Chapter 6: Single equation regressions

GROCER contains most of the usual methods for single equation regression, from the inescapable ordinary least squares to the less used Theil-Goldberger mixed estimator through Cochrane-Orcutt, instrumental variables, non linear least squares,… As explained in chapter 4, these methods can be applied to a great variety of objects, among which timeseries. In that last case, the user will often want to choose her estimation bounds: this is done through the GROCER command bounds, which is presented in part 1 of this chapter. Ordinary least squares are the object of part 2. Part 3 of this chapter deals with the estimation of models with autocorrelated residuals, part 4 with instrumental variables estimations, part 5 with non linear least squares, part 6 with limited dependent variables, part 7 with various other single variable regression methods and part 8 with forecasting. Part 9 is devoted to the precise description of the functions and their specifications.

1. Setting estimation bounds

Performing regressions with timeseries often involves the choice of a time range over which endogenous and exogenous variables are considered. This is done in GROCER by the function bounds. This function takes an even number of string arguments, each pair of arguments corresponding to a time interval. The simplest call to the function bounds is then:

```
-->bounds(date1,date2)
```

For instance, in the demo hendryericsson, you will find the following call to bounds:

```
-->bounds('1964q3','1989q2')
```

It can sometimes be useful to exclude a time period from the estimation bounds. For instance, if you want to exclude the word wars from a yearly estimation over the 20th century, you can type[1]:

```
-->bounds('1901a','1913a', '1919a','1939a', '1946a','2000a')
```

Note that the function bounds creates a variable called grocer_boundsvar in the calling environment. The bounds will then apply to all following regressions performed in the environment, and only in that environment, until a new call to function bounds is made or until the command clear(grocer_boundsvar) is executed in the environment. For instance, a unique call to function bounds in a function or a script will apply all over the function and its subroutines, if any, but not to the current working space.

2. Ordinary least squares (*)

There are no less than 4 functions performing ordinary least squares in GROCER: ols0, ols1, ols2 and ols. The most versatile one is ols, this is the one an applied econometrician should usually use. The 3 other functions are useful mainly for programming. For instance ols0 is used by lad, robust … or ols1 ! Ols1 is used by kpss (see chapter 7). Ols2 is used by var1 … and by ols !

The usual call to ols is the following one: ols(endo,exo1,…exon). The type of endo can be: a ts, a (nx1) vector or a string, equal to the name of a ts or a vector between quotes. The type of exoi can be the

[1] Note that this method is not strictly equivalent to that consisting in adding one dummy for each world war year: the coefficients will be the same, but others statistics (such as the Student T) will be different.

same or even a (nxp) matrix or a list of such elements. If ols applies to ts, then it is highly recommended, although not compulsory, to set bounds before.

You can mix different types of data in ols: vectors and ts; strings and vectors… This is practical for "quick and dirty" uses, but it should be better to limit yourself to 3 uses:

1) ols('ts0','ts1',…,'tsn') where ts0,ts1,… tsn are obviously ts and are entered between quotes in order to record their names.

2) ols('y','x1',…,'xn') where y,x1,…,xn are all (nx1) vectors and are entered between quotes in order to record their names.

3) ols('y','x') where y si a (nx1) vector and x a (nxp) matrix (also between quotes in order to record its name).

There are two specific arguments which can be entered in the list of exogenous arguments. The first one is 'cte' if you want to add a constant term to your regression (whatever type your other exogenous variables have). The second one is 'noprint' if you do not want to display on screen the results of the regression. In that case, you should record your result by typing:

```
-->myresults=ols(…);
```

The results tlist myresults will then contain all your regression results. You can display on screen the regression results later, by typing prtres(myresults) or prtuniv(myresults) (see chapter 21). You can also retrieve one of these results by typing myresults('name') where name is one of myresults fields. Note lastly that, if you do not end your command by ';', the content of the tlist will not be displayed on screen, but instead the message:

```
myresults =


ols estimation results
```

As with ts, this feature results from scilab capability of replacing the display of a whole tlist by what the user wants by the mean of a function %type_of_tlist_p where type_of_tlist is here 'results': the function, which can be found in sublibrary 'basic', is then %results_p. If you prefer to display the whole tlist, then destroy the corresponding function in the zip file Grocer_v1.2.zip and unzip it again in your Scilab library. If you want another display, then change accordingly the text of function %results_p in the zip file and unzip it again in your Scilab library.

Now:

```
-->b=myresults('beta')
```

will store in b the estimated parameter of your regression (and it will display it on screen, since you have not ended the instruction by ;). For a list of all arguments, see description in part 9 of this chapter.

If the user has not given the argument 'noprint', the results displayed on the screen will look like this (here are the results from equation # (6) in D.F Hendry et N.R Ericsson (1991): "Modeling the demand for narrow money in the United Kingdom and the United States", European Economic Review, p833-886; see hendryericsson() for GROCER implementation; comments are added for this chapter and do not appear on screen):

```
ols estimation results for dependent variable: del(lm1-lp)
// del(lm1-lp) is the name of the endogenous variable
estimation period: 1964q3-1989q2
// the estimation period is reported if there are ts in the regression, whether the user has
// given explicit bounds or not
number of observations: 100
```

```
number of variables: 5
R² =  .7616185      ajusted R² = .7515814
Overall F test: F(4,95) = 75.880204       p-value = 0
```

// Overall F test is the test that all the non constant variables in the
// regression are 0

```
standard error of the regression:  .0131293
sum of squared residuals:  .0163761
DW(0) =2.1774376
```

// DW stands for Durbin and Watson

```
Belsley, Kuh, Welsch Condition index: 114

variable                  coeff      t-statistic p value
del(lp)                    -.6870384 -5.4783422  3.509E-07
del(lagts(1,lm1-lp-ly))    -.1746071 -3.0101342   .0033444
rnet                       -.6296264 -10.46405   0
lagts(1,lm1-lp-ly)         -.0928556 -10.873398  0
cte                        .0234367  5.818553    7.987E-08
```

// for each variable (whose name has been here kept by the program), the estimated
// coefficient, its Student t and the significance level under which it will be considered as
// 0 is given.

```
      *
    *   *
```

// the 3 stars are added at the end of the printings to add clarity. If this device does not
// appeal to you, just change it in function printsep (see chapter 19).

All functions call, directly or indirectly, the function invxpx. This function uses the QR decomposition of a matrix x to calculate the inverse of matrix (x'x): this method is the one which produces the most precise estimates (see James LeSage's book, chapter 1). Since the 3.0 Scilab 3.0, there is an option in QR that allows to calculate only the k (where k is the number of columns of the matrix x), first columns of the matrix q and the k first rows of r: this device allows the qr decomposition to be almost as fast as the standard inversion.

Note that you can use alternatively the function hwhite (see for an example hwhite_d): this function has the same calling syntax and does the same thing as ols, except that the variance-covariance matrix is calculated using White's consistent covariance matrix estimator[2]. There is a companion function, called nwest, that estimates the variance-covariance matrix by the Newey and West formula.

(°)Constrained ordinary least squares are performed by function ols_cons: you can with this function estimate an ordinary least squares estimation Y= X b +U with the constraint that R b= r. The difference with function ols is that you have to enter in the list of arguments after y an argument 'R=m1' and an argument 'r=m2' where m1 and m2 are respectively a matrix or a vector. The results are stored in a tlist which contains the same fields as with ols, and fields 'R' and 'r'.

As an example, if you want to impose the constraints that b2=-b3 and b2=-b4 in the regression of 'delts(lm1-lp)' on 'delts(lp)', 'delts(lagts(1,lm1))', 'delts(lagts(1,lp))', 'delts(lagts(1,ly))', 'rnet','lagts(1,lm1-lp-ly)' and the 'cte', then run:

```
-->rols=ols_cons('delts(lm1-lp)','delts(lp)','delts(lagts(1,lm1))',...
-->'delts(lagts(1,lp))', 'delts(lagts(1,ly))','rnet', 'lagts(1,lm1-lp-ly)','cte',...
```

[2] H.E White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity, *Econometrica*, vol 48, n°4, pp. 817-838.

```
-->'R=[0 1 1 0 0 0 0 ; 0 1 0 1 0 0 0]','r=[0;0]')
```

You can perform an ordinary least squares regression on a rolling or expanding (also called recursive) window with function rolreg: in the first case the size of the estimation window remains the same and it is equal to the size given by the bounds of the regression; in the second case, data are added at the end of the sample, which expands for each regression. Such a regression can be made as an ols regression, with some differences. Fisst, you must add to the list of arguments the option 'dates=d' where d is a vector of dates between double quotes corresponding to the end date of the first regression and of the end date of the last regression (case of ts) or a vector of observations (case of vectors or matrices).

There are also numerous new options:

- you can perform expanding regression (default) or rolling regression; in that case you have to enter the option 'simu="roll"';
- you can use the Newey-West method of estimation of the variances instead of the ols one; in that case, you have to enter the option 'meth=nwest' and you may enter the length of the Barlett window used by the Newey-West estimator (option 'wind=i' where i the length of the window; the default value is floor(5*nobs^0.25) with nobs the number of observations);
- you can make h-step forecats with each successive regression; in that case you have to enter the option 'hstep=h' with h the chosen forecasting horizon; in that case, you can also choose to keep all 1 to h-step forecasts (default) or only the h-step forecasts, with the option 'mul=0'; you can also choose to make forecasts beyond the last date of the endogenous variable, by entering the option 'xsmpl=1' (this option works with ts only);

Results are stored in a tlist, which contains the main results for the rolling estimations: the estimated beta (field 'beta'), the associated tstats and p-values (fields 'tstat' and 'pvalue'), the corresponding R-squared and adjusted R-square (fields 'rsqr' and 'rbar'), the matrix of forecasts (field 'yfor') if any, forecasts at a given horizon i being stored in row i. If there are ts, bounds are stored and the matrix of forecasts is transformed into the corresponding time series (the field 'yfor_h1' is for instance the forecast at a 1 period horizon).

