Chapter 9: Multiple equations regressions

GROCER contains the old simultaneous equations estimation methods (Seemingly Unrelated Regressions, two and three stages least squares) as well as the more recent VAR ones. As regards the VAR methodology, the following procedures are currently available: VAR estimation, impulse response calculation, ecm estimation, bayesian VAR and its ecm form, VAR forecasting.

1. Simultaneous equations methods

Seemingly Unrelated Regressions, two and three stages least squares are implemented in GROCER through functions sur, twosls and threesls.

In sur, the text of each equation must be entered in function sur between quotes and function sur estimates the coefficient that these equations contain. The names of the coefficients can be given through the option 'coef=[name1,…, namen]'. They can be omitted, provided that their name is of the form ai, with i being a number, starting from 1 to n without any discontinuity.

The Function sur is rather general: it can be applied to any system of equations, provided it is linear in its coefficients. So, you can impose constraints to the coefficients in the system, provided they remain linear. This possibility is achieved through function explosys, that retrieves the matrix of exogenous variables by the mean of differentiating numerically the vector of stacked equations with respect to the coefficients at any value (since the system is linear in its coefficients).

(°) The Function sur can be applied to a model with equal or unequal numbers of observations. The default is that the number of observations is equal. In particular, if the regressions contain ts and the time bounds has not been specified, the function chooses the time bounds as the ones that cover the greatest time span for all the ts in the regressions.

If the user wants to estimate the system with unequal numbers of observations, she has therefore to make 2 things:

- first, provide the maximum time bounds over which the equations must be estimated;
- second, enter the option 'unequal' in the function sur. (°°)

Take as an example the results of the sur estimation presented in Green's book[1], which is provided by function sur_d:

```
eq1='igm=a1*fgm+a2*cgm+a3'
eq2='ich=a4*fch+a5*cch+a6'
eq3='ige=a7*fge+a8*cge+a9'
eq4='iwest=a10*fwest+a11*cwest+a12'
eq5='iuss=a13*fuss+a14*cuss+a15'

r=sur(eq1,eq2,eq3,eq4,eq5)
```

[1] See Green (1997): *Econometric Analysis*, Third Edition, Prentice Hall, New Jersey

This example provides the simplest way of performing a sur estimation: the 5 corresponding equations are written such as the coefficients are a1, a2, … until a15. If you want to impose for instance the constraint that all constants are equal (for sure, a meaningless constraint on economic grounds…), then you should write:

```
eq1='igm=a1*fgm+a2*cgm+a3'
eq2='ich=a4*fch+a5*cch+a3'
eq3='ige=a6*fge+a7*cge+a3'
eq4='iwest=a8*fwest+a9*cwest+a3'
eq5='iuss=a10*fuss+a11*cuss+a3'

r=sur(eq1,eq2,eq3,eq4,eq5)
```

Note that since we have not given the names of the coefficients and since we have imposed 4 constraints, the coefficients are now named a1, a2, … until a11. Except from the first three coefficients all coefficients have now a different name from the one they are before.

So the user may find convenient to impose its own names. This is done through the option coef=[‘name1’,…,’namek’]. For instance, with the same constraints as with the previous example, you could write:

```
eq1='igm=f1*fgm+c1*cgm+c0'
eq2='ich=f2*fch+c2*cch+ c0'
eq3='ige=f3*fge+c3*cge+ c0'
eq4='iwest=f4*fwest+c4*cwest+ c0'
eq5='iuss=f5*fuss+c5*cuss+ c0'

r=sur(eq1,eq2,eq3,eq4,eq5, 'coef=[f1,f2,f3,f4,f5,c0,c1,c2,c3,c4,c5]’)
```

The function displays on screen: the correlation matrix and the estimation results for each equation. As with other estimation functions, the results of the estimation can be saved in a tlist, as is done above, by typing name_of_tlist = sur(...).

Note that the coefficients can be given in the option ‘coef=…’ in a different order than in the equations. As an example, taken from function sur_d, here are the results from a sur estimation performed on the grun (matlab) database which you can find in siclab library macros/grocer/db :

```
-->ige  = grun(:,1);
-->fge = grun(:,2);
-->cge = grun(:,3);
// general electric

-->iwest  = grun(:,4)
-->fwest = grun(:,5);
-->cwest = grun(:,6);
// westinghouse

-->igm  = grun(:,7);
-->fgm = grun(:,8);
```

```
-->cgm = grun(:,9);
```
// general motors

```
-->ich  = grun(:,10);
-->fch = grun(:,11);
-->cch = grun(:,12);
```
// chrysler

```
-->iuss  = grun(:,13);
-->fuss = grun(:,14);
-->cuss = grun(:,15);
```
// us steel

(order follows that in Green, 1997)

```
-->eq1='igm=a1*fgm+a2*cgm+a3;'
-->eq2='ich=a4*fch+a5*cch+a6'
-->eq3='ige=a7*fge+a8*cge+a9'
-->eq4='iwest=a10*fwest+a11*cwest+a12'
-->eq5='iuss=a13*fuss+a14*cuss+a15'

-->r=sur(eq1,eq2,eq3,eq4,eq5)

sur estimation results

correlation matrix of residuals

eq 1          2          3         4         5
1  1           -.3107024  .2351589  .1359552  -.3427850
2   -.3107024 1           .0342116  .1485078  .3777833
3   .2351589   .0342116  1          .7973658  .5270897
4   .1359552   .1485078   .7973658 1          .7370832
5   -.3427850  .3777833   .5270897  .7370832 1

results for equation # 1:
igm=a1*fgm+a2*cgm+a3

number of observations: 20
number of variables: 3
standard error of the regression: 92.740765
sum of squared residuals: 146214.44
DW(0) = .9367137

variable coeff      t-statistic p value
a1        .1219526  5.5542618   2.788E-08
a2        .3894513  11.272557   0
a3       -173.03757 -1.892899    .0583713

                         *
                      *     *

results for equation # 2:
ich=a4*fch+a5*cch+a6

number of observations: 20
number of variables: 3
standard error of the regression: 13.508073
```

```
sum of squared residuals: 3101.9567
DW(0) =1.8851097

variable coeff     t-statistic p value
a4        .0674506 3.6361868    .0002767
a5        .3050661 10.789811   0
a6       2.3783076  .1885154    .8504726

                        *
                     *     *

results for equation # 3:
ige=a7*fge+a8*cge+a9

number of observations: 20
number of variables: 3
standard error of the regression: 29.549511
sum of squared residuals: 14843.951
DW(0) = .8980277

variable coeff      t-statistic p value
a7        .0370190  2.899646     .0037358
a8        .1169537  4.9618774   6.982E-07
a9       -16.376032  -.6048658   .5452681

                        *
                     *     *

results for equation # 4:
iwest=a10*fwest+a11*cwest+a12

number of observations: 20
number of variables: 3
standard error of the regression: 11.03339
sum of squared residuals: 2069.5068
DW(0) =1.1247319

variable coeff     t-statistic p value
a10       .0538605 4.8239174    .0000014
a11       .0264688  .6588708    .5099787
a12      4.4891321  .687268     .4919139

                        *
                     *     *

results for equation # 5:
iuss=a13*fuss+a14*cuss+a15

number of observations: 20
number of variables: 3
standard error of the regression: 106.77545
sum of squared residuals: 193816.96
DW(0) = .9673477

variable coeff     t-statistic p value
a13       .0886000 1.8040818    .0712185
a14       .3092972 2.4200821    .0155170
a15      138.01198 1.3449313    .1786474
```

```
                    *
                *     *
```

An example of a sur estimation with an unequal number of observations is also provided in function sur_d:

```
--> load('SCI/macros/grocer/db/suruneq_d.dat')
```
// load the database
```
-->  bounds('1a','32a')
```
// set bounds
```
--> rsur=sur('vkbus=a1*pob+a2*pibloc+a3*bus','vkpar=a4*pob+a5*pibloc+a6*par',...
--> 'vktax=a7*pob+a8*pibloc+a9*tax','vkcam=a10*pob+a11*pibloc+a12*cam',...
--> 'vkcom=a13*pob+a14*pibloc+a15*com','vkmot=a16*pob+a17*pibloc+a18*mot',...
'unequal')
```
// perform estimation with the option 'unequal' to indicate that the numbers of observations are
// unequal

And the results are the following (note that the time bounds reported differ across the equations: 1a-32a for equations 1 to 3, 1a-22a for equations 4 and 5, 3a-29a for equation 6; these correspond to the non NA values for each equations):

```
sur estimation results

correlation matrix of residuals

eq 1 2         3          4          5          6
1  1 0.5980398 -0.2295625 0.6755263  0.6824408  0.1823176
2    1         -0.5819938 0.9157323  0.7775527  0.4271621
3              1          -0.7050460 -0.1334909 0.0657512
4                         1          0.7692916  0.1426075
5                                    1          0.2108701
6                                               1

results for equation # 1:
vkbus=a1*pob+a2*pibloc+a3*bus

estimation period: 1a-32a
number of observations: 32
number of variables: 3
standard error of the regression: 91871209
sum of squared residuals: 2.448D+17
DW(0) =1.0787468

variable coeff      t-statistic p value
a1       -63.404432 -0.0299042  0.9761435
a2       -25.235514 -0.0317227  0.9746932
a3       -26876.372 -0.2138026  0.8307010

                    *
                *     *

results for equation # 2:
vkpar=a4*pob+a5*pibloc+a6*par

estimation period: 1a-32a
number of observations: 32
```

```
number of variables: 3
standard error of the regression: 4.201D+08
sum of squared residuals: 5.119D+18
DW(0) =2.1419855

variable coeff      t-statistic p value
a4       465.76502  0.1135805   0.9095704
a5       -388.14262 -0.2485644  0.8036977
a6       -9468.4379 -0.6541410  0.5130210

                        *
                     *     *

results for equation # 3:
vktax=a7*pob+a8*pibloc+a9*tax

estimation period: 1a-32a
number of observations: 32
number of variables: 3
standard error of the regression: 3.026D+08
sum of squared residuals: 2.655D+18
DW(0) =1.4838448

variable coeff      t-statistic p value
a7       -514.68819 -1.8209688  0.0686116
a8       -9.2421202 -0.0911927  0.9273395
a9       12752.947  0.5809327   0.5612858

                        *
                     *     *

results for equation # 4:
vkcam=a10*pob+a11*pibloc+a12*cam

estimation period: 1a-22a
number of observations: 22
number of variables: 3
standard error of the regression: 71590952
sum of squared residuals: 9.738D+16
DW(0) =1.9720603

variable coeff      t-statistic p value
a10      277.43581  0.1984202   0.8427163
a11      -161.65381 -0.3044544  0.7607818
a12      -12911.187 -0.4730693  0.6361638

                        *
                     *     *

results for equation # 5:
vkcom=a13*pob+a14*pibloc+a15*com

estimation period: 1a-22a
number of observations: 22
number of variables: 3
standard error of the regression: 1.697D+08
sum of squared residuals: 5.472D+17
DW(0) =1.5761791
```

```
variable coeff      t-statistic p value
a13      472.4618   2.3748738   0.0175550
a14      -293.06275 -3.8976692  0.0000971
a15      -12284.748 -12.211848  0

                        *
                     *     *

results for equation # 6:
vkmot=a16*pob+a17*pibloc+a18*mot

estimation period: 3a-29a
number of observations: 27
number of variables: 3
standard error of the regression: 2404689.8
sum of squared residuals: 1.388D+14
DW(0) =0.7715770

variable coeff      t-statistic p value
a16      9.9587431  1.3977433   0.1621902
a17      -0.4770413 -0.1414008  0.8875533
a18      -12620.854 -3.4921133  0.0004792

                        *
                     *     *
```

