Chapter 7: Multicollinearity diagnostics, specification and other tests.

GROCER contains numerous diagnostics and specification tests proposed in the literature: diagnostic of multicollinearity (part 2); outlier diagnostics (part 3) and heteroskedasticity, autocorrelation, normality, stability… specification tests (part 4), (°)tests of forecasting accuracy (part 5)(°°) as long as the simple Fisher test (part 1).

1. Fisher test.

GROCER contains a simple function, named waldf that calculates the Fisher statistics for a test of linear constraints and its p-value: this function takes as first entry the results tlist from the restricted regression and as second entry the results tlist from the unrestricted regression. If the user does not want to print the results, he has just to give as third argument 'noprint'. This function returns a tlist, that can be printed later, with function prtfish (see chapter 21). There is also a function called waldf0, that returns only the statistics, its p-value and the degrees of freedom: results are neither stored in a tlist, nor printed. This last function is useful for the programmer who wants a quicker function and who wants to use, but not print, the results of the test. This function is for instance used by waldf, or by function automatic (see chapter 13).

2. Multicollinearity. (*)

In GROCER, the condition index, the multicollinearity diagnostic proposed by Belsey, Kuh and Welsch[1], is implemented trough 3 functions: bkw, bkw_scale and bkwols. Function bkw provides the whole variance decomposition proportions table: let $\lambda 1,\ldots \lambda n$ be the eigen values of matrix X, ordered from the greatest to the smallest; then the table reports (first row of the table) the quotients of $\lambda 1$ by all λi and (next rows) the contribution of the corresponding eigenvalues to the variance of each coefficients (see Belsley, Kuh and Welsch or James Lesage's book for more details). The function bkw_scale performs the same decomposition but on variables that have been normalized to have a unit norm. Since bkw applies to the exogenous variables, all types of exogenous variables can be given, as in ols (see chapter 6): matrices, vectors, ts, the name of any of these arguments between quotes, or lists of such elements.

Function bkwols, used by ols and other single equation regression functions, provides only the condition index, that is the quotient of the greatest by the smallest eigenvalue, **applied on normalized variables**[2].

As an example, below is the result of bkw_d, which applies bkw to an artificial dataset built to show multicollinearity. This is done simply by the following command in the function bkw_d: bkw('x').

```
Belsley, Kuh, Welsch Variance-decomposition (in %)


K(x) 1 1  1  2  45
x_1  0 4  80 17 0
```

[1] Belsley, D.A., R.E. Welsch and E. Kuh, 1980. Regression Diagnostics, John Wiley & Sons, Inc. New York

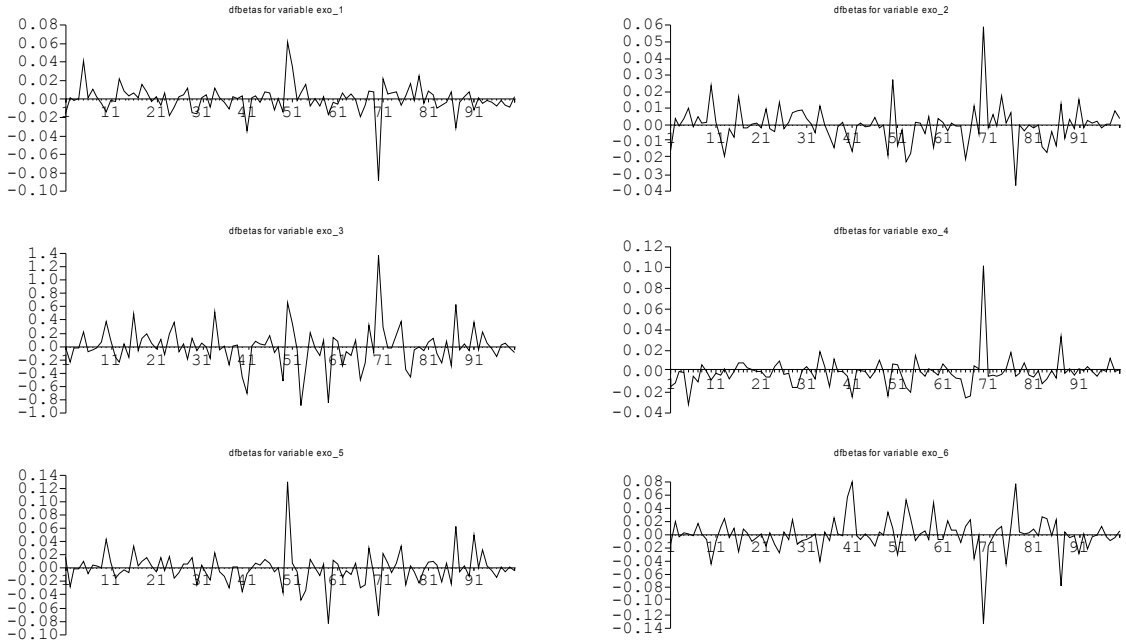
[2] This is new in version 1.1: in versions 1.0 to 1.041, bkwols was calculated on non normalized variables, which is not the best way to measure multicollinearity (a point made by Emmanuel Michaux).

```
x_2   0 0   0   0   100
x_3   0 0   0   0   100
x_4   0 19  6   75  0
x_5   0 69  13  18  0
```

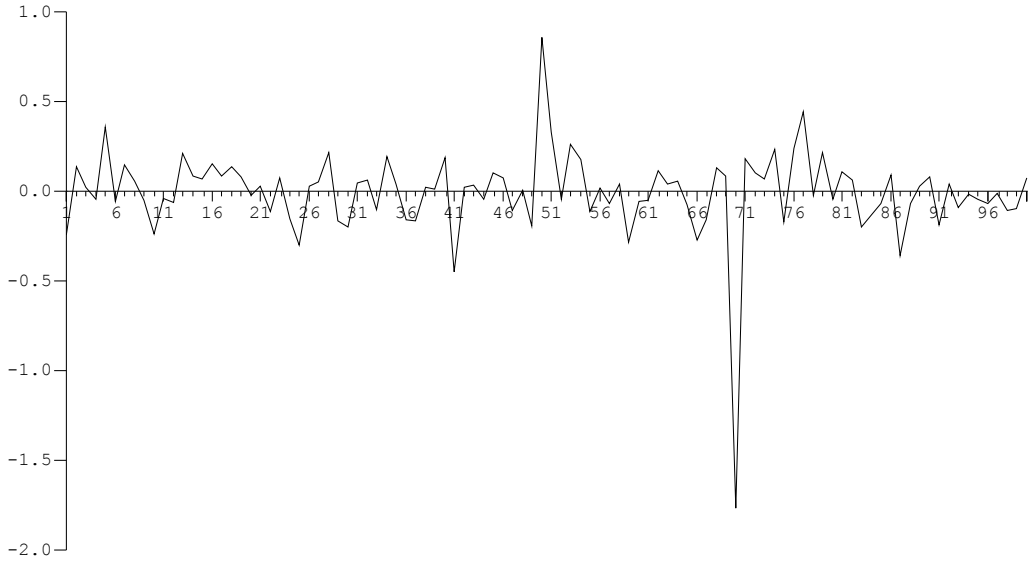
3. Outlier diagnostic.

