SHAZAM

A Brief Introduction to the Computer Program

Introduction

The aim of this note is to introduce students to the powerful econometrics computer program called SHAZAM written by Kenneth White. For more details on various aspects of the program users are directed towards the very comprehensible reference manual "Shazam: Econometrics Computer Program User's Reference Manual Version 7.0" (by Ken White publisher McGraw-Hill, some copies of this manual are in the bookshop).

While econometrics programs like Microfit and PC-Give are very 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 Shazam or TSP may well prove more useful.

Both Shazam and TSP are available on the UNIX at Warwick, both have a similar language and do similar things; however, if you wanted to undertake simulation exercises TSP is probably far more useful than Shazam. Here we will concentrate our discussion on Shazam. This package is not menu driven, but entails the user writing his or her own program using Shazam's 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 Shazam and then this is sent to Shazam to process. We will introduce some of the most commonly used commands and provide some example programs for reference.

When writing your Shazam programs it is probably advisable to write the initial program on the micro computer and then transfer that program to the mainframe as the jove editor on the mainframe is not as powerful as that on the micro. To execute the program in batch model on the mainframe you need to type

shazam < 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 Shazam program.

1. Reading in Data

To run Shazam it is best to have two files. The first file will contain the data and will be referred to as the DATA FILE from now on, the second file will contain the commands you want Shazam 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. Shazam 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 Shazam to ignore this information by put 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 Shazam 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 SHAZAM one would have at the top of your PROGRAM one of the following set of instructions

file 4 data.obs		*file 4 data.var*
sample 1 9	or	*sample 1 9*
read(4) cons inc infl		*read(4) cons inc infl/byvar*

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.

NOTE
In these two examples we are reading from unit 4 (on the file and read lines) you could have made the unit 4, 11, 12, 13,.., 99. If you are reading multiple files it is best to read them from different units.

Throughout this note we will use the variables cons, inc and infl to demonstrate the things Shazam could do, however, you may call your variables anything you like using mnemonics up to seven characters long.

If your data is time series and you know the dates for the start and finish you might want to use dates rather than just the date-anonymous numbers 1 - 9. In this case the data reading command lines for reading in observations organised by observation would be

```
time 1974 4
file 4 data.obs
sample 1974.1 1976.1
read(4) cons inc infl
```

For monthly data the opening line would have been *time 1974 12* and for annual data *time 1974 1*. In the sample line ".x" indicates a quarter or a month, x.

1.b Writing Out Data

The form for writing out data is similar to reading it in for any set of variables, one can write out the observations by observation or by variable (by observation is the default). For well presented data one can also write it out formatted. An example of writing out formatted data would be

```
* This format line controls the output, the data will be writing by observation (three numbers
* per line), each number will take up 10 spaces with 2 after the decimal point
format(3f10.2)
file 7 newdata.dat
sample 1973.1 1976.4
write(7) cons inc infl/format
```

The data file newdata.dat would then look like

```
 99.20   106.30    10.60
 99.50   107.20     9.40
100.20   108.30     8.50
```

100.30	108.70	9.80
101.20	110.60	10.20
102.00	112.30	8.60
103.20	114.10	6.50
104.10	115.20	7.00
104.00	115.30	8.00

2. Summary Statistics

To obtain summary statistics on our three variables you type

time 1974 4
file 4 data.obs
sample 1974.1 1976.1
read(4) cons inc infl
sample 1974.1 1976.1
stat cons inc infl/pcor, pcov, mean=name1, stdev=name2[1]

pcor = Prints the Correlation matrix
pcov = Prints the Covariance matrix
mean=name1 - Saves the means of the 3 series in a vector called *name1*
stdev=name2 - Saves the standard deviations in a vector called *name2*

Of course one can calculate summary statistics for any subset of the full sample of observations by simply changing the *sample* command before the *stat* instruction. You can choose to calculate summary statistics on any subset of the variables.

3. Generating Variables

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

genr save=inc-cons - Subtracts consumption from income
genr apc=cons/inc - Divides consumption by income
*genr inc2=inc**2* - Squares income
genr d1=(apc.ge.0.8) - D1=1 if APC > 0.8 and 0 otherwise
genr lapc=log(apc) - Logarithm of APC
genr cons1=lag(cons,1) - Lags consumption by 1
genr inc2=lag(inc,2) - Lags income by 2

[1] This is only a subset of the options one can use with the stat command.

genr dcons=cons-cons1	- First difference of consumption
genr u=nor(1)	- Generates u = Normal(0,1) variable
genr y=uni(2)	- Generates y= Uniform(0,2)
gen1 pi=3.1415926	- Generates a single number called pi
gen1 m1=mean1:1	- Assigns the mean of cons to a value *m1*

