TSP

A Brief Introduction to the Computer Program

Introduction

The aim of this note is to introduce students to the powerful econometrics computer program called TSP written by Bronwyn Hall. For more details on various aspects of the program users are directed towards the reference manual and user's guide. Unfortunately, to date there is only one copy of each of these held in the Computer Services Centre by Frances Halstead.

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. For more details on running Shazam one is referred to the introductory guide for Shazam. Here we will concentrate our discussion on TSP. This package is not menu driven, but entails the user writing his or her own program using TSP'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 TSP and then this is sent to TSP to process. We will introduce some of the most commonly used commands and provide some example programs for reference.

When writing your TSP 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

tsp   input program name   output filename

Input program name and output filename are supplied by you. If you do not type 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 TSP program.

1. Reading in Data

To run TSP 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 TSP 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. TSP 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 TSP to ignore this information by putting a "?" at the start of each line which contains this extra information. It is easiest to read in data organised by observation.

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
```

To read this data file into TSP one would have at the top of your PROGRAM the following set of instructions

```
smpl 1 9;
read(file='data.obs') cons inc infl;
load;
```

In this case the cons variable takes on the numbers from 99.2 to 104.0, income takes 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 TSP 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

```
freq q;
smpl 74:1 76:1;
read(file='data.obs') cons inc infl;
load;
```

For monthly data the opening line would have been *freq m* and for annual data *freq a*. 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, although it is best to write out the observations by observation. 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
smpl 1973.1 1976.4;
write(file ='newdata.dat',format='(3f10.2)') cons inc inf;
```

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

```
freq q;
smpl 74:1 76:1;
read(file='data.obs') cons inc infl;
load;
```

msd (options) cons inc infl[1]

corr = Prints the Correlation matrix
cova = Prints the Covariance matrix

It is possible to save summary statistics from this command

@corr	-Correlation Matrix
@cova	-Covariance matrix
@mean	-Mean vector
@stddev	-Standard deviation vector
@median	-Median Vector

Of course one can calculate summary statistics for any subset of the full sample of observations by simply changing the *smpl* command before the *msd* 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 TSP 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>0.8;	- D1=1 if APC > 0.8 and 0 otherwise
genr lapc=log(apc);	- Logarithm of APC
genr cons1=cons(-1);	- Lags consumption by 1
genr inc2=inc(-2);	- Lags income by 2
genr dcons=cons-cons1;	- First difference of consumption
set pi=3.1415926;	- Generates a single number called pi
set m1=@mean(1);	- Assigns the mean of cons to a value *m1*
set v12=@corr(1,2);	- Assigns the correlation between variables 1 and 2 to *v*12

To generate random numbers with a specific distribution, one has to use the *random* option.

random (cauchy, expon, laplace, poisson, t, uniform, seedin=n1, stdev=n2, df=n3, mean=n4, vcov=vn1, vmean=vn2) x1;

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

The first 6 options specify the type of distribution you would like to use, of which you specify just one.

seedin=n1	-Specifies the first seed number *n1*
stdev=n2	-Specifies the standard deviation *n2* for the normal option
df=n3	-Specifies the degrees of freedom as *n3* for the t-distribution
mean=n4	-Specifies the mean of the normal or t-distributions as *n4*
vcov=vn1	-Specifies the covariance matrix for multivariate normals as *vn1*
vmean=vn2	-Specifies the mean vector for multivariate normals as *vn2*

Finally, *x1*, denotes the series into which the random variable is to be written, if no distribution is specified TSP will use a normal random variate.

You can also work out the critical values corresponding to specified probability values for a series of different distributions, or given the critical values calculate the corresponding probability values.

cdf(bivnorm,chisq, dickeyf, f, normal, t, options) x1;

The first 6 options specify alternate distributions available. *Dickeyf* corresponds to Table 1 of MacKinnon (1990) for Dickey-Fuller statistics, *bivnorm* is the bivariate normal.

df=n1	- Degrees of freedom for chi-squared distribution=*n1*
df1=n2	- Numerator degrees of freedom for F-distribution=*n2*
df2=n3	- Denominator degrees of freedom for F-distribution=*n3*
rho=n4	- Correlation coefficient for bivariate normal *n4*
nvar=n5	- Number of variables used in Engle and Granger cointegration *n5*
constant	- Specifies whether constant exists for *dickeyf*
trend	- Specifies whether trend exists for *dickeyf*
inverse	- Specify a significance level and output is critical values

For example,

```
read px;
0.1 0.05 0.01;
cdf(inv,norm) px crit;
print px crit;
```

4. <u>OLS</u>

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

olsq(options) cons c inc infl;

This runs an OLS regression in which cons is the dependent variable and inc, infl and an intercept (which is denoted c) are the explanatory variables. The options control the amount of output results you get and enable you to save parameter estimates. Examples of the options are:

robustse	- Uses White's robust standard errors.

