Package 'pdynmc'

September 14, 2026

Type Package

Title Moment Condition Based Estimation of Linear Dynamic Panel Data Models

Version 0.9.13

Date 2026-09-14

Depends R (>= 3.6.0)

Imports data.table (>= 1.12.2), MASS (>= 7.3-51.4), Matrix (>= 1.2-17), methods(>= 3.6.2), optimx (>= 2018-07.10), stats (>= 3.6.0), Rdpack (>= 0.11-0)

Suggests pder (>= 1.0-1), testthat (>= 2.3.2), R.rsp (>= 0.43.2)

RdMacros Rdpack

Description Linear dynamic panel data modeling based on linear and nonlinear moment conditions as proposed by Holtz-Eakin, Newey, and Rosen (1988) <doi:10.2307/1913103>, Ahn and Schmidt (1995) <doi:10.1016/0304-4076(94)01641-C>, and Arellano and Bover (1995) <doi:10.1016/0304-4076(94)01642-D>. Estimation of the model parameters relies on the Generalized Method of Moments (GMM) and instrumental variables (IV) estimation, numerical optimization (when nonlinear moment conditions are employed) and the computation of closed form solutions (when estimation is based on linear moment conditions). One-step, two-step and iterated estimation is available. For inference and specification testing, Windmeijer (2005) <doi:10.1016/j.jeconom.2004.02.005> and doubly corrected standard errors (Hwang, Kang, Lee, 2021 <doi:10.1016/j.jeconom.2020.09.010>) are available. Additionally, serial correlation tests, tests for overidentification, and Wald tests are provided. Functions for visualizing panel data structures and modeling results obtained from GMM estimation are also available. The plot methods include functions to plot unbalanced panel structure, coefficient ranges and coefficient paths across GMM iterations (the latter is implemented according to the plot shown in Hansen and Lee, 2021 <doi:10.3982/ECTA16274>).

For a more detailed description of the GMM-based functionality,
please see Fritsch, Pua, Schnurbus (2021) <doi:10.32614/RJ-2021-035>.
For more details on the IV-based estimation routines,
see Fritsch, Pua, and Schnurbus (WP, 2026) and
Han and Phillips (2010) <doi:10.1017/S026646660909063X>.

License GPL (>= 2)

URL https://github.com/markusfritsch/pdynmc

BugReports https://github.com/markusfritsch/pdynmc/issues

VignetteBuilder R.rsp

Encoding UTF-8

Classification/JEL C23, C26, C87

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Markus Fritsch [aut, cre],
Joachim Schnurbus [aut],
Andrew Adrian Yu Pua [aut]

Maintainer Markus Fritsch <Markus.Fritsch@uni-Passau.de>

Repository CRAN

Date/Publication 2026-09-14 21:40:02 UTC

Contents

`ABdata` *Employment, wages, capital, and output for companies based in the UK*

Description

Unbalanced panel dataset of 140 firms from different sectors located in the UK which were observed over the years 1976 until 1984. The dataset contains the variables firm, year, sector, employment, wages, capital, and output and was used in Arellano and Bond (1991) which also provides more details on the different variables.

Usage

```
data(ABdata)
```

Format

A dataset with 1031 rows and 7 variables containing:

firm firm identifier

year year

sector sector

emp number of firm employees in the UK

wage real wage

capital gross capital stock

output industry output

References

Arellano M, Bond S (1991). "Some Tests of Specification for Panel Data: Monte Carlo Evidence and an Application to Employment Equations." *The Review of Economic Studies*, **58**(2), 277–297. doi:10.2307/2297968.

Examples

```
## Not run:
  data(ABdata, package = "pdynmc")
  n <- ABdata$emp
  w <- ABdata$wage
  \donttest{plot(y = n, x = w)}

## End(Not run)
```

AH81 *Estimator of Anderson and Hsiao (1981).*

Description

AH81 computes closed form estimator for lag parameter of linear dynamic panel data model based on Anderson & Hsiao (1981) (AH81) estimator.

Usage

```
AH81(dat, varname.i, varname.t, varname.y, eq8.2 = TRUE)
```

Arguments

dat	A dataset.
varname.i	The name of the cross-section identifier.
varname.t	The name of the time-series identifier.
varname.y	A character string denoting the name of the dependent variable in the dataset.
eq8.2	A logical variable indicating whether the estimation function is based on Equation (8.2) of Anderson and Hsiao (1981); otherwise Equation (8.1) is employed (defaults to 'TRUE').

Details

The function estimates a linear dynamic panel data model of the form

$$y_{i,t} = y_{i,t-1}\rho_1 + a_i + \varepsilon_{i,t}$$

where $y_{i,t-1}$ is the lagged dependent variable, ρ_1 is the lag parameter, a_i is an unobserved individual specific effect, and $\varepsilon_{i,t}$ is an idiosyncratic remainder component. The model structure accounts for unobserved individual specific heterogeneity and dynamics. Note that more general lag structures and further covariates are beyond the scope of the current implementation in pdynmc.

More details on the AH81 estimator and its properties are provided in Anderson and Hsiao (1981) and Anderson and Hsiao (1982).

Value

An object of class 'numeric' that contains the coefficient estimate for the lag parameter according to the two roots of the quadratic equation.

Author(s)

Joachim Schnurbus, Markus Fritsch

References

Anderson TW, Hsiao C (1981). "Estimation of Dynamic Models with Error Components." *Journal of the American Statistical Association*, **76**(375), 598–606. doi:10.1080/01621459.1981.10477691.

Anderson TW, Hsiao C (1982). "Formulation and estimation of dynamic models using panel data." *Journal of Econometrics*, **18**(1), 47–82. doi:10.1016/03044076(82)900951.

Examples

```
## Load data
data(cigDemand, package = "pdynmc")
dat <- cigDemand

## Code example
m1 <- AH81(dat = dat, varname.i = "state", varname.t = "year", varname.y = "packpc")
```

`case.names.pdynmc` *Case and Variable Names of Fitted Model.*

Description

`case.names` extracts variable names of cross-sectional and longitudinal identifiers of an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
case.names(object, ...)
```

Arguments

`object`	An object of class 'pdynmc'.
`...`	further arguments.

Value

A list containing tow character vectors with the variable names of the cross-sectional and the longitudinal identifiers from object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
case.names(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 case.names(m1)
```

---

cigDemand *Cigarette consumption in the US*

Description

Balanced panel dataset on annual cigarette consumption in the US for the 48 continental states in the years 1985 until 1995. The dataset is available from Stock and Watson (2003) and used in Stock and Watson (2019) and Fritsch et al. (2024). Gratitude is owed to Jonathan Gruber of MIT for providing the data.

