Chapter 17: Panel data estimation

GROCER contains basic panel data estimation[1]: pooled; fixed and random effects; between (part 2). The 'panel data' tlist that is the favoured method for organising data for a panel estimation is first presented (part 1) and the detailed specification of the function reported in part 3.

1. Definition of panel data.

Panel estimation works on data that have 2 dimensions: a time and an individual dimension. The best way to deal with Panel data in grocer is through a tlist whose type is 'panel data' and which has 5 components:

- a field 'dates' with reports the date of each observation
- a field 'id' which reports the numerical identity of the individual for each observation
- a field 'x' which reports the values of the variables in the panel
- a field 'nameid' which reports the name of each individual
- a field 'namex' which reports the name of each variable.

To have an example, load the panel database which is on your computer after you have loaded grocer:

```
-->load('SCI/macros/grocer/db/judgepanel.dat')

-->judgepanel
```

You can build such a database "by hand", but the best way is through the importation of a cvs file, by function impexc2bd. Your database must contain a column (or a row) labelled 'dates' and one 'id'. The dates column should contain valid grocer dates; the 'id' column should contain the names of the individuals or their numerical identities. The function impexc2bd then creates the 'panel data' tlist: if the column id contained numbers then these numbers fill the field 'id' in the tlist and the filed 'nameid' is filled with 'individual # i" where is the i's are the numerical identifier of the individuals. If the column 'id' contained names, then 'nameid' the filed 'nameid' is field with these names and the file id is filled with the content of column where each individual has been replaced by a number equal to its place in field 'nameid'.

(°)Another convenient method is provided by the function paneldb, which takes time series whose name have the following structure: first, the name of the individual, then the name of the corresponding variable, sticked together. Alternatively, the name of the variable can be given first and the name of the individual in the second place: in that case the user must give the optional arg 'conc=2' in third or fourth place. The function paneldb_d for instance build a panel tlist from individuals called 'ze', 'jp', 'ca', 'uk', 'sw', 'nw', 'sd, 'dk' and 'us' and variables called 'lxrate', 'money_s', 'rmmkt_s' and 'pic_s'. The data are therefore: zelxrate, jplxrate,..., usmmkt_s, uspic_s.

The call to function paneldb takes the following form:
paneldb(vector_of_individual_names,vector_of_variables,bounds)
where:

- vector_of_individual_names is a string vector gathering the names of the individuals;
- vector_of_ variables is a string vector gathering the names of the variables;
- bounds is a string vector of time bounds.

[1] For the most part, these functions has been built by translating and adapting matlab programs written by Carlos Alberto Castro.

For instance, creating the panel tlist of the above individuals and variables over the period 1994m1 to 2002m12 is done as follows:

```
-->pdb    =    paneldb(['ze';'jp';'ca';'uk';'sw';'nw';'sd';'dk';'us'],['money';'rmmkt';
'pic'],'bounds=[''1994m1'' ''2002m12'']');(°°)
```

2. Estimation.

GROCER provides the 3 oldest panel estimation methods: pooled, between and within, under the form of the fixed effect model. It provides also the random effect model estimation and the Hausmann test, that allows testing the validity of the random effect model against the fixed effect model.

The function pfixed takes as first argument the name of an endogenous variable. In the simplest case it takes as second argument a 'panel data' tlist: in that case the name of endogenous variable must be the names of a variable in the tlist database. If the user wants to use all variables other than this endogenous variable as exogenous variable, then there is nothing more to do. For instance estimating y=beta*x by the fixed effect method on the judgepanel 'panel data' tlist can be done as follows:

```
-->load('SCI/macros/grocer/db/judgepanel.dat');pfixed('y',judgepanel)
```