As an example, take the data on the French manufacturing production (y), the past trend (ypast) of production and future trend (yfut) of production in the business survey that you can fin in the database industrial.dat in your Scilab library macros/grocer/db:

```
-->load('SCI\macros\grocer\db\industrial.dat')
```

Then estimate recursive regressions (option 'simu="recu"') of delts(log(y)) on ypast, delts(yfut) and a constant with ending dates starting from 1991q1 to 2002q4 (option 'dates=["1991q1" "2002q4"]') at a 4 quarter horizon (option 'hstep=4') while keeping only the 4-step forecast (opyion 'mul=0'). Note that the stored forecast will go from 1992q1 until 2003q4.

```
-->ri     =     rolreg('delts(log(y))','ypast','delts(yfut)','const','dates=[''1992q1''
''2003q4'']','hstep=4','mul=0','simu=''recu''')
```

And if you want to recover the forecast at the 4th quarter horizon:

```
-->fct = ri('yfor_h4') ;
```

If you want to recover the coefficients:

```
-->fct = ri('beta') ;
```

(°°)

3. Estimation of linear models with autocorrelated residuals.

GROCER provides two popular methods for the estimation of linear models with autocorrelated residuals. The first one is Cochrane and Orcutt's method, implemented in function olsc: the calling sequence is exactly the same as with ols and so needs no more presentation. The second method, implemented in function olsar1, is the maximum likehood method: the calling sequence is also the same as ols. Note that this function uses a very convenient device of Scilab function optim, that is the possibility to impose bound constraints on an estimated parameter: since an autocorrelation coefficient is in absolute value bounded by 1, it is desirable that the estimated coefficient verifies this constraint. This is done in the call to optim by the keyword 'b', followed by the vectors of lower and upper constraints. Note also that, since only the autocorrelation coefficient is constrained, the bounds on the other parameters are set to -∞ and +∞, which are named -%inf and %inf in scilab. The call to optim in function olsar1 is then the following:

```
binf=[-1+%eps ; -%inf*ones(nvar,1)]
[like,beta] = optim(grocer_cost,'b',binf,-binf,parm);
```

where grocer_cost is the function to maximise and parm the starting values for all the parameters which are estimated.

The presentation of the results is almost exactly the same as with ols, except that the estimated first order autocorrelation of residuals is also given. Here is for instance the result of the estimation of a money demand function very similar to the one estimated before by ols:

```
-->load('grocer/bdexamples/bdhenderic.dat') ;
// set the estimation bounds
-->bounds('1964q3','1989q2')

// show the results of cochrane-Orcutt estimation
-->rolsc=olsc('del(lm1-lp)','del(lp)','rnet','lagts(1,lm1-lp-ly)','cte')

Cochrane-Orcutt estimation results for dependent variable: del(lm1-lp)
estimation period: 1964q3-1989q2
number of observations: 99
number of variables: 4
R² =  .8236925      ajusted R² = .8181249
Overall F test: F(3,95) = 147.94377       p-value = 0
standard error of the regression:  .0131803
sum of squared residuals:  .0165033
DW(0) =2.0153408
Belsley, Kuh, Welsch Condition index: 119

variable           coeff      t-statistic p value
del(lp)             -.5843923 -5.7310684  1.173E-07
rnet                -.5562022 -12.66214   0
lagts(1,lm1-lp-ly)  -.0807630 -12.966793  0
cte                 .0218459  6.9046054   5.649E-10


                        *
                    *     *

estimated rho:  -.2872218
(Student t: -2.9684277)

// then show the result of olsar1
-->rolsar1=olsar1('del(lm1-lp)','del(lp)','rnet','lagts(1,lm1-lp-ly)','cte')

ar(1) maximum likelihood estimation results for dependent variable: del(lm1-lp)
```

```
estimation period: 1964q3-1989q2
number of observations: 100
number of variables: 4
standard error of the regression:  .0131120
sum of squared residuals:  .0165048
DW(0) =2.0240002
Belsley, Kuh, Welsch Condition index: 119

variable           coeff      t-statistic p value
del(lp)             -.5843882 -5.7608732  1.007E-07
rnet                -.5561977 -12.727983  0
lagts(1,lm1-lp-ly)  -.0807022 -13.099482  0
cte                  .0218866  7.0234996   3.105E-10


                        *
                      *     *


rho     -.2872249
t-stat -2.9684632

                        *
                      *     *
```

4. Instrumental variables estimation.

Instrumental variables can be performed with two functions: iv and iv1. As with ols, the non numbered function is the most versatile one and is very flexible while the numbered function looks much more to the textbook. Because there are 4 types of object, namely the endogenous variable, the exogenous variables uncorrelated with the residuals, the exogenous variables correlated with the residuals, and the instruments, the syntax is however a little more constrained than with ols. To be more precise, the call to iv must have the following form: iv(endoy, 'exo=[var1;var2;...,vark]', 'endo=[var1;var2;...,varl]', 'ivar=[var1;var2;...,varm]'), where exo stands for exogenous variables uncorrelated with the residuals, endo for exogenous variables correlated with the residuals and ivar for instruments. The keywords exo, endo and ivar are compulsory, but the order in which they are given does not matter. For endoy, the choices are exactly the same as with ols. Each of the names vari can represent, a usual, a ts, a (nx1) vector or a (nxk) matrix. Note lastly that, as with ols, you can switch off the printings of the results with the argument 'noprint', given anywhere after the first argument…

Compared to the results of function ols, the results produced by iv present a few differences: first, the list of variables supposed to be correlated with the residuals are presented, along with the list of instruments; second, beside the traditional R^2 and $\bar{R}^2$, the program gives the generalized R^2, calculated by replacing the instrumented variables by the results of their regression on the instruments: this is the correct way to calculate it[3].

As an example, here is a call to iv and the corresponding results, taken from function iv_d:

```
-->x1 = rand(200,1,'n');
-->x2 = rand(200,1,'n');
-->iota = ones(200,1);
-->1 = zeros(200,1);
-->y2 = zeros(200,1);
```

[3] see Pesaran and Smith (1994): "A Generalized R^2 Criterion for Regression Models Using Instrumental Variables", Econometrica, vol.62, pp. 705-710.

```
-->evec = rand(200,1,'n');
```
// create simultaneously determined variables y1,y2
```
-->y1 = iota+x1+evec(i,1);
-->y2 = iota+y1+x2+evec(i,1);
```
// do iv regression
```
result2 = iv('y2','endo=y1','exo=iota;x2','iv=iota;x1;x2');

-->iv estimation results for dependent variable: y2

endogenous variable(s) in this equation is(are):
y1
```
// list of independent variables correlated with the residuals; here only y1 belongs to this category

```
instruments for this equation are:
iota,x1,x2
```
// list of instruments

```
number of observations: 200
number of variables: 3
R² =  .8741958      ajusted R² = .8729186
generalized R² =  .4167750 (*)
standard error of the regression:  .9091582
sum of squared residuals: 162.83401
DW(0) =2.1448555
Belsley, Kuh, Welsch Condition index: 2

variable coeff     t-statistic p value
y1       1.0711203 18.888354   0
iota      .9867467 11.765988   0
x2       1.1021831 17.026645   0

* warning, R² and ajusted R² are shown here to conform to commun practice, but only the
generalized R² should be interpreted
```

*

* *

The function iv1 is much more restricted (and therefore more efficient…). The call to iv1 is the following: iv1(y,endo,exo,ivar). Here y is a (nx1) vector, endo is the (nxk) matrix of dependent variables uncorrelated with the residuals, exo the (nxl) matrix of exogenous variables correlated with the residuals, and ivar the (nxp) matrix of instruments. The order of the sequence is here compulsory. This function should be used only for programming and it is used for instance by the functions twosls and, naturally, iv.

5. Non linear least squares.

Non linear least squares are programmed in GROCER in a function called -without originality!- nls. The calling syntax is still different from the ones of ols or iv. It involves the entry of the equation that the user wants to estimate, which must have the following form 'lhs = rhs(x1,…xn,a1,…,an)' where the xi are exogenous variables, constants and the ai coefficients. The simplest call to nls is then the following one: nls('lhs = rhs(x1,…xn,a1,…,an)'). Behind this simplicity hide the following conventions. First, the coefficients are named from a1 to an, starting from a1 and going to an without discontinuity. If, for instance, you have 4 coefficients in your regression, then they must be called a1,a2,a3,a4. Second, nls involves a maximisation program (optim or maxlik, see chapter 5) and therefore needs starting values: these are drawn from a normal law. These conventions can be overruled: if you want to give other names to your coefficients, then you will have to enter the option 'coef=[myname1;myname2;…;mynamen]'; if you want to provide

starting values to the maximisation program, then you will have to enter the option 'init=[v1;…;vn]' where v1,…,vn are the starting values. There must be exactly the same number of starting values as the number of parameters to estimate.

As regard the estimation, you have 2 choices: 'optim' or 'maxlik'. The default one is maxlik and you can give any option to maxlik that you want as with maxlik (see chapter 5 for more precision).

Lastly, the option 'noprint' is also available.

The output is very similar to the one provided by ols. The main differences are that the name of the endogenous variable is replaced by the text of the estimated equation and that the names of the exogenous variables are replaced by the names of the coefficients. Here is an example:

```
-->load('SCI/macros/grocer/db/smpt.dat');                    r=nls('smptr/lagts(smptr)-
1=a+b*(lagts(smptr)/lagts(2,smptr)-1-a)', 'coef=a;b','init=[0;0.5]')

   CONVERGENCE CRITERIA MET: Change in Objective Function



nls estimation results for equation:
smptr/lagts(smptr)-1=a+b*(lagts(smptr)/lagts(2,smptr)-1-a)
estimation period: 1980a-1999a
number of observations: 20
number of variables: 2
standard error of the regression:  .0070050
sum of squared residuals:  .0008833
DW(0) =1.5300827

variable coeff     t-statistic p value
a         .0074947 2.6870741    .0150592
b         .4382498 2.096523     .0504327


                         *
                      *     *
```

6. Limited dependent variables.

GROCER contains also the basic estimation functions for limited dependent variables. The first two are the logit and probit models, which apply to endogenous variables that take only a 0 or 1 value. Each model assumes that the dependency between the endogenous variable and the exogenous ones takes the following form:

$\text{Prob}(y=1)=\Phi(X,\beta)$

where Φ is the cumulative distribution function of respectively a logistic law and a normal one.

The third model is the tobit model, where the endogenous variable can take either a 0 value with positive probability or a continuum of positive values. The underlying model is then:

$y_i^* = xi*b+ui$

$y_i = 0$ if $y_i^* \leq 0$

$y_i = y_i^*$ if $y_i^* > 0$

The call to logit and probit is very similar to the call to ols. The only difference is that the user can set values others that the default ones for the authorized number of iterations by entering 'maxit=myvalue' where myvalue is the desired maximum number of iterations or the convergence criterion by entering 'tol=myvalue' where myvalue is the desired convergence criterion: the two functions use built-in numerical optimization algorithms, which need exit conditions. The call to tobit is also similar to the call to ols. The program uses maxlik, so all options to maxlik can be entered. There is two other options: if the truncation condition above applies to positive instead of negative values (that is if $y_i = 0$ if y_i *$\geq$0), then the user has to enter the option 'trunc=right'; if the user wants to use another truncation value than 0, she has to enter the option 'vtrunc=value'.

