Package: PooledMeanGroup (via r-universe)

October 31, 2024

Version 1.0

Date 2017-12-13

Title Pooled Mean Group Estimation of Dynamic Heterogenous Panels

Author Piotr Zientara [aut], Lech Kujawski [aut, cre]

Maintainer Lech Kujawski <lech.kujawski@ug.edu.pl>

Depends R (>= 3.2.3)

Description Calculates the pooled mean group (PMG) estimator for dynamic panel data models, as described by Pesaran, Shin and Smith (1999) <doi:10.1080/01621459.1999.10474156>.

License GPL (>= 2)

URL https://www.r-project.org

NeedsCompilation no

Repository CRAN

Date/Publication 2017-12-14 13:08:28 UTC

Contents

PooledMeanGroup-package . . . 2
BGtest . . . 3
ConoverMulti . . . 4
DataExp . . . 5
DiffPanel . . . 6
GQtest . . . 7
JBtest . . . 8
LagPanel . . . 9
optimPMG . . . 10
PanelNaOmit . . . 12
PMG . . . 13

Index **16**

PooledMeanGroup-package *Pooled Mean Group Estimation of Dynamic Heterogenous Panels*

Description

Calculates the pool mean group (PMG) estimator for dynamic panel data models, as described by Pesaran, Shin and Smith (1999) <doi:10.1080/01621459.1999.10474156>. This estimator enables the intercepts, short-run coefficient and error variances to differ freely across groups, but restricts the long-run coefficients to being equal. Additionally, it allows the numbers of time series observations to differ freely across groups. This software also performs diagnostic tests of error terms, such as autocorrelation, heteroscedasticity and normality. Calculates the pooled mean group (PMG) estimator for dynamic panel data models, as described by Pesaran, Shin and Smith (1999) <doi:10.1080/01621459.1999.10474156>.

Details

The DESCRIPTION file:

Package:	PooledMeanGroup
Version:	1.0
Date:	2017-12-13
Title:	Pooled Mean Group Estimation of Dynamic Heterogenous Panels
Authors@R:	c(person("Piotr", "Zientara", role="aut", email="zientara@fest.pl"), person("Lech", "Kujawski", role=c("aut",
Author:	Piotr Zientara [aut], Lech Kujawski [aut, cre]
Maintainer:	Lech Kujawski <lech.kujawski@ug.edu.pl>
Depends:	R (>= 3.2.3)
Description:	Calculates the pooled mean group (PMG) estimator for dynamic panel data models, as described by Pesaran, S
License:	GPL (>= 2)
URL:	https://www.r-project.org

Index of help topics:

BGtest	BGtest
ConoverMulti	ConoverMulti
DataExp	DataExp
DiffPanel	DiffPanel
GQtest	GQtest
JBtest	JBtest
LagPanel	LagPanel
PMG	PMG
PanelNaOmit	PanelNaOmit
PooledMeanGroup-package	Pooled Mean Group Estimation of Dynamic Heterogenous Panels
optimPMG	optimPMG

Author(s)

Lech Kujawski, Piotr Zientara Piotr Zientara [aut], Lech Kujawski [aut, cre]

Maintainer: Lech Kujawski <lech.kujawski@ug.edu.pl>

References

Pesaran, Shin and Smith (1999) <doi:10.1080/01621459.1999.10474156>

`BGtest` *BGtest*

Description

Tests autocorrelation between the current and lagged residuals. The test is a joint test of the first P autocorrelations

Usage

```
BGtest(residuals, explvariab, acor.ord)
```

Arguments

`residuals`	residuals for group i
`explvariab`	explanatory variables (regressors) for group i
`acor.ord`	order of tested autocorrelations

Details

Calculates statistics and probs of the Breusch-Godfrey autocorrelation test (with two variants: chi-squared and F

Value

Chi-squared and F statistics with probs

Author(s)