Usage

```
data(cigDemand)
```

Format

A dataset with 528 rows and 9 variables containing:

state state

year year

cpi consumer price index (US)

pop state population

packpc number of cigarette packs sold per capita and year

income state personal income (total, nominal)

tax average federal, state, and local excise taxes on cigarettes for fiscal year in cents per pack

avgprs average price during fiscal year in cents per pack (including sales taxes)

taxs average excise tax for fiscal year in cents per pack (including sales taxes)

References

Fritsch M, Pua AAY, Schnurbus J (2024). "Teaching advanced topics in econometrics using introductory textbooks: The case of dynamic panel data methods." *International Review of Economics Education*, **47**, 100297. doi:10.1016/j.iree.2024.100297.

Stock JH, Watson MM (2019). *Introduction to Econometrics*, Fourth edition. Pearson.

Stock JH, Watson MW (2003). "cig_ ch10, cig85_95, Instructional Stata datasets for econometrics cig8595." https://ideas.repec.org/p/boc/bocins/cig8595.html.

Examples

```
## Not run:
  data(cigDemand, package = "pdynmc")
  packs <- cigDemand$packpc
  tax <- cigDemand$tax
  \donttest{plot(y = packs, x = tax)}

## End(Not run)
```

coef.pdynmc *Extract Coefficient Estimates of Fitted Model.*

Description

coef.pdynmc extracts coefficient estimates of an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
coef(object, ...)
```

Arguments

object	An object of class 'pdynmc'.
...	further arguments.

Value

Extract coefficient estimates from object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
coef(m1)
```

```
## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 coef(m1)
```

`data.info` *Show Basic Structure of Panel Dataset.*

Description

`data.info` shows basic structure of a balanced/unbalanced panel dataset contained in a 'data.frame'.

Usage

```
data.info(object, i.name = NULL, t.name = NULL, ...)
```

Arguments

`object`	An object of class 'data.frame'.
`i.name`	Column name of cross-section identifier.
`t.name`	Column name of time-series identifier.
`...`	further arguments.

Value

Returns information if panel dataset contained in an object of class 'data.frame' is a balanced or unbalanced panel dataset.

Author(s)

Markus Fritsch, Joachim Schnurbus

See Also

`pdynmc` for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
data.info(dat, i.name = "firm", t.name = "year")

data.info(dat[dat$year %in% 1979:1981, ], i.name = "firm", t.name = "year")
```

`dummy.coef.pdynmc` *Extract Coefficient Estimates of Time Dummies of Fitted Model.*

Description

`dummy.coef.pdynmc` extracts coefficient estimates of time dummies of an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
dummy.coef(object, ...)
```

Arguments

`object`	An object of class 'pdynmc'.
`...`	further arguments.

Value

Extract coefficient estimates of time dummies from object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

`pdynmc` for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
dummy.coef(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 dummy.coef(m1)
```

FDLS *First Difference Least Squares (FDLS) Estimator of Han and Phillips (2010).*

Description

FDLS computes closed form estimator for lag parameter of linear dynamic panel data model based on first difference least squares (FDLS) estimator.

Usage

```
FDLS(dat, varname.i, varname.t, varname.y)
```

Arguments

dat	A dataset.
varname.i	The name of the cross-section identifier.
varname.t	The name of the time-series identifier.
varname.y	A character string denoting the name of the dependent variable in the dataset.

Details

The function estimates a linear dynamic panel data model of the form

$$y_{i,t} = y_{i,t-1}\rho_1 + a_i + \varepsilon_{i,t}$$

where $y_{i,t-1}$ is the lagged dependent variable, ρ_1 is the lag parameter, a_i is an unobserved individual specific effect, and $\varepsilon_{i,t}$ is an idiosyncratic remainder component. The model structure accounts for unobserved individual specific heterogeneity and dynamics. Note that more general lag structures and further covariates are beyond the scope of the current implementation in pdynmc.

More details on the FDLS estimator and its properties are provided in Han and Phillips (2010).

Value

An object of class 'numeric' that contains the coefficient estimate for the lag parameter according to the two roots of the quadratic equation.

Author(s)

Joachim Schnurbus, Markus Fritsch

References

Han C, Phillips PCB (2010). "GMM Estimation For Dynamic Panels With Fixed Effects And Strong Instruments At Unity." *Econometric Theory*, **26**(1), 119–151. doi:10.1017/S026646660909063X.

Examples

```
## Load data
data(cigDemand, package = "pdynmc")
dat <- cigDemand

## Code example
m1 <- FDLS(dat = dat, varname.i = "state", varname.t = "year", varname.y = "packpc")
```

`fitted.pdynmc` *Extract Fitted Values of Fitted Model.*

Description

`fitted.pdynmc` extracts fitted values of an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
fitted(object, step = object$iter, na.rm = FALSE, ...)
```

Arguments

`object`	An object of class 'pdynmc'.
`step`	An integer denoting the iteration step for which fitted values are extracted (defaults to last iteration step used for obtaining parameter estimates).
`na.rm`	A logical variable indicating whether missing values should be removed from the vector of fitted values (defaults to 'FALSE').
`...`	further arguments.

Value

Extract fitted values from object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

`pdynmc` for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
```

```
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
fitted(m1, na.rm = TRUE)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 fitted(m1, na.rm = TRUE)
```

`jtest.fct` *Hansen J-Test.*

Description

`jtest.fct` tests the validity of the overidentifying restrictions.

Usage

```
jtest.fct(object)
```

Arguments

`object`	An object of class 'pdynmc'.

Details

The null hypothesis is that the overidentifying restrictions are valid. The test statistic is computed as proposed by Hansen (1982). As noted by Bowsher (2002) and Windmeijer (2005) the test statistic is weakened by many instruments.

Value

An object of class 'htest' which contains the Hansen J-test statistic and corresponding p-value for the null hypothesis that the overidentifying restrictions are valid.

References

Bowsher CG (2002). "On testing overidentifying restrictions in dynamic panel data models." *Economics Letters*, **77**(2), 211–220. doi:10.1016/S01651765(02)001301.

Hansen LP (1982). "Large Sample Properties of Generalized Method of Moments Estimators." *Econometrica*, **50**(4), 1029–1054. doi:10.2307/1912775.

Windmeijer F (2005). "A finite sample correction for the variance of linear efficient two-step GMM estimators." *Journal of Econometrics*, **126**(1), 25–51. doi:10.1016/j.jeconom.2004.02.005.

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(140:0), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
jtest.fct(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 jtest.fct(m1)
```

model.matrix.pdynmc *Extract Instrument Matrix of Fitted Model.*

Description

model.matrix.pdynmc extracts instrument matrix of an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
model.matrix(object, sparse = TRUE, ...)
```

Arguments

object	An object of class 'pdynmc'.
sparse	Whether to return a sparse matrix (if set to 'TRUE') or a regular matrix (if set to 'FALSE').
...	further arguments.

Value

Extracts instrument matrix from an object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
```

```
    opt.meth = "none")
model.matrix(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 model.matrix(m1)
```

`mtest.fct` *Arellano and Bond Serial Correlation Test.*

Description

`mtest.pdynmc` Methods to test for serial correlation in the error terms for objects of class 'pdynmc'.