It is also possible to calculate the probability significance level (or critical values) a series of numbers *x1*, for a variety of distributions. The format of the command is

distib x1/options

The options are:

type=	-Specifies the type of distribution you wish to use the choices available include: *normal, beta, binomial, chi, f, gamma, imhof,* and *t*
df=n1	-Specifies the degrees of freedom as *n1* for the *type=t* or *type=chi* distributions
df1=n2,df2=n3	-Specifies the degrees of freedom of numerator *n2* and denominator *n3*
inverse	-Returns the critical values rather than the probabilities

4. OLS

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

ols cons inc infl/ options

This runs an OLS regression in which cons is the dependent variable and inc, infl and an intercept (which is included by default) are the explanatory variables. The options command which appear after the "/" control the amount of output results you get and enable you to save parameter estimates. Examples of the options are:

noconstant	- Surpresses the intercept to be zero.
coef=name1	- Saves the coefficients in a vector *name1*
cov=name2	- Saves the covariance matrix in a matrix *name2*
stderr=name3	- Saves the standard errors of the coefficients in the vector *name3*
hetcov	- Uses White's (1980) standard errors.
lm	- Prints the Jarque-Bera test of normality
resid=name4	- Saves the residuals in a variable called *name4*
predict=name5	- Saves the predicted values in a variable called *name5*

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

$DF - Degrees of freedom

$k - Number of coefficients
$n - Number of observations
$r2 - R-squared
$sig2 - Estimated variance of the disturbance

You would retrieve these values using by typing

ols cons inc infl/ options
gen1 sigma2=$sig2 - this saves the estimated variance of the disturbance term in sigma2

You can also test parameter restrictions and undertake diagnostic test on the OLS equation you have just estimated. An example of the instructions which would enable you to test a set joint restrictions and undertake diagnostic tests on the regression equation are

ols cons inc infl cons1/coef=b stderr=se cov=mcov
*This tests 2 restrictions as a joint null hypothesis
test
test infl+inc=0
test constant=0
end
*Diagnostic test can include :
* Autocorrelation tests (acf),
* Hetroscedasticity test (het)
* Reset tests (reset)
* Sequential structural break tests (chowtest)
* Bootstrapping of the coefficient estimates (bootsamp=number of samples, e.g.1000)
diagnos/acf,het,reset,chowtest,bootsamp=n
* Comments can go in the middle of you program, but must have a "*"
* at the start of the line
* This forms a t-ratio for the coefficient on cons1. Note that to call up an element from a vector
* you simply use the colon ":" after the vector name and the element you are interested in.
gen1 t1=(b:3-0.6)/se:3
* If your instruction goes over more than one line you continue it
* on the next line but type an "&" at the end of the first line

To call up an element from a matrix you would write

gen1 c11=mcov(1,1)

5. ARIMA Modelling

In general this process is split into 3 stages:

i. Identification

arima cons/options

The options available include

log - Takes logarithms of the series

plotac - Plots the autocorrelation function
nlag=n1 - The number of lags on the autocorrelation function is n1
plotpac - Plots the partial autocorrelation function
nlagp=n2 - The number of lags on the partial autocorrelation is n2
ndiff=n3 - Differences the data n3 times
nsdiff=n4 - Seasonal differences the data n4 times
nspan=n5 - Specifies the number of periods in the cycle

ii Estimation

arima cons/nar=n1 nma=n2 start options
0.5 -0.2 100

nar=n1 - Number of AR parameters *n1*
nma=n2 - Number of MA parameters *n2*
nsar=n3 - Number of seasonal AR parameters *n3*
nsma=n4 - Number of seasonal MA parameters *n4*
nspan=n5 - Specifies the number of periods in the cycle *n5*
noconst - Excludes the constant term
start - Indicates starting values are specified straight after the arima command in the order *nar, nma, nsar, nsma* and the *constant* last. In our example nar=1, nma=1, nsar=0, nsma=0
resid=name1 - Saves the residuals in *name1*
predict=name2 - Saves the predicted values in *name2*
coef=b1 - Saves the coefficients in *b1*

iii Forecasting

arima cons/nar=n1 nma=n2 fbeg=n6 coef=b1 fend=n7

coef=b1 - Gives the model the coefficients for forecasting *b1*
fbeg=n6 - Beginning dates for the forecast to start *n6*
fend=n7 - End date for the forecasting *n7*
predict=name4 - Saves the forecasts in *name4*

6. Nonlinear Regression

The form of this instruction is different from that of OLS in that you must explicitly write 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 Shazam 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*

```
nl 1/ncoef=5 options
eq y=alpha0+alpha1*(x1**alpha2)+(alpha2*x2+alpha3*x3)/(1-alpha4)
coef alpha0 0 alpha1 0.12 alpha2 0.5 alpha3 0.14 alpha4 0.4
```

This last line provides the initial values for the coefficient estimates. Again a series of options exist. In addition, to those available for OLS there exist options to control the method of optimisation, for example, the number of iterations and the optimisation method.

