Package 'plm'

July 23, 2025

Version 2.6-6

Date 2025-04-04

Title Linear Models for Panel Data

Depends R (>= 3.2.0)

Imports MASS, bdsmatrix, collapse (>= 1.8.9), zoo, nlme, sandwich, lattice, lmtest, maxLik, Rdpack, Formula, stats

Suggests AER, car, statmod, urca, pder, texreg, knitr, rmarkdown, fixest, lfe

Description A set of estimators for models and (robust) covariance matrices, and tests for panel data econometrics, including within/fixed effects, random effects, between, first-difference, nested random effects as well as instrumental-variable (IV) and Hausman-Taylor-style models, panel generalized method of moments (GMM) and general FGLS models, mean groups (MG), demeaned MG, and common correlated effects (CCEMG) and pooled (CCEP) estimators with common factors, variable coefficients and limited dependent variables models. Test functions include model specification, serial correlation, cross-sectional dependence, panel unit root and panel Granger (non-)causality. Typical references are general econometrics text books such as Baltagi (2021), Econometric Analysis of Panel Data (<doi:10.1007/978-3-030-53953-5>), Hsiao (2014), Analysis of Panel Data (<doi:10.1017/CBO9781139839327>), and Croissant and Millo (2018), Panel Data Econometrics with R (<doi:10.1002/9781119504641>).

License GPL (>= 2)

VignetteBuilder knitr

URL https://cran.r-project.org/package=plm, https://github.com/ycroissant/plm

BugReports https://github.com/ycroissant/plm/issues

RoxygenNote 7.3.2

RdMacros Rdpack

Encoding UTF-8

NeedsCompilation no

Author Yves Croissant [aut],
Giovanni Millo [aut],
Kevin Tappe [aut, cre],
Ott Toomet [ctb],
Christian Kleiber [ctb],
Achim Zeileis [ctb],
Arne Henningsen [ctb],
Liviu Andronic [ctb],
Nina Schoenfelder [ctb]

Maintainer Kevin Tappe <kevin.tappe@bwi.uni-stuttgart.de>

Repository CRAN

Date/Publication 2025-04-04 15:00:02 UTC

Contents

`plm-package` *plm package: linear models for panel data*

Description

plm is a package for R which intends to make the estimation of linear panel models straightforward. plm provides functions to estimate a wide variety of models and to make (robust) inference.

Details

For a gentle and comprehensive introduction to the package, please see the package's vignette.

The main functions to estimate models are:

- `plm`: panel data estimators using `lm` on transformed data,
- `pvcm`: variable coefficients models
- `pgmm`: generalized method of moments (GMM) estimation for panel data,
- `pggls`: estimation of general feasible generalized least squares models,
- `pmg`: mean groups (MG), demeaned MG and common correlated effects (CCEMG) estimators,
- `pcce`: estimators for common correlated effects mean groups (CCEMG) and pooled (CCEP) for panel data with common factors,
- `pldv`: panel estimators for limited dependent variables.

Next to the model estimation functions, the package offers several functions for statistical tests related to panel data/models.

Multiple functions for (robust) variance–covariance matrices are at hand as well.

The package also provides data sets to demonstrate functions and to replicate some text book/paper results. Use `data(package="plm")` to view a list of available data sets in the package.

Author(s)

Maintainer: Kevin Tappe <kevin.tappe@bwi.uni-stuttgart.de>

Authors:

- Yves Croissant <yves.croissant@univ-reunion.fr>
- Giovanni Millo <giovanni.millo@deams.units.it>

Other contributors:

- Ott Toomet <otoomet@gmail.com> [contributor]
- Christian Kleiber <Christian.Kleiber@unibas.ch> [contributor]
- Achim Zeileis <Achim.Zeileis@R-project.org> [contributor]
- Arne Henningsen <arne.henningsen@googlemail.com> [contributor]
- Liviu Andronic [contributor]
- Nina Schoenfelder [contributor]

See Also

Useful links:

- https://cran.r-project.org/package=plm
- https://github.com/ycroissant/plm
- Report bugs at https://github.com/ycroissant/plm/issues

Examples

```
data("Produc", package = "plm")
zz <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
          data = Produc, index = c("state","year"))
summary(zz)

# replicates some results from Baltagi (2013), table 3.1
data("Grunfeld", package = "plm")
p <- plm(inv ~ value + capital,
         data = Grunfeld, model="pooling")

wi <- plm(inv ~ value + capital,
          data = Grunfeld, model="within", effect = "twoways")

swar <- plm(inv ~ value + capital,
            data = Grunfeld, model="random", effect = "twoways")

amemiya <- plm(inv ~ value + capital,
               data = Grunfeld, model = "random", random.method = "amemiya",
               effect = "twoways")

walhus <- plm(inv ~ value + capital,
              data = Grunfeld, model = "random", random.method = "walhus",
              effect = "twoways")
```

`aneweytest` *Angrist and Newey's version of Chamberlain test for fixed effects*

Description

Angrist and Newey's version of the Chamberlain test

Usage

```
aneweytest(formula, data, subset, na.action, index = NULL, ...)
```

Arguments

formula	a symbolic description for the model to be estimated,
data	a data.frame,
subset	see lm(),
na.action	see lm(),
index	the indexes,
...	further arguments.

Details

Angrist and Newey's test is based on the results of the artifactual regression of the within residuals on the covariates for all the periods.

Value

An object of class "htest".

Author(s)

Yves Croissant

References

Angrist JD, Newey WK (1991). "Over-identification tests in earnings functions with fixed effects." *Journal of Business & Economic Statistics*, **9**(3), 317–323.

See Also

piest() for Chamberlain's test

Examples

```
data("RiceFarms", package = "plm")
aneweytest(log(goutput) ~ log(seed) + log(totlabor) + log(size), RiceFarms, index = "id")
```

Cigar *Cigarette Consumption*

Description

a panel of 46 observations from 1963 to 1992

Format

A data frame containing :

state state abbreviation

year the year

price price per pack of cigarettes

pop population

pop16 population above the age of 16

cpi consumer price index (1983=100)

ndi per capita disposable income

sales cigarette sales in packs per capita

pimin minimum price in adjoining states per pack of cigarettes

Details

total number of observations : 1380

observation : regional

country : United States

Source

Online complements to Baltagi (2001):

https://www.wiley.com/legacy/wileychi/baltagi/

Online complements to Baltagi (2013):

https://bcs.wiley.com/he-bcs/Books?action=resource&bcsId=4338&itemId=1118672321&resourceId=13452

References

Baltagi BH (2001). *Econometric Analysis of Panel Data*, 3rd edition. John Wiley and Sons ltd.

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Baltagi B, Levin D (1992). "Cigarette taxation: Raising revenues and reducing consumption." *Structural Change and Economic Dynamics*, **3**(2), 321-335. https://EconPapers.repec.org/RePEc:eee:streco:v:3:y:1992:i:2:p:321-335.

Baltagi BH, Griffin JM, Xiong W (2000). "To Pool or Not to Pool: Homogeneous Versus Heterogeneous Estimators Applied to Cigarette Demand." *The Review of Economics and Statistics*, **82**(1), 117-126. doi:10.1162/003465300558551, https://doi.org/10.1162/003465300558551.

`cipstest` *Cross-sectionally Augmented IPS Test for Unit Roots in Panel Models*

Description

Cross-sectionally augmented Im, Pesaran and Shin (IPS) test for unit roots in panel models.

Usage

```
cipstest(
  x,
  lags = 2L,
  type = c("trend", "drift", "none"),
  model = c("cmg", "mg", "dmg"),
  truncated = FALSE,
  ...
)
```

Arguments

`x`	an object of class `"pseries"`,
`lags`	integer, lag order for Dickey-Fuller augmentation,
`type`	one of `"trend"` (default), `"drift"`, `"none"`,
`model`	one of `"cmg"` (default), `"mg"`, `"dmg"`,
`truncated`	logical, specifying whether to calculate the truncated version of the test (default: `FALSE`),
`...`	further arguments passed to `critvals.cips` (non-exported function).

Details

Pesaran's (Pesaran 2007) cross-sectionally augmented version of the IPS unit root test (Im et al. 2003) (H0: `pseries` has a unit root) is a so-called second-generation panel unit root test: it is in fact robust against cross-sectional dependence, provided that the default `model="cmg"` is calculated. Else one can obtain the standard (`model="mg"`) or cross-sectionally demeaned (`model="dmg"`) versions of the IPS test.

Argument `type` controls how the test is executed:

- `"none"`: no intercept, no trend (Case I in (Pesaran 2007)),
- `"drift"`: with intercept, no trend (Case II),
- `"trend"` (default): with intercept, with trend (Case III).

Value

An object of class `"htest"`.

Author(s)

Giovanni Millo

References

Im KS, Pesaran MH, Shin Y (2003). "Testing for unit roots in heterogenous panels." *Journal of Econometrics*, **115(1)**, 53-74.

Pesaran MH (2007). "A simple panel unit root test in the presence of cross-section dependence." *Journal of Applied Econometrics*, **22**(2), 265–312.

See Also

`purtest()`, `phansitest()`

Examples

```
data("Produc", package = "plm")
Produc <- pdata.frame(Produc, index=c("state", "year"))
## check whether the gross state product (gsp) is trend-stationary
cipstest(Produc$gsp, type = "trend")
```

---

cortab *Cross–sectional correlation matrix*

Description

Computes the cross–sectional correlation matrix

Usage

```
cortab(x, grouping, groupnames = NULL, value = "statistic", ...)
```

Arguments

`x`	an object of class `pseries`
`grouping`	grouping variable,
`groupnames`	a character vector of group names,
`value`	to complete,
`...`	further arguments.