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

@res	- Residuals of the regression
@fit	- Fitted values of the regression
@rsq	- R-squared
@coef	- Coefficient estimates
@vcov	- Covariance matrix of estimated coefficients
@nob	- Number of observations
@s2	- Variance of residuals

You could retrieve these values (variables) by typing

```
olsq cons c inc infl;
set sigma2=@s2;
?this saves the estimated variance of the disturbance term in sigma2
set b1=@coef(1);
?This saves the first coefficient of the regression into the value b1
```

To see how you save matrixes see the section on matrix algebra in TSP.

5. ARIMA Modelling

In general this process is split into 3 stages:

i. Identification

bjident(options) cons;

The options available include

plotac	- Plots autocorrelation coefficients

nlag=n1	- The number of lags on the autocorrelation function is n1
plotpac	- Plots partial autocorrelation coefficients
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	- Sets the number of periods in a cycle to *n5* (if not specified frequency of data)

ii <u>Estimation</u>

bjest(options) cons start[phi(1) 0.2 theta(1) 0.3];

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, in the order *nar [phi(j)], nma [theta(k)], nsar [gamma(l)], nsma [delta(m)]* and the *constant* last, where j, k, l, m are the lag length on the parameters. In
our	example nar=1, nma=1, nsar=0, nsma=0

Having estimated the model it is possible to retrieve many of the results in the same way as we did for *olsq*

@res	- Saves the residuals
@coef	- Saves the coefficients

iii <u>Forecasting</u>

bjfrcst(options) cons;

retrieve	- Uses the model the coefficients from bjest
orgbeg=n6	- Beginning date for the forecast to start *n6*
orgend=n7	- End date for the forecasting *n7*
conbound=n8	- Specifies the probability confidence bounds as *n8*
nhoriz=n9	- Number of periods ahead to forecast *n9*

If you do not use *retrieve* you must specify the coefficients estimates after the series you want to forecast over, using the form of expression used in the *bjest* command.

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 TSP code for this equation would be:

frml eq1 y=alpha0+alpha1(x1**alpha2)+(alpha2*x2+alpha3*x3)/(1-alpha4);*
param alpha0 0 alpha1 0.12 alpha2 0.5 alpha3 0.14 alpha4 0.4;
lsq (options) eq1;

covu=vn3	- Residual covariance matrix is *vn3*
maxit=n1	- Maximum number of iterations *n1*
tol=n2	- Convergence criteria on parameters is *n2*
hetro	- Heteroscedastic consistent standard errors
step=	- Specifies the step length this can use one of five options *bard* or *bardb* or *cea* or *ceab* or *golden*
maxsqz=n3	- Specifies max number of squeezes in stepsize search as *n3*
hcov=	- Specifies the method for calculating the parameter covariance matrix as one of six options these are *b* or *n* or *g* or *d* or *w* or *r* or some combination of these, for example, *bnw* *b*=bhhh *n*=newton *g*=gauss *d*=dfp *w*= Eicker-White *r*=robust

The parameter covariance matrix is stored under the name *@vcov*, however, if a second covariance matrix is requested this is stored under the appropriate name, for example *@bhhh*.

The second to last line of the set of commands for nonlinear least squares provides the initial values for the coefficient estimates. It is again possible to retrieve a series of results from this procedure using similar commands to those for *olsq*.

7. Two Stage Least Squares

The format for this instruction is similar to that for OLS

```
genr inc1=inc(-1);
2sls(options inst=(c,infl,infl1,cons1,inc1)) cons c inc infl cons1;
```

For the *2sls* command the first variable cons is the dependent variable and inc, infl and cons1 are the right hand side explanatory variables along with the intercept. The variables in parentheses are the exogenous variables used to instrument out inc in this example. The options are again similar to those of OLS and similar variables can be saved as vectors or matrices.

8. <u>3SLS and FIML</u>

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 *frml* commands. In this section we will give examples of each of these

8.1 <u>NL2SLS</u>

To specify 2SLS use the *lsq* command and 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* should yield the same solutions). The options are denoted in the *lsq* section.

```
frml coneq cons=beta0+beta1*inc+beta2*infl+beta3*cons1;
param beta0 0.0 beta1 1.0 beta2 1.0 beta3 1.0;
lsq(options, inst=(c,infl,infl1,con1,inc1)) coneq;
```

8.2 <u>Nonlinear Systems</u>

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

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

The TSP code for this would be:

```
frml eq1 z=b0+b1*x1+b2*x2;
frml eq2 y=d0+d1*(z-b0-b1*x1-b2*x2)+d2*(b0+b1*lag(x1,1)+b2*lag(x2,1));
param b0 0.04 b1 0.1 b2 0.3 d0 -0.3 d1 0.2 d2 -0.1;
fiml(options endog=(z,y)) eq1 eq2;
```

The options are similar to those for *lsq*.