The output is also somehow different than the one of ols: numerous statictics, such as the standard error of the regression or the Durbin and Watson have not much sense and are not given. The goodness of fit is provided by the mean of two statistics: MacFadden R^2 and Estrella R^2. The log-likelihood of the model is also provided.

As an example, here is the result of the probit estimation taken from function probit_d:

```
probit estimation results for dependent variable: grade
number of observations: 32
number of variables: 4
McFadden R² =  .3774780
Estrella R² =  .9984098
log-likelihood = -12.818804
LR-ratio = 2*(Lu-Lr) = 15.545851
LR p-value  =  .0014049

variable coeff      t-statistic p value
cte      -7.4523196 -2.9311311   .0066560
psi      1.4263323  2.3970445    .0234450
tuce      .0517289   .6166263    .5424628
gpa      1.62581    2.3430625    .0264591
```

*

* *

7. Various other single variable methods

GROCER contains numerous other single variable methods, mostly translated and adapted from James LeSage 'regress' library. These methods are the following ones:

- Least absolute deviation regression: function lad.
- Ordinary least squares with Student-distributed errors: function olst.
- Ridge regression: function ridge.
- Robust regression: function robust.
- Theil-Goldbeger mixed estimator: function theil.

For all these functions, the general principles applying to ols apply: these functions always involve as an argument the name of the endogenous variable and the following arguments are either exogenous variables or the string 'noprint'; the authorized types for these variables are the same as with ols.

There are some variants however (see part 9 for a more precise dexription). Function ridge allows also another argument in the list of arguments after the endogenous one: this argument is 'theta=value' where value is the value given to the parameter used as a perturbation to the matrix (X'X). This argument is

optional: if it is not given, the default value is the one recommended by Hoerl and Kennard. Function robust has 2 compulsory arguments before the name of the endogenous variable: the name of the robust method and the corresponding weighting parameter. Lastly, function theil has 3 compulsory arguments before the name of the endogenous variable: the vector of prior mean values, the matrix transforming the vector of parameters into the prior value and the assumed variance of the prior mean values.

8. Forecasting

Forecasting is for the moment very limited in GROCER[4]. As regards single equations, the only available forecasting tool is the function statfore, which provides static forecasts, the word static meaning that all lags of the endogenous variable that appear in the list of exogenous variables are taken at their historical value and not at their forecasted values, if any. This function applies only to linear regressions: it cannot be used with results stemming from a nls estimation.

The function statfore always applies to a results tlist (let us call it res) from which it extracts the estimated coefficients. It can be called in two ways according as the results come for a regression with ts (in that case, res('prests') should be true and the names of the variables should have been given in the estimation function between quotes, so that they can be retrieved by statfore) or without ts.

In the first case, statfore must have as other arguments than the results tlist an even number of dates. The function searches in the environment for variables with the same names as those saved in the results tlist and calculates the forecast by applying the vector of coefficients to the vector of ts found in the environment. If the function does not find a variable, say var, it displays the following message :

```
undefined variable : var
```

The results is a ts.

In the second case, the user must give as second argument a matrix whose number of rows is equal to the number of forecasts and the number of columns is equal to the number of exogenous variables in the regression. The result is a vector.

As en example, let us take again hendryericsson estimation. Suppose that the estimation has been saved in tlist he (by the command he=hendryericsson_d()). Since this estimation applies to ts, a forecast must involve dates. For instance:

```
-->he_prev=statfore(r, '1985q1', '1989q3 ')
```

If you want to make a forecast with peculiar values of the exogenous variables a solution should be to enter the following command:

```
-->r2=ols(he('y'),he('x'));statfore(r2,[0.02 0.01 0.03 0.015 1])
```

[4] More sophisticated functions are left for further releases

9. The single equation functions and their specifications.

ar1_grad_____gradient for ols model with AR1 errors

CALLING SEQUENCE

[grad]=ar1_grad(param,y,x)

PARAMETERS

INPUT:
* param = parameter vector (k x 1)
* y = dependent variable vector (n x 1)
* x = explanatory variables matrix (n x m)

OUTPUT:
grad = the gradient

DESCRIPTION

Evaluates minus the gradient of the log-likelihood for ols model with AR1 errors. Param(1,1) contains rho parameter.

Example:
g=ar1_grad(parm,grocer_y,grocer_x)

This example is taken from olsar1.

ar1_like_____log-likelihood for ols model with AR1 errors

CALLING SEQUENCE

[mlike]=ar1_like(,y,x)

PARAMETERS

INPUT:
* param = parameter vector (k x 1)
* y = dependent variable vector (n x 1)
* x = explanatory variables matrix (n x m)

OUTPUT:
mlike = a scalar equal to -log(likelihood)

DESCRIPTION

Evaluates ols model with AR1 errors (minus) log-likelihood.

Example:
f=ar1_like(parm,grocer_y,grocer_x)

This example is taken from olsar1.

bounds_____setting estimation bounds

CALLING SEQUENCE

[]=bounds(b1,b2,…, b2n-1,b2n)

PARAMETERS

INPUT:
a series of dates strings

OUTPUT:
Nothing: the bounds are transferred to the upper level in a (p x 1) vector by the argument resume

DESCRIPTION

Sets the bounds for a regression. The bounds remain valid until the user leaves the environment (a working space or a function). This choice can be discussed: it is not without danger, since bounds used by a program can be different from the ones the user intended to use; it mimics the Portable Troll choice and is very convenient...

Examples:
1) bounds('1964q3','1989q2')
2) bounds('1990f52p1','1999f52d26')
3) bounds('1901a','1913a', '1919a','1939a', '1946a','2000a')

In example 1 (taken from hendryericsson_d), bounds range from the first quarter of 1963 to the third quarter of 1989. In example 2, they range from the first week ('f52' indicates that the frequency is set to 52) of 1990 until the 26th week of 1999. In example 3, bounds cover all the 20th century years, the period of the first (years 1914 to 1918) and second (years 1940 to 1945) world wars being excluded.

hwhite_____White's adjusted heteroscedastic estimation

CALLING SEQUENCE

[rhwhite]=hwhite(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* arg1 to argn = arguments which can be:

. a time series
. a real (nx1) vector
. a string equal to the name of a time series or a (nx1) real vector between quotes
. the string 'noprint' if the user doesn't want to display the results of the regression
---
OUTPUT:
rhwhite = a tlist with
. rhwhite('meth') = 'White"s heteroskedasticity consistent'
. rhwhite('y') = y data vector
. rhwhite('x') = x data matrix
. rhwhite('nobs') = nobs
. rhwhite('nvar') = nvars
. rhwhite('beta') = bhat
. rhwhite('yhat') = yhat
. rhwhite('resid') = residuals
. rhwhite('vcovar') = estimated variance-covariance matrix of beta
. rhwhite('sige') = estimated variance of the residuals
. rhwhite('sige') = estimated variance of the residuals
. rhwhite('ser') = standard error of the regression
. rhwhite('tstat') = t-stats
. rhwhite('pvalue') = pvalue of the betas
. rhwhite('dw') = Durbin-Watson Statistic
. rhwhite('prescte') = boolean indicating the presence or absence of a constant in the regression
. rhwhite('rsqr') = rsquared
. rhwhite('rbar') = rbar-squared
. rhwhite('f') = F-stat for the nullity of coefficients other than the constant
. rhwhite('pvaluef') = its significance level
. rhwhite('prescte') = boolean indicating the presence or absence of a time series in the regression
. rhwhite('namey') = name of the y variable
. rhwhite('namex') = name of the x variables
. rhwhite('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION

Computes White's adjusted heteroscedastic consistent Least-squares Regression. If the user has not given the argument 'noprint', displays on screen the results of the regression and various diagnostics
---
References: H. White 1980, Econometrica Vol. 48 pp. 818-838.

Example:
hwhite('del(lm1-lp)','del(lp)','del(lagts(1,lm1-lp-ly))','rnet','lagts(1,lm1-lp-ly)','cte')

This example, taken from hwhite_d, provides White's adjusted heteroscedastic consistent Least-squares Regression for Hendry and Ericsson (1991) equation n° 6 (results not presented by the authors).

invxpx_____nversion of X'X

CALLING SEQUENCE

[xpxi]=invxpx(x)

PARAMETERS

INPUT:
x = a (nxk) real matrix

OUTPUT:
xpxi = the (k x k) matrix equal to inv(x'x)

DESCRIPTION

Provides inversion of the matrix (X'X) using the QR decomposition

Example:
beta = invxpx(x)*(x'*y)

This example, taken from ols0, provides the ordinary least squares estimate of parameters, using the QR decomposition of x.

iv_____instrumental variables

CALLING SEQUENCE

[riv]=iv(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = dependent variable vector (nobs x 1)
* argi is either
  - the string 'exo=[var1;var2;...,vark]' where var1, var2,..., vark are exogenous variables which are therefore not instrumented
  - the string 'endo=[var1;var2;...,varl]' where var1, var2,..., varl are endogenous variables which are therefore instrumented
  - the string 'ivar=[var1;var2;...,varm]' where var1, var2,..., varm are the instruments
  - the string 'noprint' if the user doesn't want to display the results of the regression
---
OUTPUT:
results = a structure tlist with
- riv('meth') = 'iv'
- riv('nobs') = nobs
- riv('nendog') = # of endogenous
- riv('nexog') = # of exogenous
- riv('nvar') = # of endogenous + # of exogenous

- riv('y') = y data vector
- riv('beta') = bhat estimates
- riv('tstat') = t-statistics
- riv('yhat') = yhat predicted values
- riv('resid') = residuals
- riv('residtsls') = residuals calculated with the endogenous variables replaced by their regression from first stage estimation
- riv('sigu') = e'*e
- riv('sige') = e'*e/(n-k)
- riv('dw') = Durbin-Watson Statistic
- riv('prescte') = boolean indicating the presence or absence of a constant in the regression
- riv('rsqr') = rsquared
- riv('rbar') = rbar-squared
- riv('f') = F-stat for the nullity of coefficients other than the constant
- riv('pvaluef') = its significance level
- riv('grsqr') = generalized rsquared (that is which takes into account the endogeneity of some explicative variables)
- riv('prests') = boolean indicating the presence or absence of a time series in the regression
- riv('namey') = name of the y variable
- riv('namex') = name of the x variables
- riv('nameinst') = name of the instruments
- riv('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION

Computes instrumental variables estimation for a single equation. Variables can be ts, vectors or even matrices. Explicative variables enter either the "list" of instrumented variables ('endo=[var1;var2;...,varl]') or the list of exogenous variables ('exo=[var1;var2;...,vark]').