A function dfbeta calculates 4 usual diagnostics for the influence of outliers on a regression. These diagnostics are: the dfbeta itself, that measures the studentized change in the least squares estimates when each observation is sequentially omitted from the regression; the dffits which measures the effect of such changes on the fitted values; the studentized residuals and the hat-matrix diagonals. These results are stored in a results tlist and if the user has not given as an argument 'noprint', they are plotted. Dfbetas, which are in same number than the coefficients, are plotted on one graphic windows each. Dffits, studentized residuals and hat-matrix diagonals are printed on the next graphic window. Here is the result of the command (where y is a (100x1) vector, x is a (100x6) matrix and 2 outliers have been artificially put at observations 50 and 70):

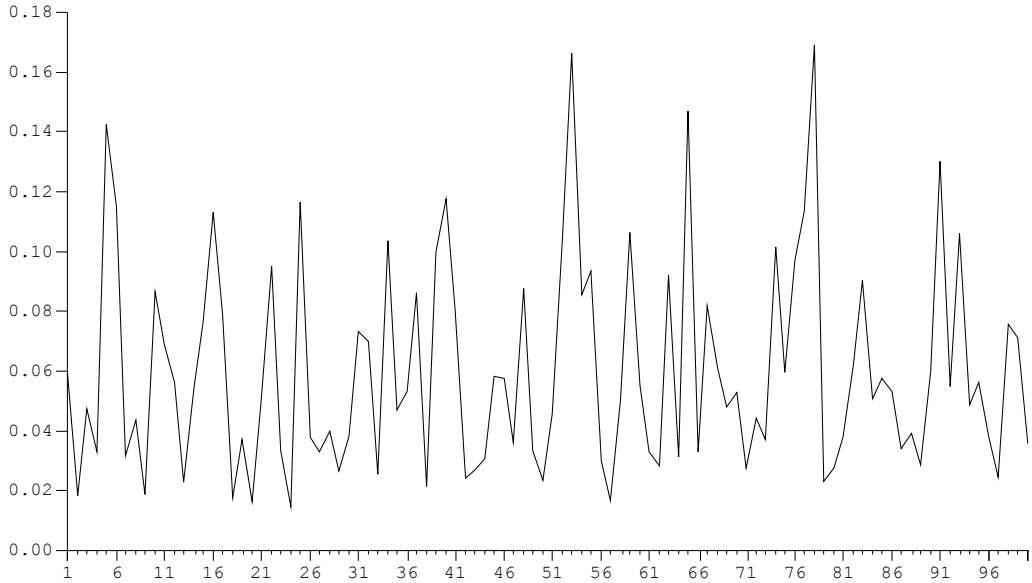
```
-->result = dfbeta('y','x');
```

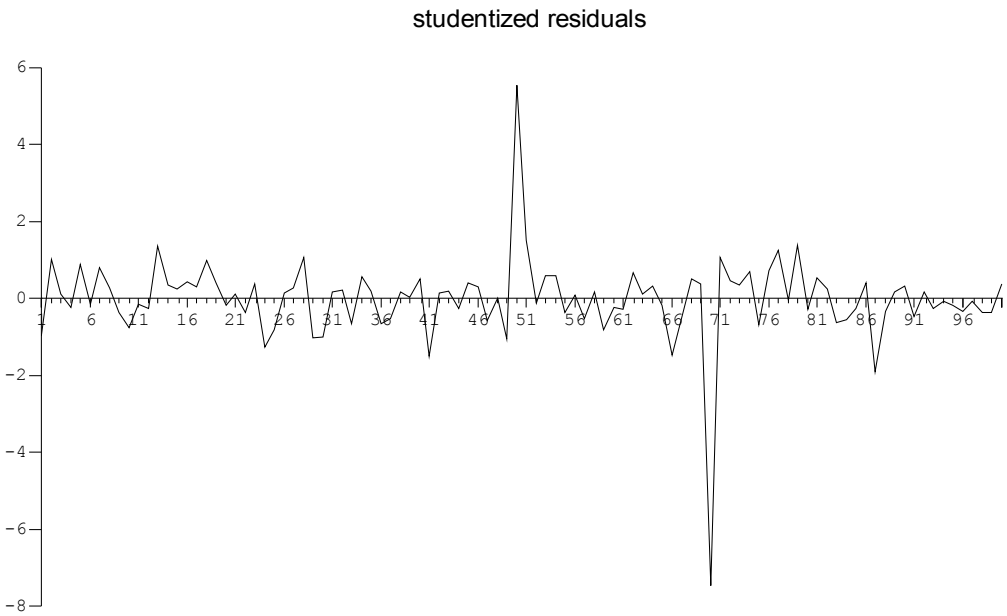


dffits



hat-matrix diagonals





4. Specification tests.

GROCER contains numerous specification tests that have been proposed in the literature. With only a few exceptions, discussed below, there are two functions performing each test. The first one, with suffix 0, provides only the statistics of the test, the corresponding p-value, the number of variables in the auxiliary regression (except for archz0, arlm0, chowtest0 and reset) and the R^2 of the auxiliary regression (except for chowtest0, hetero_sq0, and reset0). The other ones, without suffix 0, provide a tlist, that can be stored and printed later. They also display on screen the results of the test, unless the user has given the argument 'noprint'. Most tests are presented in R^2 form (to conform to an usual practice; White's famous test for instance has originally been presented in this form) and in Fisher form, which should be preferred in practice because it presents less small sample size distortion[3].

The existing tests are the following:

- Lagrange multiplier autocorrelation test of order p: arlm and arlm0;
- Arch test of order p: archz and archz0;
- Breusch and Pagan Heteroskedasticity test with second stage exogenous variables given by the user: bpagan and bpagan0;
- Xi^2 Heteroskedasticity test: hetero_sq and hetero_sq0;
- White Heteroskedasticity test: white and white0;
- Chow traditional stability test: chowtest and chowtest0;
- Chow in-sample predictive failure test: predfailin and predfailin0;
- out-of-sample predictive failure test: predfail;
- Brown, Durbin and Evans cusum and cusum-squared stability tests: cusumb (backward) and cusmf (forward);
- Jarque and Bera normality test: jbnorm and jbnorm0;
- Doornik and Hansen normality test: doornhans and doornhans0;

[3] see Kiviet (1986): "On the rigor of some mis-specification tests for modelling dynamic relationships", Review of Economic Studies, 53, 241–261.

- Ramsay RESET test for non linearity: reset and reset0.

Note the following exceptions: because of the specificity of the test, cusumb, cusumf and predfail do not exist in the "0-suffixed" form. Functions cusumf and cusmb use 2 subroutines: recresid, that calculates the recursive residuals and cusmq_tab, that extrapolates Durbin's table used to assess the confidence bands of the cusmq test.

As, an example, here are the tests of Hendry and Ericsson[4] equation #6:

```
--> bounds('1964q3','1989q2');rols=ols('del(lm1-lp)','del(lp)', 'del(lagts(1,lm1-
lp-ly))','rnet','lagts(1,lm1-lp-ly)','cte')
```

// estimation of equation #6

```
-->jbnorm(rols);
Jarque and Bera normality test:
chi2(2)=1.683552
(p -value                  =  .4309445)
```

```
-->doornhans(rols)

Doornik and Hansen normality test:
chi2(2)=1.9768208
[More (y or n ) ?]
(p -value                  =  .3721678)
```

// Jarque and Bera and Doornik and Hansen normality tests

```
-->arlm(rols,4);

Lagrange multiplier 1-4 autocorrelation test:
chi2(4)=7.1561938
(p -value                  =  .1278607)
```

```
Lagrange multiplier 1-4 autocorrelation test:
F(4,91)=1.9417464
(p -value                  =  .1102127)
```

// LM AR test with 4 lags

```
ARCH test:
chi2(4)=3.0104
(p -value                  = 0.5560865)
```

```
ARCH test:
F(4,92)=0.7780595
(p -value                  = 0.5422606)
```

// ARCH test with 4 lags[5]

[4] D.F Hendry et N.R Ericsson (1991): "Modelling the demand for narrow money in the United Kingdom and the United States", European Economic Review, p833-886.