Functions performing two stage (twosls) and three stage (threesls) least squares have the same syntax. So, only twosls is presented here. Note that the function twosls is here reserved to two stage least squares in a system of equations: two stage least squares applied on a single equation is performed through function iv (see chapter 7). As with sur, the only compulsory arguments are the texts of the equations in the system. In that case, the program determines what are the coefficients and what are, in each equation, the endogenous variables. This imposes however constraints on the way equations are written: as with sur, the coefficients must be named a1, a2,... until an without any discontinuity ; and the names of the endogenous variables must be exactly equal to the left hand side of the equations.

If you take the example which is performed in function twosls_d (which belongs to Scilab library macros/grocer/multi), you can see the simplest way of performing a Two Stage Least Squares estimation:

```
--> rt=twosls('y1=a+b*x1','y2=d+e*(y1-x2)+f*x2','coef=a;b;d;e;f')

iv estimation results for dependent variable: y2

endogenous variable(s) in this equation is(are):
y1

instruments for this equation are:
iota,x1,x2

number of observations: 200
number of variables: 3
R² =  .8105257      ajusted R² = .8086021
generalized R² =  .2762786 (*)
```

```
standard error of the regression: 1.0431856
sum of squared residuals: 214.38254
DW(0) =2.1064743
Belsley, Kuh, Welsch Condition index: 2

variable coeff     t-statistic p value
y1        .9766612 12.965309   0
iota     1.0645033 9.9609201   0
x2        .9213350 12.326011   0

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

                         *
                      *     *


iv estimation results for dependent variable: y1

there is no endogenous variable in the equation: iv=ols

number of observations: 200
number of variables: 2
R² =  .4840523      ajusted R² = .4814465
generalized R² =  .4840523 (*)
standard error of the regression: 1.0197048
sum of squared residuals: 205.88
DW(0) =2.091244
Belsley, Kuh, Welsch Condition index: 1

variable coeff     t-statistic p value
un       1.0469045 1.0469045   0
x1        .9800141  .9800141   0

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

                         *
                      *     *


iv estimation results for dependent variable: y2

endogenous variable(s) in this equation is(are):
(y1-x2)

instruments for this equation are:
un,x1,x2

number of observations: 200
number of variables: 3
R² =  .8105257      ajusted R² = .8086021
generalized R² =  .2762786 (*)
standard error of the regression: 1.0431856
sum of squared residuals: 214.38254
DW(0) =2.1064743
Belsley, Kuh, Welsch Condition index: 3
```

```
variable coeff     t-statistic p value
un       1.0645033 1.0645033   0
(y1-x2)   .9766612  .9766612   0
x2       1.8979962 1.8979962   0

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

                          *
                       *     *
```

Note that these functions call the subroutines exploeqs and syslist, whose syntax is given in this chapter, but which should note have many other uses than in twosls and threesls. Note also that function sur uses the subroutine explosys that retrieves among others from the system of equations the matrix of exogenous variables, vector of endogenous variables, the list of coefficients ... (see the exact syntax in part 3 of this chapter).

2. VAR estimation.

GROCER provides several tools to perform VAR estimations:

- VAR estimation itself: function var and its low level counterpart var1;
- Impulse response functions: irf and its subroutines: irf0, irf_asy and irf_mc1;
- Bayesian VARs: bvar and its low level counterpart bvar1 (which uses the subroutine scstd that determines the standard deviations from univariate regressions in order to scale the coefficients in the bvar estimation and theilbv that estimate the coefficients for an individual equation);
- Error Correction estimation: ecm and a bayesian one version, becm;
- Forecasting with a VAR model: varf

An ordinary VAR estimation involves function var. The function has two compulsory inputs: in the first place, the VAR order; and, at whatever place after, a string beginning with 'endo=''. This string should then be completed by the list of endogenous variables separated by commas. See for example the example in function var_d1(), the command:
results=var(2,'endo=del(log(rfa_inv)),del(log(rfa_inc)),del(log(rfa_cons))')

The lag order is 2 and the endogenous variables are del(log(rfa_inv)), del(log(rfa_inc)) and del(log(rfa_cons)). You can also add two other optional arguments: the usual argument 'noprint' if you do not want to print the results of the regression and a string beginning with 'exo=' if you want to add exogenous variables (other than the constant which is automatically included) to the VAR. This string should then be completed by the list of exogenous variables separated by commas. For example, if you want to add an exogenous variable called z and a time trend called temps[2], then you should type:

```
-->  results=var(2,'endo=del(log(rfa_inv)),del(log(rfa_inc)),del(log(rfa_cons))',
'exo=z,t')
```

As usual, the results are printed, unless the user enters 'noprint' as an input. As usual too, the results can be stored in a tlist by giving the name of this tlist as an output of the function (as it is done in the example, the output being results).

2 French world for "time"

As an example, here is the result displayed on screen of the above example:

```
var estimation results for variables
del(log(rfa_inv)), del(log(rfa_inc)) and del(log(rfa_cons))

AIC criterion: -24.549439
BIC criterion: -22.656297
Hannan-Quinn criterion: -24.286856

estimation results for dependant variable del(log(rfa_inv))
estimation period: 1960q4-1978q4
number of observations: 73
number of variables: 7
R² =  .1285615      ajusted R² = .0493398
Overall F test: F(6,66) = 1.6228074       p-value =  .1547471
standard error of the regression:  .0461479
sum of squared residuals:  .0019254
DW(0) =1.962689
Belsley, Kuh, Welsch Condition index: 165

variable                coeff      t de student p value
del(log(rfa_inv))(-1)   -.3196310 -2.5477448    .0108422
del(log(rfa_inc))(-1)   .1459888   .2675425     .7890515
del(log(rfa_cons))(-1)  .9612190  1.4469428     .1479129
del(log(rfa_inv))(-2)   -.1605511 -1.2853682    .1986636
del(log(rfa_inc))(-2)   .1146050   .2143873     .8302451
del(log(rfa_cons))(-2)  .9343938  1.4049004     .1600509
cte                     -.0167220  -.9707203    .3316876

                         *
                      *     *

estimation results for dependant variable del(log(rfa_inc))
estimation period: 1960q4-1978q4
number of observations: 73
number of variables: 7
R² =  .1141941      ajusted R² = .0336663
Overall F test: F(6,66) = 1.4180703       p-value =  .2209957
standard error of the regression:  .0117191
sum of squared residuals:  .0001242
DW(0) =1.9814536
Belsley, Kuh, Welsch Condition index: 165

variable                coeff      t de student p value
del(log(rfa_inv))(-1)   .0439311  1.3789096     .1679226
del(log(rfa_inc))(-1)   -.1527319 -1.1021991    .2703751
del(log(rfa_cons))(-1)  .2885016  1.7101505     .0872381
del(log(rfa_inv))(-2)   .0500308  1.5772812     .1147308
del(log(rfa_inc))(-2)   .0191658   .1411818     .8877264
del(log(rfa_cons))(-2)  -.0102049  -.0604199    .9518212
cte                     .0157672  3.6042716     .0003130

                         *
                      *     *

estimation results for dependant variable del(log(rfa_cons))
estimation period: 1960q4-1978q4
number of observations: 73
```

```
number of variables: 7
R² =  .2512819      ajusted R² = .1832166
Overall F test: F(6,66) = 3.6917783       p-value =  .0031839
standard error of the regression:  .0094448
sum of squared residuals:  .0000806
DW(0) =2.1458326
Belsley, Kuh, Welsch Condition index: 165

variable                coeff      t de student p value
del(log(rfa_inv))(-1)   -.0024227  -.0943543    .9248277
del(log(rfa_inc))(-1)   .2248127  2.0130521     .0441092
del(log(rfa_cons))(-1)  -.2639675 -1.9415137    .0521960
del(log(rfa_inv))(-2)   .0338804  1.3253298     .1850618
del(log(rfa_inc))(-2)   .3549124  3.2439759     .0011787
del(log(rfa_cons))(-2)  -.0222301  -.1633121    .8702727
cte                     .0129259  3.6662873     .0002461

                         *
                      *     *
```

Impulse response functions can then be calculated provided that the results have been properly saved in a results tlist. This calculation implies the call to function irf. As with most GROCER functions, there are many default input, so that the user has only to enter a few compulsory parameters. These are: in first place, the results tlist from the VAR estimation; in second place, the number of periods until which the response is calculated. By default, irf calculates then the impulse function using a Choleski decomposition and the asymptotic estimate of the variance to calculate the confidence bands. These confidence bands are calculated by default with a 5% size. These default options can be modified the following way:

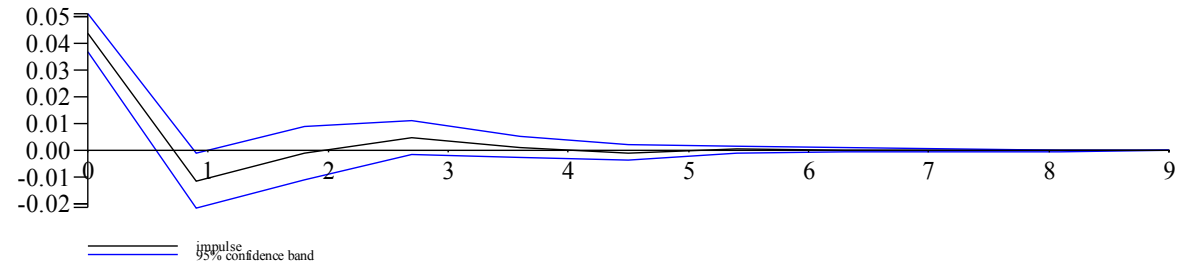
- if you want to calculate the impulse response from the original residuals (note that because these residuals are not orthogonal, the impulse functions are altogether not consistent...), then enter the option 'mres=original';
- if you want to calculate the confidence bands by the mean of Monte-Carlo simulations from the estimated coefficients, then enter the option 'meth=mc1';
- in that case, you can use the default value for the number of draws, that is 1000, or enter the option 'niter=x', where x is the number of draws you want to use; for instance, if you want to use 500 draws, the enter 'niter=500';
- lastly, if you want to use another size level for the confidence band, then enter 'size=x' where x is the size you want to use.

- The result of the irf function can be saved in a tlist by giving the name of the tlist as output. Results are graphed on the graphic screens, the response of one variable to all shocks are gathered in one scilab graphic widow. You can also, if you want so, graph all responses on the same graphic window by using function pltirf2.

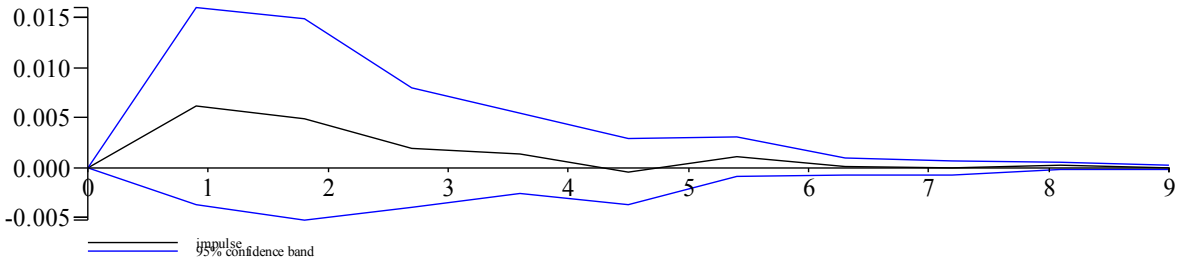
As an example here is the simplest command to estimate the impulse function of the VAR model estimated above with the resulting graphs:

```
-->irf(results,10);
```

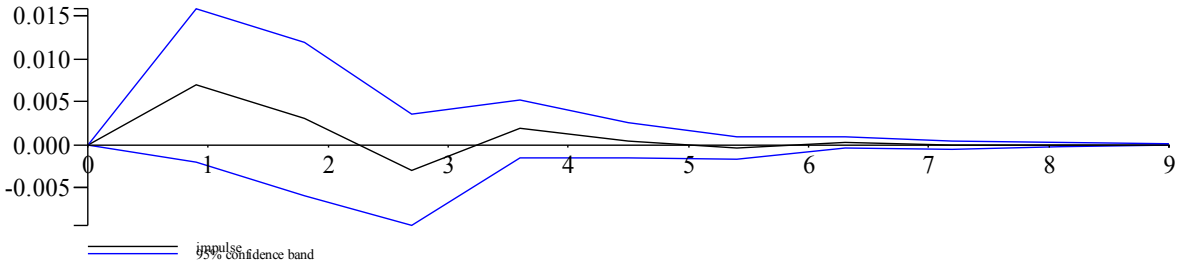
Orthog. IRF: 1 sigma changes- response of del(log(rfa_inv)) to an innovation in del(log(rfa_inc))



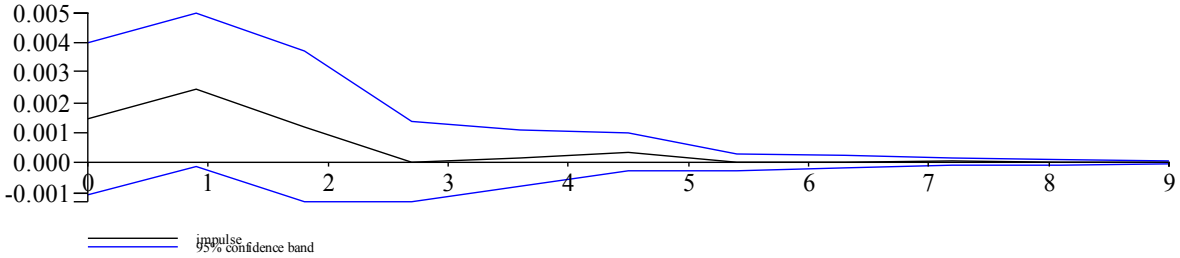
Orthog. IRF: 1 sigma changes- response of del(log(rfa_inv)) to an innovation in del(log(rfa_cons))



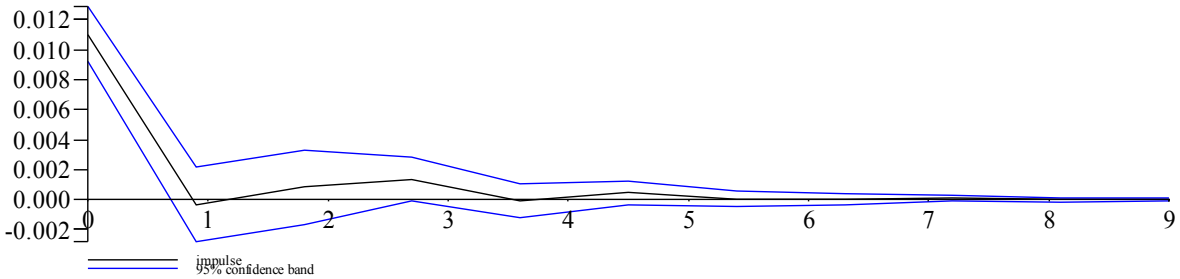
Orthog. IRF: 1 sigma changes- response of del(log(rfa_inv)) to an innovation in del(log(rfa_inv))



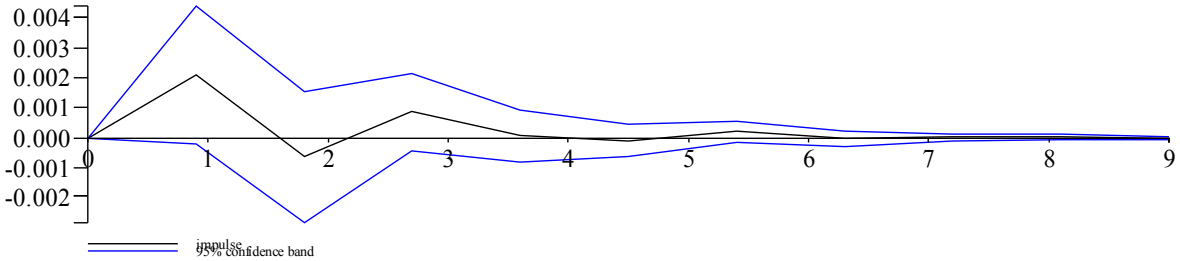
Orthog. IRF: 1 sigma changes- response of del(log(rfa_inc)) to an innovation in del(log(rfa_inc))