Example:
iv('y2','endo=y1','exo=iota;x2','iv=iota;x1;x2');

This example, taken from iv_d, provides the instrumental variables estimation for the regression of endogenous variable y2 on y1, iota and x2: y1 being endogenous, it is instrumented by iota, x2 and an exogenous variable not in the regression, x1.

iv1_____instrumental variables

CALLING SEQUENCE

[rtsls]=iv1(y,y1,x1,xall)

PARAMETERS

INPUT:
* y = dependent variable vector (nobs x 1)
* y1 = endogenous variables matrix (nobs x g) for this equation
* xexog = exogenous variables matrix for this equation
* xall = instruments for variables y1

OUTPUT:
results = a structure tlist with
- rtsls('meth') = 'tsls'
- rtsls('nobs') = nobs
- rtsls('nendog') = # of endogenous
- rtsls('nexog') = # of exogenous
- rtsls('nvar') = # of endogenous + # of exogenous
- rtsls('y') = y data vector
- rtsls('beta') = bhat estimates
- rtsls('tstat') = t-statistics
- rtsls('yhat') = yhat predicted values
- rtsls('resid') = residuals
- rtsls('residtsls') = residuals calculated with the endogenous variables replaced by their regression from first stage estimation
- rtsls('sigu') = e'*e
- rtsls('sige') = e'*e/(n-k)
- rtsls('dw') = Durbin-Watson Statistic
- rtsls('prescte') = boolean indicating the presence or absence of a constant in the regression
- rtsls('rsqr') = rsquared
- rtsls('rbar') = rbar-squared
- rtsls('f') = F-stat for the nullity of coefficients other than the constant
- rtsls('pvaluef') = its significance level
- rtsls('grsqr') = generalized rsquared (that is which takes into account the endogeneity of some explicative variables)

DESCRIPTION

Produces instrumental variables estimation when all variables are in vector or matrix form.

Example:
riv=iv1(grocer_y,grocer_endo,grocer_exo,grocer_ivar));

This example, taken from iv, provides the instrumental variables estimation for the regression of endogenous variable grocer_y on the matrix of variables supposed to be correlated with the residuals, grocer_endo, and the matrix of variables supposed to be uncorrelated with the residuals, grocer_exo: grocer_endo being endogenous, it is instrumented by grocer_ivar. This is a low level function: for empirical uses, the user can perform the same, and much more, with iv.

lad_____least absolute deviations regression

CALLING SEQUENCE

[rlad]=lad(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* argi = an argument which can be:
  . a time series
  . a real (nx1) vector
  . a string equal to the name of a time series or a (nx1) real vector between quotes
  . the string 'noprint' if the user doesn't want to display the results of the regression
  . the string 'itmax=n' where n is the maximum # of iterations (for example 'itmax=10'; default=500)
  . the string 'crit=n' where n is the convergence criterion (for example 'crit=1e-5'; default = 1e-15)

OUTPUT:
rlad = a tlist with
  . rlad('meth') = 'lad'
  . rlad('y') = y data vector
  . rlad('x') = x data matrix
  . rlad('nobs') = nobs
  . rlad('nvar') = nvars
  . rlad('b_new') = bhat
  . rlad('yhat') = yhat
  . rlad('resid') = residuals
  . rlad('vcovar') = estimated variance-covariance matrix of
    b_new
  . rlad('sige') = estimated variance of the residuals
  . rlad('sige') = estimated variance of the residuals
  . rlad('ser') = standard error of the regression
  . rlad('tstat') = t-stats
  . rlad('pvalue') = pvalue of the b_news
  . rlad('dw') = Durbin-Watson Statistic
  . rlad('prescte') = boolean indicating the presence or absence of a constant in the regression
  . rlad('namey') = name of the y variable
  . rlad('namex') = name of the x variables
  . rlad('bounds') = if there is a timeseries in the regression, the bounds of the regression
  . rlad('iter') = # of iterations
  . rlad('conv') = convergence max(abs(bnew-bold))
  . rlad('weight') = weight used to do the last ols regression

DESCRIPTION

Least absolute deviations regression. Minimizes sum(abs(y - x*b)) using re-iterated weighted least-squares where the weights are the inverse of the absolute values of the residuals. If the user has not given the argument 'noprint', the results of the regression and various diagnostics are displayed on the working space.

Examples:
1) lad(y,x)
2) lad('y','x1','x2','cte','maxit=200','noprint')

Example 1, taken from lad_d, provides the least absolute deviation regression of vector y on matrix x. Example 2 provides the least absolute deviation regression of y (which can be a ts or a vector) on x1, x2, which can be ts, vectors or matrices and on a constant; the maximum number of iteration is set to 200 and the results are not displayed.

lagbounds_____lag bounds in a regression

CALLING SEQUENCE

[bounds_out]=lagbounds(bounds_in,l)

PARAMETERS

INPUT:
* bounds_in = the original bounds
* l = the lag (positive or negative, if you want to extend bounds in the past)

OUTPUT:
bounds_out = the lagged bounds

DESCRIPTION

Cuts the bounds at the start of the period. If the lag is negative, then bounds are extended in the past for a number of peridos equal to the absolute value of the lag.

Examples:
1) lagbounds(['1985q2';'1997q4'],3)
2) grocer_boundsvar=lagbounds(grocer_boundsvarb,-grocer_l-1)

Example 1 leads to the (2x1) vector:
['1986q1'
'1997q4']
Example 2 is taken from function cadf.

logit_____logit regression

CALLING SEQUENCE

[rlogit]=logit(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* argi = an argument which can be:
  . a time series
  . a real (nx1) vector
  . a string equal to the name of a time series or a (nx1) real vector between quotes
  . the string 'noprint' if the user doesn't want to display the results of the regression
  . the string 'maxit=xx' if the user wants to set the maximum # of iterations to xx (default=100)
  . the string 'tol=xx' if the user wants to set the convergence criterion to xx (default=1e-6)

OUTPUT:
rlogit = a results tlist with
  . rlogit('meth') = 'logit'
  . rlogit('y') = y data vector
  . rlogit('x') = x data matrix
  . rlogit('nobs') = # observations
  . rlogit('nvar') = # variables
  . rlogit('beta') = bhat
  . rlogit('yhat') = yhat
  . rlogit('resid') = residuals
  . rlogit('vcovar') = estimated variance-covariance matrix of beta
  . rlogit('tstat') = t-stats
  . rlogit('pvalue') = pvalue of the betas
  . rlogit('r2mf') = = McFadden pseudo-R^2
  . rlogit('rsqr') = = Estrella R^2
  . rlogit('lratio') = LR-ratio test against intercept model
  . rlogit('lik') = unrestricted Likelihood
  . rlogit('zip') = # of 0's
  . rlogit('one) = # of 1's
  . rlogit('iter') = # of iterations
  . rlogit('crit') = convergence criterion
  . rlogit('namey') = name of the y variable
  . rlogit('namex') = name of the x variables
  . rlogit('prests') = boolean indicating the presence or absence of a time series in the regression
  . rlogit('prescte') = %f (for printings)
  . rlogit('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION

Computes Logit Regression. If the user has not given the argument 'noprint', displays on screen the results of the regression and various diagnostics.

References: Arturo Estrella (1998) 'A new measure of fit for equations with dichotomous dependent variable', JBES, Vol. 16, #2, April, 1998.

Example:
1) logit('grade','cte','psi','tuce','gpa');
2) logit('grade','cte','psi','tuce','gpa','maxit=200','crit=sqrt(%eps)','noprint');

Example 1, taken from logit_d, provides the logit regression of vector grade on a constant and the vectors psi, tuce, gpa. Example 2 does the same, except that the maximum number of iterations is set to 200 (instead of 100), the convergence criterion to sqrt(%eps) (instead of 0.000001) and the results are not displayed on screen.

mcov_____White's X'ΩX

CALLING SEQUENCE

[xuux]=mcov(x,u)

PARAMETERS

INPUT:
* x = nobs x nvar explanatory variables matrix
* u = nobs x 1 residuals

OUTPUT:
xuux such that xpx-inverse*xuux*xpx-inverse represents a heteroscedasticity consistent variance-covariance matrix.

DESCRIPTION

Computes x'*u*u'*x.

References: H. White 1980, Econometrica Vol. 48 pp. 818-838.

Example:
xuux = mcov(x,resid);

Example taken from hwhite. Should not have many other uses.

nls_____non linear least squares

CALLING SEQUENCE

[rnls]=nls(eq [,option_1,…,option_n])

PARAMETERS

INPUT:
* eq = a string representing the equation to estimate
* option_i =
  . 'coef=[namecoef1;namecoef2;...;namecoefn]' with namecoefi = name of the coefficient # i in

(default coef=['a1';'a2';...;'an'])
. 'init=[init1;init2;...;intin] if the user wants to give starting values
. any option of maxlik (see maxlik() for a list)

OUTPUT:
rnsls= a tlist with
. rnls('meth') = 'nls'
. rnls('beta') = bhat
. rnls('nobs') = nobs
. rnls('nvar') = nvars
. rnls('beta') = bhat
. rnls('resid') = residuals
. rnls('vcovar') = estimated variance-covariance matrix of beta
. rnls('sige') = estimated variance of the residuals
. rnls('sigu') = sum of squared residuals
. rnls('ser') = standard error of the regression
. rnls('tstat') = t-stats
. rnls('pvalue') = pvalue of the betas
. rnls('dw') = Durbin-Watson Statistic
. rnls('prests') = boolean indicating the presence or absence of a time series in the regression
. rnls('namey') = the equation
. rnls('namex') = name of the coefficients
. rnls('bounds') = if there is a timeseries in the regression, the bounds of the regression
. rnls('ropt') = the output tlist from maxlik (see maxlik for the list of arguments)

DESCRIPTION

Provides the non linear least squares estimation of a regression, which is given by the string eq. This string must have the form : 'lhs(y,x1,…,xk, coef1, coef2,…, coefn) = rhs(coef1,x1,…,xk,coef2,…, coefn)'. y,x1,… ,xk are respectively the endogenous variable and the exogenous ones. They can be time series or vectors. In the first case, the user can have specified the bounds; if she has not, then they are chosen to cover the longest time span. In the second case, vectors must have the same length.