8.3 <u>3SLS</u>

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

```
frml eq1 y1=b0+b1*x1+b2*x2+b3*y2;
frml eq2 y2=d0+d1*x1+d2*x1(-1)+d3*y1;
frml eq3 y3=f0+f1*x3+f2*y2+f3*x2(-1);
param 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;
3sls(inst=(c, x1,x1(-1),x2,x2(-1),x3)) eq1 eq2 eq3;
```

<u>Note</u> For expressions continuing over more than one line you do not need to tell TSP anything as it treats one line as being up to the semi-colon.

9. <u>Matrices</u>

To read a matrix into TSP you need the following instructions

```
load(nrow=3,ncol=4) a;
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

```
mmake z cons inc infl;
```

This copies the 9 observations on consumption, income and inflation into a matrix z which has dimension (9x3). Alternatively,

mform(type=general,nrow=3 ncol=3) cons x;
? This forms a matrix x as a (3x3) matrix from the 9 observations on cons as

$$X = \begin{vmatrix} 99.2 & 99.5 & 99.6 \\ 100.2 & 100.3 & 102.0 \\ 103.1 & 104.2 & 105.6 \end{vmatrix}$$

mform q=@vcov;
?Forms a matrix q as the covariance matrix from a previous regression. Similarly
copy @vcov q;
? The reverse command is
unmake z cons inc infl;

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

-	Subtraction
*	Multiplication
+	Addition
'	Transpose
#	Kronecker Multiplication
%	Hadamard product
"	Inverse

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.
eigval(matrix)	- Forms the eigenvalues of a matrix
eigvec(matrix)	- Forms the eigenvectors of a matrix
ident(ndim)	- An identity matrix with ndim rows and columns
tr(matrix)	- Forms the trace of the matrix
rank(matrix)	- Calculates the rank of a matrix
sym(matrix)	- Creates a symmetric matrix from a square matrix
gen(matrix)	-Creates a general matrix fro a symmetric diagonal matrix

For example,

? Takes the inverse of the matrix z'z
matrix a=(z'z)";
? Forms a Choleski decomposition of the matrix a
matrix c=chol(a);

10. Simulation

Having estimated a system of equations if you are interested in simulating the system then TSP is the best program to use. The basic command is

```
frml eq1 y1=b0+b1*x1+b2*x2+b3*y2;
frml eq2 y2=d0+d1*x1+d2*lag(x1,1)+d3*y1;
frml eq3 y3=f0+f1*x3+f2*y2+f3*lag(x2,1);
param 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;
smpl 2 9;
3sls(inst=(c,x1,x2,x3)) eq1 eq2 eq3;
smpl 6 9;
siml(options tag=s dynamic endog=(y1 y2 y3)) eq1 eq2 eq3;
```

This will do a dynamic simulation of a system of equations and store the simulated results in variables tagged with the letter s, that is, *y1s, y2s, y3s*, which can be used later on.

10. Do Loops

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

```
? Sets a scalars for the no. of obs, no. of trials and the random seed.
set nobs=100;
set trials=1000;
set ranseed=100;
? This sets the sample size
smpl 1 nobs;
? Defines a variable x as a uniform distribution using the sees as 100
random(seedin=100,uniform) x;
genr x=x*2;
? Calculates summary statistics on the variable x.
msd x;
? Declares the dimension on the two matrices
mform(nrow=2,ncol=1) beta=0;
mform(nrow=2,type=sym) betap=0;
regopt(noprint) @coef;
? The start of the do loop.
do tr=1 trials;
? Defines u~N(0,1)
random u;
? Defines u~Uniform[0,1]
?random(uniform) u;
```

? Standardises u so that E(u)=0, V(u)=1.
*?genr u=(u-0.5)*sqrt(12);*
? Generates $y = \alpha + \beta x + u$, $\alpha = 1, \beta = 2$.
*genr y=1+2*x+u;*
? Undertakes OLS estimation, although no results are presented.
olsq(silent) y c x;
? Sums the coefficients (and squares) from the 1000 replications as beta and betap
mat beta=beta+@coef;
mat betap=betap+@coef@coef';*
enddo;
? Mean and variance of the coefficients
mat beta=beta/trials;
*mat betap=betap/trials - beta*beta';*
regopt;
? Produces mean and se of the coefficients.
tstats(names=@rnms) beta betap;
end;
stop;

Finally we finish by introducing an entire program that was written to simulate a simple system

Error Correlation Model for imports.

?reading in the data from a file *import.dat*
freq q;
? The sample size goes out beyond the todays date to enable ex-anti forecasts. Zero vales exist
? from 1992.1-1995.4
smpl 69:3 95:4;
read(file='import.dat' format=free) share lnrp lncu lnz lnspec8;
load;
smpl 69:4 95:4;
? Transforming the data into changes
genr chshare=share-share(-1);
genr chlnrp=lnrp-lnrp(-1);
genr chlncu=lncu-lncu(-1);
genr chlnz=lnz-lnz(-1);
genr chlnspec=lnspec8-lnspec8(-1);
genr z=exp(lnz);
*genr m=share*z;*
genr mmf=(z-m)/180.57;
? Dummy variables for the civil service strike
genr d74q1=0;
genr d74q2=0;
smpl 74:1 74:1;
genr d74q1=1;
smpl 74:2 74:2;
genr d74q2=1;
? Setting up the equations for simulation. Each is in an ECM format