Lech Kujawski, Piotr Zientara

Examples

```
# creating artificial variables
x1=rnorm(30,0,1)
x2=rnorm(30,0,1)
e=rnorm(30,0,0.2)
y=1+2*x1+3*x2+e
# any model
model=lm(y~x1+x2)
# BGtest
ExpBGtest=BGtest(residuals=resid(model), explvariab=cbind(x1,x2), acor.ord=4)
ExpBGtest
```

`ConoverMulti` *ConoverMulti*

Description

Tests for homoscedasticity among subsamples k within a particular group i (note that the Conover test is a non-parametric test)

Usage

```
ConoverMulti(residuals, subsample)
```

Arguments

`residuals`	residuals for group i
`subsample`	the vector of c(s1i, s2i,..., ski), where s1i+s2i+...+ski=Ti (i.e., the vector divides a particular group i into subsamples)

Details

Calculates chi-squared statistic with a prob

Value

Chi-squared statistic with a prob

Author(s)

Lech Kujawski, Piotr Zientara

Examples

```
# creating artificial variables
x1=rnorm(30,0,1)
x2=rnorm(30,0,1)
e=rnorm(30,0,0.2)
y=1+2*x1+3*x2+e
# any model
model=lm(y~x1+x2)
# ConoverMulti
ExpConoverMulti=ConoverMulti(residuals=resid(model), subsample=c(10,10,10))
ExpConoverMulti
```

DataExp *DataExp*

Description

A dataset in the form of stacked time-series. Quarterly data cover nine countries (Poland, Bulgaria, the Czech Republic, Hungary, Latvia, Lithuania, Romania, Slovakia, Slovenia; the numbers denoting particular countries form a series i=1,2,...,9) from 2005q2 to 2013q4. The dataset contains the following (below) variables

Usage

```
data("DataExp")
```

Format

A data frame with 315 observations on the following 16 variables.

`y10` a numeric vector

`y10spread` a numeric vector

`riskavers` a numeric vector

`debt` a numeric vector

`deficit` a numeric vector

`openess` a numeric vector

`cpi` a numeric vector

`growth` a numeric vector

`crisk` a numeric vector

`urate` a numeric vector

`iip` a numeric vector

`iipnetto` a numeric vector

`cagdp` a numeric vector

`caresvs` a numeric vector

`cds` a numeric vector

`bidask` a numeric vector

Examples

```
data(DataExp)
```

`DiffPanel` *DiffPanel*

Description

Calculates first differences of a particular variable from a panel data set

Usage

```
DiffPanel(variable, quantity)
```

Arguments

`variable`	a particular variable from a panel data set in the form of stacked time series; in practice; a selected singular column from a panel data set
`quantity`	a vector of the number of time series observations in each group; in practice, it takes the form c(T1,...Tn) since the PMG allows the numbers of time series observations to differ freely across groups (if the number of time series observations in each group is the same, then c(T,...,T) and T=T1=T2=...=Tn

Details

Calculates first differences of a particular variable from a panel data set in order to bring it to stationarity. Preserves the original dimension of time series observations in each group, completing data lost due to differentiating by inserting "NA"

Value

First differences of a particular variable from a panel data set

Author(s)

Lech Kujawski, Piotr Zientara

Examples

```
# first import DataExp, i=1...9, T1=T2=...T9=35
data(DataExp)
DataExp[1:5,]
# then execute DiffPanel
y10=data.frame(y10=DataExp[,1], row.names=row.names(DataExp))
dy10=DiffPanel(variable=y10, quantity=rep(35,9))
diip=DiffPanel(variable=DataExp[,11], quantity=rep(35,9))
cbind(y10,dy10,diip)[1:5,]
```

GQtest *GQtest*

Description

Tests for homoscedasticity

Usage

```
GQtest(residuals, subsample, nep)
```

Arguments

residuals	residuals for group i
subsample	the vector of c(s1i, s2i), where s1i=Ti/2 and s2i=s1i+1 (if Ti is EVEN) or where s1i=Ti/2-0.5 and s2i=s1i+1 (if Ti is UNEVEN)
nep	the number of estimated parameters for group i