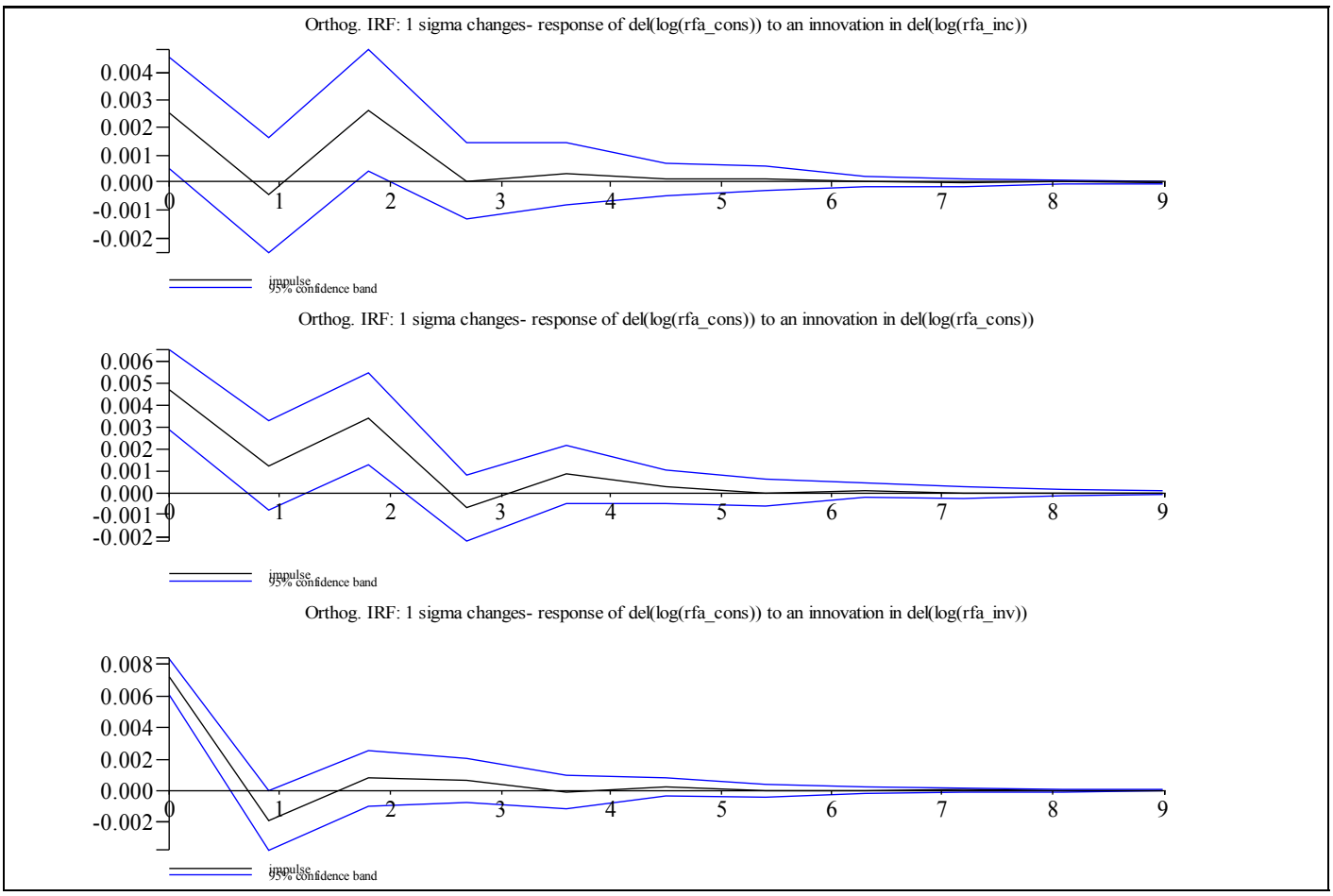


Orthog. IRF: 1 sigma changes- response of del(log(rfa_inc)) to an innovation in del(log(rfa_cons))



Orthog. IRF: 1 sigma changes- response of del(log(rfa_inc)) to an innovation in del(log(rfa_inv))





With GROCER, you can also estimate a bayesian VAR à la Doan, Litterman, Sims[3]. Doan, Litterman and Sims propose to let each coefficient of the VAR to depend on a small number of "hyperparameters", that govern the prior distribution of these parameters. Concretely, lag k of variable j in equation i is assumed to have a coefficient β_{kij} following the law:

$$\beta_{kij} \sim N(m_{kij}, \theta w(i,j) k^{-\phi} \frac{\hat{\sigma}_{uj}}{\hat{\sigma}_{ui}})$$

where θ and ϕ are parameters (called respectively the tightness and decay parameters), W=(w(i,j)) is a weighting matrix and $\hat{\sigma}_{uj}$ is the estimated standard error from a univariate autoregression involving variable j. m_{kij} is equal to 1 for the first lagged coefficient of the endogenous variables and to 0 for all other coefficients.

The estimation is done through function bvar. This function takes as inputs: first, lag order of the VAR ; second, the tightness parameter θ ; third the weighting matrix W ; fourth, the decay parameter ϕ . The following parameters are, as with an ordinary VAR, the string 'endo=…' and optionally the string 'exo=…' or the string 'noprint'. Note that you can enter the whole matrix W or you can enter a scalar w as third parameter. In that case, the matrix W is taken as:

3 see Doan, T., R. B. Litterman, and C.A. Sims (1984): « Forecasting and conditional projections using realistic prior distributions », Econometric Reviews, Vol. 3, pp. 1-100 or James Le Sage's book « Applied Econometrics using MATLAB », available at http://www.econ.utoledo.edu.

$$W = \begin{pmatrix} 1 & w & \cdots & w \\ w & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & w \\ w & \cdots & w & 1 \end{pmatrix}$$

Function bvar_d reproduces the estimations provided by Lesage in his book (cf. footnote 3). You can reproduce these results by simply typing (only the results of the first equation are here presented):

```
-->   load('SCI/macros/grocer/db/datajpl.dat');   results   =   bvar(2,0.1,0.5,1,
'endo=illinos,indiana,kentucky,michigan,ohio,pennsyvlania,tennesse,westvirginia]'
);

bvar estimation results for variables
illinos,   indiana,   kentucky,   michigan,   ohio,   pennsyvlania,   tennesse   and
westvirginia

PRIOR hyperparameters
tightness =  .1
decay = 1
symetric weights based on  .5

estimation results for dependant variable illinos
number of observations: 171
number of variables: 17
R² =  .9941671      ajusted R² = .9935611
standard error of the regression: 3.5865609
sum of squared residuals: 14.199878
DW(0) =2.0441636

variable          coeff      t de student p value
illinos(-1)       1.1348549  11.535932    0
indiana(-1)        .3904291  1.8808339     .0617045
kentucky(-1)       .0494292   .8983469     .3702712
michigan(-1)      -.0373268  -.4975042     .6194764
ohio(-1)          -.1596690 -1.6738632     .0959964
pennsyvlania(-1)   .1796105  3.5247189     .0005448
tennesse(-1)       .1563444   .7733333     .4403991
westvirginia(-1)  -.0468081 -2.0727694     .0397027
illinos(-2)       -.1612580 -1.6770889     .0953629
indiana(-2)       -.5038721 -2.5969371     .0102301
kentucky(-2)      -.0264363  -.5156394     .6067762
michigan(-2)      -.0263910  -.3770580     .7066008
ohio(-2)           .1914249  2.0634981     .0405853
pennsyvlania(-2)  -.1226781 -2.5205383     .0126385
tennesse(-2)      -.2883580 -1.4377959     .1523302
westvirginia(-2)   .0147535   .6811260     .4967185
cte               9.4546996  2.2751029     .0241487

                        *
                    *     *
```

The user can also perform ecm and becm estimations, on the same model as var and bvar estimations, but with another step: the Johansen cointegration estimation. This step can be made

before calling functions ecm and becm or inside them.

The basic arguments of functions ecm and becm are however the same as those of there counterparts var and bvar. The first argument of function ecm must then be the lag order of the ecm model and the first arguments of function becm must then be the lag order of the becm model, the tightness parameter, the weighting matrix W and the decay parameter. The next parameters can be given in any order. The only compulsory one is the string 'endo=…'.

The user can let the program determine the cointegration regressions to add to the VAR model. In that case, she has no other input to give to the function. The program calls the program johansen and chooses the number of cointegration regressions by applying the log-likelihood test with size 5%. The user can however impose the size level to be 1% or 10%, by giving the option 'plevel=0.01' or 'plevel=0.1'. She can alternatively choose to impose the number of cointegration regressions by giving the option 'nbr=n' where n is chosen number of cointegration regressions. Finally, if a johansen estimation has already been made, the user can impose the program to use the corresponding results by giving the option 'jres=res' where res is the name of the result tlist resulting from the johansen call.

As an example, here is the result of the ecm estimation applied to James Le Sage database (estimation results are reported only for the first equation):

```
-->load('SCI/macros/grocer/db/datajpl.dat')      ;      ecm(2,'illinos','indiana',
'kentucky','michigan','ohio','pennsyvlania','tennesse','westvirginia',
'plevel=0.05)

Johansen estimation results for variables:
illinos, indiana, kentucky, michigan, ohio, pennsyvlania, tennesse, westvirginia
time order: 0
# of lags: 2

NULL:    Trace Statistic Crit 90% Crit 95%  Crit 99%
r <= 0   214.38998       153.6341 159.529   171.0905
r <= 1   141.48161       120.3673 125.6185  135.9825
r <= 2   90.363179       91.109   95.7542   104.9637
r <= 3   61.554975       65.8202  69.8189   77.8202
r <= 4   37.103415       44.4929  47.8545   54.6815
r <= 5   21.069871       27.0669  29.7961   35.4628
r <= 6   10.605086       13.4294  15.4943   19.9349
r <= 7   3.1924592       2.7055   3.8415    6.6349

conclusions from the trace statistics:
at a 10% level, there are 8 cointegration relation(s)
at a 5% level, there are 2 cointegration relation(s)
at a 1% level, there are 2 cointegration relation(s)

NULL:    Max Eigenvalues Statistic  Crit 90%  Crit 95%  Crit 99%
l <= 0   72.908372                  49.2855   52.3622   58.6634
l <= 1   51.118434                  43.2947   46.2299   52.3069
l <= 2   28.808204                  37.2786   40.0763   45.8662
l <= 3   24.45156                   31.2379   33.8777   39.3693
l <= 4   16.033544                  25.1236   27.5858   32.7172
l <= 5   10.464785                  18.8928   21.1314   25.865
l <= 6   7.4126267                  12.2971   14.2639   18.52
l <= 7   3.1924592                  2.7055    3.8415    6.6349
```

```
conclusions from the maximal eigenvalues statistics:
at a 10% level, there are 8 cointegration relation(s)
at a 5% level, there are 2 cointegration relation(s)
at a 1% level, there are 1 cointegration relation(s)

                        *
                    *     *


*** cointregrating vectors from johansen estimation ***

variable     vector # 1  vector # 2
illinos       .0131635    -.0099366
indiana       -.3050443   .0399855
kentucky      .0525625    -.0273951
michigan      -.0417678   .1871467
ohio          .0176776    .0199591
pennsyvlania  -.0526335   .0250484
tennesse      .0823386    -.2737890
westvirginia  .0201478    .0216022


                        *
                    *     *


ecm estimation results for variables
del(illinos),    del(indiana),    del(kentucky),    del(michigan),    del(ohio),
del(pennsyvlania), del(tennesse) and del(westvirginia)

AIC criterion: 21.375681
BIC criterion: 28.771462
Hannan-Quinn criterion: 22.51342

estimation results for dependant variable del(illinos)
number of observations: 170
number of variables: 19
R² =  .3012112      ajusted R² = .2179119
Overall F test: F(18,151) = 3.6160106       p-value =  .0000065
standard error of the regression: 3.8478095
sum of squared residuals: 13.15089
DW(0) =2.0112372
Belsley, Kuh, Welsch Condition index: 305

variable                coeff      t de student p value
del(illinos)(-1)         .1283714  1.1426411     .2531876
del(indiana)(-1)         .5099155  2.2799889     .0226083
del(kentucky)(-1)        .0375484   .6432906     .5200355
del(michigan)(-1)        .0155465   .1993254     .8420082
del(ohio)(-1)            -.2129544 -1.828733     .0674396
del(pennsyvlania)(-1)    .1963464  3.4065628     .0006579
del(tennesse)(-1)        .1227843   .5412941     .5883049
del(westvirginia)(-1)    -.0383854 -1.4744336    .1403649
del(illinos)(-2)         -.0549987  -.5019892    .6156751
del(indiana)(-2)         .0251240   .1066415     .9150734
del(kentucky)(-2)        .0457999   .7878064     .4308100
del(michigan)(-2)        .0300744   .3836735     .7012205
```

```
del(ohio)(-2)              -.1197645 -1.0935508    .2741520
del(pennsyvlania)(-2)      .0738467  1.1793163     .2382722
del(tennesse)(-2)          -.0801044  -.3566330    .7213666
del(westvirginia)(-2)      -.0096337  -.3781445    .7053233
lag 1 of coint. vec. #1    .1057021   .3581742     .7202129
lag 1 of coint. vec. #2    -.2967561 -1.005566     .3146244
cte                       3.4340101   .8938760     .3713882

                           *
                       *     *
```