Value

A matrix with average correlation coefficients within a group (diagonal) and between groups (off-diagonal).

Examples

```
data("Grunfeld", package = "plm")
pGrunfeld <- pdata.frame(Grunfeld)
grp <- c(rep(1, 100), rep(2, 50), rep(3, 50)) # make 3 groups
cortab(pGrunfeld$value, grouping = grp, groupnames = c("A", "B", "C"))
```

Crime *Crime in North Carolina*

Description

a panel of 90 observational units (counties) from 1981 to 1987

Format

A data frame containing :

county county identifier

year year from 1981 to 1987

crmrte crimes committed per person

prbarr 'probability' of arrest

prbconv 'probability' of conviction

prbpris 'probability' of prison sentence

avgsen average sentence, days

polpc police per capita

density people per square mile

taxpc tax revenue per capita

region factor. One of 'other', 'west' or 'central'.

smsa factor. (Also called "urban".) Does the individual reside in a SMSA (standard metropolitan statistical area)?

pctmin percentage minority in 1980

wcon weekly wage in construction

wtuc weekly wage in transportation, utilities, communications

wtrd weekly wage in wholesale and retail trade

wfir weekly wage in finance, insurance and real estate

wser weekly wage in service industry

wmfg weekly wage in manufacturing

wfed weekly wage in federal government

wsta weekly wage in state government

wloc weekly wage in local government

mix offence mix: face-to-face/other

pctymle percentage of young males (between ages 15 to 24)

lcrmrte log of crimes committed per person

lprbarr log of 'probability' of arrest

lprbconv log of 'probability' of conviction

lprbpris log of 'probability' of prison sentence

lavgsen log of average sentence, days

lpolpc log of police per capita

ldensity log of people per square mile

ltaxpc log of tax revenue per capita

lpctmin log of percentage minority in 1980

lwcon log of weekly wage in construction

lwtuc log of weekly wage in transportation, utilities, communications

lwtrd log of weekly wage in wholesale and retail trade

lwfir log of weekly wage in finance, insurance and real estate

lwser log of weekly wage in service industry

lwmfg log of weekly wage in manufacturing

lwfed log of weekly wage in federal government

lwsta log of weekly wage in state government

lwloc log of weekly wage in local government

lmix log of offence mix: face-to-face/other

lpctymle log of percentage of young males (between ages 15 to 24)

Details

total number of observations : 630

observation : regional

country : United States

The variables l* (lcrmrte, lprbarr, ...) contain the pre-computed logarithms of the base variables as found in the original data set. Note that these values slightly differ from what R's log() function yields for the base variables. In order to reproduce examples from the literature, the pre-computed logs need to be used, otherwise the results differ slightly.

Source

Journal of Applied Econometrics Data Archive (complements Baltagi (2006)):

http://qed.econ.queensu.ca/jae/2006-v21.4/baltagi/

Online complements to Baltagi (2001):

https://www.wiley.com/legacy/wileychi/baltagi/

Online complements to Baltagi (2013):

https://bcs.wiley.com/he-bcs/Books?action=resource&bcsId=4338&itemId=1118672321&resourceId=13452

See also Journal of Applied Econometrics data archive entry for Baltagi (2006) at http://qed.econ.queensu.ca/jae/2006-v21.4/baltagi/.

References

Cornwell C, Trumbull WN (1994). "Estimating the economic model of crime with panel data." *Review of Economics and Statistics*, **76**, 360–366.

Baltagi BH (2006). "Estmating an economic model of crime using panel data from North Carolina." *Journal of Applied Econometrics*, **21**(4).

Baltagi BH (2001). *Econometric Analysis of Panel Data*, 3rd edition. John Wiley and Sons ltd.

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

`detect.lindep` *Functions to detect linear dependence*

Description

Little helper functions to aid users to detect linear dependent columns in a two-dimensional data structure, especially in a (transformed) model matrix - typically useful in interactive mode during model building phase.

Usage

```
detect.lindep(object, ...)

## S3 method for class 'matrix'
detect.lindep(object, suppressPrint = FALSE, ...)

## S3 method for class 'data.frame'
detect.lindep(object, suppressPrint = FALSE, ...)

## S3 method for class 'plm'
detect.lindep(object, suppressPrint = FALSE, ...)

## S3 method for class 'plm'
alias(object, ...)

## S3 method for class 'pdata.frame'
alias(
  object,
  model = c("pooling", "within", "Between", "between", "mean", "random", "fd"),
  effect = c("individual", "time", "twoways"),
  ...
)
```

Arguments

object	for detect.lindep: an object which should be checked for linear dependence (of class "matrix", "data.frame", or "plm"); for alias: either an estimated model of class "plm" or a "pdata.frame". Usually, one wants to input a model matrix here or check an already estimated plm model,
...	further arguments.
suppressPrint	for detect.lindep only: logical indicating whether a message shall be printed; defaults to printing the message, i. e., to suppressPrint = FALSE,
model	(see plm),
effect	(see plm),

Details

Linear dependence of columns/variables is (usually) readily avoided when building one's model. However, linear dependence is sometimes not obvious and harder to detect for less experienced applied statisticians. The so called "dummy variable trap" is a common and probably the best–known fallacy of this kind (see e. g. Wooldridge (2016), sec. 7-2.). When building linear models with lm or plm's pooling model, linear dependence in one's model is easily detected, at times post hoc.

However, linear dependence might also occur after some transformations of the data, albeit it is not present in the untransformed data. The within transformation (also called fixed effect transformation) used in the "within" model can result in such linear dependence and this is harder to come to mind when building a model. See **Examples** for two examples of linear dependent columns after the within transformation: ex. 1) the transformed variables have the opposite sign of one another; ex. 2) the transformed variables are identical.

During plm's model estimation, linear dependent columns and their corresponding coefficients in the resulting object are silently dropped, while the corresponding model frame and model matrix still contain the affected columns. The plm object contains an element aliased which indicates any such aliased coefficients by a named logical.

Both functions, detect.lindep and alias, help to detect linear dependence and accomplish almost the same: detect.lindep is a stand alone implementation while alias is a wrapper around stats::alias.lm(), extending the alias generic to classes "plm" and "pdata.frame". alias hinges on the availability of the package **MASS** on the system. Not all arguments of alias.lm are supported. Output of alias is more informative as it gives the linear combination of dependent columns (after data transformations, i. e., after (quasi)-demeaning) while detect.lindep only gives columns involved in the linear dependence in a simple format (thus being more suited for automatic post–processing of the information).

Value

For detect.lindep: A named numeric vector containing column numbers of the linear dependent columns in the object after data transformation, if any are present. NULL if no linear dependent columns are detected.

For alias: return value of stats::alias.lm() run on the (quasi-)demeaned model, i. e., the information outputted applies to the transformed model matrix, not the original data.

Note

function `detect.lindep` was called `detect_lin_dep` initially but renamed for naming consistency later.

Author(s)

Kevin Tappe

References

Wooldridge JM (2013). *Introductory Econometrics: a modern approach*, 5th edition. South-Western (Cengage Learning).

See Also

`stats::alias()`, `stats::model.matrix()` and especially `plm`'s `model.matrix()` for (transformed) model matrices, plm's `model.frame()`.

Examples

```
### Example 1 ###
# prepare the data
data("Cigar" , package = "plm")
Cigar[ , "fact1"] <- c(0,1)
Cigar[ , "fact2"] <- c(1,0)
Cigar.p <- pdata.frame(Cigar)

# setup a formula and a model frame
form <- price ~ 0 + cpi + fact1 + fact2
mf <- model.frame(Cigar.p, form)
# no linear dependence in the pooling model's model matrix
# (with intercept in the formula, there would be linear dependence)
detect.lindep(model.matrix(mf, model = "pooling"))
# linear dependence present in the FE transformed model matrix
modmat_FE <- model.matrix(mf, model = "within")
detect.lindep(modmat_FE)
mod_FE <- plm(form, data = Cigar.p, model = "within")
detect.lindep(mod_FE)
alias(mod_FE) # => fact1 == -1*fact2
plm(form, data = mf, model = "within")$aliased # "fact2" indicated as aliased

# look at the data: after FE transformation fact1 == -1*fact2
head(modmat_FE)
all.equal(modmat_FE[ , "fact1"], -1*modmat_FE[ , "fact2"])

### Example 2 ###
# Setup the data:
# Assume CEOs stay with the firms of the Grunfeld data
# for the firm's entire lifetime and assume some fictional
# data about CEO tenure and age in year 1935 (first observation
# in the data set) to be at 1 to 10 years and 38 to 55 years, respectively.
```

```
# => CEO tenure and CEO age increase by same value (+1 year per year).
data("Grunfeld", package = "plm")
set.seed(42)
# add fictional data
Grunfeld$CEOtenure <- c(replicate(10, seq(from=s<-sample(1:10,  1), to=s+19, by=1)))
Grunfeld$CEOage    <- c(replicate(10, seq(from=s<-sample(38:65, 1), to=s+19, by=1)))

# look at the data
head(Grunfeld, 50)

form <- inv ~ value + capital + CEOtenure + CEOage
mf <- model.frame(pdata.frame(Grunfeld), form)
# no linear dependent columns in original data/pooling model
modmat_pool <- model.matrix(mf, model="pooling")
detect.lindep(modmat_pool)
mod_pool <- plm(form, data = Grunfeld, model = "pooling")
alias(mod_pool)

# CEOtenure and CEOage are linear dependent after FE transformation
# (demeaning per individual)
modmat_FE <- model.matrix(mf, model="within")
detect.lindep(modmat_FE)
mod_FE <- plm(form, data = Grunfeld, model = "within")
detect.lindep(mod_FE)
alias(mod_FE)

# look at the transformed data: after FE transformation CEOtenure == 1*CEOage
head(modmat_FE, 50)
all.equal(modmat_FE[ , "CEOtenure"], modmat_FE[ , "CEOage"])
```

EmplUK *Employment and Wages in the United Kingdom*

Description

An unbalanced panel of 140 observations from 1976 to 1984

Format

A data frame containing :

firm firm index

year year

sector the sector of activity

emp employment

wage wages

capital capital

output output

Details

total number of observations : 1031

observation : firms

country : United Kingdom

Source

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

`ercomp` *Estimation of the error components*

Description

This function enables the estimation of the variance components of a panel model.