7. Two Stage Least Squares

The format for this instruction is similar to that for OLS

```
genr inc1=lag(inc,1)
2sls cons inc infl cons1 (infl infl1 cons1 inc1)/ options
(inst cons inc infl cons1 (infl infl1 cons1 inc1)/ options)
```

For the *2sls* command the first variable cons is the dependent variable and inc, infl and cons1 are the right hand side explanatory variables. The variables in parentheses are the exogenous variables used to instrument out inc in this example. An intercept is included as an instrument and in the regression by default. The other command *inst* has exactly the same format as *2sls*, but uses instrumental variable estimation rather than two stage least squares. The options are again similar to those of OLS.

8. 3SLS and FIML

If you plan on doing nonlinear two stage least squares (nl2sls), three stage least squares (linear or nonlinear) or nonlinear system estimation you should use the *nl* command. In this section we will give examples of each of these

8.1 NL2SLS

To specify 2SLS with the *nl* command the user must specify a set of instrumental variables to be used for estimation. In the example that follows the instruments are infl, infl1, con1, inc1 and a constant which is automatically included (note the system we are actually estimating is linear and so *2sls* or *inst* should yield the same solutions)

```
nl 1 infl, infl1, inc1, cons1/ncoef=4
eq cons=beta0+beta1*inc+beta2*infl+beta3*cons1
coef beta0 0.0 beta1 1.0 beta2 1.0 beta3 1.0
end
```

8.2 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 Shazam code for this would be:

```
nl 2/ncoef=6 options
eq z=b0+b1*x1+b2*x2
eq y=d0+d1*(z-b0-b1*x1-b2*x2)+d2*(b0+b1*lag(x1,1)+b2*lag(x2,1))
coef b0 0.04 b1 0.1 b2 0.3 d0 -0.3 d1 0.2 d2 -0.1
end
```

8.3 3SLS

The instructions for estimating 3SLS (linear or otherwise are the same). We will estimate a simple linear system of equations to illustrate the form of the commands used. In actual fact the form of the commands is the same as that for NL2SLS except there is more than one equation in the specified system

```
nl 3 x1 x2 x3/ncoef=12 options
eq y1=b0+b1*x1+b2*x2+b3*y2
eq y2=d0+d1*x1+d2*lag(x1,1)+d3*y1
eq y3=f0+f1*x3+f2*y2+f3*lag(x2,1)
coef b0 0.04 b1 0.1 b2 0.3 b3 2.0 d0 -0.3 d1 0.2 d2 -0.1 d3 1.5 f0 1.0 &
   f1 -1.3 f2 0.6 f3 1.3
end
```

8. Cointegration and Unit Roots

Shazam will also permit the undertaking of unit root test using the Dickey-Fuller (DF) and the Perron and Phillips (PP) methodologies. The format of the command is

coint cons/ options

log	- Takes logs of the data
ndiff=n1	- Differences the data *n1* times
nlag=n2	- Specifies the lag length *n2* to use calculating either the DF or PP
type=	- Specifies the type of test to be done this can be any of the 4 following
	- *df* for Dickey-Fuller
	- *pp* for Perron-Phillips
	-*resd* for DF on the residuals of a regression
	-*resp* for PP on the residuals of a regression

Shazam choose a default lag length as the maximum significant value in the autocorrelation function of the first difference of the series under investigation. This is a peculiar choice and specifying your own lag length is advisable. The results report the usual individual t-test for DF (t_α 1) and PP ($Z(t_\alpha)$ 2) as well as the appropriate joint F-type tests (Φ_3, Φ_2, Φ_1, 3 $Z(\Phi_3)$, $Z(\Phi_2)$, $Z(\Phi_1)$ 4).

9. Matrices

To read a matrix into shazam you need the following instructions

read a/rows=3 cols=4
1 0 2 3
0 3 6 2
1 1 3 6

Alternatively you can create a matrix from the different variables you have created in the program, for example

copy cons inc infl z/frow=1;9 trow=1;9

This copies the 9 observations on consumption, income and inflation into a matrix z which has dimension (9x3). If you are constructing a matrix from a variety of sources it might be useful to dimension the matrix first.

dim v 4 5

This dimensions the matrix v with 4 rows and 5 columns.