Usage

```
mtest.fct(object, order = 2, ...)
```

Arguments

`object`	An object of class 'pdynmc'.
`order`	A number denoting the order of serial correlation to test for (defaults to '2').
`...`	further arguments.

Details

The null hypothesis is that there is no serial correlation of a particular order. The test statistic is computed as proposed by Arellano and Bond (1991) and Arellano (2003).

Value

An object of class 'htest' which contains the Arellano and Bond m test statistic and corresponding p-value for the null hypothesis that there is no serial correlation of the given order.

References

Arellano M (2003). *Panel Data Econometrics*. Oxford University Press. doi:10.1093/0199245282.001.0001.

Arellano M, Bond S (1991). "Some Tests of Specification for Panel Data: Monte Carlo Evidence and an Application to Employment Equations." *The Review of Economic Studies*, **58**(2), 277–297. doi:10.2307/2297968.

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(140:0), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
mtest.fct(m1, order = 2)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 mtest.fct(m1, order = 2)
```

ninst *Extract Instrument Count of Fitted Model.*

Description

ninst is a generic function fo extracting the instrument count of an object.

Usage

```
ninst(object, ...)
```

Arguments

object	An object for which the instrument count is desired.
...	further arguments.

Value

Extracts instrument count from an object.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
ninst(m1)


## Load data
```

```
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])

m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
   use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
   include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
   fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
   varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
   include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
   w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
   opt.meth = "none")
ninst(m1)
```

`ninst.pdynmc` *Extract Instrument Count of Fitted Model.*

Description

`ninst.pdynmc` extracts instrument count of an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
ninst(object, ...)
```

Arguments

`object`	An object of class 'pdynmc'.
`...`	further arguments.

Value

Extracts instrument count from an object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

`pdynmc` for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
ninst(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 ninst(m1)
```

NLIV *Nonlinear Instrumental Variables Estimator - T-Version (NLIV).*

Description

NLIV Computes closed form solution for lag parameter of linear dynamic panel data model based on instrumental variables (IV) estimator employing nonlinear moment conditions.

Usage

```
NLIV(dat, varname.i, varname.t, varname.y, trueAR = NULL)
```

Arguments

`dat`	A dataset.
`varname.i`	The name of the cross-section identifier.
`varname.t`	The name of the time-series identifier.
`varname.y`	A character string denoting the name of the dependent variable in the dataset.
`trueAR`	A logical variable indicating whether the true autoregressive parameter is known (defaults to 'NULL'). The parameter is used to compute the terms 'A', 'B', and 'C' that can be employed to rewrite the estimating equation. For details, see Fritsch et al. (2026).

Details

The function estimates a linear dynamic panel data model of the form

$$y_{i,t} = y_{i,t-1}\rho_1 + a_i + \varepsilon_{i,t}$$

where $y_{i,t-1}$ is the lagged dependent variable, ρ_1 is the lag parameter, a_i is an unobserved individual specific effect, and $\varepsilon_{i,t}$ is an idiosyncratic remainder component. The model structure accounts for unobserved individual specific heterogeneity and dynamics. Note that more general lag structures and further covariates are beyond the scope of the current implementation in `pdynmc`.

The nonlinear IV estimator employs the original version of the nonlinear moment conditions of Ahn and Schmidt (1995). More details on the implementation, the properties of the estimator, and how to account for predetermined covariates are provided in Fritsch et al. (2026).

Value

An object of class 'numeric' that contains the coefficient estimate for the lag parameter according to the two roots of the quadratic equation.

Author(s)

Joachim Schnurbus, Markus Fritsch

References

Ahn SC, Schmidt P (1995). "Efficient estimation of models for dynamic panel data." *Journal of Econometrics*, **68**(1), 5–27. doi:10.1016/03044076(94)01641C.

Fritsch M, Pua AAY, Schnurbus J (2026). "Properties of an estimator for linear dynamic panel data models based on nonlinear moment conditions." Working Paper.

Examples

```
## Load data
data(cigDemand, package = "pdynmc")
dat <- cigDemand

## Code example
```

```
m1 <- NLIV(dat = dat, varname.i = "state", varname.t = "year", varname.y = "packpc")
```

NLIV_t *Nonlinear Instrumental Variables Estimator - t-Version (NLIV_t).*

Description

NLIV_t Computes closed form solution for lag parameter of linear dynamic panel data model based on instrumental variables (IV) estimator employing alternative formulation of nonlinear moment conditions.

Usage

```
NLIV_t(dat, varname.i, varname.t, varname.y, trueAR = NULL)
```

Arguments

dat	A dataset.
varname.i	The name of the cross-section identifier.
varname.t	The name of the time-series identifier.
varname.y	A character string denoting the name of the dependent variable in the dataset.
trueAR	A logical variable indicating whether the true autoregressive parameter is known (defaults to 'NULL'). The parameter is used to compute the terms 'A', 'B', and 'C' that can be employed to rewrite the estimating equation. For details, see Fritsch et al. (2026).

Details

The function estimates a linear dynamic panel data model of the form

$$y_{i,t} = y_{i,t-1}\rho_1 + a_i + \varepsilon_{i,t}$$

where $y_{i,t-1}$ is the lagged dependent variable, ρ_1 is the lag parameter, a_i is an unobserved individual specific effect, and $\varepsilon_{i,t}$ is an idiosyncratic remainder component. The model structure accounts for unobserved individual specific heterogeneity and dynamics. Note that more general lag structures and further covariates are beyond the scope of the current implementation in pdynmc.

The nonlinear IV estimator employs an alternative formulation of the nonlinear moment conditions of Ahn and Schmidt (1995). More details on the implementation, the properties of the estimator, and how to account for predetermined covariates are provided in Fritsch et al. (2026).

Value

An object of class 'numeric' that contains the coefficient estimate for the lag parameter according to the two roots of the quadratic equation.

Author(s)

Joachim Schnurbus, Markus Fritsch

References

Ahn SC, Schmidt P (1995). "Efficient estimation of models for dynamic panel data." *Journal of Econometrics*, **68**(1), 5–27. doi:10.1016/03044076(94)01641C.

Fritsch M, Pua AAY, Schnurbus J (2026). "Properties of an estimator for linear dynamic panel data models based on nonlinear moment conditions." Working Paper.

Examples

```
## Load data
data(cigDemand, package = "pdynmc")
dat <- cigDemand

## Code example
m1 <- NLIV_t(dat = dat, varname.i = "state", varname.t = "year", varname.y = "packpc")
```

`nobs.pdynmc` *Extract Number of Observations of Fitted Model.*

Description

`nobs.pdynmc` extracts number of observations in cross-section dimension and longitudinal dimension of an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
nobs(object, ...)
```

Arguments

`object`	An object of class 'pdynmc'.
`...`	further arguments.

Value

Extracts number of observations in cross-section dimension and longitudinal dimension of an object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
nobs(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 nobs(m1)
```

---

optimIn *Extract Input Parameters of Numeric Optimization of Fitted Model.*

Description

optimIn is a generic function for extracting input parameters of numeric optimization for an object.