With the following result:

```
panel with fixed effects estimation results for dependent variable: y
number of observations: 40
number of variables: 5
R² = 0.6902835      adjusted R² =0.6548873
Overall F test: F(4,35) = 19.50164       p-value = 1.616D-08
standard error of the regression: 3.7472999
sum of squared residuals: 491.47897
Belsley, Kuh, Welsch Condition index: 196

variable       coeff     t-statistic p value
x              1.1190286 17.603132   0
individual # 1 2.3150096 1.0162098   0.3165032
individual # 2 10.109636 4.5323537   0.0000653
individual # 3 2.385445  1.1744961   0.2481258
individual # 4 16.171492 7.4835883   9.179D-09


                         *
                     *     *
```

Note that it is here necessary to enter the name of the endogenous variable between quotes: the program needs to recover the values of the endogenous variable from the list of names in the field 'namex'. Note also that there is no need to enter the names of the exogenous variables: the program takes all variables other than 'y' as exogenous variables. If you want to use only a subset of the variables, then add the option 'x=name1,…, namep', where the namei's are the names of the exogenous variables you want to consider.

For instance, to say that you want to use the exogenous variable x in the judepanel database, run:

```
-->pfixed('y',judgepanel,'x=x')
```

With the same result as before since 'x' is the only variable other than 'y' in the database.

You can also decide to use only a subset of individuals, by entering the option 'nameid=name1,..., namen' where name1,... are names of individuals present in the database. Using only the individuals 1 to 3 in the judepanel data base is therefore done as follows:

```
-->pfixed('y',judgepanel,'nameid=individual # 1;individual # 2;individual # 3')
```

With the following result:

```
panel with fixed effects estimation results for dependent variable: y
number of observations: 30
number of variables: 4
R² = 0.7554257      adjusted R² =0.7272056
Overall F test: F(3,26) = 26.769061       p-value = 4.121D-08
standard error of the regression: 3.6002326
sum of squared residuals: 337.00354
Belsley, Kuh, Welsch Condition index: 170

variable       coeff     t-statistic p value
x              1.126237  16.024281   5.551D-15
individual # 1 2.094389  0.8605459   0.3973545
individual # 2 9.8953514 4.1588222   0.0003086
individual # 3 2.1984012 1.0225591   0.3159371
```

*

* *

Function pfixed can also work with vectors and matrices. In that case, pfixed works as ordinary regression functions: after the vector of endogenous variables, a list of exogenous variables and options. Among the options, you must enter 'id=nn' where nn is the name of a vector of individuals. You can also decide to enter the names of the corresponding individuals by the option 'nameid=nn'.

Assume for instance that you have vectors called yy, xx and ind with the y, x and id values of the tlist panel judepanel[2]. Then:

```
-->pfixed('yy','xx','id=ind')
```

yields exactly the same results as prevously.

And:

```
-->pfixed('yy','xx','id=ind','defname=paul;john;georges;ringo')
```

Gives:

```
panel with fixed effects estimation results for dependent variable: yy
number of observations: 40
number of variables: 5
R² = 0.6902835      adjusted R² =0.6548873
Overall F test: F(4,35) = 19.50164       p-value = 1.616D-08
standard error of the regression: 3.7472999
sum of squared residuals: 491.47897
Belsley, Kuh, Welsch Condition index: 196

variable coeff     t-statistic p value
```

[2] You can create these variables this way:
-->yy=judgepanel('x')(:,1); xx=judgepanel('x')(:,2); id=judgepanel('ind')

```
xx_1     1.1190286 17.603132   0
paul     2.3150096 1.0162098   0.3165032
john     10.109636 4.5323537   0.0000653
georges  2.385445  1.1744961   0.2481258
ringo    16.171492 7.4835883   9.179D-09
```

*
* *

The function pbetween, that provides between estimation works exactly the same manner.