Usage

```
ercomp(object, ...)

## S3 method for class 'plm'
ercomp(object, ...)

## S3 method for class 'pdata.frame'
ercomp(
  object,
  effect = c("individual", "time", "twoways", "nested"),
  method = NULL,
  models = NULL,
  dfcor = NULL,
  index = NULL,
  ...
)

## S3 method for class 'formula'
ercomp(
  object,
  data,
  effect = c("individual", "time", "twoways", "nested"),
  method = NULL,
  models = NULL,
  dfcor = NULL,
  index = NULL,
  ...
)
```

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

Arguments

object	a formula or a plm object,
...	further arguments.
effect	the effects introduced in the model, see plm() for details,
method	method of estimation for the variance components, see plm() for details,
models	the models used to estimate the variance components (an alternative to the previous argument),
dfcor	a numeric vector of length 2 indicating which degree of freedom should be used,
index	the indexes,
data	a data.frame,
x	an ercomp object,
digits	digits,

Value

An object of class "ercomp": a list containing

- sigma2 a named numeric with estimates of the variance components,
- theta contains the parameter(s) used for the transformation of the variables: For a one-way model, a numeric corresponding to the selected effect (individual or time); for a two-ways model a list of length 3 with the parameters. In case of a balanced model, the numeric has length 1 while for an unbalanced model, the numerics' length equal the number of observations.

Author(s)

Yves Croissant

References

Amemiya T (1971). "The Estimation of the Variances in a Variance–Components Model." *International Economic Review*, **12**, 1–13.

Nerlove M (1971). "Further Evidence on the Estimation of Dynamic Economic Relations from a Time–Series of Cross–Sections." *Econometrica*, **39**, 359–382.

Swamy PAVB, Arora SS (1972). "The Exact Finite Sample Properties of the Estimators of Coefficients in the Error Components Regression Models." *Econometrica*, **40**, 261–275.

Wallace TD, Hussain A (1969). "The Use of Error Components Models in Combining Cross Section With Time Series Data." *Econometrica*, **37**(1), 55–72.

See Also

plm() where the estimates of the variance components are used if a random effects model is estimated

Examples

```
data("Produc", package = "plm")
# an example of the formula method
ercomp(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc,
       method = "walhus", effect = "time")
# same with the plm method
z <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
         data = Produc, random.method = "walhus",
         effect = "time", model = "random")
ercomp(z)
# a two-ways model
ercomp(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc,
       method = "amemiya", effect = "twoways")
```

---

fixef.plm *Extract the Fixed Effects*

Description

Function to extract the fixed effects from a plm object and associated summary method.

Usage

```
## S3 method for class 'plm'
fixef(
  object,
  effect = NULL,
  type = c("level", "dfirst", "dmean"),
  vcov = NULL,
  ...
)

## S3 method for class 'fixef'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)

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

```
## S3 method for class 'summary.fixef'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)

## S3 method for class 'pggls'
fixef(
  object,
  effect = NULL,
  type = c("level", "dfirst", "dmean"),
  vcov = NULL,
  ...
)
```

Arguments

effect	one of "individual", "time", or "twoways", only relevant in case of two–ways effects models (where it defaults to "individual"),
type	one of "level", "dfirst", or "dmean",
vcov	a variance–covariance matrix furnished by the user or a function to calculate one (see **Examples**),
...	further arguments.
x, object	an object of class "plm", an object of class "fixef" for the print and the summary method,
digits	digits,
width	the maximum length of the lines in the print output,

Details

Function fixef calculates the fixed effects and returns an object of class c("fixef", "numeric"). By setting the type argument, the fixed effects may be returned in levels ("level"), as deviations from the first value of the index ("dfirst"), or as deviations from the overall mean ("dmean"). If the argument vcov was specified, the standard errors (stored as attribute "se" in the return value) are the respective robust standard errors. For two-way fixed-effect models, argument effect controls which of the fixed effects are to be extracted: "individual", "time", or the sum of individual and time effects ("twoways"). NB: See **Examples** for how the sum of effects can be split in an individual and a time component. For one-way models, the effects of the model are extracted and the argument effect is disrespected.

The associated summary method returns an extended object of class c("summary.fixef", "matrix") with more information (see sections **Value** and **Examples**).

References with formulae (except for the two-ways unbalanced case) are, e.g., Greene (2012), Ch. 11.4.4, p. 364, formulae (11-25); Wooldridge (2010), Ch. 10.5.3, pp. 308-309, formula (10.58).

Value

For function `fixef`, an object of class `c("fixef", "numeric")` is returned: It is a numeric vector containing the fixed effects with attribute `se` which contains the standard errors. There are two further attributes: attribute `type` contains the chosen type (the value of argument `type` as a character); attribute `df.residual` holds the residual degrees of freedom (integer) from the fixed effects model (plm object) on which `fixef` was run. For the two-way unbalanced case, only attribute `type` is added.

For function `summary.fixef`, an object of class `c("summary.fixef", "matrix")` is returned: It is a matrix with four columns in this order: the estimated fixed effects, their standard errors and associated t–values and p–values. For the two-ways unbalanced case, the matrix contains only the estimates. The type of the fixed effects and the standard errors in the summary.fixef object correspond to was requested in the `fixef` function by arguments `type` and `vcov`, respectively.

Author(s)

Yves Croissant

References

Greene WH (2012). *Econometric Analysis*, 7th edition. Prentice Hall.

Wooldridge JM (2010). *Econometric Analysis of Cross–Section and Panel Data*, 2nd edition. MIT Press.

See Also

`within_intercept()` for the overall intercept of fixed effect models along its standard error, `plm()` for plm objects and within models (= fixed effects models) in general. See `ranef()` to extract the random effects from a random effects model.

Examples

```
data("Grunfeld", package = "plm")
gi <- plm(inv ~ value + capital, data = Grunfeld, model = "within")
fixef(gi)
summary(fixef(gi))
summary(fixef(gi))[ , c("Estimate", "Pr(>|t|)")] # only estimates and p-values

# relationship of type = "dmean" and "level" and overall intercept
fx_level <- fixef(gi, type = "level")
fx_dmean <- fixef(gi, type = "dmean")
overallint <- within_intercept(gi)
all.equal(overallint + fx_dmean, fx_level, check.attributes = FALSE) # TRUE

# extract time effects in a twoways effects model
gi_tw <- plm(inv ~ value + capital, data = Grunfeld,
          model = "within", effect = "twoways")
fixef(gi_tw, effect = "time")

# with supplied variance-covariance matrix as matrix, function,
```

```
# and function with additional arguments
fx_level_robust1 <- fixef(gi, vcov = vcovHC(gi))
fx_level_robust2 <- fixef(gi, vcov = vcovHC)
fx_level_robust3 <- fixef(gi, vcov = function(x) vcovHC(x, method = "white2"))
summary(fx_level_robust1) # gives fixed effects, robust SEs, t- and p-values

# calc. fitted values of oneway within model:
fixefs <- fixef(gi)[index(gi, which = "id")]
fitted_by_hand <- fixefs + gi$coefficients["value"]   * gi$model$value +
                           gi$coefficients["capital"] * gi$model$capital

# calc. fittes values of twoway unbalanced within model via effects:
gtw_u <- plm(inv ~ value + capital, data = Grunfeld[-200, ], effect = "twoways")
yhat <- as.numeric(gtw_u$model[ , 1] - gtw_u$residuals) # reference
pred_beta <- as.numeric(tcrossprod(coef(gtw_u), as.matrix(gtw_u$model[ , -1])))
pred_effs <- as.numeric(fixef(gtw_u, "twoways")) # sum of ind and time effects
all.equal(pred_effs + pred_beta, yhat) # TRUE