Usage

```
optimIn(object, ...)
```

Arguments

object	An object for which input parameters of numeric optimization are desired.
...	further arguments.

Value

optimIn extracts input parameters used in numeric optimization from object.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
optimIn(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
```

```
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "BFGS")
optimIn(m1)
```

---

optimIn.pdynmc *Extract Input Parameters of Numeric Optimization of Fitted Model.*

Description

optimIn.pdynmc extracts input parameters of numeric optimization for an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
optimIn(object, step = object$iter, ...)
```

Arguments

object	An object of class 'pdynmc'.
step	An integer denoting the iteration step for which input parameters are extracted (defaults to last iteration step used for obtaining parameter estimates).
...	further arguments.

Value

Extracts input parameters of numeric optimization from object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
```

```
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
optimIn(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "BFGS")
 optimIn(m1)
```

`package-pdynmc` *pdynmc: A package for moment conditions based estimation of linear dynamic panel data models*

Description

The pdynmc package provides four categories of functions that are available to the user: A function for model fitting, functions for visualizing estimation results and panel data structures, functions for specification testing, and functions that extract and summarize particular information from fitted model objects.

`pDensTime.plot` *Plot Empirical Density of a Column of a Panel Dataset over Time.*

Description

`pDensTime.plot` Plot the empirical density of a column of an object of class ‘data.frame‘ containing a panel dataset across time periods/aggregates of time periods.

Usage

```
pDensTime.plot(
  object,
  var.name,
  i.name,
  t.name,
  aggregate.t = NULL,
  plot.quantiles = TRUE,
  plot.mean_ci = TRUE,
  plot.extrema = TRUE,
  col.set = c("gray", "navy", "darkorange1", "red"),
  ...
)
```

Arguments

object	An object of class 'data.frame'.
var.name	Column name of the variable that is plotted (see 'Details').
i.name	Column name of cross-section identifier.
t.name	Column name of time-series identifier.
aggregate.t	Argument of data type 'numeric'. If argument is specified, the corresponding number of time periods is merged (approximately); (defaults to 'NULL').
plot.quantiles	Argument of data type 'logical', indicating whether the 5%- and 95%-quantiles and the quartiles should be plotted (as specified by 'col.set[2]'; defaults to 'TRUE').
plot.mean_ci	Argument of data type 'logical', indicating whether the mean and the approximate confidence intervals (mean plus/minus 2 standard deviations) should be plotted (as specified by 'col.set[3]'; defaults to 'TRUE').
plot.extrema	Argument of data type 'logical', indicating whether the minimal and maximal observed value (per time period/group) should be plotted (as specified by 'col.set[4]'; defaults to 'TRUE').
col.set	Vector of length 4 with entries of data type 'character' used to visualize the entities of 'pDensTime.plot' (see 'Details'); must be a valid argument to 'col2rgb'; defaults to 'c("gray", "navy", "darkorange1", "red")'.
...	further arguments.

Value

Returns a plot that visualizes the empirical density for a column of a panel dataset contained in an object of class 'data.frame'. The variable of interest is plotted on the ordinate, the longitudinal dimension on the abscissa. For each time period or aggregate of time periods, one empirical density is computed and plotted. Corresponding summary statistics on empirical quantiles and the sample size per longitudinal dimension are included in the plot.

Author(s)

Markus Fritsch, Joachim Schnurbus

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])

## Minimal set of arguments
pDensTime.plot(object = ABdata, var.name = "emp", i.name = "firm", t.name = "year")

## All arguments explicitly stated
pDensTime.plot(object = ABdata, var.name = "emp", i.name = "firm", t.name = "year",
 aggregate.t = NULL, plot.quantiles = TRUE, plot.mean_ci = TRUE, plot.extrema = TRUE,
 col.set  = c("gray", "navy", "darkorange1", "red"))

## Aggregation over time periods (3 time periods per group)
pDensTime.plot(object = ABdata, var.name = "emp", i.name = "firm", t.name = "year",
 aggregate.t = 3)

## Employ alternative colouring scheme
pDensTime.plot(object = ABdata, var.name = "emp", i.name = "firm", t.name = "year",
 col.set  = c("pink", "blue", "purple", "black"))

## Plot only density, mean, and asymptotic confidence interval
pDensTime.plot(object = ABdata, var.name = "emp", i.name = "firm", t.name = "year",
 plot.quantiles = FALSE, plot.extrema = FALSE)
```

---

pdynmc *Generalized Method of Moments (GMM) Estimation of Linear Dynamic Panel Data Models.*

Description

pdynmc fits a linear dynamic panel data model based on moment conditions with the Generalized Method of Moments (GMM).

Usage

```
pdynmc(
  dat = NULL,
  varname.i = NULL,
  varname.t = NULL,
  use.mc.diff = NULL,
  use.mc.lev = NULL,
  use.mc.nonlin = NULL,
```

```
use.mc.nonlinAS = NULL,
inst.collapse = FALSE,
inst.stata = FALSE,
include.y,
varname.y = NULL,
lagTerms.y = NULL,
maxLags.y = NULL,
include.x = FALSE,
varname.reg.end = NULL,
lagTerms.reg.end = NULL,
maxLags.reg.end = NULL,
varname.reg.pre = NULL,
lagTerms.reg.pre = NULL,
maxLags.reg.pre = NULL,
varname.reg.ex = NULL,
lagTerms.reg.ex = NULL,
maxLags.reg.ex = NULL,
inst.reg.ex.expand = TRUE,
include.x.instr = FALSE,
varname.reg.instr = NULL,
include.x.toInstr = FALSE,
varname.reg.toInstr = NULL,
fur.con = FALSE,
fur.con.diff = NULL,
fur.con.lev = NULL,
varname.reg.fur = NULL,
lagTerms.reg.fur = NULL,
include.dum = FALSE,
dum.diff = NULL,
dum.lev = NULL,
varname.dum = NULL,
col_tol = 0.65,
w.mat = "iid.err",
w.mat.stata = FALSE,
std.err = "corrected",
estimation = "iterative",
max.iter = 100,
iter.tol = 0.01,
inst.thresh = NULL,
opt.meth = "BFGS",
hessian = FALSE,
optCtrl = list(kkt = FALSE, kkttol = .Machine$double.eps^(1/3), kkt2tol =
  .Machine$double.eps^(1/3), starttests = TRUE, dowarn = TRUE, badval = (0.25) *
 .Machine$double.xmax, usenumDeriv = FALSE, reltol = 1e-12, maxit = 200, trace = TRUE,
 follow.on = FALSE, save.failures = TRUE, maximize = FALSE, factr = 1e+07, pgtol = 0,
   all.methods = FALSE),
custom.start.val = FALSE,
start.val = NULL,
```

```
  start.val.lo = -1,
  start.val.up = 1,
  seed.input = 42
)
```