[5] A correction has been made from the version 1.0 to the 1.1 one to the degrees of freedom of the test, following a similar correction made to pc-gets by D.F Hendry and H.M. Krolzig.

```
-->hetero_sq(rols);
Xi² heteroskedasticity test:
F(8,86)=1.3573147
(p -value                  =  .2267385)
```

// heteroskedasticity test from the regression of the square of ols residuals on the exogenous
// variables and their square

```
-->white(rols);

White heteroskedasticity test:
chi2(14)=15.475674
(p -value                  =  .3464303)


White heteroskedasticity test:
F(14,80)=1.0462364
(p -value                  =  .4180556)
```

// White's heteroskedasticity test

```
-->reset(rols,2);

power 2 non linearity RESET test:
F(1,94)= .0821166
(p -value                  =  .7750799)
```

// Ramsey RESET test

(°)Grocer contains the reliability diagnostic proposed by Hendry and Krolzig in their software pcgets: it has been incorporated in grocer equivalent to pcgets, the function automatic, but it can be used as well independently. The reliability diagnostic rests on the idea that if a variable belongs to a model, then its student T is diverging away from 0; so if a variable belongs to the model, it should be significant over the whole sample as well over both halves of the sample. Therefore the reliability function calculates the Student's t of each variable and according to the periods over which they are significant returns the confidence one can have in the chosen variable.

The simplest call to function reliability involves the two following arguments:

- a results tlist taken from an ols estimation;
- a (4x1) vector w giving the percentages the user wants to give to the following cases: a variable significant only on a sub-sample; a variable significant only on the sub-samples; a variable significant only on the whole sample; a variable significant on the whole sample and on a sub_sample; note that when a variable is significant over these 3 samples, it is assumed to be 100% reliable and when it is significant over none of these, it is assumed to have 0% reliability.

Calculating the reliability of the variables in hendryericsson demo can be done as follows:

```
-->r=hendryericsson();reliability(r,[0.3 0.6 0.4 0.7])
```

which gives the following result:

```
variable                  reliability

delts(lp)                 1
```

```
delts(lagts(1,lm1-lp-ly)) 0.7
rnet                      1
lagts(1,lm1-lp-ly)        1
cte                       1
```

The user can also enter two options: as a third argument, the size level (by default, as in the above example, it is set to 0.05); and as a fourth argument, 'noprint' if she does not want to print the results.

5. Tests of forecasting properties.

Grocer contains three tests proposed to evaluate forecasts: the test of directional accuracy proposed by Pesaran & Timmerman[6] (function ddchange); the tests of Diebold-Mariano (function diebmar) and Clark and West (function clwest) that test the equivalence of two non-nested and nested forecasts respectively.

The test of directional accuracy tests if a forecast is good at predicting the sign of a variable. The function dchange takes therefore as its first argument the variable to forecast and as its second argument the forecast itself. There is also a third, optional, argument, which is 'noprint', that the user can give if she does not want to print the results. Take for instance the change in the "level of future own production in the next three month" and the "level of past own production over the past three months" in database BusinessSurvey.dat in your Scilab library macros\grocer\db: the first series should be a good predictor of the second series three months ahead. Using Pesaran & Timmerman's test to check this property can therefore be done as follows:

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

-->rd = dchange('delts(dpp)','lagts(3,delts(dpf))');
```

With the following result:

```
Pesaran-Timmermann test of directional change

H0 : dpp & lagts(3,dpf) are independently distributed
estimation period: 1976m7-2004m10

right propor. pred. Stat. Sn  p-value
0.6                 3.6919749 0.0001113

                        *
                    *     *
```

And the test confirms that the sign of the change of the assessment of industrial firms on the "level of future own production in the next three month" is a good predictor of the sign of the "level of past own production over the past three months". Results are stored in a tlist which contains the values of the 2 series (fields 'y' and 'yc'), their names (fields 'namey' and 'nameyc'), the percentage of signs that are correctly predicted (field 'P'), the value of the test (field 'Pstar') and its pvalue (field 'pvalue'). In the example above rd('pvalue') gives therefore the test pvalue.

[6] Pesaran, M.H. and A. Timmerman (1992), "A Simple Nonparmetric Test of Predictive Performance", *Journal of Business and Economic Statistics*, Vol. 10, 4, 451-465.

The test proposed by Diebold and Mariano[7] to compare the predictive accuracy of two competing forecasts is done by function diebmar, which implements also the small sample correction proposed by Harvey, Leybourne and Newbold[8]. The simplest call to function diebmar takes the following form:
r=diebmar(y,yf1,yf2)
where y is the series to be forecasted, yf1 the benchmark forecast and yf2 the forecast which performance is to be compared with the benchmark one.

If the user wants to apply Harvey, Leybourne and Newbold small sample correction, the she must enter a fourth argument and set it to %t:
r=diebmar(y,yf1,yf2,%t)
(r=diebmar(y,yf1,yf2,%f) is therefore equivalent to r=diebmar(y,yf1,yf2)).

If the user wants to change the default truncation point of the Barlett window used to compute the variance of the difference between the 2 forecasts, she has to enter the desired truncation point (say trunc) of the Barlett window:
r=diebmar(y,yf1,yf2,%t,trunc) if she wants to apply Harvey, Leybourne and Newbold small sample correction and r=diebmar(y,yf1,yf2,%f,trunc) if she does not want to.

A last option is available: if the user does not want to print the results, then she has to enter 'noprint' (or %f) as a sixth argument.

As an example, take the French industrial production and again the he "level of future own production in the next three month" and the "level of past own production over the past three months" in database Industrial.dat in your Scilab library macros\grocer\db and, from function dibemar_d the one-step ahead forecast made from an AR(2) (series y_bench) as the benchmark series and the one-step ahead forecast made with these series as the competing series (series y_indic).
then:

```
-->bounds('2000q1','2003q4')
-->rdb=diebmar('delts(log(y))','y_bench','y_indc')
```

is the easiest call in this context to diebmar.

It provides:

```
        Diebold-Mariano test

Benchmark serie: y_bench
Competing serie: y_indc
Truncation lag of the Barlett window:10
forecasting period: 2000q1-2003q4

Bench. MSE Comp. MSE DM-Stat.  p-value
0.0001463  0.0000827 2.0755579 0.0189674

                          *
                      *     *
```

Making Harvey, Leybourne and Newbold small sample correction is made as follows:

```
-->rdb=diebmar(delts(log(y)),y_bench,y_indic,%t)
```

[7] Diebold F. X. et Mariano R. S. (1995). "Comparing Predictive Accuracy", *Journal of Business and Economic Statistics*, 13, pp. 253-265.
[8] Harvey, D.I., S.J. Leybourne & P. Newbold (1998): "Tests for Forecast Encompassing", *Journal of Business and Economic Statistics*, 16, 254-259

with the following result:

```
        Diebold-Mariano test with small sample correction
truncation lag of the Barlett window:10
forecasting period: 2000q1-2003q4

Bench. MSE Comp. MSE DM-Stat.  p-value
0.0001463  0.0000827 0.8406743 0.2068655

                         *
                      *     *
```

Lastly, imposing a truncation at period 4 to the Barlett widow is done as follows:

```
-->rdb=diebmar(delts(log(y)),y_bench,y_indic,%t,4)
```

with the following result:

```
        Diebold-Mariano test with small sample correction
truncation lag of the Barlett window:4
forecasting period: 2000q1-2003q4

Bench. MSE Comp. MSE DM-Stat. p-value
0.0001463  0.0000827 1.238839 0.1172207

                         *
                      *     *
```

Contrary to Diebold and Mariano test, Clark and West[9] test applies to nested models, with the constrained model the reference (H0) model and the more general one the competing one. Because of the possibility to take the martingale difference as the H0 model, the syntax of function clwest is somehow diffrent from the one of function diebmar. The simplest call to clwest is however similar:
r=clwest(y,yf1,yf2)
If you want to keep the names of the variables, then enter yf1 and yf2 between quotes (you can also enter them directly in a vector ['yf1' 'yf2'])

You can also, as in Clark and West first paper[10], directly test a forecast against a martingale hypothesis and run:
r=clwest(y,yf1,'mdh')
where 'mdh' stands for Martingale Difference Hypothesis

Lastly you can also apply Clark and West test to overlapping forecasts; you have in that case to enter the option 'overlap=xx' with xx is the number of overlapping terms in the forecast.

And as usual you can enter 'noprint' if you do not want to enter the results of the test.

As an example, take again the industrial database:

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

Make rolling forecasts with 1979q1 as the starting period of estimation and ending estimation going from 1992q1 to 2003Q4:

[9]Todd E. Clark & Kenneth D. West (2006), "Approximately normal test for equal predictive accuracy in nested models", NBER Technical Paper, 326
[10]Todd E. Clark & Kenneth D. West (2005), "Using out-of-sample mean squared prediction errors to test the martingale difference hypothesis", *Journal of Econometrics*.