Details

Calculates F statistic with a prob

Value

F statistic with a prob

Author(s)

Lech Kujawski, Piotr Zientara

Examples

```
# creating artificial variables
x1=rnorm(30,0,1)
x2=rnorm(30,0,1)
e=rnorm(30,0,0.2)
y=1+2*x1+3*x2+e
# any model
model=lm(y~x1+x2)
#BGtest
ExpGQtest=GQtest(residuals=resid(model), subsample=c(15,16), nep=3)
ExpGQtest
```

JBtest *JBtest*

Description

Tests for normality

Usage

```
JBtest(residuals)
```

Arguments

residuals	residuals for group i

Details

Calculates chi-squared statistic with a prob

Value

Chi-squared statistic with a prob

Author(s)

Lech Kujawski, Piotr Zientara

Examples

```
# creating artificial variables
x1=rnorm(30,0,1)
x2=rnorm(30,0,1)
e=rnorm(30,0,0.2)
y=1+2*x1+3*x2+e
# any model
model=lm(y~x1+x2)
#JBtest
ExpJBtest=JBtest(residuals=resid(model))
ExpJBtest
```

LagPanel *LagPanel*

Description

Provides the first lag of a particular variable from a panel data set

Usage

```
LagPanel(variable, quantity)
```

Arguments

variable	a particular variable from a panel data set in the form of stacked time series; in practice; a selected singular column from a panel data set
quantity	a vector of the number of time series observations in each group; in practice, it takes the form c(T1,...Tn) since the PMG allows the numbers of time series observations to differ freely across groups (if the number of time series observations in each group is the same, then c(T,...,T) and T=T1=T2=...=Tn

Details

Provides the first lag of a particular variable from a panel data set. Preserves the original dimension of time series observations in each group, completing data lost due to lagging by inserting "NA"

Value

A lagged particular variable from a panel data set

Author(s)

Lech Kujawski, Piotr Zientara

Examples

```
# first import DataExp, i=1...9, T1=T2=...T9=35
data(DataExp)
DataExp[1:5,]
# then execute LagPanel
y10=data.frame(y10=DataExp[,1], row.names=row.names(DataExp))
ly10=LagPanel(variable=y10, quantity=rep(35,9))
ldebt=LagPanel(variable=DataExp[,4], quantity=rep(35,9))
cbind(y10,ly10,ldebt)[1:5,]
```

optimPMG *optimPMG*

Description

Estimates parameters of long-run and short-run relationships. Makes use of a "back-substitution" algorithm, as described by Pesaran, Shin and Smith (1999). Also estimates the information matrix as well as standard errors of estimations, as indicated in Equation 13 (Pesaran, Shin and Smith, 1999). Calculates Student's t-distribution type statistics, probs and confidence intervals. Also performs diagnostic tests of error terms, such as the Breusch-Godfrey autocorrelation test, the Goldfeld-Quandt heteroscedasticity test, the Conover nonparametric test of homogeneity of variance and the Jarque-Bera normality test

Usage

```
optimPMG(dLL, maxIter, TetaStart, vecSR, vecLR, dataset, quantity, const)
```

Arguments

dLL	a parameter indicating the convergence criterion; an optimization algorithm is stopped when an increase in concentrated log-likelihood function (Equation 8 in Pesaran, Shin and Smith (1999)) is less than dLL; the default value is dLL=10^-10
maxIter	a maximum number of iterations; the default value is 200
TetaStart	a vector of first (initial) Teta values, from which the algorithm starts searching for parameters ensuring the maximization of log-likelihood function
vecSR	a list of vectors containing the column numbers of variables in short-run relationships for each group (alternatively a list of vectors containing the variables names instead of column numbers). In each vector of the list the first number must indicate dy (i.e., the dependant variable)
vecLR	a vector containing the column numbers of variables in long-run relationships (alternatively a vector containing the variables names instead of column numbers). The first number must indicate ly (i.e., the lagged dependant variable)
dataset	a panel data set in the form of stacked time series, containing variables of long-run and short-run relationships (i.e., including differentiated and lagged variables)
quantity	a vector of the number of time series observations in each group; in practice, it takes the form c(T1,...,Tn) since the PMG allows the numbers of time series observations to differ freely across groups (if the number of time series observations in each group is the same, then c(T,...,T) and T=T1=T2=...=Tn
const	logical. If TRUE (the default value), the intercept term is added to the model (i.e., to the short-run relationship)