# Splits of summed up individual and time effects:
# use one "level" and one "dfirst"
ii <- index(gtw_u)[[1L]]; it <- index(gtw_u)[[2L]]
eff_id_dfirst <- c(0, as.numeric(fixef(gtw_u, "individual", "dfirst")))[ii]
eff_ti_dfirst <- c(0, as.numeric(fixef(gtw_u, "time",       "dfirst")))[it]
eff_id_level <- as.numeric(fixef(gtw_u, "individual"))[ii]
eff_ti_level <- as.numeric(fixef(gtw_u, "time"))[it]

all.equal(pred_effs, eff_id_level  + eff_ti_dfirst) # TRUE
all.equal(pred_effs, eff_id_dfirst + eff_ti_level)  # TRUE
```

`Gasoline` *Gasoline Consumption*

Description

A panel of 18 observations from 1960 to 1978

Format

A data frame containing :

country a factor with 18 levels

year the year

lgaspcar logarithm of motor gasoline consumption per car

lincomep logarithm of real per-capita income

lrpmg logarithm of real motor gasoline price

lcarpcap logarithm of the stock of cars per capita

Details

total number of observations : 342

observation : country

country : OECD

Source

Online complements to Baltagi (2001):

https://www.wiley.com/legacy/wileychi/baltagi/

Online complements to Baltagi (2013):

https://bcs.wiley.com/he-bcs/Books?action=resource&bcsId=4338&itemId=1118672321&resourceId=13452

References

Baltagi BH (2001). *Econometric Analysis of Panel Data*, 3rd edition. John Wiley and Sons ltd.

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Baltagi BH, Griffin JM (1983). "Gasoline demand in the OECD: An application of pooling and testing procedures." *European Economic Review*, **22**(2), 117 - 137. ISSN 0014-2921, https://doi.org/10.1016/0014-2921(83)90077-6.

Grunfeld *Grunfeld's Investment Data*

Description

A balanced panel of 10 observational units (firms) from 1935 to 1954

Format

A data frame containing :

firm observation

year date

inv gross Investment

value value of the firm

capital stock of plant and equipment

Details

total number of observations : 200

observation : production units

country : United States

Note

The Grunfeld data as provided in package plm is the same data as used in Baltagi (2001), see **Examples** below.

NB:
Various versions of the Grunfeld data circulate online. Also, various text books (and also varying among editions) and papers use different subsets of the original Grunfeld data, some of which contain errors in a few data points compared to the original data used by Grunfeld (1958) in his PhD thesis. See Kleiber/Zeileis (2010) and its accompanying website for a comparison of various Grunfeld data sets in use.

Source

Online complements to Baltagi (2001):

https://www.wiley.com/legacy/wileychi/baltagi/

https://www.wiley.com/legacy/wileychi/baltagi/supp/Grunfeld.fil

Online complements to Baltagi (2013):

https://bcs.wiley.com/he-bcs/Books?action=resource&bcsId=4338&itemId=1118672321&resourceId=13452

References

Baltagi BH (2001). *Econometric Analysis of Panel Data*, 3rd edition. John Wiley and Sons ltd.

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Grunfeld Y (1958). *The determinants of corporate investment*. Ph.D. thesis, Department of Economics, University of Chicago.

Kleiber C, Zeileis A (2010). "The Grunfeld Data at 50." *German Economic Review*, **11**, 404-417. https://doi.org/10.1111/j.1468-0475.2010.00513.x.

website accompanying the paper with various variants of the Grunfeld data: https://www.zeileis.org/grunfeld/.

See Also

For the complete Grunfeld data (11 firms), see AER::Grunfeld, in the AER package.

Examples

```
## Not run:
# Compare plm's Grunfeld data to Baltagi's (2001) Grunfeld data:
  data("Grunfeld", package="plm")
  Grunfeld_baltagi2001 <- read.csv("http://www.wiley.com/legacy/wileychi/
    baltagi/supp/Grunfeld.fil", sep="", header = FALSE)
  library(compare)
  compare::compare(Grunfeld, Grunfeld_baltagi2001, allowAll = T) # same data set

## End(Not run)
```

`has.intercept` *Check for the presence of an intercept in a formula or in a fitted model*

Description

The presence of an intercept is checked using the formula which is either provided as the argument of the function or extracted from a fitted model.

Usage

```
has.intercept(object, ...)

## Default S3 method:
has.intercept(object, data = NULL, ...)

## S3 method for class 'formula'
has.intercept(object, data = NULL, ...)

## S3 method for class 'Formula'
has.intercept(object, rhs = NULL, data = NULL, ...)

## S3 method for class 'panelmodel'
has.intercept(object, ...)

## S3 method for class 'plm'
has.intercept(object, rhs = 1L, ...)
```

Arguments

`object`	a `formula`, a `Formula` or a fitted model (of class `plm` or `panelmodel`),
`...`	further arguments.
`data`	default is `NULL` and only needs to be changes to a data set if the formula contains a dot (`.`) to allow evaluation of the dot,
`rhs`	an integer (length > 1 is possible), indicating the parts of right hand sides of the formula to be evaluated for the presence of an intercept or `NULL` for all parts of the right hand side (relevant for the `Formula` and the `plm` methods),

Value

a logical

`Hedonic` *Hedonic Prices of Census Tracts in the Boston Area*

Description

A cross-section

Format

A dataframe containing:

mv median value of owner–occupied homes

crim crime rate

zn proportion of 25,000 square feet residential lots

indus proportion of no–retail business acres

chas is the tract bounds the Charles River?

nox annual average nitrogen oxide concentration in parts per hundred million

rm average number of rooms

age proportion of owner units built prior to 1940

dis weighted distances to five employment centers in the Boston area

rad index of accessibility to radial highways

tax full value property tax rate ($/$10,000)

ptratio pupil/teacher ratio

blacks proportion of blacks in the population

lstat proportion of population that is lower status

townid town identifier

Details

number of observations : 506

observation : regional

country : United States

Source

Online complements to Baltagi (2001):

https://www.wiley.com/legacy/wileychi/baltagi/

Online complements to Baltagi (2013):

https://bcs.wiley.com/he-bcs/Books?action=resource&bcsId=4338&itemId=1118672321&resourceId=13452

References

Baltagi BH (2001). *Econometric Analysis of Panel Data*, 3rd edition. John Wiley and Sons ltd.

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Besley DA, Kuh E, Welsch RE (1980). *Regression diagnostics: identifying influential data and sources of collinearity*. John Wiley and Sons ltd. Wiley series in probability and statistics.

Harrison D, Rubinfeld DL (1978). "Hedonic housing prices and the demand for clean air." *Journal of Environmental Economics and Management*, **5**, 81-102.

`index.plm` *Extract the indexes of panel data*

Description

This function extracts the information about the structure of the individual and time dimensions of panel data. Grouping information can also be extracted if the panel data were created with a grouping variable.

Usage

```
## S3 method for class 'pindex'
index(x, which = NULL, ...)

## S3 method for class 'pdata.frame'
index(x, which = NULL, ...)

## S3 method for class 'pseries'
index(x, which = NULL, ...)

## S3 method for class 'panelmodel'
index(x, which = NULL, ...)
```

Arguments

`x`	an object of class `"pindex"`, `"pdata.frame"`, `"pseries"` or `"panelmodel"`,
`which`	the index(es) to be extracted (see details),
`...`	further arguments.

Details

Panel data are stored in a `"pdata.frame"` which has an `"index"` attribute. Fitted models in `"plm"` have a `"model"` element which is also a `"pdata.frame"` and therefore also has an `"index"` attribute. Finally, each series, once extracted from a `"pdata.frame"`, becomes of class `"pseries"`, which also has this `"index"` attribute. `"index"` methods are available for all these objects. The argument `"which"` indicates which index should be extracted. If `which = NULL`, all indexes are extracted. `"which"` can also be a vector of length 1, 2, or 3 (3 only if the pdata frame was constructed with an

additional group index) containing either characters (the names of the individual variable and/or of the time variable and/or the group variable or "id" and "time") and "group" or integers (1 for the individual index, 2 for the time index, and 3 for the group index (the latter only if the pdata frame was constructed with such).)

Value

A vector or an object of class c("pindex","data.frame") containing either one index, individual and time index, or (any combination of) individual, time and group indexes.

Author(s)

Yves Croissant

See Also

pdata.frame(), plm()

Examples

```
data("Grunfeld", package = "plm")
Gr <- pdata.frame(Grunfeld, index = c("firm", "year"))
m <- plm(inv ~ value + capital, data = Gr)
index(Gr, "firm")
index(Gr, "time")
index(Gr$inv, c(2, 1))
index(m, "id")

# with additional group index
data("Produc", package = "plm")
pProduc <- pdata.frame(Produc, index = c("state", "year", "region"))
index(pProduc, 3)
index(pProduc, "region")
index(pProduc, "group")
```

---

is.pbalanced *Check if data are balanced*

Description

This function checks if the data are balanced, i.e., if each individual has the same time periods

Usage

```
is.pbalanced(x, ...)

## Default S3 method:
is.pbalanced(x, y, ...)
```

```
## S3 method for class 'data.frame'
is.pbalanced(x, index = NULL, ...)