```
-->bounds('1979q1','2003q4');
-->rs = rolreg('delts(log(y))','lagts(ypast)','delts(lagts(yfut))','cte',...
-->      'dates=[''1992q1'' ''2003q4'']','hstep=1')
```

Then, test the predictive accuracy of these forecasts against a martingal hypothesis (note that the bounds are imposed :

```
-->y_short = rs('yhat') ;
-->bounds('1992q1','2003q4')
-->rscl=clwest('delts(log(y))','y_short','bench=''mdh''') ;
```

With the following result:

```
     Clark-West test of martingal difference hypothesis

Random walk series: delts(log(y))
Series of forecasts: y_short
Forecasting period: 1992q1-2003q4

MSE-adj.  CW-Stat.  p-value
0.0001913 4.3884761 0.0000057

                          *
                       *     *
```

The test is clearly rejected: the forecast has more predictive accuracy than a martingale.

(°°)

6. The tests functions and their specifications.

archz_____ARCH test

CALLING SEQUENCE

[rarch]=archz(results,p,np)

PARAMETERS

INPUT:
* results = results tlist from a first stage estimation
* p = # of lag of squared residuals in the second stage estimation
* np = 'noprint' if the user does not want to print the results
---

OUTPUT:
rarch= a typed list with :
. rarch('meth') = 'arch'
. rarch('r1st') = results of the first step regression (allows the "traceability" of the results)
. rarch('chistat') = the value of the chi2 statistics
. rarch('chi_pvalue') = the corresponding p-value
. rarch('chi_df') = the corresponding degrees of freedom
. rarch('fstat') = the value of the Fisher statistics

. rarch('pfstat') = the corresponding p-value
. rarch('dfnum') = degrees of freedom of the numerator
. rarch('dfden') = degrees of freedom of the denominator

DESCRIPTION

Computes a test for AutoRegressive Conditional Heteroskedasticty (ARCH) of order p. Results are stored in a tlist and displayed on screen.

Example:
rarch=archz(rols,4)

This example is taken from hendryericsson. It provides the ARCH test of order 4 for Hendry and Ericsson equation #6, whose estimation result has been saved in tlist rols.

archz0_____ARCH test

CALLING SEQUENCE

[f,f_pvalue,r2]=archz0(results,p,np)

PARAMETERS

INPUT:
* results = results tlist from a first stage estimation
* p = # of lag of squared residuals in the second stage estimation
* np = 'noprint' if the user does not want to print the results
---
OUTPUT:
* f = value of the Goldfeld-Quandt F test
* f_pvalue = its p-value
* r2 = R^2 of the auxilliary regression

DESCRIPTION

Computes a test for ARCH(p). Results are saved as numbers and not displayed on screen.

Example:
[f,f_pvalue,r2]=archz0(results,p)

This example is taken from archz. Useful mainly for programming purpose (since archz does much more).

arlm_____Lagrange multiplier autocorrelation test

CALLING SEQUENCE

[resulbp]=arlm(resul1,p,np)

PARAMETERS

INPUT:
* resul1 = results tlist from a first stage estimation
* p = # of lag of residuals in the second stage estimation
* np = 'noprint' if the user does not want to print the results
---
OUTPUT:
resulbp = results tlist with:
. resulbp('meth') = 'archtest'
. resulbp('resul1st') = resul1
. resulbp('f') = fstat
. resulbp('p') = p
. resulbp('df') = df
. resulbp('f_pvalue')=f_pvalue

DESCRIPTION

Computes a Lagrange multiplier autocorrelation test of order p (see Godfrey, L. G. (1978). Testing for higher order serial correlation in regression equations when the regressors include lagged dependent variables. Econometrica, 46, 1303–1313). Results are stored in a tlist and displayed on screen.

Example:
rbp=arlm(rols,4)

This example is taken from hendryericsson. It provides the Lagrange multiplier autocorrelation test of order 4 for Hendry and Ericsson preferred equation, whose estimation result has been saved in tlist rols.

arlm0_____Lagrange multiplier autocorrelation test

CALLING SEQUENCE

[fstat,f_pvalue,r2]=arlm0(resulols,p,np)

PARAMETERS

INPUT:
* resul1 = results tlist from a first stage estimation
* p = # of lag of residuals in the second stage estimation
* np = unused argument (but put here for compatibility with other testing functions)

OUTPUT:
* fstat = value of the statistic
* f_pvalue = its p-value

DESCRIPTION

Computes a Lagrange multiplier autocorrelation test of order p (see Godfrey, L. G. (1978). Testing for higher order serial correlation in regression equations when the regressors include lagged dependent variables. Econometrica, 46, 1303–1313).. Contrary to arlm, output does not take the form of a tlist, but of the Fisher test and its p value and nothing is printed.

Example:
[fstat,f_pvalue,r2]=arlm0(resul1,p)

This example is taken from arlm. Useful mainly for programming purpose (since arlm does much more).

bkw_____BKW multicollinearity diagnostic

CALLING SEQUENCE

[condindex]=bkw(arg1,…,argn)

PARAMETERS

INPUT:
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 print the results of the regression
---
OUTPUT:
condindex = the condition number

DESCRIPTION

Computes and prints BKW collinearity variance-decomposition proportions matrix.

Example:
bkw('x')

This example is taken from bkw_d. Here the exogenous variables take the form of a matrix x; it is entered between quotes, so the exogenous variables will be called x_1, x_2,…

bkw_scale_____BKW multicollinearity diagnostic

CALLING SEQUENCE

[condindex]=bkw_scale(arg1,…,argn)

PARAMETERS

INPUT:
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 print the results of the regression
---
OUTPUT:
condindex = the condition number

DESCRIPTION

Computes and prints BKW collinearity variance-decomposition proportions matrix, each column being normalized to have a unit norm

Example:
Bkw_scale('x')

On the data used in bkw_d, the exogenous variables take the form of a matrix x; it is entered between quotes, so the exogenous variables will be called x_1, x_2,…

bkwols_____BKW multicollinearity diagnostic

CALLING SEQUENCE

[condindex]= bkwols(arg1,…,argn)

PARAMETERS

INPUT:
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 print the results of the regression
---
OUTPUT:
condindex = the condition number

DESCRIPTION

Computes BKW COND index of a x matrix, each column being normalized to have a unit norm. Contrary to bkw_scale, this function does not print the variance-decomposition matrix.

Example:
condindex=bkwols(x)

This example is taken from ols. Useful mainly for programming purpose (since bkw_scale does much more).

bpagan_____Breusch and Pagan heteroskedasticity test

CALLING SEQUENCE

[resulbp]=bpagan(resulols,arg1,…,argn)

PARAMETERS

INPUT:
* resulols = results tlist from a first stage estimation
* argi = arguments of the second stage regression, 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 print the results of the regression
---
OUTPUT:
resulbp = a results tlist with:
- resulbp('meth') = 'bpagan'
- resulbp('resul1st') = results tlist of the first stage regression
- resulbp('u2') = sum of square residuals from the first stage regression
- resulbp('namex2') = names of the exogenous variables in the second stage regression
- resulbp('x2') = vector of the exogenous variables in the second stage regression
- resulbp('f') = Breush-Pagan LM-statistic (Fisher form)
- resulbp('p') = p-value of the test
- resulbp('dfnum') = degrees of freedom of the numerator
- resulbp('dfden') = degrees of freedom of the denominator
- resulbp('f_pvalue) = p-value of the test

DESCRIPTION

Computes Breusch and Pagan heteroskedasticity test with the exogenous second stage variables provided by the user (see Breusch, T.S. and A.R. Pagan (1979): "A simple test for heteroskedasticity and random
coefficient variation", Econometrica 47, 1287-1294). Results are stored in a tlist and displayed on screen.

Example:
bpagan(rols,'del(lp)','del(lagts(1,lm1-lp-ly))','rnet','lagts(1,lm1-lp-ly)','cte','(del(lp))^2', '(del(lagts(1,lm1-lp-ly)))^2','(rnet)^2','(lagts(1,lm1-lp-ly))^2')