Details

Estimates parameters of long-run and short-run relationships. Also estimates the information matrix as well as standard errors of estimations, as indicated in Equation 13 (Pesaran, Shin and Smith, 1999). Calculates Student's t-distribution type statistics, probs and confidence intervals. Also performs diagnostic tests of error terms, such as the Breusch-Godfrey autocorrelation test, the Goldfeld-Quandt heteroscedasticity test and the Conover nonparametric test of homogeneity of variance and the Jarque-Bera normality test

Value

`$LogL`	the concentrated log-likelihood function
`$dLogL`	the incresase of concentrated log-likelihood function in last iteration
`$i`	the number of iterations performed to achieve convergence
`$LR`	the estimated parameters of long-run relationships
`$SR`	the estimated parameters of short-run relationships
`$DiagTests`	results of diagnostic tests
`$residuals`	residuals

Author(s)

Lech Kujawski, Piotr Zientara

References

Pesaran, Shin and Smith (1999) <doi:10.1080/01621459.1999.10474156>

Examples

```
# first import DataExp, i=1...9, T1=T2=...T9=35
data(DataExp)
DataExp[1:5,]
# then prepare lags and diffs using LagPanel and DiffPanel
y10=data.frame(y10=DataExp[,1], row.names=row.names(DataExp))
cpi=data.frame(cpi=DataExp[,7], row.names=row.names(DataExp))
dy10=DiffPanel(variable=y10, quantity=rep(35,9))
dopeness=DiffPanel(variable=DataExp[,6], quantity=rep(35,9))
ly10=LagPanel(variable=y10, quantity=rep(35,9))
diip=DiffPanel(variable=DataExp[,11], quantity=rep(35,9))
dcrisk=DiffPanel(variable=DataExp[,9], quantity=rep(35,9))
ldcrisk=LagPanel(variable=dcrisk, quantity=rep(35,9))
dcpi=DiffPanel(variable=DataExp[,7], quantity=rep(35,9))
ddcpi=DiffPanel(variable=dcpi, quantity=rep(35,9))
ldebt=LagPanel(variable=DataExp[,4], quantity=rep(35,9))
# create homogenous preliminary dataset (containing "NA" as a result of DiffPanel, LagPanel)
dataPanel=cbind(y10, dy10, ly10, DataExp[,6], dopeness, diip,
DataExp[,11], ldcrisk, DataExp[,9], ddcpi, DataExp[,7])
dataPanel=data.frame(dataPanel)
names(dataPanel)=c("y10", "dy10", "ly10", "openess", "dopeness", "diip",
"iip", "ldcrisk", "crisk", "ddcpi", "cpi")
```

```
dataPanel[1:5,]
# prepare dataset and quantity for PMG or optimPMG functions using PanelNaOmit
dataPanel=PanelNaOmit(dataset=dataPanel, quantity=rep(35,9))
dataPanel$dataset[1:5,]
dataPanel$quantity
# optimPMG
OptimPmgExp=optimPMG(
dLL=10^-10,
maxIter=200,
TetaStart=rep(x=1, times=4), # note that length(TetaStart)=length(vecLR)-1
vecSR=list(SR1=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR2=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR3=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR4=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR5=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR6=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR7=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR8=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR9=c("dy10","dopeness","diip","ldcrisk","ddcpi")),
vecLR=c("ly10","openess","iip","crisk","cpi"),
dataset=dataPanel$dataset,
quantity=dataPanel$quantity,
const=TRUE)
OptimPmgExp
```