## S3 method for class 'pdata.frame'
is.pbalanced(x, ...)

## S3 method for class 'pseries'
is.pbalanced(x, ...)

## S3 method for class 'pggls'
is.pbalanced(x, ...)

## S3 method for class 'pcce'
is.pbalanced(x, ...)

## S3 method for class 'pmg'
is.pbalanced(x, ...)

## S3 method for class 'pgmm'
is.pbalanced(x, ...)

## S3 method for class 'panelmodel'
is.pbalanced(x, ...)
```

Arguments

x	an object of class pdata.frame, data.frame, pseries, panelmodel, or pgmm,
...	further arguments.
y	(only in default method) the time index variable (2nd index variable),
index	only relevant for data.frame interface; if NULL, the first two columns of the data.frame are assumed to be the index variables; if not NULL, both dimensions ('individual', 'time') need to be specified by index as character of length 2 for data frames, for further details see pdata.frame(),

Details

Balanced data are data for which each individual has the same time periods. The returned values of the is.pbalanced(object) methods are identical to pdim(object)$balanced. is.pbalanced is provided as a short cut and is faster than pdim(object)$balanced because it avoids those computations performed by pdim which are unnecessary to determine the balancedness of the data.

Value

A logical indicating whether the data associated with object x are balanced (TRUE) or not (FALSE).

See Also

punbalancedness() for two measures of unbalancedness, make.pbalanced() to make data balanced; is.pconsecutive() to check if data are consecutive; make.pconsecutive() to make data consecutive (and, optionally, also balanced).
pdim() to check the dimensions of a 'pdata.frame' (and other objects), pvar() to check for individual and time variation of a 'pdata.frame' (and other objects), pseries(), data.frame(), pdata.frame().

Examples

```
# take balanced data and make it unbalanced
# by deletion of 2nd row (2nd time period for first individual)
data("Grunfeld", package = "plm")
Grunfeld_missing_period <- Grunfeld[-2, ]
is.pbalanced(Grunfeld_missing_period)     # check if balanced: FALSE
pdim(Grunfeld_missing_period)$balanced    # same

# pdata.frame interface
pGrunfeld_missing_period <- pdata.frame(Grunfeld_missing_period)
is.pbalanced(Grunfeld_missing_period)

# pseries interface
is.pbalanced(pGrunfeld_missing_period$inv)
```

---

is.pconsecutive *Check if time periods are consecutive*

Description

This function checks for each individual if its associated time periods are consecutive (no "gaps" in time dimension per individual)

Usage

```
is.pconsecutive(x, ...)

## Default S3 method:
is.pconsecutive(x, id, time, na.rm.tindex = FALSE, ...)

## S3 method for class 'data.frame'
is.pconsecutive(x, index = NULL, na.rm.tindex = FALSE, ...)

## S3 method for class 'pseries'
is.pconsecutive(x, na.rm.tindex = FALSE, ...)

## S3 method for class 'pdata.frame'
is.pconsecutive(x, na.rm.tindex = FALSE, ...)
```

```
## S3 method for class 'panelmodel'
is.pconsecutive(x, na.rm.tindex = FALSE, ...)
```

Arguments

x	usually, an object of class pdata.frame, data.frame, pseries, or an estimated panelmodel; for the default method x can also be an arbitrary vector or NULL, see **Details**,
...	further arguments.
id, time	only relevant for default method: vectors specifying the id and time dimensions, i. e., a sequence of individual and time identifiers, each as stacked time series,
na.rm.tindex	logical indicating whether any NA values in the time index are removed before consecutiveness is evaluated (defaults to FALSE),
index	only relevant for data.frame interface; if NULL, the first two columns of the data.frame are assumed to be the index variables; if not NULL, both dimensions ('individual', 'time') need to be specified by index for is.pconsecutive on data frames, for further details see pdata.frame(),

Details

(p)data.frame, pseries and estimated panelmodel objects can be tested if their time periods are consecutive per individual. For evaluation of consecutiveness, the time dimension is interpreted to be numeric, and the data are tested for being a regularly spaced sequence with distance 1 between the time periods for each individual (for each individual the time dimension can be interpreted as sequence t, t+1, t+2, ... where t is an integer). As such, the "numerical content" of the time index variable is considered for consecutiveness, not the "physical position" of the various observations for an individuals in the (p)data.frame/pseries (it is not about "neighbouring" rows). If the object to be evaluated is a pseries or a pdata.frame, the time index is coerced from factor via as.character to numeric, i.e., the series as.numeric(as.character(index(<pseries/pdata.frame>)[[2]]))] is evaluated for gaps.

The default method also works for argument x being an arbitrary vector (see **Examples**), provided one can supply arguments id and time, which need to ordered as stacked time series. As only id and time are really necessary for the default method to evaluate the consecutiveness, x = NULL is also possible. However, if the vector x is also supplied, additional input checking for equality of the lengths of x, id and time is performed, which is safer.

For the data.frame interface, the data is ordered in the appropriate way (stacked time series) before the consecutiveness is evaluated. For the pdata.frame and pseries interface, ordering is not performed because both data types are already ordered in the appropriate way when created.

Note: Only the presence of the time period itself in the object is tested, not if there are any other variables. NA values in individual index are not examined but silently dropped - In this case, it is not clear which individual is meant by id value NA, thus no statement about consecutiveness of time periods for those "NA-individuals" is possible.

Value

A named logical vector (names are those of the individuals). The i-th element of the returned vector corresponds to the i-th individual. The values of the i-th element can be:

TRUE	if the i-th individual has consecutive time periods,
FALSE	if the i-th individual has non-consecutive time periods,
"NA"	if there are any NA values in time index of the i-th the individual; see also argument na.rm.tindex to remove those.

Author(s)

Kevin Tappe

See Also

make.pconsecutive() to make data consecutive (and, as an option, balanced at the same time) and make.pbalanced() to make data balanced.
pdim() to check the dimensions of a 'pdata.frame' (and other objects), pvar() to check for individual and time variation of a 'pdata.frame' (and other objects), lag() for lagged (and leading) values of a 'pseries' object.

pseries(), data.frame(), pdata.frame(), for class 'panelmodel' see plm() and pgmm().

Examples

```
data("Grunfeld", package = "plm")
is.pconsecutive(Grunfeld)
is.pconsecutive(Grunfeld, index=c("firm", "year"))

# delete 2nd row (2nd time period for first individual)
# -> non consecutive
Grunfeld_missing_period <- Grunfeld[-2, ]
is.pconsecutive(Grunfeld_missing_period)
all(is.pconsecutive(Grunfeld_missing_period)) # FALSE

# delete rows 1 and 2 (1st and 2nd time period for first individual)
# -> consecutive
Grunfeld_missing_period_other <- Grunfeld[-c(1,2), ]
is.pconsecutive(Grunfeld_missing_period_other) # all TRUE

# delete year 1937 (3rd period) for _all_ individuals
Grunfeld_wo_1937 <- Grunfeld[Grunfeld$year != 1937, ]
is.pconsecutive(Grunfeld_wo_1937) # all FALSE

# pdata.frame interface
pGrunfeld <- pdata.frame(Grunfeld)
pGrunfeld_missing_period <- pdata.frame(Grunfeld_missing_period)
is.pconsecutive(pGrunfeld) # all TRUE
is.pconsecutive(pGrunfeld_missing_period) # first FALSE, others TRUE


# panelmodel interface (first, estimate some models)
mod_pGrunfeld <- plm(inv ~ value + capital, data = Grunfeld)
mod_pGrunfeld_missing_period <- plm(inv ~ value + capital, data = Grunfeld_missing_period)
```

```
is.pconsecutive(mod_pGrunfeld)
is.pconsecutive(mod_pGrunfeld_missing_period)

nobs(mod_pGrunfeld) # 200
nobs(mod_pGrunfeld_missing_period) # 199


# pseries interface
pinv <- pGrunfeld$inv
pinv_missing_period <- pGrunfeld_missing_period$inv

is.pconsecutive(pinv)
is.pconsecutive(pinv_missing_period)

# default method for arbitrary vectors or NULL
inv <- Grunfeld$inv
inv_missing_period <- Grunfeld_missing_period$inv
is.pconsecutive(inv, id = Grunfeld$firm, time = Grunfeld$year)
is.pconsecutive(inv_missing_period, id = Grunfeld_missing_period$firm,
                                    time = Grunfeld_missing_period$year)

# (not run) demonstrate mismatch lengths of x, id, time
# is.pconsecutive(x = inv_missing_period, id = Grunfeld$firm, time = Grunfeld$year)

# only id and time are needed for evaluation
is.pconsecutive(NULL, id = Grunfeld$firm, time = Grunfeld$year)
```

`is.pseries` *Check if an object is a pseries*

Description

This function checks if an object qualifies as a pseries

Usage

```
is.pseries(object)
```

Arguments

`object`	object to be checked for pseries features

Details

A `"pseries"` is a wrapper around a "basic class" (numeric, factor, logical, character, or complex).

To qualify as a pseries, an object needs to have the following features:

- class contains `"pseries"` and there are at least two classes (`"pseries"` and the basic class),
- have an appropriate index attribute (defines the panel structure),
- any of `is.numeric`, `is.factor`, `is.logical`, `is.character`, `is.complex` is `TRUE`.

Value

A logical indicating whether the object is a pseries (TRUE) or not (FALSE).

See Also

pseries() for some computations on pseries and some further links.

Examples

```
# Create a pdata.frame and extract a series, which becomes a pseries
data("EmplUK", package = "plm")
Em <- pdata.frame(EmplUK)
z <- Em$output

class(z) # pseries as indicated by class
is.pseries(z) # and confirmed by check

# destroy index of pseries and re-check
attr(z, "index") <- NA
is.pseries(z) # now FALSE
```

LaborSupply *Wages and Hours Worked*

Description

A panel of 532 observations from 1979 to 1988

Format

A data frame containing :

lnhr log of annual hours worked

lnwg log of hourly wage

kids number of children

age age

disab bad health

id id

year year

Details

number of observations : 5320

Source

Online complements to Ziliak (1997).

Journal of Business Economics and Statistics web site: https://amstat.tandfonline.com/loi/ubes20/.

References

Colin Cameron A, K. Trivedi P (2005). *Microeconometrics: Methods and Applications*. Cambridge University Press. ISBN 0521848059, doi:10.1017/CBO9780511811241.

Ziliak JP (1997). "Efficient Estimation with Panel Data When Instruments Are Predetermined: An Empirical Comparison of Moment-Condition Estimators." *Journal of Business & Economic Statistics*, **15**(4), 419–431. ISSN 07350015.

`lag.plm` *lag, lead, and diff for panel data*

Description

lag, lead, and diff functions for class pseries.

Usage

```
lead(x, k = 1L, ...)