For instance:

```
-->pbetween('y',judgepanel)
```

Yields:

```
between estimation results for dependent variable: y
number of observations: 4
number of variables: 2
R² = 0.1697118      adjusted R² =-0.2454323
Overall F test: F(1,2) = 0.4088022       p-value = 0.5880391
standard error of the regression: 8.1593738
sum of squared residuals: 133.15076
Belsley, Kuh, Welsch Condition index: 471

variable coeff      t-statistic p value
x        1.4854628  0.6393764   0.5880391
cte      -2.7627452 -0.0413898  0.9707455
```

*
* *

The function prandom works also the same manner, but with another option: the choice of the method used to estimate the variance matrix of the residuals. Four variants are available: 'wallace', 'swamy', 'amemiya' or 'nerlove'. The default one is 'swamy'. So:

```
-->prandom('y',judgepanel)
```

is equivalent to:

```
-->prandom('y',judgepanel,'glsmeth=swamy')
```

and yields:

```
panel with random effects estimation results for dependent variable: y
number of observations: 40
number of variables: 2
R² = 0.8933859      adjusted R² =0.8905802
Overall F test: F(1,38) = 318.42551       p-value = 0
standard error of the regression: 3.6989064
sum of squared residuals: 519.91253
Belsley, Kuh, Welsch Condition index: 77

variable coeff     t-statistic p value
```

```
x        1.1193028 17.844481   0
cte      7.7375343 1.7543544   0.0874316


**************
random effects

individual     estimated effect
individual # 1 -5.3163646
individual # 2 2.3140912
individual # 3 -5.2461648
individual # 4 8.2484383

                        *
                    *     *
```

Using for instance Nerlove's method (the only one to guarantee a definite-positive variance matrix) is done as follows:

```
-->prandom('y',judgepanel,'glsmeth=nerlove')
```

Which yields to the following result, very close here to the one obtained with swamy's method:

```
panel with random effects estimation results for dependent variable: y
number of observations: 40
number of variables: 2
R² = 0.8977716      adjusted R² =0.8950814
Overall F test: F(1,38) = 333.71669       p-value = 0
standard error of the regression: 3.6112543
sum of squared residuals: 495.56398
Belsley, Kuh, Welsch Condition index: 174

variable coeff     t-statistic p value
x        1.119068  18.267914   0
cte      7.7442655 0.7361266   0.4661737


**************
random effects

individual     estimated effect
individual # 1 -5.4140062
individual # 2 2.3570346
individual # 3 -5.3436013
individual # 4 8.4005729

                        *
                    *     *
```

The last available method is the pooled estimation method, with the function ppooled. The functioning is again very similar to pfixed. The only slight difference is that you must add a constant to the regression if you want to have one in the regression.

For instance:

```
-->ppooled('y',judgepanel,'cte')
```

which yields:

```
panel pooled estimation results for dependent variable: y
number of observations: 40
number of variables: 2
R² = 0.7147287      adjusted R² =0.7072216
Overall F test: F(1,38) = 95.206529       p-value = 6.727D-12
standard error of the regression: 6.9565952
sum of squared residuals: 1838.9802
Belsley, Kuh, Welsch Condition index: 96

variable coeff     t-statistic p value
x        1.1315894 9.7573833   6.727D-12
cte      7.385193  2.1083127   0.0416536
```

*

* *

There is on last thing that can be done in grocer: testing the random effect model against the fixed effect one. This is done by function phaussman, which takes as first entry a results tlist from a fixed effect estimation a second entry a results tlist from a random effect estimation. The third entry is optional and is the symbolic name of the file where the results are printed (default is: %io(2) - the working space window).

For instance:

```
-->r1   =   pfixed('y',judgepanel,'noprint');r2   =   prandom('y',judgepanel,'noprint');
phaussman(r1,r2)
```

Yields:

```
***** Haussman Test *******
Ho: Random Effects
Ha: Fixed Effects
Statistic = 0.0007047
P-value = 0.9788224
***************************
```

(°) If you are interested in the fit of the panel regression, the you have to apply the function panelfit to your tlist result (from a pooled, fixed effect or a random effect estimation).