This example is taken from bpagan_d. The first stage result comes from the estimation of Hendry and Ericsson equation # 6. The second stage exogenous variables are the squares of all exogenous variables of the first stage regression.

bpagan0_____Breusch and Pagan heteroskedasticity test

CALLING SEQUENCE

[nvar2,f,f_pvalue,r2]=bpagan0(u2,x2,nvar)

PARAMETERS

INPUT:
* u2 = the first stage regression squared residuals
* x2 = the exogenous variables of the Breusch and Pagan second stage regression
* nvar = number of exogenous variables in the first satge regression
---
OUTPUT:
* nvar2 = # of exogenous variables of the Breusch and Pagan second stage regression
* f = value of the Goldfeld-Quandt F test
* f_pvalue = its p-value
* r2 = R^2 of the auxilliary regression

DESCRIPTION

Computes Breusch and Pagan heteroskedasticity test with the exogenous second stage variables provided by the user (see Breusch, T.S. and A.R. Pagan (1979): "A simple test for heteroskedasticity and random
coefficient variation", Econometrica 47, 1287-1294). Results are given only in constant form and not printed.

Example:
[nvar2,f,f_pvalue]=bpagan0(u2,x2)

This example is taken from function bpagan. Useful mainly for programming purpose (since bkw does much more).

chowtest_____Chow usual test

CALLING SEQUENCE

[resulch]=chowtest(resulols,n1,np)

PARAMETERS

INPUT:
* resulols = results tlist from a first stage estimation
* n1 = # of observations of the first sub-period
* np = 'noprin' if the user does not want to display the results
---
OUTPUT:
resulch = a results tlist with:
. resulch('meth') = chow
. resulch('rols') = the tlist results from first stage estimation
. resulch('fstat') = value of the test statistic
. resulch('f_pvalue') = its p-value
. resulch('dfnum') = # dof of the numerator
. resulch('dfden') = # dof of the denominator
. resulch('cut') = date (if there is a ts in the regression) or index of the observation (if there is no ts in the regression) of the cut

DESCRIPTION

Computes Chow usual test with a break at observation n1. Results are saved in a tlist and displayed on screen.

Example:
r=ols('del(lm1-lp)','lagts(lm1-lp-ly)','del(lp)','rnet','del(lagts(lm1-lp-ly))','cte'); chowtest(r,50)

This example is taken from function chowtest_d. The Chow test on Hendry and Ericsson equation # 6 is provided for a break at observation # 50.

chowtest0_____Chow usual test

CALLING SEQUENCE

[fstat,f_pvalue,dfnum,dfden]=chowtest0(resulols,n1,np)

PARAMETERS

INPUT:
* resulols = a results tlist from a first stage estimation
* n1 = # of observations of the first sub-period

OUTPUT:
* fstat = value of the test statistic
* f_pvalue = its p-value
* dfnum = # dof of the numerator
* dfden = # dof of the denominator

DESCRIPTION

Computes Chow usual test with a break at observation n1. Results are saved as numbers and not displayed on screen.

Example:
[fstat,f_pvalue,dfnum,dfden]=chowtest0(resulols,n1)

This example is taken from function chowtest. Useful mainly for programming purpose (since chowtest does much more).

(°)**clwest_____Clark and West Test for Equal Predictive Accuracy in Nested Models**

CALLING SEQUENCE

rcw = clwest(namey,arg1,...,argn)

PARAMETERS

INPUT:
* namey = observed vector or ts
WARNING: in case of of martingale difference test combined with overlapping regression (for computational purpose):
(1) this series should be in first difference: overlapping terms are built by summation
Ex: If y(t)-y(t-4) = d(y(t))+d(y(t-1))+d(y(t-2))+d(y(t-3)) is the series to be forecasted it's d(y) that must be entered
(2) the series of forecasts should be in the overlapping form (y(t)-y(t-4) in the example)
* arg1,...,argn: arguments that can be:
- 1 or 2 time series according to the test specified (first series should be the benchmark in case of nested model)
- a real (nx1) or (nx2) vector
- a string equal to the name of a time series or a (nx1)/(nx2) real vector or matrix between quotes
- a list of such elements
- 'bench = "mdh'" if the user want to test martingale difference hypothesis otherwise 2 nested model are tested (default, 'nm')
-'overlap=xx' : if the user number of overlapping terms if the user estimated overlapping regression
- the string 'noprint' if the user doesn't want to display the results of the regression
---
OUTPUT:
rcw = a results tlist with:
- rcw('meth') = 'Clark-West' / 'Clark-West - martingal difference'
- rcw('y') = input forecasted series
- rcw('x') = forecasts
- rcw('namey') = name of forecasted serie
- rcw('namex') = names of forecasts
- rcw('stat') = value of the Clark-West statistics
- rcw('pvalue') = p-value of the Clark-West statistics
- rcw('overlap') = number of overalapping terme if any (0 otherwise)

- rcw('prests') = flag for the presence of time series
- rcw('bounds') = if there is a timeseries in the forecast, the bounds of the regression

DESCRIPTION

A function that computes Clark & West (2006) approximately normal test for equal predictive accuracy in nested model and it's special case to test martingale difference hypothesis (Clark & West, 2005).

Examples:
1) rscl=clwest('delts(log(y))','y_short') ;
2) rlcl=clwest('delts(log(y))','y_long','overlap=4') ;
3) rscl=clwest('delts(log(y))','y_short2','y_short','bench="nm"') ;

Example 1 taken from function clwest_d(). The forecasted series is delts(log(y)).
In example 2, forecasts are computed by mean of a long horizon (overlapping) regression.
In example 3, 2 nested model are compared where 'y_short2' is defined as the benchmark series
(°°)

cusumb_____backward cusum stability tests

CALLING SEQUENCE

[rcusum]=cusumb(y,arg1,…,argn)

PARAMETERS

* grocer_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 (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 print the results of the regression
. the string 'size=x' where x=0.01, 0.05 or 0.1 is the size chosen for the test (default =0.05)

OUTPUT:
rcusum = a results tlist with:
. rcusum('meth')='cusum'
. rcusum('nobs')= # of observations
. rcusum('nvar')= # of variables
. rcusum('y')= y data vector
. rcusum('x')= x matrix vector
. rcusum('rres')= vector of recursive residuals
. rcusum('cusum')= cusum test
. rcusum('cusum_l90')= the lower value of its 90% confidence interval
. rcusum('cusum_u90')= the upper value of its 90% confidence interval
. rcusum('cusum_l95')= the lower value of its 95% confidence interval
. rcusum('cusum_u95')= the upper value of its 95% confidence interval
. rcusum('cusum_l99')= the lower value of the 99% confidence interval
. rcusum('cusum_u99')= the upper value of the 99% confidence interval

. rcusum('cusums')= squared cusum test
. rcusum('cusums_l90')= the lower value of its 90% confidence interval
. rcusum('cusums_u90')= the upper value of its 90% confidence interval
. rcusum('cusums_l95')= the lower value of its 95% confidence interval
. rcusum('cusums_u95')= the upper value of its 95% confidence interval
. rcusum('cusums_l99')= the lower value of the 99% confidence interval
. rcusum('cusums_u99')= the upper value of the 99% confidence interval
. rcusum('prests')=boolean indicating the presence or absence of a time series in the regression
. rcusum('namey') = name of the y variable
. rcusum('namex') = name of the x variables
. rcusum('bounds') = if there is a timeseries in the regression, the bounds of the test (which are the bounds of the regression, less the k first dates)

DESCRIPTION

Computes backward cusum and cusum-squares test and plots the tests values along with their 5% confidence bands (see Brown R.L, J. Durbin and J.M Evans (1975):"Techniques for Testing the Constancy of Regression Relationship over Time", Journal of the Royal Statistical Society, Series B, n°2, 149-192).

Example:
load('grocer/bdexamples/bdhenderic.dat') ; bounds('1964q3','1985q2') ; r=cusumb('del(lm1-lp)','del(lp)','del(lagts(1,lm1-lp-ly))','rnet','lagts(1,lm1-lp-ly)','cte')

This example is taken from function cusum_d. The example provides the cusum backward stability test for the estimation of Hendry and Ericsson equation # 6.

cusumf_____forward cusum stability tests

CALLING SEQUENCE

[rcusum]=cusumf(y,arg1,…,argn)

PARAMETERS

INPUT:
* grocer_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 the to print the results of the regression
. the string 'size=x' where x=0.01, 0.05 or 0.1 is the size chosen for the teste (default =0.05)

OUTPUT:
rcusum = a results tlist with:
. rcusum('meth')='cusum'
. rcusum('nobs')= # of observations
. rcusum('nvar')= # of variables
. rcusum('y')= y data vector
. rcusum('x')= x matrix vector
. rcusum('rres')= vector of recursive residuals
. rcusum('cusum')= cusum test
. rcusum('cusum_l90')= the lower value of its 90% confidence interval
. rcusum('cusum_u90')= the upper value of its 90% confidence interval
. rcusum('cusum_l95')= the lower value of its 95% confidence interval
. rcusum('cusum_u95')= the upper value of its 95% confidence interval
. rcusum('cusum_l99')= the lower value of the 99% confidence interval
. rcusum('cusum_u99')= the upper value of the 99% confidence interval
. rcusum('cusums')= squared cusum test
. rcusum('cusums_l90')= the lower value of its 90% confidence interval
. rcusum('cusums_u90')= the upper value of its 90% confidence interval
. rcusum('cusums_l95')= the lower value of its 95% confidence interval
. rcusum('cusums_u95')= the upper value of its 95% confidence interval
. rcusum('cusums_l99')= the lower value of the 99% confidence interval
. rcusum('cusums_u99')= the upper value of the 99% confidence interval
. rcusum('prests')=boolean indicating the presence or absence of a time series in the regression
. rcusum('namey') = name of the y variable
. rcusum('namex') = name of the x variables
. rcusum('bounds') = if there is a timeseries in the regression, the bounds of the test (which are the bounds of the regression, less the k first dates)

DESCRIPTION

Computes forward cusum and cusum-squares test and plots the tests values along with their 5% confidence bands (see Brown R.L, J. Durbin and J.M Evans (1975):"Techniques for Testing the Constancy of Regression Relationship over Time", Journal of the Royal Statistical Society, Series B, n°2, 149-192).

Example:
load('grocer/bdexamples/bdhenderic.dat') ; bounds('1964q3','1985q2') ; r=cusumf('del(lm1-lp)','del(lp)','del(lagts(1,lm1-lp-ly))','rnet','lagts(1,lm1-lp-ly)','cte')

This example is taken from function cusum_d. The example provides the cusum forward stability test for the estimation of Hendry and Ericsson equation # 6.

cusumq_tab_____cusum squared confidence band

CALLING SEQUENCE

[s] = cusumsq_tab(nobs)

PARAMETERS

INPUT:
nobs = # of observations

OUTPUT:
s = (3x1) vector of (1%, 5%,10%) confidence bounds

REFERENCES
* Durbin 1969 'Test for Serial Correlation in Regression Analysis based on the Periodogramm of Least Squares Residuals', Biometrika, 56, 1-15
(°)* Edgerton, David & Wells, Curt, 1994. "Critical Values for the Cusumsq Statistic in Medium and Large Sized Samples",Oxford Bulletin of Economics and Statistics, BlackwellPublishing, vol. 56(3), pages 355-65, August.(°°)

DESCRIPTION

Gives the 1%, 5% and 10% stat level of the cusum-squared statistics. (°)Durbin's table is used when values exist and Edgerton and Wells ones for all other values of nobs.(°°)

Example:
s=cusumsq_tab(nobs)

This example is taken from function cusumb. Used in cusumb and cusumf.

(°)
dchange_____Test for directional accuracy

CALLING SEQUENCE

rdb = dchange(y,yb,prt)

PARAMETERS

INPUT:
* y = vector or ts of observable or benchmark series
* yb = vector or ts of competing or forecast series
* prt = %f if the user does not want to print the results (optional)

OUTPUT:
rdb = a results tlist with:
- rchange('meth') = 'direct. change'
- rchange('y') = benchmark or observable serie s
- rchange('yc') = competing series or forecasted series
- rchange('namey') = name of benchmark or observable series
- rchange('nameyc') = name of the competiting series or forecasted series
- rchange('P') = proportion of time the sign of the observable value is correctly forecasted
- rchange('Pstar') = concordance index
- rchange('pvalue')= pvalue of the statistic

DESCRIPTION

A function that computes Pesaran & Timmerman nonparametric test for directional accuracy.

Example:
rdb=diebmar('dpp','lagts(3,dpf)')

Example taken from function dchange(): tests in French business survey if changes in "level of future own production" are a good predictors of that of "level of past own production".
(°°)

dfbeta_____BKW influential observation diagnostics

CALLING SEQUENCE

[results]=dfbeta(y, arg1,…,argn)

PARAMETERS

INPUT:
* y = either an ols results tlist or a 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:
  . the string 'noprint' if the user doesn't want to print the results of the regression
  or, and only if y is not an ols results tlist:
  . a time series
  . a real (nx1) vector
  . a string equal to the name of a time series or a (nx1) real vector between quotes
---
OUTPUT:
results = a results tlist with
- results('meth') = 'dfbeta'
- results('nobs') = # of observations
- results('nvar') = # of variables in x-matrix
- results('dfbeta') = df betas
- results('dffits') = df fits
- results('hatdi') = hat-matrix diagonals
- results('stud') = studentized residuals
- result('namex') = name of the x variables
- result('namey') = name of the y variable
- result('prests') = boolean indicating the presence or absence of a time series in the regression

DESCRIPTION

Computes BKW (influential observation diagnostics) dfbetas, dffits, hat-matrix, studentized residuals. Plots the corresponding results.

Example:
result = dfbeta('y','x')

This example is taken from function dfbeta_d. The example provides dfbetas, dffits, hat-matrix, studentized residuals for a regression whose endogenous variable is y and exogenous variables is a matrix (as in dfbeta_d), a vector or a ts x.

(°)**diebmar_____Diebold and Mariano test for comparing predictive accuracy**

CALLING SEQUENCE

rdb = diebmar(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = observed vector TS: 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:
- 2 time series (the first series is the benchmark model)
- a real (nx2) vector
- a real (nx2) matrix
- a string equal to the name of a time series or a (nx2) real vector or matrix between quotes
- a list of such elements
- 'smallc =1' if the user want to perform a small sample correction will be done (optional; default = 0: no small sample correction)
- 'trunc = xx' truncation point of the bartlett window (optional; default = floor(5*nobs^0.25))
- the string 'noprint' if the user doesn't want to display the results of the regression
---
OUTPUT:
rdb = a results tlist with
- rdb('meth') = 'diebmar'
- rdb('y') = input forecasted series
- rdb('x') = forecasts resp. under the null hypothesis & under the alternative hypothesis
- rdb('namey') = name of forecasted serie
- rdb('namex') = names of forecasts resp. under the null hypothesis & under the alternative hypothesis
- rdb('mse') = vector of MSE
- rdb('stat') = value of the Diebold-Mariano statistics
- rdb('pvalue') = p-value of the Diebold-Mariano statistics
- rdb('smallc') = flag for a small sample correction
- rdb('trunc') = truncation lag of the Barlett window
- rdb('prests') = flag for the presence of time series
- rdb('bounds') = if there is a time series in the forecast, the bounds of the regression

DESCRIPTION

A function that computes Diebold and Mariano test for predictive accuracy while allowing for small sample correction as proposed by Harvey et al.

Example:
1) rdb=diebmar('delts(log(y))','y_bench','y_indic','smallc=1','trunc=4")

2) rdb=diebmar('delts(log(y))','y_bench','y_indic','smallc=1')
3) rdb=diebmar('delts(log(y))','y_bench','y_indic')

Examples taken from function diebmar_d(). The forecasted series is delts(log(y)), the benchmark series is y_bench, the alternative one is y_indic.

doornhans_____Doornik and Hansen normality test

CALLING SEQUENCE

[rdoornhans]=doornhans(res,np)

PARAMETERS

INPUT:
* res = a result tlist
* np= the string 'noprint' if the user doesn't want to print the results of the test
---
OUTPUT:
rdoornhans= a typed list with :
. rdoornhans('meth') = 'doornhans'
. rdoornhans('r1st') = results of the first step regression (allows the "tracability" of the results)
. rdoornhans('chistat') = the value of the chi2 statistics
. rdoornhans('chi_pvalue') = the corresponding p-value
. rdoornhans('chi_df') = the corresponding degrees of freedom

DESCRIPTION

Computes "omnibus" normality test (see J. Doornik and H. Hansen (1994) : "A practical test for univariate and multivariate normality", Discussion Paper, Nuffield College). Results are stored in a tlist and displayed on screen if the user has not given as second argument 'noprint'.

Example:
doornhans(rols)

This example is taken from function hendryericsson. The example provides Doornik and Hansen normality test for the estimation of Hendry and Ericsson equation # 6, whose results have been stored in tlist rols.

doornhans0_____Doornik and Hansen normality test

CALLING SEQUENCE

[dh,pn]=doornhans0(res,np)

PARAMETERS

INPUT:
* res = a result tlist
* np = unused argument put for compatibility with other testing functions
---
OUTPUT:
* dh = the value of the test
* pn = its p-value

DESCRIPTION

Computes "omnibus" normality test (see J. Doornik and H. Hansen (1994) : "A practical test for univariate and multivariate normality", Discussion Paper, Nuffield College). Results are stored as numbers and are not displayed on screen.

Example:
[dh,pn]=doornhans0(res)

This example is taken from function doornhans. Useful mainly for programming purpose (since doornhans does much more).

hetero_sq_____Xi² heteroskedasticity test

CALLING SEQUENCE

[resulh]=hetero_sq(r,np)

PARAMETERS

INPUT:
* r = results tlist from a first stage estimation
* np = 'noprint' if the user does not want to print the results
---
OUTPUT:
resulh = a results tlist with:
- resulth('meth') = 'hagan'
- resulth('resul1st') = results tlist of the first stage regression
- resulth('f') = Xi² Hetero statistics
- resulth('p') = p-value of the test
- resulth('dfnum') = degrees of freedom of the numerator
- resulth('dfden') = degrees of freedom of the denominator
- resulth('f_pvalue) = p-value of the test

DESCRIPTION

Computes the value of the Xi² hetero test. Results are stored in a tlist and displayed on screen if the user has not given as second argument ‘noprint’.

Example:
hetero_sq(rols)

This example is taken from function hendryericsson. The example provides the Xi² heteroskedasticity test for Hendry and Ericsson equation # 6, whose results have been stored in tlist rols.

hetero_sq0_____Xi² heteroskedasticity test

CALLING SEQUENCE

[f,f_pvalue,nvar2]=hetero_sq0(r,np)

PARAMETERS

INPUT:
* r = results tlist from a first stage estimation
* np = unused argument put for compatibility with other testing functions
---
OUTPUT:
* f = value of the Xi² hetero F test
* f_pvalue = its p-value

DESCRIPTION

Computes the value of the Xi² hetero test. Results are stored as numbers and are not displayed on screen.

Example:
[f,f_pvalue,nvar2]=hetero_sq0(r)

This example is taken from function hetero_sq. Useful mainly for programming purpose (since hetero_sq does much more).

jbnorm_____Jarque and Bera normality test

CALLING SEQUENCE

[rjbnorm]=jbnorm(res,np)

PARAMETERS

INPUT:
* res = a result tlist
* np= the string 'noprint' if the user doesn't want to print the results of the test.

OUTPUT:
rjbnorm= a typed list with :
  . rjbnorm('meth') = 'jbnorm'

. rjbnorm('r1st') = results of the first step regression (allows the "traceability" of the results)
. rjbnorm('chistat') = the value of the chi2 statistics
. rjbnorm('chi_pvalue') = the corresponding p-value
. rjbnorm('chi_df') = the corresponding degrees of freedom

DESCRIPTION

Computes Jarque and Bera normality test (see Jarque, C. M., and Bera, A. K. (1980). ' Efficient tests for normality, homoskedasticity and serial independence of regression residuals', Economics Letters, 6, 255–259). Results are stored in a tlist and displayed on screen if the user has not given as second argument 'noprint'.

Example:
jbnorm(rols)

This example is taken from hendryericsson. The example provides the normality test for Hendry and Ericsson equation # 6, whose results have been stored in tlist rols.

jbnorm0_____Jarque and Bera normality test

CALLING SEQUENCE

[jb,pn]=jbnorm0(res,np)

PARAMETERS

INPUT:
* res = a result tlist
* np= unused argument put for compatibility with other testing functions
---
OUTPUT:
* jb = the value of the test
* jn = its p-value

DESCRIPTION

Computes Jarque and Bera normality test (see Jarque, C. M., and Bera, A. K. (1980). ' Efficient tests for normality, homoskedasticity and serial independence of regression residuals', Economics Letters, 6, 255–259). Results are stored as numbers and are not displayed on screen.

Example:
[jb,pn]=jbnorm0(res)

This example is taken from jbnorm. Useful mainly for programming purpose (since jbnorm does much more).

predfail_____out of sample predictive failure test

CALLING SEQUENCE

[rpredf]=predfail(grocer_res,arg1,…,argn)

PARAMETERS

INPUT:
* grocer_res = the tlist results resulting from a regression
* argi
  . the string 'noprint' if the user doesn't want to print the results of the regression
  . either:
    - a constant, which idicates the # of periods over which the forecast is done
    - a date, which indicates the date until which the forecast is done (if the estimation has been made with ts)

OUTPUT:
grocer_rpredf = a results tlist with:
  . grocer_rpredf('meth') = 'rprefail'
  . grocer_rpredf('res0') = results tlist of the originating estimation
  . grocer_rpredf('chistat') = value of the test statistics
  . grocer_rpredf('pchi') = its p-level
  . grocer_rpredf('p') = the # of forecasts
  . grocer_rpredf('prests') = boolean indicating the presence or absence of a time series in the regression
  . grocer_rpredf('prests') = the residuals over the forecasting period
  . grocer_rpredf('bounds') = the bounds of the forecasting period

DESCRIPTION

Computes out-of-sample predictive failure test. Results are stored in a tlist and displayed on screen if the user has not given as second argument 'noprint'.

Example:
rpredf=predfail(rols,16)

This example is taken from predfail_d. The example provides the predictive failure test with 16 periods ahead for Hendry and Ericsson equation # 6, whose results have been stored in tlist rols.

predfailin_____Chow predictive failure test

CALLING SEQUENCE

[resulch]=predfailin(resulols,n1)

PARAMETERS

INPUT:
* resulols = a results tlist from a first stage estimation
* n1 = # of observations of the sub-period

OUTPUT:
resulch = a results tlist with:
. resulch('meth') = predfail
. resulch('rols') = the tlist results from first stage estimation
. resulch('fstat') = value of the test statistic
. resulch('f_pvalue') = its p-value
. resulch('dfnum') = # dof of the numerator
. resulch('dfden') = # dof of the denominator
. resulch('cut') = date (if there is a ts in the regression) or index of the observation (if there is no ts in the regression) of the break

DESCRIPTION

Computes in-sample Chow predictive failure test. Results are stored in a tlist and displayed on screen if the user has not given as second argument 'noprint'.

Example:
predfailin(r,50)

