcv1, vcv) start end/4
*
ratio(degrees=6, other options)
# 1 to 3
# 4 to 6
```

the options available are:

smpl=	- Sample period.
mcorr=	- Multiplier correction.

The supplementary card lists the residuals from the first and second VAR models, respectively. Based on the VAR model estimated one can then formulate a structural VAR model using Bernanke's method for identifying the structural parameters, based on prior information. Alternatively, one can undertake a Choleski decomposition of the VAR, for example:

```
system 1 to 3
variables cons inc infl
lags 1 to 4
deterministic constant trend
end(system)
estimate(outsigma=vcv1, vcv)
# 3 0 0 3
# 1 0 0 1
# 2 0 0 2
```

and this undertakes a Choleski decomposition in the order *infl-cons-inc*. The format of the command line is

 # equation series newstart column

where column refers to the column position in the covariance matrix of the residuals.

11.2 Impulse Responses

This generates the responses of a system to a specified set of shocks and these can be used to either generate a moving average representation of the VAR or to compute dynamic multipliers.

The form of the command is:

impulse(options) neq steps shocks to vcv1
 # eqn response newstart column

neq	- Number of equations.
steps	- Number of steps ahead to compute responses.
shocks to	- Referring to the component to be shocked in supplementary cards.
vcv1	- Covariance matrix of residuals.

In the supplementary card

eqn	- Equation name.
response	- The series for the response of the dependent variable.
newstart	- Starting entry for the series *response*.
column	- Column entry in *vcv1* matrix.

For example one may have

system 1 to 3
variables cons inc infl
lags 1 to 4
deterministic constant trend
end(system)
estimate(outsigma=vcv1, vcv)
impulse 3 20 2 vcv1
 # 3 impi 1 3
 # 1 impf 1 1
 # 2 impc 1 2

Shocks to the orthogonal innovations in component 1 (2nd listed), decomposition in the order 3-2-1, with responses stored as impi, impf, impc.

11.3 Error Decomposition

Decomposes the variance of the VAR model into that contributing from various components.

The format of the instruction is:

errors neq steps vcv1
 # eqn stderr newstart column

neq	- Number of equations
steps	- Number of forecast steps
vcv1	- Covariance matrix of the residuals

In the supplementary card

eqn - Equation name
stderr - errors will fill this series with the standard errors of forecasts
newstart - Starting entry for *stderr*

List of RATS Procedures

More information on each of these is available in the manual.

Filename	Description of Program
BAYESTST.SRC	Sims' Basyesian unit root testing procedure.
BERNANKE.SRC	Bernanke-Sims decomposition procedure.
BJEST.SRC	Estimates an ARIMA forecasting model.
BJFORE.SRC	Computes forecasts from an ARIMA model.
BJIDENT.SRC	Procedure for identifying Box-Jenkins forecasting models.
BJTHEIL.SRC	Computes Theil U statistics for an ARIMA model.
BSOPTION.SRC	Black-Scholes option pricing formula.
CNDITION.SRC	Conditional forecasting for VAR's.
CROSSPEC.SRC	Computes and graphs the coherence and phase of two series.
CUMPDGM.SRC	Performs a cumulated periodogram test for serial correlation.
DFUNIT.SRC	Dickey-Fuller unit root tests.
DISTRIB.SRC	Distributes a series to a higher frequency.
DIVISIA.SRC	Computes a divisia index.
FPE.SRC	Estimates a VAR by the FPE criterion.
GAIN.SRC	Alternative version of CROSSPEC, computes the gain and coherence.
GPH.SRC	Geweke-Porter-Hudak fractional differencing.
HPFILTER.SRC	Executes a Hodrick-Prescott filter.
INTERPOL.SRC	Interpolates a series to a higher frequency.
MSEA.SRC	Computes a spectrum using the Maximum Entropy method.
MIXED.SRC	General mixed estimation.
MIXVAR.SRC	Computes mixed estimates for a single VAR.
OLSTRUNC.SRC	Estimates a linear model for truncated data sets.
PDL.SRC	Estimates a polynomial distributed lag regression.
PERSIST.SRC	Estimates the sum of a univariate moving average representation coefficient.
PPUNIT.SRC	Phillips-Perron unit root tests.
RECRESID.SRC	Computes a series of recursive residuals for a regression.
SPECFORE.SRC	Forecasts using spectral techniques.
SPRECTRUM.SRC	Computes and graphs a spectral density.
STOCKWAT.SRC	Computes unit root and trend tests using the Stock-Watson methods.
TVARSET.SRC	Time-varying parameters VAR setup procedure.
VAR1LAG.SRC	Estimares a one lag VAR.