The syntax is the following (see function panelfit_d):
panhat = panelfit(res,paneldb)

where:

- res is a panel tlist result;
- paneldb is the panel database used in the estimation. (°°)

3. The panel estimation functions and their specifications.

paneldb_____Build a panel tlist

CALLING SEQUENCE
rpdb = paneldb(ind,var,arg1,...,argn)

PARAMETERS

INPUT:
* ind = vector of name for individual
* var = vector of generic name for variable
* arg1,...,argn = arguments which can be:
 - 'conc = 1' if concatenation must begins by individual names (2 instead, default =1)
 - 'bounds =["bounds1" "bounds2"]' bounds for data

OUTPUT:
rpdb = a tlist in "panel format"
 - rpdb('panel data') = type of the list
 - rpdb('dates') = bounds of data
 - rpdb('id') = individual indentifiant
 - rpdb('x') = stacked data in matrix format
 - rpdb('nameid') = names of individual indentifiant
 - rpdb('namex') = generic names of data

DESCRIPTION
Build a panel tlist from existing data.

Example
lco = ['ze';'jp';'ca';'uk';'sw';'nw';'sd';'dk';'us'];
lvar = ['lxrate';'money_s';'rmmkt_s';'pic_s'];
pdb = paneldb(lco,lvar,'bounds=["1994m1" "2002m12"]');

Example taken from function paneldb_d. The database contains 4 variables for 9 countries in TS format. For instance, zelxrate is the logarithm of the exchange rate of the euro against the dollar. They are transformed in a tlist that can be used in any panel estimation method available in grocer.

panelfit_____Fitted values in a panel regression

CALLING SEQUENCE
panhat = panelfit(res,paneldb)

PARAMETERS

INPUT:
* res = tlist result of panel estimation
* paneldb = tlist containing data in panel format (use of tlist as input allow user to make out-of-sample forecasts by using a database with different time span, see paneldb.sci for the construction of a panel tlist)

OUTPUT:
* panhat = a tlist with
  . panhat('meth') ='panel fit'
  . panhat('yhat') = matrix of fitted data
  . panhat('bounds') = bounds of the adjustment

DESCRIPTION
Computes the fitted values of a panel regression (works with pooled, fixed effect & random effect estimation).

Example
load('SCI\macros\grocer\db\judgepanel.dat');
r1 = pfixed('y',judgepanel);
rfit = panelfit(r1,judgepanel);

Example taken from function panelfit_d. The example uses the examples provided by Judge's textbook, uses the fixed effect estimation method and then provides the fit of the model.

pfixed_____Fixed Estimation for Panel Data

CALLING SEQUENCE
res=pfixed(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes (this last case is the only one authorized if you are using a 'panel data' tlist, see below)
* arg1:
  - either a 'panel data' tlist (generally imported from a .csv database by function impexc2bd)
  - or an endogenous variable taking the form of 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
* arg2 to argn:
  - if arg1 was a 'panel data' tlist
  then:
    other input are optional and can be:
    * either 'x=name1;...;namep' where name1,...,namep are a subset of the names of the variables that are in the database
    * the string 'nameid=name1,..., namen' where name1,... are names of individuals present in the database
    * the string 'glsmeth=n' where n is the name of a method available to estimate the gls parameters, that is 'wallace', 'swamy', 'amemiya' or 'nerlove' (default: 'swamy')
    * or the string 'noprint' if the user does not want to print the estimation results
  - if arg1 was an endogenous variable
  then either:
    * a time series
    * a real (nxk) matrix
    * a (kx1) string vector of names of time series, vectors or matrices

* the string 'id=v' where v is the vector of individuals attached to the y and x data (this argument must be present somewhere in the list of variables arguments)
* the string 'glsmeth=n' where n is the name of a method available to estimate the gls parameters, that is 'wallace', 'swamy', 'amemiya' or 'nerlove' (default: 'swamy')
* the string 'noprint' if the user doesn't want to print the results of the regression