The equation must be the first argument of nls. All other arguments can be given in whatever order.

The user can specify the name of the coefficients. In that case, the string 'coef = [namecoef1;namecoef2;...;namecoefn]' must be entered in nls. If the user has not given the names of her coefficients, the function assumes that all variables beginning with the character 'a' are coefficients, that the first one is named a1 and that the following ones are a2,… without any discontinuity: you can name your coefficients a1,a2,a3; but not a1,a2,a4.

The user can specify starting values for the maximisation program. In that case, the parameter 'init = [init_1;init_2;...;inti_n]' must be entered in nls. If the user does not give starting values, they are drawn from a normal law.

Lastly, the user can give all maxlik() options as in maxlik().

Examples

All these examples use the ts smptr in the database smpt.dat. smptr is an annual timeseries, from 1978 to 1999.

1) r=nls('smptr/lagts(smptr)-1=a1+a2*(lagts(smptr)/lagts(2,smptr)-1-a1)')
2) r=nls('smptr/lagts(smptr)-1=a1+a2*(lagts(smptr)/lagts(2,smptr)-1-a1','coef=[a1 ;a2])')
3) r=nls('smptr/lagts(smptr)-1=a1+a2*(lagts(smptr)/lagts(2,smptr)-1-a1)','init=rand(2,1,"normal"')
4) bounds('1980a','1999a'); r=nls('smptr/lagts(smptr)-1=a1+a2*(lagts(smptr)/lagts(2,smptr)-1-a1)')
5) r=nls('smptr/lagts(smptr)-1=a+b*(lagts(smptr)/lagts(2,smptr)-1-a1)','coef=[a;b]')
6)[r]=nls('smptr/lagts(smptr)-1=a1+a2*(lagts(smptr)/lagts(2,smptr)-1-a1)','init=[0.5;0.5]','maxit=1000')
7) s=series(smptr) ; s0=s(3:22) ; s1=s(2:21) ; s2=s(1:20) ;r=nls('s0./s1-1=a1+a2*(s1./s2-1-a1)'

Examples 1 to 4 are strictly equivalent: they provide the estimation of the equation 'smptr/lagts(smptr)-1=a1+a2*(lagts(smptr)/lagts(2,smptr)-1-a1)' over the period 1980-1999 (note that since there are 2 lags in the equation, the bounds are adjusted to withdraw the first 2 years of smptr), starting with random values (note the double quotes before and the triple quotes after normal).

Example 5 does the same as examples 1 to 4, except that the coefficient are named a and b.

Example 6 does the same as examples 1 to 4, except that starting values are set to [0.5 ;0.5] and that the maximum # of iterations is set to 1000.

Example 7 does the same as examples 1 to 4, except that it uses the vector representation of the ts (trickier, but quicker).

nwest_____Estimation with Newey-West correction on standard errors

CALLING SEQUENCE

[rnwest]=nwest(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* varargin = arguments which can be:
  . a time series
  . a real (nx1) vector
  . a string equal to the name of a time series or a (nx1) real vector between quotes
  . the string 'noprint' if the user doesn't want to display the results of the regression
  . the string 'win=n' where n is the length of the Barlett window (default = floor(5*nobs^0.25))

OUTPUT:
rnwest = a tlist with
  . rnwest('meth') = 'Newey-West's heteroskedastic-autocorrelation consistent'
  . rnwest('y') = y data vector
  . rnwest('x') = x data matrix

. rnwest('nobs') = nobs
. rnwest('nvar') = nvars
. rnwest('beta') = bhat
. rnwest('yhat') = yhat
. rnwest('resid') = residuals
. rnwest('vcovar') = estimated variance-covariance matrix of beta
. rnwest('sige') = estimated variance of the residuals
. rnwest('sige') = estimated variance of the residuals
. rnwest('ser') = standard error of the regression
. rnwest('tstat') = t-stats
. rnwest('pvalue') = pvalue of the betas
. rnwest('dw') = Durbin-Watson Statistic
. rnwest('prescte') = boolean indicating the presence or absence of a constant in the regression
. rnwest('rsqr') = rsquared
. rnwest('rbar') = rbar-squared
. rnwest('f') = F-stat for the nullity of coefficients other than the constant
. rnwest('pvaluef') = its significance level
. rnwest('prests') = boolean indicating the presence or absence of a time series in the regression
. rnwest('namey') = name of the y variable
. rnwest('namex') = name of the x variables
. rnwest('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION

Computes Newey-West's adjusted heteroskedastic and autocorrelation consistent (HAC) Least-squares Regression. High level function that works with ts, matrices,…

Example:
load('SCI/macros/grocer/db/bdhenderic.dat') ; bounds('1964q3','1989q2') ; rn=nwest('delts(lm1-lp)', 'delts(lp)','delts(lagts(1,lm1-lp-ly))','rnet','lagts(1,lm1-lp-ly)','cte')

Provides the Newey-West's adjusted heteroskedastic and autocorrelation consistent (HAC) Least-squares Regression of equation (6) in Hendry and Ericsson (1991).

nwest1_____Estimation with Newey-West correction on standard errors

CALLING SEQUENCE

[rnwest]=nwest1(y,x,win)

PARAMETERS

INPUT:
* y = a real (nx1) vector
* x = a real (nxk) matrix
* win = a scalar (length of the Barlett window)

OUTPUT:
rnwest = a tlist with
. rnwest('meth') = 'Newey-West's heteroskedastic-autocorrelation consistent'
. rnwest('y') = y data vector
. rnwest('x') = x data matrix
. rnwest('nobs') = nobs
. rnwest('nvar') = nvars
. rnwest('beta') = bhat
. rnwest('yhat') = yhat
. rnwest('resid') = residuals
. rnwest('vcovar') = estimated variance-covariance matrix of beta
. rnwest('sige') = estimated variance of the residuals
. rnwest('sige') = estimated variance of the residuals
. rnwest('ser') = standard error of the regression
. rnwest('tstat') = t-stats
. rnwest('pvalue') = pvalue of the betas
. rnwest('dw') = Durbin-Watson Statistic
. rnwest('prescte') = boolean indicating the presence absence of a constant in the regression
. rnwest('rsqr') = rsquared
. rnwest('rbar') = rbar-squared
. rnwest('f') = F-stat for the nullity of coefficients other than the constant
. rnwest('pvaluef') = its significance level
. rnwest('prescte') = boolean indicating the presence or absence of a time series in the regression

DESCRIPTION

Computes Newey-West's adjusted heteroskedastic and autocorrelation consistent (HAC) Least-squares Regression. Low level function that works only on vectors and matrices.

Example:
rnwest=nwest1(grocer_y,grocer_x,grocer_win)

Example taken from function nwest1.

ols_____ordinary least squares

CALLING SEQUENCE

[rols]=ols(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* varargin = arguments which can be:
. a time series
. a real (nx1) vector

. a string equal to the name of a time series or a (nx1) real vector between quotes
. the string 'noprint' if the user doesn't want to display the results of the regression
---
OUTPUT:
rols = a results tlist with
. rols('meth') = 'ols'
. rols('y') = y data vector
. rols('x') = x data matrix
. rols('nobs') = # observations
. rols('nvar') = # variables
. rols('beta') = bhat
. rols('yhat') = yhat
. rols('resid') = residuals
. rols('vcovar') = estimated variance-covariance matrix of beta
. rols('sige') = estimated variance of the residuals
. rols('sigu') = sum of squared residuals
. rols('ser') = standard error of the regression
. rols('tstat') = t-stats
. rols('pvalue') = pvalue of the betas
. rols('dw') = Durbin-Watson Statistic
. rols('condindex') = multicolinearity cond index
. rols('llike') = the log-likelihood
. rols('prescte') = boolean indicating the presence or absence of a constant in the regression
. rols('rsqr') = rsquared
. rols('rbar') = rbar-squared
. rols('f') = F-stat for the nullity of coefficients other than the constant
. rols('pvaluef') = its significance level
. rols('prests') = boolean indicating the presence or absence of a time series in the regression
. rols('namey') = name of the y variable
. rols('namex') = name of the x variables
. rols('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION

The most general GROCER function performing least-squares regression. Endogenous variable must be given first, as a vector, a ts, between quotes (if the user wants to keep the name of the variable in the tlist result and for the printings) or not. Exogenous variables are given after, in one of the formats authorized for the endogenous one, or in matrix format. The program displays on screen various results (coefficients, tstat, R^2, Durbin and Watson,…) except if the user has entered the argument 'noprint' anywhere after the first argument.

Examples:
1) bounds('1964q3','1989q2'); rols=ols('del(lm1-lp)','del(lp)','del(lagts(1,lm1-lp-ly))','rnet', 'lagts(1,lm1-lp-ly)', 'cte');
2) bounds('1964q3','1989q2'); rols=ols('del(lm1-lp)',['del(lp)','del(lagts(1,lm1-lp-ly))','rnet', 'lagts(1,lm1-lp-ly)','cte']);
3) bounds('1964q3','1989q2'); rols=ols(del(lm1-lp),del(lp),del(lagts(1,lm1-lp-ly)),rnet, lagts(1,lm1-lp-ly),cte);
4) bounds('1964q3','1989q2'); rols=ols(del(lm1-lp),del(lp),del(lagts(1,lm1-lp-ly)),rnet, lagts(1,lm1-lp-ly),cte, 'noprint');

5) y=grand(100,1, 'nor',0,1); x=grand(100,4, 'nor',0,1);ols('y', 'x')
6) ols(y, x);

Examples 1 to 3 provide the same results except that the third example does not keep the names of the variables, which are named 'endogenous', 'exogenous # 1', 'exogenous # 2', 'exogenous # 3', 'exogenous # 4', 'exogenous # 5'. Example 4 gives exactly the same tlist result as example 3, but the results are not displayed on screen (this can be done later by typing prtuniv(rols)). Examples 5 and 6 give the same results, except that the endogenous variables is named 'y' in example 5 and 'endogenous' in example 6 and the exogenous variables 'x_1', 'x_2', 'x_3', 'x_4' in example 5 and 'exogenous # 1', 'exogenous # 2', 'exogenous # 3', 'exogenous # 4' in example 6.