The same results should have been obtained by typing:

```
-->load('SCI/macros/grocer/db/datajpl.dat')      ;      ecm(2,'illinos','indiana',
'kentucky','michigan','ohio','pennsyvlania','tennesse','westvirginia', 'nbr=2')
```

or:

```
-->load('SCI/macros/grocer/db/datajpl.dat')         ;resjoh=        johansen(0,2,
'illinos','indiana','kentucky','michigan','ohio','pennsyvlania','tennesse','westv
irginia');                ecm(2,'illinos','indiana','kentucky','michigan','ohio',
'pennsyvlania','tennesse','westvirginia', 'jres=resjoh', 'plevel=0.05')
```

Lastly, you can make a forecast with any of the four above VAR methods (var, bvar, ecm, becm), with function varf. The first input is the name of the result tlist from the VAR estimation: to use varf, it is necessary to have saved before the results of the estimation. The second input is the forecasting sample. For the forecasting sample, you can either choose the beginning and the ending period and give a (2x1) vector, or only the ending period and in that case the program assumes that the beginning period is the period next after the end of the estimation period. If your regression involves times series, you can choose to enter the forecasting period as a real vector (the compulsory choice if your VAR involves only real vectors) or as a dates vector. For instance if you want to make a forecast on period 2003a-2010a with a var estimation performed on period 1970a-2002a, then the second argument can be [1;8], or 8, or ['2003a'; '2010a'], or '2010a'.

There are two other arguments that can be given to function varf. The first one is the traditional 'noprint'. The second one is 'xx=mymat', where mymat is a matrix with a number of rows equal to the number of forecasting periods and a number of columns equal to the number of exogenous variables in the VAR model: such an input is compulsory if there are exogenous variables and these variables are not all timeseries.

Function varf_d provides an example, with James Le Sage data transformed into timeseries (database datajplts.dat):

```
-->load('SCI/macros/grocer/db/datajplts.dat');bounds('1982m7','1994m12');
varjls=var(2,'endo=illinos,indiana,kentucky,michigan,ohio,pennsyvlania,tennesse,w
estvirginia','noprint');rf=varf(varjls,'1995m12');

forecasting results for the VAR  over the period 1995m1-1995m12
***************************************************************

obs\variable illinos    indiana    kentucky   michigan   ohio        pennsyvlania
tennesse  westvirginia
1995m1            142.64411  67.452595  274.26197  84.533449  144.7345    204.78406
```

```
45.064243 284.22777
1995m2            142.42276  67.422959  273.8345   84.71155   144.13083  204.97743
44.912297 282.51181
1995m3            142.85986  67.755563  273.96736  84.973191  144.36529  206.24513
44.958798 283.16639
1995m4            143.59306  68.150474  274.47086  85.287951  144.80037  207.34555
45.075311 283.47151
1995m5            144.35492  68.517613  275.12682  85.489829  145.40222  208.36631
45.215349 284.00646
1995m6            145.03503  68.797829  275.77349  85.54111   146.02265  209.18658
45.334048 284.54313
1995m7            145.59638  68.98995   276.34218  85.467011  146.60536  209.83262
45.424522 285.10155
1995m8            146.05528  69.114296  276.81728  85.317768  147.13048  210.35387
45.491398 285.65858
1995m9            146.44798  69.1988    277.22028  85.144766  147.60617  210.80872
45.545782 286.20725
1995m10           146.81183  69.267958  277.58552  84.986117  148.05205  211.24656
45.598326 286.74669
1995m11           147.1753   69.338582  277.94626  84.862964  148.4883   211.69976
45.656729 287.27951
1995m12           147.5552   69.419474  278.32695  84.781708  148.93007  212.18434
45.72507  287.81067
```

*

* *

3. The multiple equtions functions and their specifications.

becm_____bayesian error correction model estimation

CALLING SEQUENCE
[rbecm]=becm(nlag,tight,weight,decay,arg1,...,argn)

PARAMETERS

INPUT:
* nlag = the lag length
* tight = Litterman's tightness hyperparameter
* weight = Litterman's weight (matrix or scalar)
* decay = Litterman's lag decay = lag^(-decay)
* 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
  . 'jres=xx' where xx is the name of a johansen results tlist (optional: if not given, is estimated by the function)
  . 'nbr=xx' where xx is the # of cointegration vectors to keep from the johansen estimation (optional: if not given, is calculated by the function with a level equal to plevel)
  . 'plevel=xx' where xx=0.01, 0.05 or 0.1 is the significance level for the cointegrating vectors

(optional: if not given, is set to 0.05; useless if the option 'nbr=xx' is used)

OUTPUT:
rbecm = a results tlist with:
. rbecm('meth') = 'becm'
. rbecm('y') = y data vector
. rbecm('x') = x data matrix
. rbecm('nobs') = # observations
. rbecm('nvar') = # exogenous variables
. rbecm('neqs') = # endogenous variables
. rbecm('tight') = Litterman's tightness hyperparameter
. rbecm('weight') = Litterman's weight (matrix or scalar)
. rbecm('decay') = Litterman's lag decay = lag^(-decay)
. rbecm('resid') = residuals, with rbecm('resid')(:,i): residuals for equation # i
. rbecm('beta') = bhat, with rbecm('beta')(:,i): coefficients for equation # i
. rbecm('rsqr') = rsquared, with rbecm('rsqr')(i) rsquared for equation # i
. rbecm('f') = F-stat for the nullity of other than the constant rbecm('f')(i): F-stat for equation # i
. rbecm('pvaluef') = their significance level rbecm('pvaluef')(i): significance for equation # i
. rbecm('rbar') = rbar-squared with rbecm('rbar')(i) r-bar-squared for equation # i
. rbecm('sigu') = sums of squared residuals with rbecm('sigu')(:,i): sum of residuals for equation # i
. rbecm('ser') = standard errors of the regression rbecm('ser')(i): standard error equation # i
. rbecm('tstat') = t-stats, with rbecm('tstat')(:,i): t-stat for equation # i
. rbecm('pvalue')= pvalue of the betas, rbecm('pvalue')(:,i): p-value equation # i
. rbecm('dw') = Durbin-Watson Statistic, rbecm('dw')(i): DW for equation # i
. rbecm('boxq') = Box Q-stat, with rbecm('boxq')(i): Box Q-stat for equation # i
. rbecm('sigma') = (neqs x neqs) var-covar matrix of the regression
. rbecm('prests') = boolean indicating the presence or absence of a time series in the regression
. rbecm('namey') = name of the y variable
. rbecm('nx') = # of x variables
. rbecm('nb_coint_relat') = # of cointegration relations
. rbecm('namex') = name of the cointegrating variables (if any)
. rbecm('bounds') = if there is a timeseries in the regression, the bounds of the regression
. rbecm('jres') = the result tlist from the johansen step

DESCRIPTION
Performs error bayesian correction model estimation. The user can specify the names of the tlist resulting from johansen estimation, the number of cointegration relations to take into account or the significance level to use to select the number of cointegration relations, but if these parameters are not provided, the function takes default values and, if necessary, performs johansen estimation. The lags of the var models are estimated with bayesian prior given by the parameters tight, weight and decay.

Examples:
results = becm(2,0.1,1,0.5,'illinos','indiana','kentucky','michigan'...
,'ohio','pennsyvlania','tennesse','westvirginia');

Example taken from function becm_d(); endogenous variables are 'illinos','indiana','kentucky','michigan', 'ohio','pennsyvlania','tennesse','westvirginia'; # of lags set to 2;

tight, weight and decay are set to 0.1,1 and 0.5.

bvar _____bayesian VAR Estimation

CALLING SEQUENCE
rbvar=bvar1(nlag,tight,weight,decay,y,x)

PARAMETERS

INPUT:
* nlag = the lag length
* tight = Litterman's tightness hyperparameter
* weight = Litterman's weight (matrix or scalar)
* decay = Litterman's lag decay = lag^(-decay)
* varargin = arguments which can be:
  . 'endo=[var1 var2 ... varn]' with vari: the ith endogenous variable in the var
  . 'exo=[var1 var2 ... varn]' with vari: the ith exogenous variable in the var
  . the string 'noprint' if the user doesn't want to print the results of the regression
---
OUTPUT:
rbvar = a results tlist with:
  . rbvar('meth') = 'bvar'
  . rbvar('y') = y data vector
  . rbvar('x') = x data matrix
  . rbvar('nobs') = # observations
  . rbvar('nvar') = # exogenous variables
  . rbvar('tight') = Litterman's tightness hyperparameter
  . rbvar('weight') = Litterman's weight (matrix or scalar)
  . rbvar('decay') = Litterman's lag decay = lag^(-decay)
  . rbvar('neqs') = # endogenous variables
  . rbvar('resid') = residuals, with rbvar('resid')(:,i): residuals for equation # i
  . rbvar('beta') = bhat, with rbvar('beta')(:,i): coefficients for equation # i
  . rbvar('rsqr') = rsquared, with rbvar('rsqr')(i) : rsquared for equation # i
  . rbvar('overallf') = F-stat for the nullity of coefficients other than the constant with: rbvar('f')(i): F-stat for equation # i
  . rbvar('pvaluef') = their significance level with: rbvar('pvaluef')(i): significance level for equation # i
  . rbvar('rbar') = rbar-squared
  . rbvar('sigu') = sums of squared residuals with rbvar('sigu')(:,i): sum of squared residuals for equation # i
  . rbvar('ser') = standard errors of the regression with rbvar('ser')(i): standard error for equation # i
  . rbvar('tstat') = t-stats, with rbvar('tstat')(:,i): t-stat for equation # i
  . rbvar('pvalue')= pvalue of the betas, with rbvar('pvalue')(:,i): p-value for equation # i
  . rbvar('dw') = Durbin-Watson Statistic, with: rbvar('dw')(i): DW for equation # i
  . rbvar('sigma') = (neqs x neqs) var-covar matrix of the regression
  . rbvar('nx') = # of x variables
  . rbvar('namey') = name of the y variable

. rbvar('namex') = name of the x variables (if any)
. rbvar('prests') = boolean indicating the presence or absence of a time series in the regression
. rbvar('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION
Performs error bayesian VAR estimation. The lags of the var models are estimated with bayesian prior given by the parameters tight, weight and decay.