---

OUTPUT:
res = a tlist with
. res('meth')='panel with random effects'
. res('y') = y data vector
. res('x') = x data matrix
. res('nobs') = nobs
. res('nvar') = nvars
. res('beta') = bhat
. res('yhat') = yhat
. res('resid') = residuals
. res('vcovar') = estimated variance-covariance matrix of beta
. res('sige') = estimated variance of the residuals
. res('sige') = estimated variance of the residuals
. res('ser') = standard error of the regression
. res('tstat') = t-stats
. res('pvalue') = pvalue of the betas
. res('condindex') = multicolinearity cond index
. res('prescte') = boolean indicating the presence or absence of a constant in the regression
. res('rsqr') = rsquared
. res('rbar') = rbar-squared
. res('f') = F-stat for the nullity of coefficients other than the constant
. res('pvaluef') = its significance level
. res('gls estimation method') = the gls method used
. res('random effects') = the estimation of the individual effects
. res('res0') = residuals from the original model
. res('alfa') = the gls parameters

DESCRIPTION
Performs Fixed Effects Estimation for Panel Data. This is a high level function that can work with various data types, but mainly with a *panel data* tlist, created for example by the importation of an excel database.

Example
load('SCI\macros\grocer\db\judgepanel.dat'); r0 = pfixed('y',judgepanel);

The tlist judgepanel contains the endogenous variable y and the exogenous variable x.

pbetween_____Between Estimation for Panel Data

CALLING SEQUENCE
res=pbetween(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes (this last case is the only one authorized if you are using a 'panel data' tlist, see below)
* arg1:
 - either a 'panel data' tlist (generally imported from a .csv database by function impexc2bd)
 - or an endogenous variable taking the form of 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
* arg2 to argn:
 - if arg1 was a 'panel data' tlist
 then:
 other input are optional and can be:
 * either 'x=name1;...;namep' where name1,...,namep are a subset of the names of the variables that are in the database
 * the string 'nameid=name1,..., namen' where name1,... are names of individuals present in the database
 * the string 'glsmeth=n' where n is the name of a method available to estimate the gls parameters, that is 'wallace', 'swamy', 'amemiya' or 'nerlove' (default: 'swamy')
 * or the string 'noprint' if the user does not want to print the estimation results
 - if arg1 was an endogenous variable
 then either:
 * a time series
 * a real (nxk) matrix
 * a (kx1) string vector of names of time series, vectors or matrices
 * the string 'id=v' where v is the vector of individuals attached to the y and x data (this argument must be present somewhere in the list of variables arguments)
 * the string 'glsmeth=n' where n is the name of a method available to estimate the gls parameters, that is 'wallace', 'swamy', 'amemiya' or 'nerlove' (default: 'swamy')
 * the string 'noprint' if the user doesn't want to print the results of the regression
---
OUTPUT:
res = a tlist with
 . res('meth')='panel with random effects'
 . res('y') = y data vector
 . res('x') = x data matrix
 . res('nobs') = nobs
 . res('nvar') = nvars
 . res('beta') = bhat
 . res('yhat') = yhat
 . res('resid') = residuals
 . res('vcovar') = estimated variance-covariance matrix of beta
 . res('sige') = estimated variance of the residuals
 . res('sige') = estimated variance of the residuals
 . res('ser') = standard error of the regression
 . res('tstat') = t-stats
 . res('pvalue') = pvalue of the betas
 . res('condindex') = multicolinearity cond index
 . res('prescte') = boolean indicating the presence or absence of a constant in the regression
 . res('rsqr') = rsquared
 . res('rbar') = rbar-squared

. res('f') = F-stat for the nullity of coefficients other than the constant
. res('pvaluef') = its significance level
. res('gls estimation method') = the gls method used
. res('random effects') = the estimation of the individual effects
. res('res0') = residuals from the original model
. res('alfa') = the gls parameters

DESCRIPTION
Performs Between Estimation for Panel Data. This is a high level function that can work with various data types, but mainly with a *panel data* tlist, created for example by the importation of an excel database.