ols_cons_____constrained ordinary least squares

CALLING SEQUENCE

[rols]=ols(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* arg1,…,argn = arguments which can be:
  . a time series
  . a real (nx1) vector
  . a string equal to the name of a time series or a (nx1) real vector between quotes
  . the string 'noprint' if the user doesn't want to display the results of the regression
  . the string 'R=m' where m is the R matrix in Rb = r
  . the string 'r=m' where m is the r matrix in Rb = r

OUTPUT:
rols = a results tlist with
  . rols('meth') = 'ols'
  . rols('y') = y data vector
  . rols('x') = x data matrix
  . rols('nobs') = # observations
  . rols('nvar') = # variables
  . rols('beta') = bhat
  . rols('yhat') = yhat
  . rols('resid') = residuals
  . rols('vcovar') = estimated variance-covariance matrix of beta
  . rols('sige') = estimated variance of the residuals
  . rols('sigu') = sum of squared residuals
  . rols('ser') = standard error of the regression
  . rols('tstat') = t-stats
  . rols('pvalue') = pvalue of the betas
  . rols('dw') = Durbin-Watson Statistic
  . rols('condindex') = multicolinearity cond index
  . rols('prescte') = boolean indicating the presence or absence of a constant in the regression

. rols('R') = the R matrix in Rb=r
. rols('r') = the r matrix in Rb=r
. rols('rsqr') = rsquared
. rols('llike') = the log-likelihood
. rols('rbar') = rbar-squared
. rols('f') = F-stat for the nullity of coefficients other than the constant
. rols('pvaluef') = its significance level
. rols('prests') = boolean indicating the presence or absence of a time series in the regression
. rols('namey') = name of the y variable
. rols('namex') = name of the x variables
. rols('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION

The high-level function that provides the constrained ordinary least squares, when the constraint takes the form Rb=r. Endogenous variable must be given first, as a vector, a ts, between quotes (if the user wants to keep the name of the variable in the tlist result and for the printings) or not. Exogenous variables are given after, in one of the formats authorized for the endogenous one, or in matrix format. The matrices R and r must be entered under the form 'R=m1' and 'r=m2' where m1 and m2 are their respective values. The program displays on screen various results (coefficients, tstat, R-squared, Durbin and Watson,...) except if the user has entered the argument 'noprint' anywhere after the first argument.

Examples:
load('SCI/macros/grocer/db/bdhenderic.dat') ; bounds('1964q3','1989q2') ;rols=ols_cons('delts(lm1-lp)','delts(lp)','delts(lagts(1,lm1))','delts(lagts(1,lp))','delts(lagts(1,ly))','rnet', 'lagts(1,lm1-lp-ly)','cte','R=[0 1 1 0 0 0 0 ; 0 1 0 1 0 0 0]','r=[0;0]')

The example shows the estimation of Hendry and Ericsson (1991) model by imposing the constraints on delts(lagts(1,lm1)), 'delts(lagts(1,lp))' and 'delts(lagts(1,ly))' in order to obtain delts(lagts(1,lm1-ly-lp)). Calling b the vector of coefficients, the constraints are:
b2=-b3 and b2=-b4
hence:
R= [0 1 1 0 0 0 0]
[0 1 0 1 0 0 0]
and:
r = [0]
As expected, results are the same as those provided by function hendryericsson!

ols0_____ordinary least squares

CALLING SEQUENCE

[bhat]=ols0(y,x)

PARAMETERS

INPUT:
* y = dependent variable vector (nobs x 1)
* x = independent variables matrix (nobs x nvar)

OUTPUT:
bhat = estimated beta in y=x*beta+u

DESCRIPTION

The most basic least-squares regression: provides only the estimated beta, bhat. Used mainly in other programs.

Example:
1) beta=ols0(y,x)

This example is taken from hwhite.

ols1_____ordinary least squares

CALLING SEQUENCE

[rols]=ols1(y,x)

PARAMETERS

INPUT:
* y = dependent variable vector (nobs x 1)
* x = independent variables matrix (nobs x nvar)

OUTPUT:
a tlist with:
rols('meth') = 'ols'
rols('y') = y data vector
rols('x') = x data matrix
rols('nobs') = nobs
rols('nvar') = nvars
rols('beta') = bhat
rols('tstat') = t-stats
rols('pvalue') = pvalue of the betas
rols('resid') = residuals
rols('vcovar') = estimated variance-covariance matrix of beta
rols('sige') = estimated variance of the residuals
rols('sigu') = sum of squared residuals
rols('ser') = standard error of the regression
rols('yhat') = yhat

DESCRIPTION

One of the numerous functions performing ordinary least squares: this one assumes that x et y are already a matrix and a vector and does not provide DW, R^2 statistics. Used in other programs.

Example:
r=ols1(y,ptrend(t,nobs))

This example is taken from kpss.

ols2_____ordinary least squares

CALLING SEQUENCE

[rols]=ols2(y,x)

PARAMETERS

INPUT:
* y = dependent variable vector (nobs x 1)
* x = independent variables matrix (nobs x nvar)

OUTPUT:
rols = a tlist with
. rols('meth') = 'ols'
. rols('y') = y data vector
. rols('x') = x data matrix
. rols('nobs') = nobs
. rols('nvar') = nvars
. rols('beta') = bhat
. rols('yhat') = yhat
. rols('resid') = residuals
. rols('vcovar') = estimated variance-covariance matrix of beta
. rols('sige') = estimated variance of the residuals
. rols('sige') = estimated variance of the residuals
. rols('ser') = standard error of the regression
. rols('tstat') = t-stats
. rols('pvalue') = pvalue of the betas
. rols('dw') = Durbin-Watson Statistic
. rols('condindex') = multicolinearity cond index
. rols('prescte') = boolean indicating the presence or absence of a constant in the regression
. rols('rsqr') = rsquared
. rols('rbar') = rbar-squared
. rols('f') = F-stat for the nullity of coefficients other than the constant
. rols('pvaluef') = its significance level

DESCRIPTION

Least-squares regression when variables looks like in the textbook. The results tlist contains all the results recorded by ols, except the fields 'namey', 'namex', 'prests', and, naturally, 'bounds'.

Example:
rols=ols2(grocer_y,grocer_x)

This example is taken from ols.

ols2_cons_____constrained ordinary least squares

CALLING SEQUENCE

rols=ols2_cons(y,x,R,r)

PARAMETERS

INPUT:
* y = dependent variable vector (nobs x 1)
* x = independent variables matrix (nobs x nvar)
* R = a (nc x k) matrix
* r = a (nc x 1) vector

OUTPUT:
rols = a results tlist with
. rols('meth') = 'ols'
. rols('y') = y data vector
. rols('x') = x data matrix
. rols('nobs') = # observations
. rols('nvar') = # variables
. rols('beta') = bhat
. rols('yhat') = yhat
. rols('resid') = residuals
. rols('vcovar') = estimated variance-covariance matrix of beta
. rols('sige') = estimated variance of the residuals
. rols('sigu') = sum of squared residuals
. rols('ser') = standard error of the regression
. rols('tstat') = t-stats
. rols('pvalue') = pvalue of the betas
. rols('dw') = Durbin-Watson Statistic
. rols('condindex') = multicolinearity cond index
. rols('prescte') = boolean indicating the presence or absence of a constant in the regression
. rols('R') = the R matrix in Rb=r
. rols('r') = the r matrix in Rb=r
. rols('rsqr') = rsquared
. rols('llike') = the log-likelihood
. rols('rbar') = rbar-squared
. rols('f') = F-stat for the nullity of coefficients other than the constant
. rols('pvaluef') = its significance level

DESCRIPTION

The low-level function that provides the constrained ordinary least squares, when the constraint takes the form Rb=r. Endogenous variable y and r in 'Rb=r' must be given in a vector form and x and R in 'Rb=r' in a matrix form. The program does not display the results, but they can be found in the output results tlist.

Examples:
load('SCI/macros/grocer/db/bdhenderic.dat') ; bounds('1964q3','1989q2') ;[y,junk,x]=explouniv('delts(lm1-lp)',['delts(lp)','delts(lagts(1,lm1))','delts(lagts(1,lp))','delts(lagts(1,ly))','rnet','lagts(1,lm1-lp-ly)','cte']);
r=ols2_cons(y,x,[0 1 1 0 0 0 0 ; 0 1 0 1 0 0 0],[0;0])
<p>
The example shows the estimation of Hendry and Ericsson (1991) model by imposing the constraints on delts(lagts(1,lm1)), 'delts(lagts(1,lp))' and 'delts(lagts(1,ly))' in order to obtain delts(lagts(1,lm1-ly-lp)) (see help for ols_cons). Estimation results are stored in tlist r.

olsar1_____maximum likelihood estimation of an autocorrelated model

CALLING SEQUENCE

[result]=olsar1(namey, arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* argi = an argument which can be:
  . a time series
  . a real (nxp) vector
  . a string equal to the name of a time series or a (nxp) real vector between quotes
  . the string 'noprint' if the user doesn't want the to display the results of the regression
---
OUTPUT:
rolsar1 = a results tlist with
  . rolsar1('meth') = ' ar(1) maximum likelihood'
  . rolsar1('y') = y data vector
  . rolsar1('x') = x data matrix
  . rolsar1('nobs') = # observations
  . rolsar1('nvar') = # variables
  . rolsar1('beta') = bhat
  . rolsar1('yhat') = yhat
  . rolsar1('resid') = residuals
  . rolsar1('vcovar') = estimated variance-covariance matrix of beta
  . rolsar1('sige') = estimated variance of the residuals
  . rolsar1('sigu') = sum of squared residuals
  . rolsar1('ser') = standard error of the regression
  . rolsar1('tstat') = t-stats
  . rolsar1('pvalue') = pvalue of the betas
  . rolsar1('dw') = Durbin-Watson Statistic
  . rolsar1('condindex') = multicolinearity cond index
  . rolsar1('prescte') = boolean indicating the presence or absence of a constant in the regression
  . rolsar1('rsqr') = rsquared
  . rolsar1('rbar') = rbar-squared
  . rolsar1('f') = F-stat for the nullity of coefficients other than the constant
  . rolsar1('pvaluef') = its significance level

. rolsar1('prests') = boolean indicating the presence or absence of a time series in the regression
. rolsar1('namey') = name of the y variable
. rolsar1('namex') = name of the x variables
. rolsar1('bounds') = if there is a timeseries in the regression, the bounds of the regression
. rolsar1('rho') = estimated first order autocorrelation of residuals
. rolsar1('trho') = its Student t
. rolsar1('like') = log-likelihood of the regression

DESCRIPTION

Computes maximum likelihood ols regression for AR1 errors, using Cochrane-Orcutt estimates as starting values. Endogenous variable must be given first, as a vector, a ts, between quotes (if the user wants to keep the name of the variable in the tlist result and for the printings) or not. Exogenous variables are given after, in one of the formats authorized for the endogenous one, or in matrix format. The program displays on screen various results (coefficients, tstat, R^2, Durbin and Watson, first order autocorrelation of residuals,…) except if the user has entered the argument 'noprint' anywhere after the first argument.