PanelNaOmit *PanelNaOmit*

Description

Prepares a panel data set for further calculations by eliminating "NA" and modifying quantity or a vector of the number of time series observations in each group

Usage

```
PanelNaOmit(dataset, quantity)
```

Arguments

dataset	a panel data set in the form of stacked time series, containing variables of long-run and short-run relationships (i.e., including differentiated and lagged variables from DiffPanel or LagPanel)
quantity	a vector of the number of time series observations in each group; in practice, it takes the form c(T1,...Tn) since the PMG allows the numbers of time series observations to differ freely across groups (if the number of time series observations in each group is the same, then c(T,...,T) and T=T1=T2=...=Tn

Details

Eliminates "NA" and modifies quantity or a vector of the number of time series observations in each group

Value

`$dataset`	panel data set for further calculations modified by eliminating "NA"
`$quantity`	modified vector of the number of time series observations in each group

Author(s)

Lech Kujawski, Piotr Zientara

Examples

```
# first import DataExp, i=1...9, T1=T2=...T9=35
data(DataExp)
DataExp[1:5,]
# then prepare lags and diffs using LagPanel and DiffPanel
y10=data.frame(y10=DataExp[,1], row.names=row.names(DataExp))
cpi=data.frame(cpi=DataExp[,7], row.names=row.names(DataExp))
dy10=DiffPanel(variable=y10, quantity=rep(35,9))
dopeness=DiffPanel(variable=DataExp[,6], quantity=rep(35,9))
ly10=LagPanel(variable=y10, quantity=rep(35,9))
diip=DiffPanel(variable=DataExp[,11], quantity=rep(35,9))
dcrisk=DiffPanel(variable=DataExp[,9], quantity=rep(35,9))
ldcrisk=LagPanel(variable=dcrisk, quantity=rep(35,9))
dcpi=DiffPanel(variable=DataExp[,7], quantity=rep(35,9))
ddcpi=DiffPanel(variable=dcpi, quantity=rep(35,9))
ldebt=LagPanel(variable=DataExp[,4], quantity=rep(35,9))
# create homogenous preliminary dataset (containing "NA") after DiffPanel, LagPanel
dataPanel=cbind(y10, dy10, ly10, DataExp[,6], dopeness, diip,
DataExp[,11], ldcrisk, DataExp[,9], ddcpi, DataExp[,7])
dataPanel=data.frame(dataPanel)
names(dataPanel)=c("y10", "dy10", "ly10", "openess", "dopeness", "diip",
"iip", "ldcrisk", "crisk", "ddcpi", "cpi")
dataPanel[1:5,]
# prepare dataset and quantity for PMG or optimPMG functions using PanelNaOmit
dataPanel=PanelNaOmit(dataset=dataPanel, quantity=rep(35,9))
dataPanel$dataset[1:5,]
dataPanel$quantity
```

PMG *PMG*

Description

Having particular long-run parameters (exp. start values) estimates parameters of short-run relationships as well as standard errors of estimations, Student's t-distribution type statistics, probs, confidence intervals. Also performs diagnostic tests of error terms, such as autocorrelation, heteroscedasticity and normality

Usage

```
PMG(paramTeta, vecSR, vecLR, dataset, quantity, const)
```

Arguments

paramTeta	the vector of parameters of long-run relationships, as outlined in Equation 7 (Pesaran, Shin and Smith, 1999)
vecSR	a list of vectors containing the column numbers of variables in short-run relationships for each group (alternatively a list of vectors containing the variables names instead of column numbers). In each vector of the list the first number must indicate dy (i.e., the dependant variable)
vecLR	a vector containing the column numbers of variables in long-run relationships (alternatively a vector containing the variables names instead of column numbers). The first number must indicate ly (i.e., the lagged dependant variable)
dataset	a panel data set in the form of stacked time series, containing variables of long-run and short-run relationships (i.e., including differentiated and lagged variables)
quantity	a vector of the number of time series observations in each group; in practice, it takes the form c(T1,...,Tn) since the PMG allows the numbers of time series observations to differ freely across groups (if the number of time series observations in each group is the same, then c(T,...,T) and T=T1=T2=...=Tn
const	logical. If TRUE (the default value), the intercept term is added to the model (i.e., to the short-run relationship)