Example
load('SCI\macros\grocer\db\judgepanel.dat'); r0 = pbetween('y',judgepanel);

The tlist judgepanel contains the endogenous variable y and the exogenous variable x.

pfixed1_____Fixed Estimation for Panel Data

CALLING SEQUENCE
res= pfixed1(y,index,x,z)

PARAMETERS

INPUT:
* y = a (nobs x neqs) matrix of all of the individual's observations vertically concatenated. This matrix must include in the first column the dependent variable, the independent variables must follow accordingly.
* index = index vector that identifies each observation with an individual
e.g. 1 (first 2 observations for individual # 1)
1
2 (next 1 observation for individual # 2)
3 (next 3 observations for individual # 3)
3
3
* z = optional matrix of exogenous variables, dummy variables.

OUTPUT:
res = a tlist with
. rpanel('meth')='panel with fixed effects'
. rpanel('y') = y data vector
. rpanel('x') = x data matrix
. rpanel('nobs') = nobs
. rpanel('nvar') = nvars
. rpanel('beta') = bhat
. rpanel('yhat') = yhat
. rpanel('resid') = residuals
. rpanel('vcovar') = estimated variance-covariance matrix of beta
. rpanel('sige') = estimated variance of the residuals
. rpanel('sigu') = sum of squared residuals

. rpanel('ser') = standard error of the regression
. rpanel('tstat') = t-stats
. rpanel('pvalue') = pvalue of the betas
. rpanel('condindex') = multicolinearity cond index
. rpanel('prescte') = boolean indicating the presence or absence of a constant in the regression
. rpanel('lliked') = log-likelihood
. rpanel('rsqr') = rsquared
. rpanel('rbar') = rbar-squared
. rpanel('f') = F-stat for the nullity of coefficients other than the constant
. rpanel('pvaluef') = its significance level

DESCRIPTION
Performs Fixed Effects Estimation for Panel Data. This is the low level version that works with only matrices.

Example
res=pfixed1(grocer_y,grocer_id,grocer_x)

Example taken from function pfixed.

phausmann_____Haussman test for Panel Data

CALLING SEQUENCE
res= phaussman(result1,result2,out)

PARAMETERS

INPUT:
* results1 = a tlist returned by pfixed()
* results2 = a tlist returned by prandom()
* prt = 'noprint' if the user does not want to print the result on the screen
---
OUTPUT:
res = a results tlist with
* res('meth') = 'panel haussman'argi1
* res('pfixed res') = the tlist returned by pfixed()
* res('prandom res') = the tlist returned by prandom()
* res('stat') = the Haussman statistics
* res('pvalue') = the corresponding p-value

DESCRIPTION
Performs Haussman test, used for testing the specification of the fixed or random effects model.

Example
phaussman(r1,r2);

Example taken from function panel_d.

ppooled_____Pooled Least Squares for Panel Data

CALLING SEQUENCE
res= ppooled(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes (this last case is the only one authorized if you are using a 'panel data' tlist, see below)
* arg1:
 - either a 'panel data' tlist (generally imported from a .csv database by function impexc2bd)
 - or an endogenous variable taking the form of 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
* arg2 to argn:
 - if arg1 was a 'panel data' tlist
 then:
 other input are optional and can be:
 * either 'x=name1;...;namep' where name1,...,namep are a subset of the names of the variables that are in the database
 * the string 'nameid=name1,..., namen' where name1,... are names of individuals present in the database
 * the string 'glsmeth=n' where n is the name of a method available to estimate the gls parameters, that is 'wallace', 'swamy', 'amemiya' or 'nerlove' (default: 'swamy')
 * or the string 'noprint' if the user does not want to print the estimation results
 - if arg1 was an endogenous variable
 then either:
 * a time series
 * a real (nxk) matrix
 * a (kx1) string vector of names of time series, vectors or matrices
 * the string 'id=v' where v is the vector of individuals attached to the y and x data (this argument must be present somewhere in the list of variables arguments)
 * the string 'glsmeth=n' where n is the name of a method available to estimate the gls parameters, that is 'wallace', 'swamy', 'amemiya' or 'nerlove' (default: 'swamy')
 * the string 'noprint' if the user doesn't want to print the results of the regression
 ---
OUTPUT:
res = a tlist with
 . res('meth') = 'panel pooled'
 . res('y') = y data vector
 . res('x') = x data matrix
 . res('nobs') = nobs
 . res('nvar') = nvars
 . res('beta') = bhat
 . res('yhat') = yhat
 . res('resid') = residuals
 . res('vcovar') = estimated variance-covariance matrix of beta
 . res('sige') = estimated variance of the residuals
 . res('sigu') = sum of squared residuals