Example:
load('grocer/bdexamples/bdhenderic.dat') ; bounds('1964q3','1989q2') ; rolsar1=olsc('del(lm1-lp)','del(lp)','rnet','lagts(1,lm1-lp-ly)','cte')

This example, taken from olsar1_d, shows the estimation of first order autocorrelated model on Hendry and Ericsson (1991) preferred specification from which the variable 'del(lagts(1,lm1-lp-ly))' has been withdrawn.

olsc_____Cochrane-Orcutt estimation of an autocorrelated model

CALLING SEQUENCE

[rolsc]=olsc(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* argi = an argument which can be:
. a time series
. a real (nxp) vector
. a string equal to the name of a time series or a (nxp) real vector between quotes
. the string 'noprint' if the user doesn't want the to display the results of the regression
---
OUTPUT:
rolsc = a tlist with
. rolsc('meth') = 'Cochrane-Orcutt'
. rolsc('y') = y data vector
. rolsc('x') = x data matrix
. rolsc('nobs') = nobs
. rolsc('nvar') = nvars
. rolsc('beta') = bhat

. rolsc('yhat') = yhat
. rolsc('resid') = residuals
. rolsc('vcovar') = estimated variance-covariance matrix of beta
. rolsc('sige') = estimated variance of the residuals
. rolsc('sige') = estimated variance of the residuals
. rolsc('ser') = standard error of the regression
. rolsc('tstat') = t-stats
. rolsc('pvalue') = pvalue of the betas
. rolsc('dw') = Durbin-Watson Statistic
. rolsc('prescte') = boolean indicating the presence or absence of a constant in the regression
. rolsc('rsqr') = rsquared
. rolsc('rbar') = rbar-squared
. rolsc('f') = F-stat for the nullity of coefficients other than the constant
. rolsc('pvaluef') = its significance level
. rolsc('prescte') = boolean indicating the presence or absence of a time series in the regression
. rolsc('namey') = name of the y variable
. rolsc('namex') = name of the x variables
. rolsc('bounds') = if there is a timeseries in the regression, the bounds of the regression
. rolsc('rho') = the autocorrelation coefficient of the residuals
. rolsc('trho') = its T-stat
. rolsc('iterout') = a (niter x 3) matrix giving for each iteration the estimated rho, the convergence criterion and the iteration itself

DESCRIPTION

Computes Cochrane-Orcutt ols regression for AR1 errors. Endogenous variable must be given first, as a vector, a ts, between quotes (if the user wants to keep the name of the variable in the tlist result and for the printings) or not. Exogenous variables are given after, in one of the formats authorized for the endogenous one, or in matrix format. The program displays on screen various results (coefficients, tstat, R^2, Durbin and Watson, first order autocorrelation of residuals,…) except if the user has entered the argument 'noprint' anywhere after the first argument.

Example:
load('grocer/bdexamples/bdhenderic.dat') ; bounds('1964q3','1989q2') ; rolsc=olsc('del(lm1-lp)','del(lp)','rnet','lagts(1,lm1-lp-ly)','cte')

This example, taken from olsar1_d, shows the estimation of first order autocorrelated model on Hendry and Ericsson (1991) preferred specification from which the variable 'del(lagts(1,lm1-lp-ly))' has been withdrawn.

olst_____ ols with t-distributed errors

CALLING SEQUENCE

[rolst]=olst(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* argi = an argument which can be:
  . a time series
  . a real (nxp) vector
  . a string equal to the name of a time series or a (nxp) real vector between quotes
  . the string 'noprint' if the user doesn't want the to display the results of the regression

---

rolst = a tlist with
  . rolst('meth') = 'olst'
  . rolst('y') = y data vector
  . rolst('x') = x data matrix
  . rolst('nobs') = nobs
  . rolst('nvar') = nvars
  . rolst('beta') = bhat
  . rolst('yhat') = yhat
  . rolst('resid') = residuals
  . rolst('vcovar') = estimated variance-covariance matrix of beta
  . rolst('sige') = estimated variance of the residuals
  . rolst('ser') = standard error of the regression
  . rolst('tstat') = t-stats
  . rolst('pvalue') = pvalue of the betas
  . rolst('dw') = Durbin-Watson Statistic
  . rolst('namey') = name of the y variable
  . rolst('namex') = name of the x variables
  . rolst('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION

Computes ols with t-distributed errors, using iterated re-weighted least-squares to find maximum likelihood estimates. Endogenous variable must be given first, as a vector, a ts, between quotes (if the user wants to keep the name of the variable in the tlist result and for the printings) or not. Exogenous variables are given after, in one of the formats authorized for the endogenous one, or in matrix format. The program displays on screen various results (coefficients, tstat, R^2, Durbin and Watson, first order autocorrelation of residuals,…) except if the user has entered the argument 'noprint' anywhere after the first argument.

Example:
lresult = olst(y,x,'maxit=1000','crit=.0001');

This example, taken from olst_d, shows the estimation of a model where residuals are supposed to follow a Student law (in olst_d, the residuals are drawn from the Student law through the use of function tdis_rnd – see chapter 12 for a description of this function)

probit_____probit regression

CALLING SEQUENCE

[rprobit]=probit(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes; all values should be 0 or 1.
* argi = an argument which can be:
  . a time series
  . a real (nxp) vector
  . a string equal to the name of a time series or a (nxp) real vector between quotes
  . the string 'noprint' if the user doesn't want the to display the results of the regression
---
rprobit = a results tlist with
  . rprobit('meth') = 'probit'
  . rprobit('y') = y data vector
  . rprobit('x') = x data matrix
  . rprobit('nobs') = # observations
  . rprobit('nvar') = # variables
  . rprobit('beta') = bhat
  . rprobit('yhat') = yhat
  . rprobit('resid') = residuals
  . rprobit('vcovar') = estimated variance-covariance matrix of beta
  . rprobit('tstat') = t-stats
  . rprobit('pvalue') = pvalue of the betas
  . rprobit('r2mf') = = McFadden pseudo-R^2
  . rprobit('rsqr') = = Estrella R^2
  . rprobit('lratio') = LR-ratio test against intercept model
  . rprobit('lik') = unrestricted Likelihood
  . rprobit('zip') = # of 0's
  . rprobit('one') = # of 1's
  . rprobit('iter') = # of iterations
  . rprobit('crit') = convergence criterion
  . rprobit('namey') = name of the y variable
  . rprobit('namex') = name of the x variables
  . rprobit('prests') = boolean indicating the presence or absence of a time series in the regression
  . rprobit('prescte') = %f (for printings)
  . rprobit('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION

Computes Probit Regression. Endogenous variable must be given first, as a vector, a ts, between quotes (if the user wants to keep the name of the variable in the tlist result and for the printings) or not. Exogenous variables are given after, in one of the formats authorized for the endogenous one, or in matrix format. The program displays on screen various results (coefficients, tstat, R^2, Durbin and Watson, first order autocorrelation of residuals,…) except if the user has entered the argument 'noprint' anywhere after the first argument.

Examples:
1) r = probit('grade','cte','psi','tuce','gpa');
2) r = probit('grade','cte','psi','tuce','gpa', 'tol=sqrt(%eps)', 'noprint', 'maxit=200')

Example 1, taken from probit_d, shows the estimation of a probit model taken from Greene (1997). Example 2 does the same, except that the options tol and maxit are set to non default values and that the results are not displayed.

res2ts_____result vector object to timeseries

CALLING SEQUENCE

[tsout]=res2ts(res,nameobject)

PARAMETERS

INPUT:
* res = a results tlist
* nameobject = the name of a tlist field that can be transformed into ts ('y','resid' or 'yhat')

OUTPUT:
tsout = the corresponding timeseries

DESCRIPTION

Transforms into a timeseries a vector object in a results tlist according to the bounds saved in the tlist.

Example:
1) load('SCI/macros/grocer/db/bdhenderic.dat') ;bounds('1964q3','1989q2');
rols=ols('delts(lm1-lp)','delts(lp)','delts(lagts(1,lm1-lp-ly))','rnet','lagts(1,lm1-lp-ly)','cte') ;
he_yhat=res2ts(rols,'yhat')
2) rols_resid=res2ts(rols,'resid')

In example 1, the yhat from Hendry and Ericsson's estimate is transformed into the corresponding ts.
Example 2 is takane from function cadf.

ridge_____ridge regression

CALLING SEQUENCE

[rridge]=ridge(namey, arg1,…,argn)

PARAMETERS

INPUT:
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes; all values should be 0 or 1.
* argi = an argument which can be:
  . a time series
  . a real (nxp) vector
  . a string equal to the name of a time series or a (nxp) real vector between quotes
  . the string 'noprint' if the user doesn't want the to display the results of the regression
  . the string 'theta=xx' if the user wants to enter theta's value (default is the one recommended by Hoerl and Kennard)

OUTPUT:
rridge = a tlist with
  . rridge('meth') = 'ridge'
  . rridge('y') = y data vector
  . rridge('x') = x data matrix
  . rridge('nobs') = nobs
  . rridge('nvar') = nvars
  . rridge('beta') = bhat
  . rridge('yhat') = yhat
  . rridge('resid') = residuals
  . rridge('vcovar') = estimated variance-covariance matrix of beta
  . rridge('sige') = estimated variance of the residuals
  . rridge('sige') = estimated variance of the residuals
  . rridge('ser') = standard error of the regression
  . rridge('tstat') = t-stats
  . rridge('pvalue') = pvalue of the betas
  . rridge('dw') = Durbin-Watson Statistic
  . rridge('prescte') = boolean indicating the presence or absence of a constant in the regression
  . rridge('rsqr') = rsquared
  . rridge('rbar') = rbar-squared
  . rridge('f') = F-stat for the nullity of coefficients other than the constant
  . rridge('pvaluef') = its significance level
  . rridge('prescte') = boolean indicating the presence or absence of a time series in the regression
  . rridge('namey') = name of the y variable
  . rridge('namex') = name of the x variables
  . rridge('bounds') = if there is a timeseries in the
    regression, the bounds of the regression
  . rridge('theta') = the scale factor theta

DESCRIPTION

Computes Hoerl-Kennard Ridge Regression, using either the theta value entered by the user or the default one recommended by Hoerl and Kennard. Endogenous variable must be given first, as a vector, a ts, between quotes (if the user wants to keep the name of the variable in the tlist result and for the printings) or not. Exogenous variables are given after, in one of the formats authorized for the endogenous one, or in matrix format. The program displays on screen various results (coefficients, tstat, R^2, Durbin and Watson, first order autocorrelation of residuals,…) except if the user has entered the argument 'noprint' anywhere after the first argument.