Example:
results = bvar(2,0.1,1,0.5,'illinos','indiana','kentucky','michigan'...,'ohio','pennsyvlania' 'tennesse','westvirginia');

Example taken from function bvar_d(); endogenous variables are 'illinos','indiana','kentucky','michigan', 'ohio','pennsyvlania','tennesse','westvirginia'; # of lags set to 2; tight, weight and decay are set to 0.1,1 and 0.5.

bvar1 _____bayesian VAR Estimation

CALLING SEQUENCE
[rbvar]=bvar1(nlag,tight,weight,decay,y,x)

PARAMETERS

INPUT:
* nlag = the lag length
* tight = Litterman's tightness hyperparameter
* weight = Litterman's weight (matrix or scalar)
* decay = Litterman's lag decay = lag^(-decay)
* y = (nobs x neqs) matrix of endogenous variables
* x = (nobs x nx) matrix of exogenous variables (optional)

OUTPUT:
rbvar = a results tlist with:
. rbvar('meth') = 'bvar'
. rbvar('y') = y data vector
. rbvar('x') = x data matrix
. rbvar('nvar') = # exogenous variables
. rbvar('nobs') = # observations
. rbvar('neqs') = # endogenous variables
. rbvar('nlag') = # lags
. rbvar('tight') = Litterman's tightness hyperparameter
. rbvar('weight') = Litterman's weight (matrix or scalar)
. rbvar('decay') = Litterman's lag decay = lag^(-decay)
. rbvar('beta') = bhat, with rbvar('beta')(:,i): coefficients for equation # i
. rbvar('tstat') = t-stats, with rbvar('tstat')(:,i): t-stat for equation # i
. rbvar('pvalue')= pvalue of the betas, with rbvar('pvalue')(:,i): p-value for equation # i
. rbvar('resid') = residuals, with rbvar('resid')(:,i): residuals for equation # i

. rbvar('yhat') = yhat, with rbvar('yhat')(:,i): residuals for equation # i
. rbvar('sige') = estimated variances rbvar('sige')(i): variance for equation # i
. rbvar('ser') = standard errors of the regression with rbvar('ser')(i): standard error equation # i
. rbvar('dw') = Durbin-Watson Statistic, with: rbvar('dw')(i): DW for equation # i
. rbvar('rsqr') = rsquared, with rbvar('rsqr')(i): rsquared for equation # i
. rbvar('rbar') = rbar-squared
. rbvar('sigma') = (neqs x neqs) var-covar matrix of the regression
. rbvar('nx') = # exogenous variables
. rbvar('prescte') = boolean indicating the presence or absence of a constant in the regression
---
NOTE: constant vector automatically included

DESCRIPTION
Estimates a bayesian VAR of order p. Applies when the variables are already in matrix form. Does note display any result on screen (see bvar for a more complete function)

Examples:
1) rbvar = bvar1(nlag,tight,weight,decay,y)
2) rbecm=bvar1(nlag,tight,weight,decay,dy,x)

Examples taken from functions bvar and becm: these functions should be mainly be used in other functions.

ecm_____Error Correction Regression

CALLING SEQUENCE
recm=ecm(p,arg1,...,argn)

PARAMETERS

INPUT:
* p = # of lags
* argi = an argument which can be:
- the name of an endogenous variable
- 'jres=xx' where xx is the name of a johansen results tlist (optional: if not given, is estimated by the function)
- 'nbr=xx' where xx is the # of cointegration vectors to keep from the johansen estimation (optional: if not given, is calculated by the function with a level equal to plevel)
- 'plevel=xx' where xx=0.01, 0.05 or 0.1 is the significance level for the cointegrating vectors optional: if not given, is set to 0.05; useless if the option 'nbr=xx' is used)

OUTPUT:
recm = a results tlist with:
. recm('meth') = 'var'
. recm('y') = y data vector
. recm('x') = x data matrix

. recm('nobs') = # observations
. recm('nvar') = # exogenous variables
. recm('neqs') = # endogenous variables
. recm('resid') = residuals, with recm('resid')(:,i): residuals for equation # i
. recm('beta') = bhat, with recm('beta')(:,i): coefficients for equation # i
. recm('rsqr') = rsquared, with recm('rsqr')(i) : rsquared for equation # i
. recm('overallf') = F-stat for the nullity of coefficients other than the constant with: recm('overallf')(i): F-stat for equation # i
. recm('pvaluef') = their significance level with: recm('pvaluef')(i): significance level for equation # i
. recm('rbar') = rbar-squared
. recm('sigu') = sums of squared residuals with recm('sigu')(:,i): sum of squared error for equation # i
. recm('tstat') = t-stats, with recm('tstat')(:,i): t-stat for equation # i
. recm('pvalue')= pvalue of the betas, with recm('pvalue')(:,i): p-value for equation # i
. recm('dw') = Durbin-Watson Statistic, with: recm('dw')(i): DW for equation # i
. recm('condindex') = multicolinearity cond index, with recm('condindex')(i): cond index for equation # i
. recm('boxq') = Box Q-stat, with recm('boxq')(i): Box Q-stat for equation # i
. recm('aic') = Akaïke information criterion
. recm('bic') = Schwartz information criterion
. recm('hq') = Hannan-Quinn information criterion
. recm('namey') = name of the y variable
. recm('nx') = # of x variables
. recm('namex') = name of the cointegration relations (if any)
. recm('prests') = boolean indicating the presence or absence of a time series in the regression
. recm('nb_coint_relat') = # of cointegration relations
. recm('jres') = results of johansen estimation
. recm('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION

Performs error correction model estimation. The user can specify the names of the tlist resulting from johansen estimation, the number of cointegration relations to take into account or the significance level to use to select the number of cointegration relations, but if these parameters are not provided, the function takes default values and, if necessary, performs johansen estimation.

Example

1) result = ecm(2,'illinos','indiana','kentucky','michigan','ohio','pennsyvlania','tennesse','westvirginia')
2) result = ecm(2,'illinos','indiana','kentucky','michigan','ohio','pennsyvlania','tennesse','westvirginia', 'plevel=0 05')
3) result = ecm(2,'illinos','indiana','kentucky','michigan','ohio','pennsyvlania','tennesse','westvirginia', 'nbr=2)

First example is taken from function ecm_d(); endogenous variables are 'illinos','indiana','kentucky','michigan', 'ohio','pennsyvlania','tennesse','westvirginia'; example 1 let the function set the number of cointegration relations to the one chosen by a Johansen estimation at a 5%

level; since there are 2 cointegration relations at a 5% level, all 3 examples give the same result.

eq2xcol _____Transformation of a system of equations into a matrix vector

CALLING SEQUENCE

[x,boundsvarb,prests]=eq2xcol(grocer_a,listeq)

PARAMETERS

INPUT:
* grocer_a = the vector of coefficients where the equations must be evaluated
* listeq = the list of equations in string form

OUTPUT:
* x = the X matrix in the regression represented by the system of equations embedded in listeq
* boundsvarb = the bounds of the regressions (if any)
* prests = a boolean indicating whether there is a ts in the equations

DESCRIPTION

Transforms a list of strings into the column vector equal to the evaluation of the list of equations contained in grocer_listeq at point grocer_a. The coefficients must be named grocer_a(i) in the equations. Used in function explosys.

Examples
1) [grocer_x,grocer_boundsvarb,grocer_prests]=eq2xcol(ones(grocer_ncoef,1),grocer_listeq)
2) x=-numz0(eq2xcol,ones(grocer_ncoef,1),grocer_ncoef,ones(grocer_ncoef,grocer_totalnobs),1, grocer_listeq)'

These two examples are taken from explosys. The main interest is in example 2: eq2xcol is given as an entry in function numz0 in order to calculate the x matrix as the first derivative of the vector of equations with respect to the coefficients: it is evaluated at vector ones(grocer_ncoef,1).

exploeqs _____Transformation of 2sls or 3sls equations

CALLING SEQUENCE
[xall,lx1,ly1,endoeq,nameinst,lindx1,lindy1]=exploeqs(neqs,lexo,endo,z,namez)

PARAMETERS

INPUT:
* neqs = # of equations
* lexo = list of independent variables in each equation
* endo = list of dependent variables in each equation
* z = matrix of values of the dependent variables

* namez = matrix of names of the dependent variables

OUTPUT:
* xall = matrix of the values of the instruments
* lx1 = list of the exogenous variables for each equation
* ly1 = list of the endogenous variables for each equation
* endoeq = list of he names of the endogenous variables for each equation
* nameinst = matrix of the names of the instruments
* lindx1 = list of the indexes of exogenous variables for each equation
* lindy1 = list of the indexes of endogenous variables for each equation

DESCRIPTION
Retrieves from the equations the list of endogenous, exogenous and instruments for a Two-Stage or Three-Stage Least-squares Regression.