Details

Having particular long-run parameters estimates parameters of short-run relationships. Also estimates the information matrix as well as standard errors of estimations, as indicated in Equation 13 (Pesaran, Shin and Smith, 1999). Calculates Student's t-distribution type statistics, probs and confidence intervals. Also performs diagnostic tests of error terms, such as the Breusch-Godfrey autocorrelation test, the Goldfeld-Quandt heteroscedasticity test and the Conover nonparametric test of homogeneity of variance and the Jarque-Bera normality test

Value

$LogL	the concentrated log-likelihood function
$LR	parameters of long-run relationships
$SR	the estimated parameters of short-run relationships
$DiagTests	results of diagnostic tests
$residuals	residuals

Author(s)

Lech Kujawski, Piotr Zientara

References

Pesaran, Shin and Smith (1999) <doi:10.1080/01621459.1999.10474156>

Examples

```
# first import DataExp, i=1...9, T1=T2=...T9=35
data(DataExp)
DataExp[1:5,]
# then prepare lags and diffs using LagPanel and DiffPanel
y10=data.frame(y10=DataExp[,1], row.names=row.names(DataExp))
cpi=data.frame(cpi=DataExp[,7], row.names=row.names(DataExp))
dy10=DiffPanel(variable=y10, quantity=rep(35,9))
dopeness=DiffPanel(variable=DataExp[,6], quantity=rep(35,9))
ly10=LagPanel(variable=y10, quantity=rep(35,9))
diip=DiffPanel(variable=DataExp[,11], quantity=rep(35,9))
dcrisk=DiffPanel(variable=DataExp[,9], quantity=rep(35,9))
ldcrisk=LagPanel(variable=dcrisk, quantity=rep(35,9))
dcpi=DiffPanel(variable=DataExp[,7], quantity=rep(35,9))
ddcpi=DiffPanel(variable=dcpi, quantity=rep(35,9))
ldebt=LagPanel(variable=DataExp[,4], quantity=rep(35,9))
# create homogenous preliminary dataset (containing "NA" as a result of DiffPanel, LagPanel)
dataPanel=cbind(y10, dy10, ly10, DataExp[,6], dopeness, diip,
DataExp[,11], ldcrisk, DataExp[,9], ddcpi, DataExp[,7])
dataPanel=data.frame(dataPanel)
names(dataPanel)=c("y10", "dy10", "ly10", "openess", "dopeness", "diip",
"iip", "ldcrisk", "crisk", "ddcpi", "cpi")
dataPanel[1:5,]
# prepare dataset and quantity for PMG or optimPMG functions using PanelNaOmit
dataPanel=PanelNaOmit(dataset=dataPanel, quantity=rep(35,9))
dataPanel$dataset[1:5,]
dataPanel$quantity
# PMG
PmgExp=PMG(
paramTeta=c(-14.22768, -23.84427, -0.75717, 27.57753),
vecSR=list(SR1=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR2=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR3=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR4=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR5=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR6=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR7=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR8=c("dy10","dopeness","diip","ldcrisk","ddcpi"),
SR9=c("dy10","dopeness","diip","ldcrisk","ddcpi")),
vecLR=c("ly10","openess","iip","crisk","cpi"),
dataset=dataPanel$dataset,
quantity=dataPanel$quantity,
const=TRUE)
PmgExp
```

Index

∗ **package, PMG, pooled mean group, dynamic panel models**
PooledMeanGroup-package, 2
∗ **panel data**
DataExp, 5

BGtest, 3

ConoverMulti, 4

DataExp, 5
DiffPanel, 6

GQtest, 7

JBtest, 8

LagPanel, 9

optimPMG, 10

PanelNaOmit, 12
PMG, 13
PooledMeanGroup *(*PooledMeanGroup-package*)*, 2
PooledMeanGroup-package, 2