Arguments

dat	A dataset.
varname.i	The name of the cross-section identifier.
varname.t	The name of the time-series identifier.
use.mc.diff	A logical variable indicating whether moment conditions from equations in differences (i.e. instruments in levels) should be used.
use.mc.lev	A logical variable indicating whether moment conditions from equations in levels (i.e. instruments in differences) should be used.
use.mc.nonlin	A logical variable indicating whether nonlinear (quadratic) moment conditions should be used.
use.mc.nonlinAS	A logical variable indicating whether only the nonlinear (quadratic) moment conditions in the form proposed by Ahn and Schmidt (1995) should be used (is set to 'TRUE' when nonlinear moment conditions are employed).
inst.collapse	A logical variable indicating whether to collapse the set of moment conditions as proposed by (Roodman 2009) (defaults to 'FALSE').
inst.stata	A logical variable indicating whether to use the moment conditions from equations in levels as in Stata implementations xtabond2 Roodman (2018) and xtdpdgmm Kripfganz (2019).
include.y	A logical variable indicating whether instruments should be derived from the lags of the dependent variable.
varname.y	A character string denoting the name of the dependent variable in the dataset.
lagTerms.y	An integer indicating the number of lags of the dependent variable. Note that setting 'lagTerms.y' to zero excludes the dependent variable from the right-hand-side of the model specification.
maxLags.y	An integer indicating the maximum number of lags of the dependent variable from which instruments should be derived.
include.x	A logical variable indicating whether instruments should be derived from the covariates. Setting the argument to 'TRUE' requires specifying whether the covariates are endogenous, predetermined, or (strictly) exogenous (defaults to 'FALSE').
varname.reg.end	One or more character strings denoting the covariate(s) in the dataset to be treated as endogenous (defaults to 'NULL').
lagTerms.reg.end	One or more integers indicating the number of lags of the endogenous covariate(s). One integer per covariate needs to be given in the same order as the covariate names (defaults to 'NULL').

maxLags.reg.end
: One or more integers indicating the maximum number of lags of the endogenous covariate(s) used for deriving instruments.

varname.reg.pre
: One or more character strings denoting the covariate(s) in the dataset to be treated as predetermined (defaults to 'NULL').

lagTerms.reg.pre
: One or more integers indicating the number of lags of the predetermined covariate(s).

maxLags.reg.pre
: One or more integers indicating the maximum number of lags of the predetermined covariate(s) used for deriving instruments. One integer per covariate needs to be given in the same order as the covariate names (defaults to 'NULL').

varname.reg.ex
: One or more character strings denoting the covariate(s) in the dataset to be treated as (strictly) exogenous (defaults to 'NULL').

lagTerms.reg.ex
: One or more integers indicating the number of lags of the (strictly) exogenous covariate(s). One integer per covariate needs to be given in the same order as the covariate name (defaults to 'NULL').

maxLags.reg.ex
: One or more integers indicating the maximum number of lags of the (strictly) exogenous covariate(s) used for deriving instruments.

inst.reg.ex.expand
: A logical variable that allows for using all past, present, and future observations of 'varname.reg.ex' to derive instruments (defaults to 'TRUE'). If set to 'FALSE', only past and present time periods are used to derive instruments.

include.x.instr
: A logical variable that allows to include additional IV-type instruments (i.e., include covariates which are used as instruments but for which no parameters are estimated; defaults to 'FALSE').

varname.reg.instr
: One or more character strings denoting the covariate(s) in the dataset treated as instruments in estimation (defaults to 'NULL'). Note that the instrument type needs to be specified by including the names of the covariate(s) in any of the arguments 'varname.reg.end', 'varname.reg.pre', or 'varname.reg.ex'.

include.x.toInstr
: A logical variable that allows to instrument covariate(s) (i.e., covariates which are not used as instruments but for which parameters are estimated; defaults to 'FALSE').

varname.reg.toInstr
: One or more character strings denoting the covariate(s) in the dataset to be instrumented (defaults to 'NULL'). Note that the names of the covariate(s) should not be included in any other function argument.

fur.con
: A logical variable indicating whether further control variables (covariates) are included (defaults to 'FALSE').

fur.con.diff
: A logical variable indicating whether to include further control variables in equations from differences (defaults to 'NULL').

`fur.con.lev` A logical variable indicating whether to include further control variables in equations from level (defaults to 'NULL').

`varname.reg.fur` One or more character strings denoting covariate(s) in the dataset to treat as further controls (defaults to 'NULL').

`lagTerms.reg.fur` One or more integers indicating the number of lags of the further controls. One integer per further control needs to be given in the same order as the corresponding variable names (defaults to 'NULL').

`include.dum` A logical variable indicating whether dummy variables for the time periods are included (defaults to 'FALSE').

`dum.diff` A logical variable indicating whether dummy variables are included in the equations in first differences (defaults to 'NULL').

`dum.lev` A logical variable indicating whether dummy variables are included in the equations in levels (defaults to 'NULL').

`varname.dum` One or more character strings from which time dummies should be derived (can be different from varname.t; defaults to 'NULL').

`col_tol` A numeric variable in [0,1] indicating the absolute correlation threshold for collinearity checks (columns are omitted when pairwise correlations are above the threshold; defaults to 0.65).

`w.mat` One of the characters '"iid.err"', '"identity"', '"zero.cov"' indicating the type of weighting matrix to use (defaults to '"iid.err"').

`w.mat.stata` A logical variable that slightly adjusts the weighting matrix according to the Stata function xtdpdgmm (defaults to 'FALSE').

`std.err` One of the character '"corrected"', '"unadjusted"', '"dbl.corrected"'. The second and third options compute corrected standard error according to Windmeijer (2005) and Hwang et al. (2021), respectively (defaults to '"corrected"').

`estimation` One of the characters '"onestep"', '"twostep"', '"iterative"'. Denotes the number of iterations of the parameter procedure (defaults to '"twostep"').

`max.iter` An integer indicating the maximum number of iterations (defaults to 'NULL'; if estimation is set to '"iterative"', 'max.iter' defaults to 100).

`iter.tol` A numeric variable in [0,1] indicating the tolerance for determining convergence of the iterative approach (defaults to 'NULL'; if estimation is set to '"iterative"', iter.tol defaults to 0.01).

`inst.thresh` An integer denoting above which instrument count a generalized inverse is used to invert the weighting matrix (defaults to 'NULL').

`opt.meth` A character string denoting the numerical optimization procedure. When no nonlinear moment conditions are employed in estimation, closed form estimates can be computed by setting the argument to '"none"' (defaults to '"BFGS"'; for details on the further available optimizers see the documentation of package **optimx**).