```
frml shareeq   chshare=+(a11*chshare(-1))+(a12*chlnrp(-1))
+(a13*chlncu(-1))+(a14*chlnz(-1))+(a15*chlnspec(-1))
+(a16*d74q1)+(a17*d74q2)
+(a18*(-(122.298*share(-2))-(27.56984*lnrp(-2))+
(3.419760*lncu(-2))+(37.97275*lnspec8(-2))))
+a19;

frml lnrpeq    chlnrp=(a21*chshare(-1))+(a22*chlnrp(-1))
+(a23*chlncu(-1))+(a24*chlnz(-1))+(a25*chlnspec(-1))
+(a28*(-(122.298*share(-2))-(27.56984*lnrp(-2))+
(3.419760*lncu(-2))+(37.97275*lnspec8(-2))))
+(a26*d74q1)+(a27*d74q2)+a29;

frml lncueq    chlncu=(a31*chshare(-1))+(a32*chlnrp(-1))
+(a33*chlncu(-1))+(a34*chlnz(-1))+(a35*chlnspec(-1))
+(a36*d74q1)+(a37*d74q2)+a39;

frml lnzeq     chlnz=(a41*chshare(-1))+(a42*chlnrp(-1))
(a43*chlncu(-1))+(a44*chlnz(-1))+(a45*chlnspec(-1))
+(a48*(-(122.298*share(-2))-(27.56984*lnrp(-2))+
(3.419760*lncu(-2))+(37.97275*lnspec8(-2))))
+(a46*d74q1)+(a47*d74q2)+a49;

frml lnspeceq chlnspec=(a51*chshare(-1))+(a52*chlnrp(-1))
+(a53*chlncu(-1))+(a54*chlnz(-1))+(a55*chlnspec(-1))
+(a56*d74q1)+(a57*d74q2)+a59;

frml zeq z=((180.57*mmf*share)/(1-share))+(180.57*mmf);

?Identities to establish link between differences and level variables
ident id1 share=chshare+share(-1);
ident id2 lnrp=chlnrp+lnrp(-1);
ident id3 lncu=chlncu+lncu(-1);
ident id4 lnz=chlnz+lnz(-1);
ident id5 lnspec8=chlnspec+lnspec8(-1);
ident id6 lnz=log(z);
ident id7 chlnz=lnz-lnz(-1);
ident id8 share=m/exp(lnz);

? Parameter estimates for each equation
param a11 -0.43916 a12 -0.066254 a13 0.021537 a14 -0.038422
a15 -0.027328 a16 0.010208 a17 0.015751 a18 0.0029057 a19 -0.064920;

param a21 0.16512 a22 0.16188 a23 0.056162 a24 -0.046466 a25 0.21892
a26 0.033172 a27 0.052307 a28 0.0067978 a29 -0.16601;

param a31 -0.025365 a32 1.1129 a33 -0.0084039 a34 1.8386 a35 -4.4398
a36 -0.68352 a37 0.65138 a38 0.000 a39 0.017896;
```

```
param a41 -0.64799 a42 0.047760 a43 0.074871 a44 -0.064702
a45 -0.46849 a46 -0.025607 a47 0.13081 a48 0.010480 a49 -0.23815;

param a51 0.039135 a52 -0.023531 a53 0.0035479 a54 -0.021786
a55 0.76469 a56 0.0081825 a57 0.0024403 a58 0.000 a59 0.0014916;

?Setting the sample period for simulation
smpl 80:1 88:4;
siml (tag=s endog=(chshare chlnrp chlncu chlnz chlnspec share lnrp lncu lnz
lnspec8) noprint, noprnsim, dynamic) shareeq lnrpeq lncueq lnzeq lnspeceq
id1 id2 id3 id4 id5;
print shares lnrps lncus lnzs lnspec8s;
write(file='newsim.dat',format='(5f10.5)') shares lnrps lncus lnzs
lnspec8s;

stop;
end;
```