. res('ser') = standard error of the regression
. res('tstat') = t-stats
. res('pvalue') = pvalue of the betas
. res('condindex') = multicolinearity cond index
. res('prescte') = boolean indicating the presence or absence of a constant in the regression
. res('rsqr') = rsquared
. res('rbar') = rbar-squared
. res('f') = F-stat for the nullity of coefficients other than the constant
. res('pvaluef') = its significance level
. res('prests') = boolean indicating the presence or absence of a time series in the regression
. res('namey') = name of the y variable
. res('namex') = name of the x variables

DESCRIPTION
Performs Pooled Least Squares for Panel Data. This is a high level function that can work with various data types, but mainly with a *panel data* tlist, created for example by the importation of an excel database.

Example
load('SCI\macros\grocer\db\judgepanel.dat'); r0 = ppooled('y',judgepanel,'cte');

Example taken from function panel_d. The tlist judgepanel contains the endogenous variable y and the exogenous variable x. A constant has been added to the list of this exogenous.

ppooled1_____Pooled Least Squares for Panel Data

CALLING SEQUENCE
res=ppooled1(y,x)

PARAMETERS

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

OUTPUT:
res = a tlist with
. res('meth') = 'panel pooled'
. res('y') = y data vector
. res('x') = x data matrix
. res('nobs') = nobs
. res('nvar') = nvars
. res('beta') = bhat
. res('yhat') = yhat
. res('resid') = residuals
. res('vcovar') = estimated variance-covariance matrix of beta
. res('sige') = estimated variance of the residuals
. res('sige') = estimated variance of the residuals
. res('ser') = standard error of the regression

. res('tstat') = t-stats
. res('pvalue') = pvalue of the betas
. res('condindex') = multicolinearity cond index
. res('prescte') = boolean indicating the presence or absence of a constant in the regression
. res('rsqr') = rsquared
. res('rbar') = rbar-squared
. res('f') = F-stat for the nullity of coefficients other than the constant
. res('pvaluef') = its significance level

DESCRIPTION
Performs Pooled Least Squares for Panel Data. This is the low level version that works with only matrices.

Example
res=ppooled1(grocer_y,grocer_x)

Example taken from function ppooled.

prandom_____Random Effects Estimation

CALLING SEQUENCE
res=prandom(namey,arg1,…,argn)

PARAMETERS

INPUT:
* namey = a real (nx1) vector or a string equal to the name of a time series or a (nx1) real vector between quotes (this last case is the only one authorized if you are using a 'panel data' tlist, see below)
* arg1:
- either a 'panel data' tlist (generally imported from a .csv database by function impexc2bd)
- or an endogenous variable taking the form of 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
* arg2 to argn:
- if arg1 was a 'panel data' tlist
then:
other input are optional and can be:
* either 'x=name1;...;namep' where name1,...,namep are a subset of the names of the variables that are in the database
* the string 'nameid=name1,..., namen' where name1,... are names of individuals present in the database
* or the string 'noprint' if the user does not want to print the estimation results
- if arg1 was an endogenous variable
then either:
* a time series
* a real (nxk) matrix
* a (kx1) string vector of names of time series, vectors or matrices
* the string 'id=v' where v is the vector of individuals attached to the y and x data (this argument must be present somewhere in the list of variables arguments)
* the string 'noprint' if the user doesn't want to print the results of the regression
---