Example
[xall,lx1,ly1,endoeq,nameinst,lindx1,lindy1]=exploeqs(grocer_neqs,grocer_lexo,grocer_endo,grocer_z,grocer_namez)

This examples is taken from twosls.

explosys _____Transformation of sur equations

CALLING SEQUENCE
[x,y,boundsvarb,prests,listcoef,namey,ncoefeqs]= explosys(namecoef,specara,speccarb,arg1,...,argn)

PARAMETERS

INPUT:
* namecoef = column vector of coefficients name
* speccara = column vector of characters that must be found after the name of a coefficient
* speccarb = column vector of characters that must be found before the name of a coefficient
* argi = an equation containing the coefficients named in namecoef
---
OUTPUT:
* x = (nobs x k) matrix of exogenous variable in the complete model
* y = (nobs x 1) vector of endogenous variable in the complete model
* boundsvarb = bounds (if any) of the regression
* prests = a boolean indicating the presence or absence of ts in the regressions
* listcoef = list of indexes of the coefficients in each equation
* namey = column vector of the names of the endogenous variables
* ncoefeqs = column vector of the # of coefficients in each equation

DESCRIPTION
From a series of equations retrieve the matrix of exogenous variables, taking into acount the constraints (if any) imposed on the coefficients; the equations must be linear in the coefficients

Example
[x,y,grocer_boundsvarb,grocer_prests,listcoef,namey,ncoefeqs]=
explosys(grocer_namecoef,grocer_speccara,grocer_speccarb,varargin(:))

This example is taken from sur.

irf _____Impulse Response Function

CALLING SEQUENCE

res=irf(results,S,arg1,..,argn)

PARAMETERS

INPUT:
* results = results tlist returned by VAR
* S = scalar for number of periods in IRF
* argi = optional argument which can be:
  - 'mres=x' where:
    x = chol1 (cholesky decomposition)
    x = chol2 (triangular factorisation)
    x = original (original residuals)
      (default = chol1)
  - 'meth=x' where:
    x = asym (asymptotic formula)
    x = mc1 (Monte-Carlo simulations using draws from the coefficients)
      (default = asym)
  - 'niter=x' where x= # iterations for the Monte-Carlo simulations (if any; default=1000)
  - 'size=x' where x = significance level for the confidence band (default =0.05)

OUTPUT:
res = a results tlist with:
  - res('meth') = 'irf'
  - res('mres') = decomposition method
  - res('T') = # of periods represented
  - res('IRF') = ((S+1) x T) impulse response functions
  - res('IRF_LOW') = ((S+1) x T) lower range of impulse response confidence band
  - res('IRF_UPP') = ((S+1) x T) upper range of impulse response confidence band
  - res('PHI') = (N*p x T) matrix of coefficients
  - res('resvar') = results tlist of the originating VAR
  - res('msg') = message inidicating the nature of the decomposition
  - res('size') = size of the confidence band

DESCRIPTION

Calculates Impulse Response Function for VAR

Examples
1) results=var(2,'endo=del(log(rfa_inv)),del(log(rfa_inc)),del(log(rfa_cons))');
[resirf]=irf(results,10,'mres=chol1','meth=asym')
2) [resirf]=irf(results,10,'mres=original','meth=mc1','niter=1000')

Examples taken from function var_d(). In the first one, impulse response is calulated for 10 periods, with shcoks calculated from a Choleski decomposition, with the asymptotic formula. In the second one, impulse response is calulated for 10 periods, with original unorthogonalized shocks, using 1000 draws from the coefficients estimated law.

irf0_____Impulse Response Function

CALLING SEQUENCE

[IRF,PHI]=irf0(beta,S,N,p,P))

PARAMETERS

INPUT:
* beta = estimated parameters from a VAR
* S = # of periods
* N = dimension of the VAR
* p = # of lags
* P = matrix such that P*e = u
  where u is the residual from the VAR regression; e is the residual to be shocked

---
OUTPUT:
* IRF = ((S+1) x T) impulse response functions
* PHI = (N*p x T) matrix of coefficients

DESCRIPTION

Calculates Impulse Response Function for VAR (low level function).

Example
m=irf0(sbeta,S,N,p,P)

Example taken from function irf_mc1(). Used here because of its greater speed.

irf_asy_____Impulse Response Function

CALLING SEQUENCE

[irf_low,irf_upp]=irf_asy(results,mres,P,IRF,PHI,S,siz)

PARAMETERS

INPUT:
* results = results tlist returned by VAR
* mres = decomposition method
* P = matrix such that P*e = u
  where u is the residual from the VAR regression e is the residual to be shocked
* IRF = ((S+1) x T) impulse response functions
* PHI = (N*p x T) matrix of coefficients
* S = # of periods
* siz = size of the confidence band

OUTPUT:
* irf_low = ((S+1) x T) lower range of impulse response confidence band
* irf_upp = ((S+1) x T) upper range of impulse response confidence band

DESCRIPTION

Calculates standard error of Impulse Response Function for VAR by means of the asymptotic formula. Note: a much simpler method is to use irf with option 'meth=asym'

Example
[irf_low,irf_upp]=irf_asy(results,mres,P,IRF,PHI,S,siz)

Example taken from function irf(). This is equivalent to the simpler call: irf(results,S,'meth= asym','mres=name of mres','size=siz')

irf_mc1_____Impulse Response Function

CALLING SEQUENCE

[irf_low,irf_upp]=irf_mc1(res,mres,S,N,p,niter,siz)

PARAMETERS

INPUT:
* res = results tlist returned by VAR
* mres = decomposition method
* N = # of endogenous variables in the VAR
* S = # of periods
* p = # of lags in the VAR
* niter = # of draws in the simulation
* siz = size of the confidence band

OUTPUT:
* irf_low = ((S+1) x T) lower range of impulse response confidence band
* irf_upp = ((S+1) x T) upper range of impulse response confidence band

DESCRIPTION

Calculates standard error of Impulse Response Function for VAR by means of Monte-Carlo simulations on the coefficient matrix.

Example
[irf_low,irf_upp]=irf_mc1(results,mres,S,N,p,niter,siz)

Example taken from function irf(). This is equivalent to the simpler call: irf(results,S,'mres=name of mres','size=siz','meth=mc1','niter=value of niter'))

scstd _____bvar scaling factor

CALLING SEQUENCE
scale=scstd(y,nobs,nlag)

PARAMETERS

INPUT:
* y = an (nobs x neqs) matrix of y-vectors in levels.
* nobs = # of observations in y
* nlag = the lag length

OUTPUT:
scale = std deviation of the residuals

DESCRIPTION
Determines bvar() function scaling factor using a univariate AR model (called by bvar1() only)

Example
scale(j,1) = scstd(ytmp,nobs,nlag);

Example taken from function bvar1 (!)

sur _____Zellner Seemingly Unrelated Regressions

CALLING SEQUENCE
rsur=sur(arg1,...,argn)

PARAMETERS

INPUT:
argi : an argument which can be:
. equations written 'vary=coef1*varx1+...+coefi*varxi'
where:

- coefi = the name of a coefficient
- varxi = the name of a variable

. 'coef=xx' where xx is a vector of coefficients names
. 'niter=x' where x is the max # iterations authorized (optional; default =100)
. 'crit=x' where x is the convergence criterion
. the string 'noprint' if the user doesn't want to print the results of the regression
. 'unequal' if the equations do not all cover the entered the time span given by the bounds (so, the user must have bounds by the use of the function bounds) and the user wants to estimate all equations over the largest time span included in the one covered by the bounds.

OUTPUT:
rsur =a results tlist with:
- rsur('meth') = 'sur'
- rsur('nobs') = # of observations
- rsur('neqs') = # of estimated equations
- rsur('ncoef') = # of estimated coefficients
- rsur('beta') = bhat
- rsur('tstat') = t-stats
- rsur('pvalue') = pvalue of the betas
- rsur('sigma') = covariance matrix of the residuals
- rsur('sigu') = (1 x neqs) sum of squared residuals
- rsur('sigu') = (1 x neqs) sum of squared residuals
- rsur('dw') = (1 x neqs) Durbin-Watson
- rsur('prests') = boolean indicating the presence or absence of a time series in the regression
- rsur('namecoef') = (ncoef x 1) mame of the coeffcients
- rsur('namey') = name of endogenous variables
- rsur('eqs') = list of the neqs equations
- rsur('coefs') = list of the coefs names in each equation

DESCRIPTION
Computes Zellner Seemingly Unrelated Regression. Some coefficients can be common to several equations. There can be spaces in the text of the equations. If you want to introduce a constant in your equation, you can omit the '*varxi' in the text of the equation. The exogenous variable can be expressed anyway. The only constraint is that the model must be linear in its coefficients

EXAMPLE
1) eq1='igm=a1*fgm+a2*cgm+a3'; eq2='ich=a4*fch+a5*cch+a6'; eq3='ige=a7*fge+a8*cge+a9';
eq4='iwest=a10*fwest+a11*cwest+a12'; eq5='iuss=a13*fuss+a14*cuss+a15';
r=sur(eq1,eq2,eq3,eq4,eq5)
2) sur('y1=c1*x1+c2*x2','y2=c1*x1+c3*log(x4)','coeff=[c1,c2,c3]','itmax=20','noprint')

Example 1 is taken from function sur_d. Here coefficients names are a1 to a15 and are not given as input of the function. # of iterations is set to default. Results are printed. Example 2 makes use of all options. Note that the coefficient c1 is imposed to be the same in equations for y1 and y2.

syslist _____recovers objects names in a system of equations

CALLING SEQUENCE
[lx,listcoef,ncoefeqs,xx,namexos,boundsvarb]=syslist(l,coef,boundsvarb)

PARAMETERS

INPUT:
* eq = a string of the form :
* 'varendo=coefi*varex1+...+coefj*varexk' with varexi possibly lacking
* coef = a string vector of the form coef=['coef1';...;'coefn']

OUTPUT:
* namey = the name of the rhs variable
* ncoefeqi = a (nx1) vector of the indexes of coefi,...,coefj in coef
* lexo = a (nx1) vector of names of the exogenous variables

DESCRIPTION
Recovers from an equation the name of the endogenous variable, the indexes of the coefficients (in a vector) and the names of the exogenous variables (also in a vector)