Example:
1) r = ridge('del(lm1-lp)','del(lp)','del(lagts(1,lm1-lp-ly))','rnet','lagts(1,lm1-lp-ly)','cte')
2) r = ridge('del(lm1-lp)','del(lp)','del(lagts(1,lm1-lp-ly))','rnet','lagts(1,lm1-lp-ly)','cte', 'theta=0.001')

These examples shows the results of a ridge regression on Hendry and Ericsson's preferred regression, using for theta the default parameter in example 1 and 0.001 in example 2.

robust_____robust regression

CALLING SEQUENCE

[rrobust]=robust(wfunc,wparm, namey, arg1,…,argn)

PARAMETERS

INPUT:
* wfunc = 'huber' for Huber's t function
  'ramsay' for Ramsay's E function
  'andrew' for Andrew's wave function
  'tukey' for Tukey's biweight
* wparm = weighting function parameter
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* argi = an argument which can be:
  . a time series
  . a real (nxp) vector
  . a string equal to the name of a time series or a (nxp) real vector between quotes
  . the string 'noprint' if the user doesn't want to display the results of the regression
---
OUTPUT:
rrobust = a tlist with
  . rrobust('meth') = 'robust'+ 'huber', 'ramsay', 'andrew' or 'tukey'
  . rrobust('y') = y data vector
  . rrobust('x') = x data matrix
  . rrobust('nobs') = nobs
  . rrobust('nvar') = nvars
  . rrobust('beta') = bhat
  . rrobust('yhat') = yhat
  . rrobust('resid') = residuals

. rrobust('vcovar') = estimated variance-covariance matrix of beta
. rrobust('sige') = estimated variance of the residuals
. rrobust('sige') = estimated variance of the residuals
. rrobust('ser') = standard error of the regression
. rrobust('tstat') = t-stats
. rrobust('pvalue') = pvalue of the betas
. rrobust('dw') = Durbin-Watson Statistic
. rrobust('prescte') = boolean indicating the presence or absence of a constant in the regression
. rrobust('rsqr') = rsquared
. rrobust('rbar') = rbar-squared
. rrobust('f') = F-stat for the nullity of coefficients other than the constant
. rrobust('pvaluef') = its significance level
. rrobust('prescte') = boolean indicating the presence or absence of a time series in the regression
. rrobust('namey') = name of the y variable
. rrobust('namex') = name of the x variables
. rrobust('bounds') = if there is a timeseries in the regression, the bounds of the regression
. rrobust('wparm') = wparm
. rrobust('iter') = # of iterations
. rrobust('weight') = nobs - vector of weights
. rrobust('convg') = convg criterion

DESCRIPTION

Computes robust regression using iteratively reweighted least-squares. The first argument control the weighting scheme. The second argument controls the weighting parameter. Endogenous variable must be given third, as a vector, a ts, between quotes (if the user wants to keep the name of the variable in the tlist result and for the printings) or not. Exogenous variables are given after, in one of the formats authorized for the endogenous one, or in matrix format. The program displays on screen various results (coefficients, tstat, R^2, Durbin and Watson, first order autocorrelation of residuals,…) except if the user has entered the argument 'noprint' anywhere after the first argument.

Example:
1) r = robust('huber', 0.000338','del(lm1-lp)','del(lp)','del(lagts(1,lm1-lp-ly))','rnet', 'lagts(1,lm1-lp-ly)','cte')
2) r = robust('andrew', 0.000338','del(lm1-lp)','del(lp)','del(lagts(1,lm1-lp-ly))','rnet', 'lagts(1,lm1-lp-ly)','cte', 'noprint')

These examples shows the results of a robust regression on Hendry and Ericsson's preferred regression, using huber's weighting scheme in example 1 and andrew's one in example 2. Results are not displayed in example 2.

rolreg_____rolling/recursive regression and h-step ahead forecasts

CALLING SEQUENCE

rreg = rolreg(namey0,arg1,…,argn)

PARAMETERS

INPUT:
* namey0 = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* arg1,…,argn = arguments which can be:
  . a time series
  . a real (nxp) vector
  . a string equal to the name of a time series or a (nxp) real vector between quotes
  . a list of such elements
  . 'simu ="roll"' or "recu"' : if the user want to use a rolling or recursive window (default=)
  . 'hstep = xx': number of out-of-sample forecasts (which determines the frequency of re-estimation of the model)
  . 'mul = 0' : if the user wants to keep only the forecast at the hstep horizon
  . 'dates = ["xx","yy"]': end of first estimation + end of last estimation
  . 'const' or 'cte' if the user want a constant in the regression
  . 'meth=nwest' / 'ols'
  . 'win = xx" truncation window for newey-west estimation
  . 'xsmpl = 1' : allows more forecasts than the number of y observations if there is enough exogenous obs (x)

OUTPUT:
rreg = a results tlist with
- rreg('meth') = 'rolling' / 'recursive'
- rreg('beta') = rolling/recursive betas
- rreg('tstat') = rolling/recursive tstats
- rreg('pvalue') = rolling/recursive pvalue of the betas
- rreg('rsqr') = rolling/recursive rsquared
- rreg('rbar') = rolling/recursive rbar-squared
- rreg('yhat') = out-of-sample forecasts
- rreg('prescte') = boolean indicating the presence or absence of a constant in the regression
- rreg('bounds') = begin-end of first estimation & end of sample
- rreg('nfor') = vector of number of realized forecast and number of desired forecast

DESCRIPTION

Computes rolling or recursive estimation and makes out-of-sample forecasts.

Example:
1) ri = rolreg('delts(log(y))','ypast','delts(yfut)','cte','dates=["1992q1" "2003q4"]','hstep=4','mul=1', 'simu="recu"')

(2) ri = rolreg('delts(log(y))','ypast','delts(yfut)','cte','dates=["1992q1" "2003q4"]','hstep=1','simu="roll"')
Out-of-sample forecasts 1 quarter ahead with rolling estimation

These examples are taken from function rolreg_d. In example 1, out-of-sample forecasts 4 quarters ahead,are performed form 1992q1 to 2003q4 forecasts from 1 to 4 quarters are stored, with recursive estimation.

statfore_____static forecast

CALLING SEQUENCE

[p]=statfore(res,arg1,…,argn)

PARAMETERS

INPUT:
* res = result tlist
* argi =
  - (optional) subperiod over which the forecast is done if variables are ts
  - a (mxk) matrix of exogenous variables if they are not ts

OUTPUT:
p = prevision

DESCRIPTION

Provides a static forecast from an equation estimated by ols(), hwhite(), olst(), lad(),... It can be used also with results of olsc, but does not take into account the autocorrelation of residuals. If the estimation has been performed with ts, then the function assumes that the variables have been entered between quotes and that their name have therefore been saved in the tlist; in that case, the corresponding data must be present in the environment.

Examples:
1) r = hendryericsson();statfore(r, '1985q1', '1989q3 ')
2) r2=ols(r('y'),r('x'));statfore(r2,[0.02 0.01 0.03 0.015 1])

Example 1 shows the static forecast made on period 1985q1 to 1989q3 with Hendry and Ericsson preferred specification. Example 2 shows a static forecast made with the same model, but estimated now with the corresponding vectors (r('y') and r('x')), and with exogenous values equal to [0.02 0.01 0.03 0.015 1].

theil_____Theil-Goldberger mixed estimator

CALLING SEQUENCE

[results]=theil(rvec, rmat, v, namey, arg1,…,argn)

PARAMETERS

INPUT:
* rvec = a vector of prior mean values, (c in description)
* rmat = a matrix of rank(r) (R in description)
* v = prior variance-covariance (var-cov(U) in description)
* namey = a time series, a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes
* argi = arguments which can be:
  . a time series

. a real (nxp) vector
. a string equal to the name of a time series or a (nxp) real vector between quotes
. the string 'noprint' if the user doesn't want to display the results of the regression
---
OUTPUT:
results = a results tlist with
. results('meth') = 'Theil-Goldberger'
. results('beta') = bhat
. results('y') = y data vector
. results('x') = x data matrix
. results('nobs') = # observations
. results('nvar') = # variables
. results('yhat') = yhat
. results('resid') = residuals
. results('vcovar') = estimated variance-covariance matrix of beta
. results('sige') = estimated variance of the residuals
. results('sigu') = sum of squared residuals
. results('ser') = standard error of the regression
. results('tstat') = t-stats
. results('pvalue') = pvalue of the betas
. results('dw') = Durbin-Watson Statistic
. results('condindex') = multicolinearity cond index
. results('prescte') = boolean indicating the presence or
absence of a constant in the regression
. results('rsqr') = rsquared
. results('rbar') = rbar-squared
. results('f') = F-stat for the nullity of coefficients other than the constant
. results('pvaluef') = its significance level
. results('prests') = boolean indicating the presence or absence of a time series in the regression
. results('namey') = name of the y variable
. results('namex') = name of the x variables
. results('bounds') = if there is a timeseries in the regression, the bounds of the regression
. results('pmean') = prior means
. results('pstd') = prior std deviations

DESCRIPTION

Computes Theil-Goldberger mixed estimator

$$y = X B + E, \quad E = N(0,sige*IN)$$
$$c = R B + U, \quad U = N(0,v)$$

Example:
r = theil(ones(5,1),eye(5,5),0.1*eye(5,5),'endo','exo')

This example shows the estimation of endogenous variables 'endo' on exogenous variables 'exo' (which should be a (nobsx5) constant matrix), with the relatively tight prior (v=0.1*eye(5,5)) that each coefficient should be equal to 1 (since c=ones(5,1) and R=eye(5,5))