## S3 method for class 'pseries'
lag(x, k = 1L, shift = c("time", "row"), ...)

## S3 method for class 'pseries'
lead(x, k = 1L, shift = c("time", "row"), ...)

## S3 method for class 'pseries'
diff(x, lag = 1L, shift = c("time", "row"), ...)
```

Arguments

`x`	a `pseries` object,
`k`	an integer, the number of lags for the `lag` and `lead` methods (can also be negative). For the `lag` method, a positive (negative) `k` gives lagged (leading) values. For the `lead` method, a positive (negative) `k` gives leading (lagged) values, thus, `lag(x, k = -1L)` yields the same as `lead(x, k = 1L)`. If `k` is an integer with length > 1 (`k = c(k1, k2, ...)`), a `matrix` with multiple lagged `pseries` is returned,
`...`	further arguments (currently none evaluated).
`shift`	character, either `"time"` (default) or `"row"` determining how the shifting in the `lag`/`lead`/`diff` functions is performed (see Details and Examples).
`lag`	integer, the number of lags for the `diff` method, can also be of length > 1 (see argument `k`) (only non–negative values in argument `lag` are allowed for `diff`),

Details

This set of functions perform lagging, leading (lagging in the opposite direction), and differencing operations on `pseries` objects, i. e., they take the panel structure of the data into account by performing the operations per individual.

Argument `shift` controls the shifting of observations to be used by methods `lag`, `lead`, and `diff`:

- `shift = "time"` (default): Methods respect the numerical value in the time dimension of the index. The time dimension needs to be interpretable as a sequence t, t+1, t+2, ... where t is an integer (from a technical viewpoint, `as.numeric(as.character(index(your_pdata.frame)[[2]]))` needs to result in a meaningful integer).
- `shift = "row"`: Methods perform the shifting operation based solely on the "physical position" of the observations, i.e., neighbouring rows are shifted per individual. The value in the time index is not relevant in this case.

For consecutive time periods per individual, a switch of shifting behaviour results in no difference. Different return values will occur for non-consecutive time periods per individual ("holes in time"), see also Examples.

Value

- An object of class `pseries`, if the argument specifying the lag has length 1 (argument `k` in functions `lag` and `lead`, argument `lag` in function `diff`).
- A matrix containing the various series in its columns, if the argument specifying the lag has length > 1.

Note

The sign of `k` in `lag.pseries` results in inverse behaviour compared to `stats::lag()` and `zoo::lag.zoo()`.

Author(s)

Yves Croissant and Kevin Tappe

See Also

To check if the time periods are consecutive per individual, see `is.pconsecutive()`.

For further function for 'pseries' objects: `between()`, Between(), `Within()`, `summary.pseries()`, `print.summary.pseries()`, `as.matrix.pseries()`.

Examples

```
# First, create a pdata.frame
data("EmplUK", package = "plm")
Em <- pdata.frame(EmplUK)

# Then extract a series, which becomes additionally a pseries
z <- Em$output
class(z)
```

```
# compute the first and third lag, and the difference lagged twice
lag(z)
lag(z, 3L)
diff(z, 2L)

# compute negative lags (= leading values)
lag(z, -1L)
lead(z, 1L) # same as line above
identical(lead(z, 1L), lag(z, -1L)) # TRUE

# compute more than one lag and diff at once (matrix returned)
lag(z, c(1L,2L))
diff(z, c(1L,2L))

## demonstrate behaviour of shift = "time" vs. shift = "row"
# delete 2nd time period for first individual (1978 is missing (not NA)):
Em_hole <- Em[-2L, ]
is.pconsecutive(Em_hole) # check: non-consecutive for 1st individual now

# original non-consecutive data:
head(Em_hole$emp, 10)
# for shift = "time", 1-1979 contains the value of former 1-1977 (2 periods lagged):
head(lag(Em_hole$emp, k = 2L, shift = "time"), 10L)
# for shift = "row", 1-1979 contains NA (2 rows lagged (and no entry for 1976):
head(lag(Em_hole$emp, k = 2L, shift = "row"), 10L)
```

`make.dummies` *Create a Dummy Matrix*

Description

Contrast-coded dummy matrix (treatment coding) created from a factor

Usage

```
make.dummies(x, ...)

## Default S3 method:
make.dummies(x, base = 1L, base.add = TRUE, ...)

## S3 method for class 'data.frame'
make.dummies(x, col, base = 1L, base.add = TRUE, ...)

## S3 method for class 'pdata.frame'
make.dummies(x, col, base = 1L, base.add = TRUE, ...)
```

Arguments

x	a factor from which the dummies are created (x is coerced to factor if not yet a factor) for the default method or a data data frame/pdata.frame for the respective method.
...	further arguments.
base	integer or character, specifies the reference level (base), if integer it refers to position in levels(x), if character the name of a level,
base.add	logical, if TRUE the reference level (base) is added to the return value as first column, if FALSE the reference level is not included.
col	character (only for the data frame and pdata.frame methods), to specify the column which is used to derive the dummies from,

Details

This function creates a matrix of dummies from the levels of a factor in treatment coding. In model estimations, it is usually preferable to not create the dummy matrix prior to estimation but to simply specify a factor in the formula and let the estimation function handle the creation of the dummies.

This function is merely a convenience wrapper around stats::contr.treatment to ease the dummy matrix creation process shall the dummy matrix be explicitly required. See Examples for a use case in LSDV (least squares dummy variable) model estimation.

The default method uses a factor as main input (or something coercible to a factor) to derive the dummy matrix from. Methods for data frame and pdata.frame are available as well and have the additional argument col to specify the the column from which the dummies are created; both methods merge the dummy matrix to the data frame/pdata.frame yielding a ready-to-use data set. See also Examples for use cases.

Value

For the default method, a matrix containing the contrast-coded dummies (treatment coding), dimensions are n x n where n = length(levels(x)) if argument base.add = TRUE or n = length(levels(x)-1) if base.add = FALSE; for the data frame and pdata.frame method, a data frame or pdata.frame, respectively, with the dummies appropriately merged to the input as last columns (column names are derived from the name of the column used to create the dummies and its levels).

Author(s)

Kevin Tappe

See Also

stats::contr.treatment(), stats::contrasts()

Examples

```
library(plm)
data("Grunfeld", package = "plm")
Grunfeld <- Grunfeld[1:100, ] # reduce data set (down to 5 firms)
```

```
## default method
make.dummies(Grunfeld$firm) # gives 5 x 5 matrix (5 firms, base level incl.)
make.dummies(Grunfeld$firm, base = 2L, base.add = FALSE) # gives 5 x 4 matrix

## data frame method
Grun.dummies <- make.dummies(Grunfeld, col = "firm")

## pdata.frame method
pGrun <- pdata.frame(Grunfeld)
pGrun.dummies <- make.dummies(pGrun, col = "firm")

## Model estimation:
## estimate within model (individual/firm effects) and LSDV models (firm dummies)
# within model:
plm(inv ~ value + capital, data = pGrun, model = "within")

## LSDV with user-created dummies by make.dummies:
form_dummies <- paste0("firm", c(1:5), collapse = "+")
form_dummies <- formula(paste0("inv ~ value + capital + ", form_dummies))
plm(form_dummies, data = pGrun.dummies, model = "pooling") # last dummy is dropped

# LSDV via factor(year) -> let estimation function generate dummies:
plm(inv ~ value + capital + factor(firm), data = pGrun, model = "pooling")
```

`make.pbalanced` *Make data balanced*

Description

This function makes the data balanced, i.e., each individual has the same time periods, by filling in or dropping observations

Usage

```
make.pbalanced(
  x,
  balance.type = c("fill", "shared.times", "shared.individuals"),
  ...
)

## S3 method for class 'pdata.frame'
make.pbalanced(
  x,
  balance.type = c("fill", "shared.times", "shared.individuals"),
  ...
)

## S3 method for class 'pseries'
```

```
make.pbalanced(
  x,
  balance.type = c("fill", "shared.times", "shared.individuals"),
  ...
)

## S3 method for class 'data.frame'
make.pbalanced(
  x,
  balance.type = c("fill", "shared.times", "shared.individuals"),
  index = NULL,
  ...
)
```

Arguments

x	an object of class pdata.frame, data.frame, or pseries;
balance.type	character, one of "fill", "shared.times", or "shared.individuals", see **Details**,
...	further arguments.
index	only relevant for data.frame interface; if NULL, the first two columns of the data.frame are assumed to be the index variables; if not NULL, both dimensions ('individual', 'time') need to be specified by index as character of length 2 for data frames, for further details see pdata.frame(),

Details

(p)data.frame and pseries objects are made balanced, meaning each individual has the same time periods. Depending on the value of balance.type, the balancing is done in different ways:

- balance.type = "fill" (default): The union of available time periods over all individuals is taken (w/o NA values). Missing time periods for an individual are identified and corresponding rows (elements for pseries) are inserted and filled with NA for the non–index variables (elements for a pseries). This means, only time periods present for at least one individual are inserted, if missing.
- balance.type = "shared.times": The intersect of available time periods over all individuals is taken (w/o NA values). Thus, time periods not available for all individuals are discarded, i. e., only time periods shared by all individuals are left in the result).
- balance.type = "shared.individuals": All available time periods are kept and those individuals are dropped for which not all time periods are available, i. e., only individuals shared by all time periods are left in the result (symmetric to "shared.times").

The data are not necessarily made consecutive (regular time series with distance 1), because balancedness does not imply consecutiveness. For making the data consecutive, use make.pconsecutive() (and, optionally, set argument balanced = TRUE to make consecutive and balanced, see also **Examples** for a comparison of the two functions.

Note: Rows of (p)data.frames (elements for pseries) with NA values in individual or time index are not examined but silently dropped before the data are made balanced. In this case, it cannot be

inferred which individual or time period is meant by the missing value(s) (see also **Examples**). Especially, this means: NA values in the first/last position of the original time periods for an individual are dropped, which are usually meant to depict the beginning and ending of the time series for that individual. Thus, one might want to check if there are any NA values in the index variables before applying make.pbalanced, and especially check for NA values in the first and last position for each individual in original data and, if so, maybe set those to some meaningful begin/end value for the time series.

Value

An object of the same class as the input x, i.e., a pdata.frame, data.frame or a pseries which is made balanced based on the index variables. The returned data are sorted as a stacked time series.

Author(s)

Kevin Tappe

See Also

is.pbalanced() to check if data are balanced; is.pconsecutive() to check if data are consecutive; make.pconsecutive() to make data consecutive (and, optionally, also balanced).
punbalancedness() for two measures of unbalancedness, pdim() to check the dimensions of a 'pdata.frame' (and other objects), pvar() to check for individual and time variation of a 'pdata.frame' (and other objects), lag() for lagging (and leading) values of a 'pseries' object.
pseries(), data.frame(), pdata.frame().

Examples