OUTPUT:
. res('meth')='panel with random effects'
. res('y') = y data vector
. res('x') = x data matrix
. res('nobs') = nobs
. res('nvar') = nvars
. res('beta') = bhat
. res('yhat') = yhat
. res('resid') = residuals
. res('vcovar') = estimated variance-covariance matrix of beta
. res('sige') = estimated variance of the residuals
. res('sige') = estimated variance of the residuals
. res('ser') = standard error of the regression
. res('tstat') = t-stats
. res('pvalue') = pvalue of the betas
. res('condindex') = multicolinearity cond index
. res('prescte') = boolean indicating the presence or absence of a constant in the regression
. res('rsqr') = rsquared
. res('rbar') = rbar-squared
. res('f') = F-stat for the nullity of coefficients other than the constant
. res('pvaluef') = its significance level
. res('gls estimation method') = the gls method used
. res('random effects') = the estimation of the individual effects
. res('res0') = residuals from the original model
. res('alfa') = the gls parameters
. res('prests') = boolean indicating the presence or absence of a time series in the regression
. res('namey') = name of the y variable
. res('namex') = name of the x variables

DESCRIPTION
Performs Random Effects Estimation for Panel Data (for balanced or unbalanced data). This is a high level function that can work with various data types, but mainly with a *panel data* tlist, created for example by the importation of an excel database.

Example
load('SCI\macros\grocer\db\judgepanel.dat'); r2 = prandom('y',judgepanel);

Example taken from function panel_d. The database used has the form of a tlist. The name of the endogenous variable in this database is y.

prandom1_____Random Effects Estimation

CALLING SEQUENCE
res=prandom1(meth,y,index,x,z)

PARAMETERS

INPUT:
* meth = the gls estimation method used ('wallace', 'swamy', 'amemiya' or 'nerlove')

```
* y  = a (nobs x 1) vector of endogenous variable
* index = a (nobs x 1) index vector that identifies each
  observation with an individual
  e.g. 1  (first 2 observations  for individual # 1)
       1
       2  (next  1 observation   for individual # 2)
       3  (next  3 observations  for individual # 3)
       3
       3
* x = matrix of exogenous variables
* z = optional matrix of exogenous variables, dummy variables.

------------------------------------------------------------
OUTPUT:
res = a tlist with
  . res('meth')='panel with random effects'
  . res('y')     = y data vector
  . res('x')     = x data matrix
  . res('nobs')  = nobs
  . res('nvar')  = nvars
  . res('beta')  = bhat
  . res('yhat')  = yhat
  . res('resid') = residuals
  . res('vcovar') = estimated variance-covariance matrix of beta
  . res('sige')  = estimated variance of the residuals
  . res('sige')  = estimated variance of the residuals
  . res('ser')  = standard error of the regression
  . res('tstat') = t-stats
  . res('pvalue') = pvalue of the betas
  . res('condindex') = multicolinearity cond index
  . res('prescte') = boolean indicating the presence or absence of a constant in the regression
  . res('rsqr')  = rsquared
  . res('rbar')  = rbar-squared
  . res('f')    = F-stat for the nullity of coefficients other than the constant
  . res('pvaluef') = its significance level
  . res('gls estimation method') = the gls method used
  . res('random effects') = the estimation of the individual effects
  . res('res0') = residuals from the original model
  . res('alfa') = the gls parameters
```

DESCRIPTION

Performs Random Effects Estimation for Panel Data (for balanced or unbalanced data). This is the low level version that works with only matrices.

Example

res=prandom1(grocer_meth,grocer_y,grocer_id,grocer_x)

Example taken from function prandom.