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

-	Subtraction
*	Multiplication
+	Addition
'	Transpose
@	Kronecker Multiplication
|	Concatenation

In addition to these there exists a series of other operations

chol(matrix)	- Does a choleski decomposition of matrix
det(matrix)	- Takes the determinant of matrix
diag(matrix)	- Creates an Nx1 vector of diagonal elements from an NxN matrix. Or other way round
eigval(matrix)	- Forms the eigenvalues of a matrix
eigvec(matrix)	- Forms the eigenvectors of a matrix
iden(ndim)	- An identity matrix with ndim rows and columns
inv(matrix)	- Inverts a matrix
lag(matrix,n)	- Each column of matrix is lagged n times
nor(nrow, ncol)	- Normally distributed numbers are created with rows and columns specified by nrow and ncol
samp(matrix,nrows)	- Creates a new matrix with nrows from random sampling the old matrix
seas(nobs,seas)	- Seasonal dummy variables are created with the number of observations and number of seasons determined by nobs and seas
trace(matrix)	- Forms the trace of the matrix

For example,

* Takes the inverse of the matrix z'z
matrix a=inv(z'z)
* Forms a Choleski decomposition of the matrix a
matrix c=chol(a)
* Forms two series with 30 observations of random normals
matrix u=nor(30,2)

10. <u>Do Loops</u>

It is also possible to program in Shazam invariably using the do-loop option. In this example we write a Monte Carlo program for the parameters of the OLS regression.

* This sets the sample size
sample 1 20
* This dimensions the matrix
dim b 2 1000

```
* This generates an x vector
genr x=uni(2)
* Avoiding too much output
set nodoecho
*start the do loop
do #=1,1000
* generates a series of normal errors
genr u=nor(1)
* constructing the y series with known coefficients
genr y=1+0.5*x+u
* running the OLS regression and saving the coefficients in the
* dimensioned matrix b. Note b:2 is the constant (or intercept)
?ols y x/coef=b:#
* Stops the do loop
endo
matrix b=b'
sample 1 1000
* Summary stats on the coefficient vector
stat b
*quits the program
stop
```

Finally we finish by introducing an entire program that was written to estimate the Barro (1977) money demand and unemployment equations correcting for the generated regressor problem in the unemployment equation.

```
time 1939 1
* dm = % growth in money
* fedv = normal government expenditure
* mil = No. doing military service
* minw = Minimum wage
* un = Number unemployed
file 4 barro.dat
smpl 1939 1985
read(4) dm fedv mil minw un/byvar
* generating lags of the variables and other transformations
genr dm1=lag(dm,1)
genr dm2=lag(dm,2)
genr dm3=lag(dm,3)
genr dm4=lag(dm,4)
genr un1=lag(un,1)
genr un2=lag(un,2)
genr un3=lag(un,3)
genr inter=1.0
genr fedv1=lag(fedv,1)
genr fedv2=lag(fedv,2)
```

```
smpl 1941 1973
* Money growth equation
ols dm dm1 dm2 fedv un1/resid=dmr
gen1 sig2n=$sig2
* Unanticipated money growth = dmr
genr dmr1=lag(dmr,1)
genr dmr2=lag(dmr,2)
smpl 1946 1973
* Unemployment = fn(unanticipated money growth, min wage and military service)
ols un dmr dmr1 dmr2 minw mil/coef=gamma
* Tests the restriction that minw and mil are jointly equal to zero
test
test mil=0
test minw=0
end
gen1 sig2e=$sig2
* The matrix of explanatory variables in the money equation + lags
copy dm1 dm2 fedv un1 inter w/frow=1946;1973 trow=1;28
copy dm2 dm3 fedv1 un2 inter w1/frow=1946;1973 trow=1;28
copy dm3 dm4 fedv2 un3 inter w2/frow=1946;1973 trow=1;28
* Form the matrix of explanatory variable in the unemployment equation
copy dmr dmr1 dmr2 minw mil inter phi/frow=1946;1973 trow=1;28
* Now forming the Correct covariance matrix for the unemployment equation
matrix iphi=inv(phi'*phi)
* Incorrect OLS covariance estimates of unemployment equation
matrix vols=sig2e*iphi
* Forming the biasedness component to get consistent standard errors
matrix iww=inv(w'*w)
matrix iwwwt=iww*w'
matrix sum=gamma:1*w+gamma:2*w1+gamma:3*w2
matrix s1=sum*iwwwt
matrix s1s1=sig2n*s1*s1'
matrix iphiphi=iphi*phi'
* Correct OLS covariance matrix
matrix cov=vols+iphiphi*s1s1*iphiphi'
print cov
stop
```