```
# take data and make it unbalanced
# by deletion of 2nd row (2nd time period for first individual)
data("Grunfeld", package = "plm")
nrow(Grunfeld)                              # 200 rows
Grunfeld_missing_period <- Grunfeld[-2, ]
pdim(Grunfeld_missing_period)$balanced    # check if balanced: FALSE
make.pbalanced(Grunfeld_missing_period)   # make it balanced (by filling)
make.pbalanced(Grunfeld_missing_period, balance.type = "shared.times") # (shared periods)
nrow(make.pbalanced(Grunfeld_missing_period))
nrow(make.pbalanced(Grunfeld_missing_period, balance.type = "shared.times"))

# more complex data:
# First, make data unbalanced (and non-consecutive)
# by deletion of 2nd time period (year 1936) for all individuals
# and more time periods for first individual only
Grunfeld_unbalanced <- Grunfeld[Grunfeld$year != 1936, ]
Grunfeld_unbalanced <- Grunfeld_unbalanced[-c(1,4), ]
pdim(Grunfeld_unbalanced)$balanced        # FALSE
all(is.pconsecutive(Grunfeld_unbalanced)) # FALSE

g_bal <- make.pbalanced(Grunfeld_unbalanced)
pdim(g_bal)$balanced        # TRUE
unique(g_bal$year)          # all years but 1936
```

```
nrow(g_bal)                # 190 rows
head(g_bal)                # 1st individual: years 1935, 1939 are NA

# NA in 1st, 3rd time period (years 1935, 1937) for first individual
Grunfeld_NA <- Grunfeld
Grunfeld_NA[c(1, 3), "year"] <- NA
g_bal_NA <- make.pbalanced(Grunfeld_NA)
head(g_bal_NA)        # years 1935, 1937: NA for non-index vars
nrow(g_bal_NA)        # 200

# pdata.frame interface
pGrunfeld_missing_period <- pdata.frame(Grunfeld_missing_period)
make.pbalanced(Grunfeld_missing_period)

# pseries interface
make.pbalanced(pGrunfeld_missing_period$inv)

# comparison to make.pconsecutive
g_consec <- make.pconsecutive(Grunfeld_unbalanced)
all(is.pconsecutive(g_consec)) # TRUE
pdim(g_consec)$balanced        # FALSE
head(g_consec, 22)             # 1st individual:   no years 1935/6; 1939 is NA;
                               # other indviduals: years 1935-1954, 1936 is NA
nrow(g_consec)                 # 198 rows

g_consec_bal <- make.pconsecutive(Grunfeld_unbalanced, balanced = TRUE)
all(is.pconsecutive(g_consec_bal)) # TRUE
pdim(g_consec_bal)$balanced        # TRUE
head(g_consec_bal)                 # year 1936 is NA for all individuals
nrow(g_consec_bal)                 # 200 rows

head(g_bal)                        # no year 1936 at all
nrow(g_bal)                        # 190 rows
```

`make.pconsecutive` *Make data consecutive (and, optionally, also balanced)*

Description

This function makes the data consecutive for each individual (no "gaps" in time dimension per individual) and, optionally, also balanced

Usage

```
make.pconsecutive(x, ...)

## S3 method for class 'data.frame'
make.pconsecutive(x, balanced = FALSE, index = NULL, ...)
```

```
## S3 method for class 'pdata.frame'
make.pconsecutive(x, balanced = FALSE, ...)

## S3 method for class 'pseries'
make.pconsecutive(x, balanced = FALSE, ...)
```

Arguments

x	an object of class pdata.frame, data.frame, or pseries,
...	further arguments.
balanced	logical, indicating whether the data should *additionally* be made balanced (default: FALSE),
index	only relevant for data.frame interface; if NULL, the first two columns of the data.frame are assumed to be the index variables; if not NULL, both dimensions ('individual', 'time') need to be specified by index as character of length 2 for data frames, for further details see pdata.frame(),

Details

(p)data.frame and pseries objects are made consecutive, meaning their time periods are made consecutive per individual. For consecutiveness, the time dimension is interpreted to be numeric, and the data are extended to a regularly spaced sequence with distance 1 between the time periods for each individual (for each individual the time dimension become a sequence t, t+1, t+2, ..., where t is an integer). Non–index variables are filled with NA for the inserted elements (rows for (p)data.frames, vector elements for pseries).

With argument balanced = TRUE, additionally to be made consecutive, the data also can be made a balanced panel/pseries. Note: This means consecutive AND balanced; balancedness does not imply consecutiveness. In the result, each individual will have the same time periods in their time dimension by taking the min and max of the time index variable over all individuals (w/o NA values) and inserting the missing time periods. Looking at the number of rows of the resulting (pdata.frame) (elements for pseries), this results in `nrow(make.pconsecutive(<.>, balanced = FALSE)) <= nrow(make.pconsecutive(<.>, balanced = TRUE))`. For making the data only balanced, i.e., not demanding consecutiveness at the same time, use make.pbalanced() (see **Examples** for a comparison)).

Note: rows of (p)data.frames (elements for pseries) with NA values in individual or time index are not examined but silently dropped before the data are made consecutive. In this case, it is not clear which individual or time period is meant by the missing value(s). Especially, this means: If there are NA values in the first/last position of the original time periods for an individual, which usually depicts the beginning and ending of the time series for that individual, the beginning/end of the resulting time series is taken to be the min and max (w/o NA values) of the original time series for that individual, see also **Examples**. Thus, one might want to check if there are any NA values in the index variables before applying make.pconsecutive, and especially check for NA values in the first and last position for each individual in original data and, if so, maybe set those to some meaningful begin/end value for the time series.

Value

An object of the same class as the input x, i.e., a pdata.frame, data.frame or a pseries which is made time–consecutive based on the index variables. The returned data are sorted as a stacked time series.

Author(s)

Kevin Tappe

See Also

is.pconsecutive() to check if data are consecutive; make.pbalanced() to make data only balanced (not consecutive).
punbalancedness() for two measures of unbalancedness, pdim() to check the dimensions of a 'pdata.frame' (and other objects), pvar() to check for individual and time variation of a 'pdata.frame' (and other objects), lag() for lagged (and leading) values of a 'pseries' object.
pseries(), data.frame(), pdata.frame().

Examples

```
# take data and make it non-consecutive
# by deletion of 2nd row (2nd time period for first individual)
data("Grunfeld", package = "plm")
nrow(Grunfeld)                              # 200 rows
Grunfeld_missing_period <- Grunfeld[-2, ]
is.pconsecutive(Grunfeld_missing_period)   # check for consecutiveness
make.pconsecutive(Grunfeld_missing_period) # make it consecutiveness


# argument balanced:
# First, make data non-consecutive and unbalanced
# by deletion of 2nd time period (year 1936) for all individuals
# and more time periods for first individual only
Grunfeld_unbalanced <- Grunfeld[Grunfeld$year != 1936, ]
Grunfeld_unbalanced <- Grunfeld_unbalanced[-c(1,4), ]
all(is.pconsecutive(Grunfeld_unbalanced)) # FALSE
pdim(Grunfeld_unbalanced)$balanced        # FALSE

g_consec_bal <- make.pconsecutive(Grunfeld_unbalanced, balanced = TRUE)
all(is.pconsecutive(g_consec_bal)) # TRUE
pdim(g_consec_bal)$balanced        # TRUE
nrow(g_consec_bal)                 # 200 rows
head(g_consec_bal)                 # 1st individual: years 1935, 1936, 1939 are NA

g_consec <- make.pconsecutive(Grunfeld_unbalanced) # default: balanced = FALSE
all(is.pconsecutive(g_consec)) # TRUE
pdim(g_consec)$balanced        # FALSE
nrow(g_consec)                 # 198 rows
head(g_consec)                 # 1st individual: years 1935, 1936 dropped, 1939 is NA


# NA in 1st, 3rd time period (years 1935, 1937) for first individual
```

```
Grunfeld_NA <- Grunfeld
Grunfeld_NA[c(1, 3), "year"] <- NA
g_NA <- make.pconsecutive(Grunfeld_NA)
head(g_NA)        # 1936 is begin for 1st individual, 1937: NA for non-index vars
nrow(g_NA)        # 199, year 1935 from original data is dropped