`hessian` A logical variable indicating if the hessian matrix should be approximated in optimization (defaults to 'FALSE').

`optCtrl` A list of arguments that are passed to **optimx**. For details on the arguments and the available options see the package documentation.

`custom.start.val` A logical variable indicating whether prespecified starting values for the parameters are provided by the user (defaults to 'FALSE'; if set to 'TRUE', starting values need to be provided via argument 'start.val').

`start.val` A vector of numeric variables denoting the starting values for the parameter vector for numeric optimization (defaults to 'NULL').

`start.val.lo` A numeric variable denoting the lower limit for drawing starting values with uniform density (defaults to -1; ignored if 'custom.start.val' is set to 'TRUE').

`start.val.up` A numeric variable denoting the lower limit for drawing starting values with uniform density (defaults to 1; ignored if 'custom.start.val' is set to 'TRUE').

`seed.input` An integer used as seed for drawing starting values (defaults to 42; required if custom.start.val is set to 'FALSE').

Details

The function estimates a linear dynamic panel data model of the form

$$y_{i,t} = y_{i,t-1}\rho_1 + \boldsymbol{x}'_{i,t}\boldsymbol{\beta} + a_i + \varepsilon_{i,t}$$

where $y_{i,t-1}$ is the lagged dependent variable, ρ_1 is the lag parameter, $\boldsymbol{x}_{i,t}$ are further covariates, $\boldsymbol{\beta}$ are the corresponding parameters, a_i is an unobserved individual specific effect, and $\varepsilon_{i,t}$ is an idiosyncratic remainder component. The model structure accounts for unobserved individual specific heterogeneity and dynamics. Note that the specification given above is simplified for illustratory purposes and more general lag structures are allowed in `pdynmc`.

Estimation of the model parameters in `pdynmc` is based on moment conditions with the generalized method of moments (GMM). The moment conditions employed in estimation can be linear and nonlinear in parameters and estimation is carried out iteratively. In case only linear moment conditions are used in estimation, closed form solutions can be for computing parameter estimates – while when nonlinear moment conditions are employed, parameter estimation relies on numerical optimization of the objective function.

'pdynmc' provides an implementation of some of the functionality available in the Stata library xtdpdgmm Kripfganz (2019) and allows for '"onestep"', '"twostep"', and '"iterative"' GMM estimation based on the moment conditions of Holtz-Eakin et al. (1988), Arellano and Bover (1995), and Ahn and Schmidt (1995). Standard error corrections according to Windmeijer (2005) and Hwang et al. (2021) are available. For further details, please see Fritsch et al. (2021).

Value

An object of class 'pdynmc' with the following elements:

`coefficients` a vector containing the coefficient estimates

`data` a list of elements on which computation of the model fit is based

`dep.clF` a list of vectors containing the dependent variable for the cross-sectional observations

`dat.clF` a list of matrices containing the covariates for the cross-sectional observations

`w.mat`	a list of weighting matrices for the different estimation steps
`H_i`	a matrix used to create the weighting matrix for the first estimation step
`par.optim`	a list of vectors containing the parameter estimates obtained from numerical optimization for the estimation steps
`ctrl.optim`	a list of control parameters used in numerical optimization for the estimation steps
`par.clForm`	a list of vectors containing the parameter estimates obtained from the closed form for the estimation steps
`iter`	a scalar denoting the number of iteration steps carried out to obtain parameter estimates
`fitted.values`	a list for each estimation step that contains a list of vectors of fitted values for each cross-sectional observation
`residuals`	a list for each estimation step that contains a list of vectors of residuals for each cross-sectional observation
`vcov`	a list of matrices containing the variance covariance matrix of the parameter estimates for each estimation step
`stderr`	a list of vectors containing the standard errors of the parameter estimates for each estimation step
`zvalue`	a list of vectors containing the z scores for the parameter estimates for each estimation step
`pvalue`	a list of vectors containing the p-values for the parameter estimates for each estimation step

It has 'case.names', 'coef', 'dum.coef', 'fitted', 'model.matrix', 'ninst', 'nobs', 'optimIn', 'plot', 'print','residuals', 'summary', 'variable.names', 'vcov', and 'wmat' methods.

Author(s)

Markus Fritsch

References

Ahn SC, Schmidt P (1995). "Efficient estimation of models for dynamic panel data." *Journal of Econometrics*, **68**(1), 5–27. doi:10.1016/03044076(94)01641C.

Arellano M, Bover O (1995). "Another look at the instrumental variable estimation of error-components models." *Journal of Econometrics*, **68**(1), 29–51. doi:10.1016/03044076(94)01642D.

Fritsch M, Pua AAY, Schnurbus J (2021). "pdynmc: A Package for Estimating Linear Dynamic Panel Data Models Based on Nonlinear Moment Conditions." *The R Journal*, **13**(1), 218–231. doi:10.32614/RJ2021035.

Holtz-Eakin D, Newey WK, Rosen HS (1988). "Estimating Vector Autoregressions with Panel Data." *Econometrica*, **56**(6), 1371–1395. doi:10.2307/1913103.

Hwang J, Kang B, Lee S (2021). "A doubly corrected robust variance estimator for linear GMM."

Journal of Econometrics, **forthcoming**. doi:10.1016/j.jeconom.2020.09.010.

Kripfganz S (2019). "XTDPDGMM: Stata module to perform generalized method of moments estimation of linear dynamic panel data models." https://econpapers.repec.org/RePEc:boc:bocode:s458395.

Roodman D (2009). "A Note on the Theme of Too Many Instruments." *Oxford Bulletin of Economics and Statistics*, **71**(1), 135–158. doi:10.1111/j.14680084.2008.00542.x.

Roodman D (2018). "xtabond2: Stata module to extend xtabond dynamic panel data estimator." https://econpapers.repec.org/software/bocbocode/s435901.htm.

Windmeijer F (2005). "A finite sample correction for the variance of linear efficient two-step GMM estimators." *Journal of Econometrics*, **126**(1), 25–51. doi:10.1016/j.jeconom.2004.02.005.