Example
1) [nyi,ncoefi,lexoi]=eqlist('y1=c1*x1+c2*x2',['c1';'c2'])
2) [grocer_nyi,grocer_ncoefi,grocer_lexoi]=eqlist(varargin(grocer_i),grocer_coef)

Example 1 extracts in nyi y1, in ncoefi [1;2] and in lexoi ['x1';'x2']. Example 2 is taken from function twosls.

theilbv _____Theil-Goldberger for bvar model

CALLING SEQUENCE
results=theilbv(nlag,tight,weight,decay,arg1,...,argn)

PARAMETERS

INPUT:
* y = nobs
* x 1 input vector
* x = nobs x nvar input explanatory variables matrix
* nobs = # of observations
* neqs = # of equations
* eqn = # equation number
* nlag = the lag
* length of the VAR
* theta = Litterman's tightness hyperparameter
* weight = Litterman's weight (matrix or scalar)
* decay = Litterman's lag decay = lag^(-decay)

* scale = scaling vector (determined in bvar)
* scale2 = scaling vector (determined in bvar)
* nx = # of deterministic variables excluding constant term

OUTPUT:
results = a results tlist with:
. results('meth') = 'bvar'
. results('beta') = bhat
. results('tstat') = t-statistics
. results('tprob') = t-probabilities
. results('yhat') = yhat
. results('resid') = residuals
. results('sige') = e'*e/(n-k)
. results('rsqr') = rsquared
. results('rbar') = rbar-squared
. results('nobs') = nobs
. results('nvar') = nvar

DESCRIPTION
Performs Theil-Goldberger for bvar model. Used in function bvar1.

Example
bresult = theilbv(yvec,ymat,nlag,neqs,i,tight,weight,decay,scale2,scale,nx)

Example taken from function bvar1. Should not have many other uses!

threesls _____Three-Stage Least-squares Regression

CALLING SEQUENCE
[results]=threesls(arg1,...,argn)

PARAMETERS

INPUT:
argi: an argument which can be:
* an equation of the following form:
'vary=coef1*varx1+...+coefi*varxi'
where:
- coefi = the name of a coefficient
- varxi = the name of a variable
* 'coef=coef1;coef2;...coefn' where coef1,...,coefn are the names of the coefficients in the system (optional; default: 'coef=a1;...,an')
* 'endo =[endo1;...;endon]' where endo1,...,endon are the names of the endogenous variables (optional; necessary if the names of the endogenous variables in the rhs of the equations are not the same as those of the lhs; default: the names of all the lhs sides of the equations)
* 'noprint' if you do not want to print the results

OUTPUT:
results =a results tlist with:
- results('meth') = 'threesls'
- results('namecoef') = the matrix of the names of the coefficients
- results('riv1'),...,results('rivn) = the results of the iv estimation for each equation (see iv for more details)

DESCRIPTION
Computes Three-Stage Least-squares Regression

Example
r=threesls('y1=a1+b1*x1','y2=a2+b2*x2+c2*y1','y3=a3+b3*x3+c3*y2+d3*x2','coef=a1;a2;a3;b1;b2;b3;c1;c2;c3;d3')

Example taken from function threesls_d. The equations are 'y1=a1+b1*x1','y2=a2+b2*x2+c2*y1' and 'y3=a3+b3*x3+c3*y2+d3*x2'.Coefficients are a1, a2, a3, b1, b2, b3, c1, c2, c3, d3. Their name is given to the function through the input 'coef=a1;a2;a3;b1;b2;b3;c1;c2;c3;d3'.

twosls _____ Two--Stage Least-squares Regression

CALLING SEQUENCE
[results]=twosls(arg1,...,argn)

PARAMETERS

INPUT:
argi : an argument which can be:
* an equation of the following form:
  'vary=coef1*varx1+...+coefi*varxi'
  where:
  - coefi = the name of a coefficient
  - varxi = the name of a variable
* 'coef=coef1;coef2;...coefn' where coef1,...,coefn are the names of the coefficients in the system (optional; default: 'coef=a1;...,an')
* 'endo =[endo1;...;endon]' where enod1,...,endon are the names of the endogenous variables (optional; necessary if the names of the endogenous variables in the rhs of the equations are not the same as those of the lhs; default: the names of all the lhs sides of the equations)
* 'noprint' if you do not want to print the results

OUTPUT:
results =a results tlist with:
- results('meth') = 'tsls'
- results('namecoef') = the matrix of the names of the coefficients
- results('riv1'),...,results('rivn) = the results of the iv estimation for each equation (see iv for more details)

DESCRIPTION
Computes Two-Stage Least-squares Regression.

Example
rt=twosls('y1=a+b*x1','y2=d+e*(y1-x2)+f*x2','coef=a;b;d;e;f')

Example taken from function twosls_d. The equations are 'y1=a+b*x1' and 'y2=d+e*(y1-x2)+f*x2'. Coefficients are a, b, d, e and f. Their name is given to the function through the input 'coef=a;b;d;e;f'.

var_____VAR Regression

CALLING SEQUENCE
rvar=var(p,arg1,...argn)

PARAMETERS

INPUT:
* p = the lag length of the VAR
* argi = an argument which can be:
  - 'endo=[var1 var2 ... varn]' with vari: the ith endogenous variable in the var
  - 'exo=[var1 var2 ... varn]' with vari: the ith exogenous variable in the var
  - the string 'noprint' if the user doesn't want to print the results of the regression
---
OUTPUT:
rvar = a results tlist with:
. rvar('meth') = 'var'
. rvar('y') = y data vector
. rvar('x') = x data matrix
. rvar('nobs') = # observations
. rvar('nvar') = # exogenous variables
. rvar('neqs') = # endogenous variables
. rvar('resid') = residuals, with rvar('resid')(:,i): residuals for equation # i
. rvar('beta') = bhat, with rvar('beta')(:,i): coefficients for equation # i
. rvar('rsqr') = rsquared, with rvar('rsqr')(i) : rsquared for equation # i
. rvar('overallf') = F-stat for the nullity of coefficients other than the constant with: rvar('overallf')(i): F-stat for equation # i
. rvar('pvaluef') = their significance level with: rvar('pvaluef')(i): significance level for equation # i
. rvar('rbar') = rbar-squared
. rvar('sigu') = sums of squared residuals with rvar('sigu')(:,i): sum of squared error for equation # i
. rvar('tstat') = t-stats, with rvar('tstat')(:,i): t-stat for equation # i
. rvar('pvalue')= pvalue of the betas, with rvar('pvalue')(:,i): p-value for equation # i
. rvar('dw') = Durbin-Watson Statistic, with: rvar('dw')(i): DW for equation # i
. rvar('condindex') = multicolinearity cond index, with rvar('condindex')(i): cond index for equation # i
. rvar('boxq') = Box Q-stat, with rvar('boxq')(i): Box Q-stat for equation # i

. rvar('aic') = Akaïke information criterion
. rvar('bic') = Schwartz information criterion
. rvar('hq') = Hannan-Quinn information criterion
. rvar('prests') = boolean indicating the presence or absence of a timeseries in the regression
. rvar('namey') = name of the y variable
. rvar('nx') = # of x variables
. rvar('namex') = name of the x variables (if any)
. rvar('bounds') = if there is a timeseries in the regression, the bounds of the regression

DESCRIPTION
Estimates a VAR of order p on endogenous variables, with the eventual presence of exogenous variables and, by default, displays on screen the estimation results.

Example
results=var(2,'endo=del(log(rfa_inv)),del(log(rfa_inc)),del(log(rfa_cons))')

Example taken from function var_d1() estimate a var model, taken from Lütkepohl's book "Introduction to multiple time series analysis", with three endogenous variables (the differentiated logarithm of investment, GDP and consumption) with 2 lags and no exogenous variables.

var1_____VAR Regression

CALLING SEQUENCE
rvar=var1(y,p,x)

PARAMETERS

INPUT:
y = an (nobs x neqs) matrix of y-vectors
nlag = the lag length
x = optional matrix of variables (nobs x nx) (NOTE: constant vector automatically included)

OUTPUT:
rvar = a results tlist with:
. rvar('meth') = 'var'
. rvar('y') = y data vector
. rvar('x') = x data matrix
. rvar('nobs') = # observations
. rvar('nvar') = # exogenous variables
. rvar('neqs') = # endogenous variables
. rvar('resid') = residuals, with rvar('resid')(:,i): residuals for equation # i
. rvar('beta') = bhat, with rvar('beta')(:,i): coefficients for equation # i
. rvar('rsqr') = rsquared, with rvar('rsqr')(i) : rsquared for equation # i
. rvar('overallf') = F-stat for the nullity of coefficients other than the constant with: rvar('overallf')(i): F-stat for equation # i
. rvar('pvaluef') = their significance level with: rvar('pvaluef')(i): significance level for equation # i

. rvar('rbar') = rbar-squared
. rvar('sigu') = sums of squared residuals with rvar('sigu')(:,i): sum of squared error for equation # i
. rvar('tstat') = t-stats, with rvar('tstat')(:,i): t-stat for equation # i
. rvar('pvalue')= pvalue of the betas, with rvar('pvalue')(:,i): p-value for equation # i
. rvar('dw') = Durbin-Watson Statistic, with: rvar('dw')(i): DW for equation # i
. rvar('condindex') = multicolinearity cond index, with rvar('condindex')(i): cond index for equation # i
. rvar('boxq') = Box Q-stat, with rvar('boxq')(i): Box Q-stat for equation # i
. rvar('aic') = Akaïke information criterion
. rvar('bic') = Schwartz information criterion
. rvar('hq') = Hannan-Quinn information criterion

DESCRIPTION
Estimates a VAR of order p on endogenous variables, with the eventual presence of exogenous variables. Contrary to var, do not display on screen the estimation results. Endogenous and exogenous variables must already be in a matrix form.

Examples
1) rvar=var1(grocer_y,grocer_p,x)
2) recm=var1(dy,grocer_nlag,x)

These examples are taken from functions var and ecm; endogenous variables are in matrices grocer_y and dy respectively; the number of lags are grocer_p and grocer_nlag respectively; and exogenous variables are in matrix x.