This example is taken from predfailin_d. The example provides the predictive failure test with a break at observation # 50 for Hendry and Ericsson equation # 6, whose results have been stored in tlist r.

predfailin0_____Chow predictive failure test

CALLING SEQUENCE

[fstat,f_pvalue,dfnum,dfden]=predfailin0(res,n1,np)

PARAMETERS

INPUT:
* resulols = a results tlist from a first stage estimation
* n1 = # of observations of the sub-period

OUTPUT:
* fstat = value of the test statistic
* f_pvalue = its p-value
* dfnum = # dof of the numerator
* dfden = # dof of the denominator

DESCRIPTION

Computes in-sample Chow predictive failure test. Results are stored as numbers and are not displayed on screen.

Example:
[fstat,f_pvalue,dfnum,dfden]=predfailin0(resulols,n1)

This example is taken from predfailin. Useful mainly for programming purpose (since jbnorm does much more).

recresid_____recursive residuals

CALLING SEQUENCE

[rresid]=recresid(y,x)

PARAMETERS

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

OUTPUT:
rresid = recursive residuals (first k-obs equal zero)

DESCRIPTION

Computes recursive residuals.

Example:
vv = recresid(grocer_y,grocer_x)

This example is taken from cusumf.

(°)**reliability_____reliability of the variables in a regression**

CALLING SEQUENCE

reso=reliability(res,w,p,np)

PARAMETERS

INPUT:
* res = the results of a -univariate- regression
* w = a (1 x 4) vector of reliability outcome:
  - w(1) = significant only on a sub-sample
  - w(2) = significant only on the sub-samples

- w(3) = significant only on the whole sample
- w(4) = significant on a sub-period and on a sub_sample

* p = a size level
* np = 'noprint' if the user does not want to print the results

---

OUTPUT:
reso = a tlist with:
- res('meth')='reliability'
- res('namey')= a string, the name of the endogenous variable in the regression
- res('namex')= the string vector of the names of the exogenous variables in the regression
- res('reliab')= the real vector of the reliability of the coefficients

DESCRIPTION

Estimates the reliability of the variables in a regression

Example:
r=hendryericsson();reliability(r,[0.3 0.6 0.4 0.7],0.05)

From Hendry and Ericsson estimation, calculates the reliability of the coefficients at a 5% level, using Hendry and Krolzig proposed parametrisation (w=[0.3 0.6 0.4 0.7]). (°°)

reset_____Ramsey RESET test

CALLING SEQUENCE

[resulres]=reset(resul1,power,np)

PARAMETERS

INPUT:
* resul1 = results tlist from a first stage estimation
* power = degree of non linearity
* np = 'noprint' if the user does not want to print the results

---

OUTPUT:
resulres = results tlist with:
. resulres('meth') = 'reset'
. resulres('resul1st') = resul1
. resulres('f') = fstat
. resulres('p') = p
. resulres('df') = df
. resulres('f_pvalue')=f_pvalue

DESCRIPTION

Computes Ramsey (1969) RESET test for non linearity of power 2 to p (see Ramsay (1969) :"Tests for specification errors in classical linear least-squares regression analysis", Journal of the Royal Statistical Society, Series B, n°2, 350-371). Results are stored in a tlist and displayed on screen if the user has not given as second argument 'noprint'.

Example:
reset(rols,2)

This example is taken from hendryericsson. The example provides RESET test for Hendry and Ericsson equation # 6, whose results have been stored in tlist rols.

reset0_____Ramsey RESET test

CALLING SEQUENCE

[f,f_pvalue]=reset0(r,npow,np)

PARAMETERS

INPUT:
* r = the first stage regression tlist result
* npow = degree of non linearity

OUTPUT:
* f = value of the Goldfeld-Quandt F test
* f_pvalue = its p-value

DESCRIPTION

Computes Ramsey (1969) RESET test for non linearity of power 2 to p (see Ramsay (1969) :"Tests for specification errors in classical linear least-squares regression analysis", Journal of the Royal Statistical Society, Series B, n°2, 350-371). Results are given only in constant form and not printed.

Example:
[fstat,f_pvalue]=reset0(resul1,power)

This example is taken from function reset. Useful mainly for programming purpose (since reset does much more).

waldf_____Wald F-test

CALLING SEQUENCE

[resulwa]=waldf(resultr,resultu,np)

PARAMETERS

INPUT:
* resultr = results tlist from ols() restricted regression
* resultu = results tlist from ols() unrestricted regression
* np= 'noprint' if the user does not want to print the results
---
OUTPUT:
resulbp = a results tlist with:
- resultbp('meth') = 'waldf'
- resultbp('runcons') = results tlist of the un restricted regression
- resultbp('rcons') = results tlist of the restricted regression
- resultbp('f') = Wald F-statistic
- resultbp('dfnum') = degrees of freedom of the numerator
- resultbp('dfden') = degrees of freedom of the denominator
- resultbp('f_pvalue') = p-value of the test

DESCRIPTION

Computes Wald F-test for two regressions. Results are stored in a tlist and displayed on screen if the user has not given as second argument 'noprint'.

Example:
rwaldf=waldf(r1,r2)

This example provides the Fisher satistics corresponding to the restriction r1 imposes over r2.

waldf0_____ Wald F-test

CALLING SEQUENCE

[fstat,fprb,dfnum,dfden]=waldf0(resultr,resultu)

PARAMETERS

INPUT:
* resultr = results tlist from ols() restricted regression
* resultu = results tlist from ols() unrestrcted regression

OUTPUT:
* fstat = {(essr - essu)/#restrict}/{essu/(nobs-nvar)}
* fprb = marginal probability for fstat

DESCRIPTION

Computes Wald F-test for two regressions. Results are stored as numbers and are not displayed on screen.

Example:
[fstat,f_pvalue,dfnum,dfden]=waldf0(resultr,resultu)

This example is taken from function waldf. Useful mainly for programming purpose (since jbnorm does much more).

white_____White's heteroskedasticity test

CALLING SEQUENCE

[resulwhite]=white(resulols,np)

PARAMETERS

INPUT:
* resulols = results tlist from a first stage estimation returns a tlist
* np = 'noprint' if the user does not want to display the results (optional argument!)

OUTPUT:
resulwhite = a results tlist with:
- resulwhite('meth') = 'white'
- resultwhite('resul1st') = results tlist of the first stage regression
- resulwhite('f') = White statistic (Fisher form)
- resulwhite('p') = p-value of the test
- resulwhite('dfnum') = degrees of freedom of the numerator
- resulwhite('dfden') = degrees of freedom of the denominator
- resulwhite('f_pvalue') = p-value of the test
- resulwhite('chistat') = p-value of the test
- resulwhite('f_chistat') = p-value of the test

DESCRIPTION

Computes the value of White's heteroskedasticity test (see White, H. (1980) "A heteroskedastic-consistent covariance matrix estimator and a direct test for heteroskedasticity", Econometrica, 48, 817–838). Results are stored in a tlist and displayed on screen if the user has not given as second argument 'noprint'.

Example:
r=white(he)

This example stores in tlist r, and display on screen, White's test on regression results stored in he.

white0_____White's heteroskedasticity test

CALLING SEQUENCE

[f,f_pvalue,nvar2,r2]=white0(r,np)

PARAMETERS

INPUT:
* r = results tlist from a first stage estimation
* np = 'noprint' if the user does not want to print the results
---
OUTPUT:
* f = value of the Xi^2 hetero F test
* f_pvalue = its p-value
* nvar2 = # of exogenous variables of the White second stage regression
* r2 = R^2 of the auxiliary regression

DESCRIPTION

Computes the value of White's heteroskedasticity test (see White, H. (1980) "A heteroskedastic-consistent covariance matrix estimator and a direct test for heteroskedasticity", Econometrica, 48, 817–838). Results are stored as numbers and are not displayed on screen.

Example:
[f,f_pvalue,nvar2,r2]=white0(resulols)

This example is taken from function white. Useful mainly for programming purpose (since white does much more).