See Also

wald.fct for Wald tests, jtest.fct for the Hansen J test, and mtest.fct for serial correlation tests. optimx for details on alternative routines and options for numerical optimization

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
         use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
         include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
         fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
         varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
         include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
         w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
         opt.meth = "none")
summary(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Arellano and Bond (1991) estimation in Table 4, column (a1)
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
         use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
         include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
         fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
         varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
```

```
        include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
        w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
        opt.meth = "none")
 summary(m1)

## Arellano and Bond (1991) estimation in Table 4, column (a2)
 m2 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
        use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
        include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
        fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
        varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
        include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
        w.mat = "iid.err", std.err = "corrected", estimation = "twostep",
        opt.meth = "none")
 summary(m2)

## Arellano and Bond (1991) twostep estimation extended by nonlinear moment
## conditions
 m3 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
        use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = TRUE,
        include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
        fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
        varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
        include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
        w.mat = "iid.err", std.err = "corrected", estimation = "twostep",
        opt.meth = "BFGS")
 summary(m3)

## Arellano and Bond (1991) iterative estimation extended by nonlinear moment
## conditions
 m4 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
        use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = TRUE,
        include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
        fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
        varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
        include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
        w.mat = "iid.err", std.err = "corrected", estimation = "iterative",
        max.iter = 4, opt.meth = "BFGS")
 summary(m4)

## Arellano and Bond (1991) twostep estimation extended by linear moment
## conditions from equations in levels
 m5 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
        use.mc.diff = TRUE, use.mc.lev = TRUE, use.mc.nonlin = FALSE,
        include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
        fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
        varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
        include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
        w.mat = "iid.err", std.err = "corrected", estimation = "twostep",
        opt.meth = "none")
 summary(m5)
```

plot.pdynmc *Plot Coefficient Estimates and Corresponding Ranges of Fitted Model.*

Description

plot.pdynmc Plot methods for objects of class 'pdynmc'. The available plot options visualize: Fitted values versus residuals, coefficient ranges across GMM iterations, coefficient paths and objective function values across GMM iterations as proposed by Hansen and Lee (2021).

Usage

```
## S3 method for class 'pdynmc'
plot(
  x,
  type = "fire",
  include.dum = FALSE,
  include.fur.con = FALSE,
  col.coefRange = 1,
  col.coefInitial = "darkgrey",
  col.coefEst = "royalblue",
  omit1step = FALSE,
  boxplot.coef = FALSE,
  co = NULL,
  add.se.approx = NULL,
  conf.lev = 0.95,
  ...
)
```

Arguments

x	An object of class 'pdynmc'. The function requires twostep or iterative GMM estimates.
type	Whether to plot fitted values against residuals (argument 'fire'; default), first differenced fitted values against first differenced residuals (argument 'fire.fd'), coefficient ranges (argument 'coef.range'; this requires twostep or iterative GMM estimates), path of coefficient estimates across GMM iterations (argument 'coef.path'; this requires twostep or iterative GMM estimates).
include.dum	Include estimates of parameters corresponding to time dummies (defaults to 'black'; requires 'type = coef.range').
include.fur.con	Include estimates of parameters corresponding to further controls (defaults to 'FALSE'; requires 'type = coef.range').
col.coefRange	Specify color for plotting range of coefficient estimates (defaults to 'NULL'; requires 'type = coef.range').

`col.coefInitial`
: Specify color for plotting initial coefficient estimates (defaults to 'darkgrey'; requires 'type = coef.range').

`col.coefEst`
: Specify color for plotting coefficient estimate (defaults to 'royalblue'; requires 'type = coef.range').

`omit1step`
: Omit coefficient estimates from one-step GMM estimation in coefficient range plot. The argument can be employed to exclude the randomly drawn starting values from the plotted coefficient range when coefficient estimates were obtained from numerical optimization methods (defaults to 'FALSE'). Set to 'TRUE' to exert the option; this argument requires iterative GMM estimates and argument 'type = coef.range'.

`boxplot.coef`
: Whether to draw boxplots for coefficient estimates (defaults to 'FALSE'); requires iterative GMM with at least 10 iterations and argument 'type = coef.range'. Proceed with caution as this argument is experimental.

`co`
: Character string denoting the variable name(s) for which to plot the path of coefficient estimate(s) across GMM iterations (defaults to 'NULL') as proposed in Hansen and Lee (2021); if no coefficient name is given, all coefficient paths are plotted; requires at least two iterations and argument 'type = coef.path'.

`add.se.approx`
: A logical variable indicating if standard errors should be added to the plot of the path of coefficient estimate(s) across GMM iterations (defaults to 'NULL'); requires at least two iterations and argument 'type = coef.path'. This option is only available when plotting a single coefficient path (i.e., when 'co' contains only a single variable name).

`conf.lev`
: A numeric variable indicating the confidence level for approximating standard errors in the plot of the path of coefficient estimate(s) across GMM iterations (defaults to 0.95; sensible values lie in the interval]0,1[); requires argument 'type = coef.path' and argument 'add.se.approx = TRUE'.

`...`
: further arguments.

Value

Plot fitted values against residuals ('type = fire') or coefficient estimates and coefficient estimate ranges ('type = coef.range') for object of class 'pdynmc'. The latter plot requires twostep or iterative GMM estimates.

Author(s)

Markus Fritsch and Joachim Schnurbus

References

Hansen BE, Lee S (2021). "Inference for Iterated GMM Under Misspecification." *Econometrica*, **89**(3), 1419–1447. doi:10.3982/ECTA16274.

See Also

`pdynmc` for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "twostep",
    opt.meth = "none")
plot(m1)
plot(m1, type = "coef.range")
plot(m1, type = "coef.path")


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "iterative",
    opt.meth = "none")
 plot(m1)
 plot(m1, type = "coef.range")
 plot(m1, type = "coef.path")
```

`print.pdynmc` *Print Fitted Model Object.*

Description

`print.pdynmc` prints objects of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	An object of class 'pdynmc'.
digits	An integer indicating the maximum number of digits to display in the object.
...	further arguments.

Value

Print objects of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
m1


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
```

```
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
m1
```

`print.summary.pdynmc` *Print Summary of Fitted Model Object.*

Description

`print.summary.pdynmc` prints the summary for objects of class 'pdynmc'.

Usage

```
## S3 method for class 'summary.pdynmc'
print(
  x,
  digits = max(3, getOption("digits") - 3),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

`x`	An object of class 'summary.pdynmc'.
`digits`	An integer indicating the maximum number of digits to display in the object.
`signif.stars`	Argument is defined as in `options`.
`...`	further arguments.

Value

Print information on objects of class 'summary.pdynmc'.

Author(s)

Markus Fritsch

See Also

`pdynmc` for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
summary(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 summary(m1)
```

`residuals.pdynmc` *Extract Residuals of Fitted Model.*

Description

`residuals.pdynmc` extracts residuals from an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
residuals(object, step = object$iter, na.rm = FALSE, ...)
```

Arguments

object	An object of class 'pdynmc'.
step	An integer denoting the iteration step for which fitted values are extracted (defaults to last iteration step used for obtaining parameter estimates).
na.rm	A logical variable indicating whether missing values should be removed from the vector of fitted values (defaults to 'FALSE').
...	further arguments.

Value

Extract residuals from object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
residuals(m1, na.rm = TRUE)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
```

```
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
residuals(m1, na.rm = TRUE)
```

`sargan.fct` *Sargan test.*

Description

`sargan.fct` tests the validity of the overidentifying restrictions.

Usage

```
sargan.fct(object)
```

Arguments

`object`	An object of class 'pdynmc'.

Details

The null hypothesis is that the overidentifying restrictions are valid. The test statistic is computed as proposed by Sargan (1958). As noted by Bowsher (2002) and Windmeijer (2005), the test statistic is weakened by many instruments and inconsistent in the presence of heteroscedasticity according to Roodman (2009a).

Value

An object of class 'htest' which contains the Sargan test statistic and corresponding p-value for the null hypothesis that the overidentifying restrictions are valid.

References

Bowsher CG (2002). "On testing overidentifying restrictions in dynamic panel data models." *Economics Letters*, **77**(2), 211–220. doi:10.1016/S01651765(02)001301.

Roodman D (2009a). "How to do xtabond2: An Introduction to Difference and System GMM in Stata." *Stata Journal*, **9**(1), 86–136. https://www.stata-journal.com/article.html?article=st0159.

Sargan JD (1958). "The Estimation of Economic Relationships Using Instrumental Variables." *Econometrica*, **26**(3), 393–415. doi:10.2307/1907619.

Windmeijer F (2005). "A finite sample correction for the variance of linear efficient two-step GMM estimators." *Journal of Econometrics*, **126**(1), 25–51. doi:10.1016/j.jeconom.2004.02.005.

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(140:0), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
sargan.fct(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 sargan.fct(m1)
```

---

strucUPD.plot *Plot on Structure of Unbalanced Panel Dataset.*

Description

strucUPD.plot Plot on cross-section and longtudinal structure of an object of class ‘data.frame‘ containing an unbalanced panel dataset.

Usage

```
strucUPD.plot(
  object,
  i.name = NULL,
  t.name = NULL,
  col.range = c("gold", "darkblue"),
  plot.name = "Unbalanced panel structure",
  ...
)
```

Arguments

object	An object of class 'data.frame'.
i.name	Column name of cross-section identifier.
t.name	Column name of time-series identifier.
col.range	A vector of at least two colors used to visualize the structure of the unbalanced panel data set (defaults to 'gold' and 'darkblue'); must be a valid argument to col2rgb.
plot.name	A vector indicating the title of the plot (defaults to 'Unbalanced panel structure').
...	further arguments.

Value

Returns a plot for an unbalanced panel dataset contained in an object of class 'data.frame' that visualizes the structure of the data. Cross-section dimension is plotted on the ordinate, longitudinal dimension on the abscissa. Each cross-sectional observation is represented by a bar. Breaks in the bars represent missing longitudinal observations.

Author(s)

Markus Fritsch, Joachim Schnurbus

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])

## Code example
strucUPD.plot(dat, i.name = "firm", t.name = "year")
```

summary.pdynmc *Summary for Fitted Model Object.*

Description

summary.pdynmc generates the summary for objects of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
summary(object, ...)
```

Arguments

object	An object of class 'pdynmc'.
...	further arguments.

Value

Object of class 'summary.pdynmc'.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
summary(m1, na.rm = TRUE)
```

```
## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 summary(m1)
```

`variable.names.pdynmc` *Extract Names of Explanatory Variables of Fitted Model.*

Description

`variable.names.pdynmc` extracts explanatory variables from an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
variable.names(object, ...)
```

Arguments

`object`	An object of class 'pdynmc'.
`...`	further arguments.

Value

Extract explanatory variables from an object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

`pdynmc` for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
variable.names(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 variable.names(m1)
```

---

vcov.pdynmc *Extract Variance Covariance Matrix of Fitted Model.*

Description

vcov.pdynmc extracts variance covariance matrix of the paramter estimates from an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
vcov(object, step = object$iter, ...)
```

Arguments

object	An object of class 'pdynmc'.
step	An integer denoting the iteration step for which fitted values are extracted (defaults to last iteration step used for obtaining parameter estimates).
...	further arguments.

Value

Extract variance covariance matrix of the parameter estimates from an object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
vcov(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
```

```
vcov(m1)
```

`wald.fct` *Wald Test.*

Description

`wald.fct` computes F test statistics and corresponding p-values for 'pdynmc' objects.

Usage

```
wald.fct(object, param)
```

Arguments

`object`	An object of class 'pdynmc'.
`param`	A character string that denotes the null hypothesis. Choices are time.dum (i.e., all time dummies are jointly zero), slope (i.e., all slope coefficients are jointly zero), and all (i.e., all dummies and slope coefficients are jointly zero).

Details

The three available null hypothesis are: All time dummies are jointly zero, all slope coefficients are jointly zero, all times dummies and slope coefficients are jointly zero.

Value

An object of class 'htest' which contains the F test statistic and corresponding p-value for the tested null hypothesis.

See Also

`pdynmc` for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(140:0), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
```

```
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
wald.fct(param = "all", m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

## Further code example
 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 wald.fct(m1, param = "all")
```

`wmat` *Extract Weighting Matrix of Fitted Model.*

Description

`wmat` is a generic function for extracting the weighting matrix of an object.

Usage

```
wmat(object, ...)
```

Arguments

`object`	An object for which the weighting matrix is desired.
`...`	further arguments.

Value

Extract weighting matrix from an object.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
wmat(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
 wmat(m1)
```

---

wmat.pdynmc *Extract Weighting Matrix of Fitted Model.*

Description

wmat.pdynmc extracts weighting matrix from an object of class 'pdynmc'.

Usage

```
## S3 method for class 'pdynmc'
wmat(object, step = object$iter, ...)
```

Arguments

object	An object of class 'pdynmc'.
step	An integer denoting the iteration step for which fitted values are extracted (defaults to last iteration step used for obtaining parameter estimates).
...	further arguments.

Value

Extract weighting matrix from an object of class 'pdynmc'.

Author(s)

Markus Fritsch

See Also

pdynmc for fitting a linear dynamic panel data model.

Examples

```
## Load data
data(ABdata, package = "pdynmc")
dat <- ABdata
dat[,c(4:7)] <- log(dat[,c(4:7)])
dat <- dat[c(1:140), ]

## Code example
m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
wmat(m1)


## Load data
 data(ABdata, package = "pdynmc")
 dat <- ABdata
 dat[,c(4:7)] <- log(dat[,c(4:7)])

 m1 <- pdynmc(dat = dat, varname.i = "firm", varname.t = "year",
    use.mc.diff = TRUE, use.mc.lev = FALSE, use.mc.nonlin = FALSE,
    include.y = TRUE, varname.y = "emp", lagTerms.y = 2,
```

```
    fur.con = TRUE, fur.con.diff = TRUE, fur.con.lev = FALSE,
    varname.reg.fur = c("wage", "capital", "output"), lagTerms.reg.fur = c(1,2,2),
    include.dum = TRUE, dum.diff = TRUE, dum.lev = FALSE, varname.dum = "year",
    w.mat = "iid.err", std.err = "corrected", estimation = "onestep",
    opt.meth = "none")
wmat(m1)
```

Index