# pdata.frame interface
pGrunfeld_missing_period <- pdata.frame(Grunfeld_missing_period)
make.pconsecutive(Grunfeld_missing_period)


# pseries interface
make.pconsecutive(pGrunfeld_missing_period$inv)


# comparison to make.pbalanced (makes the data only balanced, not consecutive)
g_bal <- make.pbalanced(Grunfeld_unbalanced)
all(is.pconsecutive(g_bal)) # FALSE
pdim(g_bal)$balanced        # TRUE
nrow(g_bal) # 190 rows
```

`Males` *Wages and Education of Young Males*

Description

A panel of 545 observations from 1980 to 1987

Format

A data frame containing :

nr identifier

year year

school years of schooling

exper years of experience (computed as `age-6-school`)

union wage set by collective bargaining?

ethn a factor with levels `black, hisp, other`

married married?

health health problem?

wage log of hourly wage

industry a factor with 12 levels

occupation a factor with 9 levels

residence a factor with levels `rural_area, north_east, northern_central, south`

Details

total number of observations : 4360

observation : individuals

country : United States

Source

Journal of Applied Econometrics data archive http://qed.econ.queensu.ca/jae/1998-v13.2/vella-verbeek/.

References

Vella F, Verbeek M (1998). "Whose wages do unions raise? A dynamic model of unionism and wage rate determination for young men." *Journal of Applied Econometrics*, **13**, 163–183.

Verbeek M (2004). *A Guide to Modern Econometrics*, 2nd edition. Wiley.

`model.frame.pdata.frame`

model.frame and model.matrix for panel data

Description

Methods to create model frame and model matrix for panel data.

Usage

```
## S3 method for class 'pdata.frame'
model.frame(
  formula,
  data = NULL,
  ...,
  lhs = NULL,
  rhs = NULL,
  dot = "previous"
)

## S3 method for class 'pdata.frame'
formula(x, ...)

## S3 method for class 'plm'
model.matrix(object, ...)

## S3 method for class 'pdata.frame'
model.matrix(
  object,
 model = c("pooling", "within", "Between", "Sum", "between", "mean", "random", "fd"),
```

```
  effect = c("individual", "time", "twoways", "nested"),
  rhs = 1,
  theta = NULL,
  cstcovar.rm = NULL,
  ...
)
```

Arguments

data	a formula, see **Details**,
...	further arguments.
lhs	inherited from package Formula::Formula() (see there),
rhs	inherited from package Formula::Formula() (see there),
dot	inherited from package Formula::Formula() (see there),
x	a model.frame
object, formula	an object of class "pdata.frame" or an estimated model object of class "plm",
model	one of "pooling", "within", "Sum", "Between", "between", "random", "fd" and "ht",
effect	the effects introduced in the model, one of "individual", "time", "twoways" or "nested",
theta	the parameter for the transformation if model = "random",
cstcovar.rm	remove the constant columns, one of "none", "intercept", "covariates", "all"),

Details

The lhs and rhs arguments are inherited from Formula, see there for more details.
The model.frame methods return a pdata.frame object suitable as an input to plm's model.matrix.
The model.matrix methods builds a model matrix with transformations performed as specified by the model and effect arguments (and theta if model = "random" is requested), in this case the supplied data argument should be a model frame created by plm's model.frame method. If not, it is tried to construct the model frame from the data. Constructing the model frame first ensures proper NA handling, see **Examples**.

Value

The model.frame methods return a pdata.frame.
The model.matrix methods return a matrix.

Author(s)

Yves Croissant

See Also

pmodel.response() for (transformed) response variable.
Formula::Formula() from package Formula, especially for the lhs and rhs arguments.

Examples

```
# First, make a pdata.frame
data("Grunfeld", package = "plm")
pGrunfeld <- pdata.frame(Grunfeld)

# then make a model frame from a formula and a pdata.frame
form <- inv ~ value
mf <- model.frame(pGrunfeld, form)

# then construct the (transformed) model matrix (design matrix)
# from model frame
modmat <- model.matrix(mf, model = "within")

## retrieve model frame and model matrix from an estimated plm object
fe_model <- plm(form, data = pGrunfeld, model = "within")
model.frame(fe_model)
model.matrix(fe_model)

# same as constructed before
all.equal(mf, model.frame(fe_model), check.attributes = FALSE) # TRUE
all.equal(modmat, model.matrix(fe_model), check.attributes = FALSE) # TRUE
```

---

mtest *Arellano–Bond Test of Serial Correlation*

Description

Test of serial correlation for models estimated by GMM

Usage

```
mtest(object, ...)

## S3 method for class 'pgmm'
mtest(object, order = 1L, vcov = NULL, ...)
```

Arguments

object	an object of class "pgmm",
...	further arguments (currently unused).
order	integer: the order of the serial correlation,
vcov	a matrix of covariance for the coefficients or a function to compute it,

Details

The Arellano–Bond test is a test of correlation based on the residuals of the estimation. By default, the computation is done with the standard covariance matrix of the coefficients. A robust estimator of a covariance matrix can be supplied with the `vcov` argument.

Note that `mtest` computes like DPD for Ox and xtabond do, i.e., uses for two-steps models the one-step model's residuals which were used to construct the efficient two-steps estimator, see (Arellano and Bond 2012), p. 32, footnote 7; As noted by (Arellano and Bond 1991) (p. 282), the m statistic is rather flexible and can be defined with any consistent GMM estimator which gives leeway for implementation, but the test's asymptotic power depends on the estimator's efficiency. (Arellano and Bond 1991) (see their footnote 9) used DPD98 for Gauss ((Arellano and Bond 1998)) as did (Windmeijer 2005) (see footnote 10) for the basis of his covariance correction, both with a slightly different implementation. Hence some results for `mtest` with two-step models diverge from original papers, see examples below.

Value

An object of class `"htest"`.

Author(s)

Yves Croissant

References

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

Arellano M, Bond S (1998). "Dynamic panel data estimation using DPD98 for GAUSS: a guide for users." unpublished, https://ifs.org.uk/publications/dpd-gauss.

Arellano M, Bond S (2012). "Panel data estimation using DPD for Ox." unpublished, https://www.doornik.com/download/oxmetrics7/Ox_Packages/dpd.pdf.

Windmeijer F (2005). "A Finite Sample Correction for the Variance of Linear Efficient Two–Steps GMM Estimators." *Journal of Econometrics*, **126**, 25–51.

See Also

`pgmm()`, `vcovHC.pgmm()`

Examples

```
data("EmplUK", package = "plm")
# Arellano/Bond 1991, Table 4, column (a1)
ab.a1 <- pgmm(log(emp) ~ lag(log(emp), 1:2) + lag(log(wage), 0:1)
              + lag(log(capital), 0:2) + lag(log(output), 0:2) | lag(log(emp), 2:99),
              data = EmplUK, effect = "twoways", model = "onestep")
mtest(ab.a1, 1L)
mtest(ab.a1, 2L, vcov = vcovHC)
```

```
# Windmeijer (2005), table 2, onestep with corrected std. err
ab.b.onestep <- pgmm(log(emp) ~ lag(log(emp), 1:2) + lag(log(wage), 0:1)
                     + log(capital) + lag(log(output), 0:1) | lag(log(emp), 2:99),
                     data = EmplUK, effect = "twoways", model = "onestep")
mtest(ab.b.onestep, 1L, vcov = vcovHC)
mtest(ab.b.onestep, 2L, vcov = vcovHC)

# Arellano/Bond 1991, Table 4, column (a2)
ab.a2 <- pgmm(log(emp) ~ lag(log(emp), 1:2) + lag(log(wage), 0:1)
             + lag(log(capital), 0:2) + lag(log(output), 0:2) | lag(log(emp), 2:99),
             data = EmplUK, effect = "twoways", model = "twosteps")
mtest(ab.a2, 1L)
mtest(ab.a2, 2L) # while a la Arellano/Bond (1991) -0.434
```

`nobs.plm` *Extract Total Number of Observations Used in Estimated Panelmodel*

Description

This function extracts the total number of 'observations' from a fitted panel model.

Usage

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

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

Arguments

`object`	a `panelmodel` object for which the number of total observations is to be extracted,
`...`	further arguments.

Details

The number of observations is usually the length of the residuals vector. Thus, `nobs` gives the number of observations actually used by the estimation procedure. It is not necessarily the number of observations of the model frame (number of rows in the model frame), because sometimes the model frame is further reduced by the estimation procedure. This is, e.g., the case for first–difference models estimated by `plm(..., model = "fd")` where the model frame does not yet contain the differences (see also **Examples**).

Value

A single number, normally an integer.

See Also

pdim()

Examples

```
# estimate a panelmodel
data("Produc", package = "plm")
z <- plm(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp,data=Produc,
         model="random", subset = gsp > 5000)

nobs(z)       # total observations used in estimation
pdim(z)$nT$N  # same information
pdim(z)      # more information about the dimensions (no. of individuals and time periods)

# illustrate difference between nobs and pdim for first-difference model
data("Grunfeld", package = "plm")
fdmod <- plm(inv ~ value + capital, data = Grunfeld, model = "fd")
nobs(fdmod)      # 190
pdim(fdmod)$nT$N # 200
```

Parity *Purchasing Power Parity and other parity relationships*

Description

A panel of 104 quarterly observations from 1973Q1 to 1998Q4

Format

A data frame containing :

country country codes: a factor with 17 levels

time the quarter index, 1973Q1-1998Q4

ls log spot exchange rate vs. USD

lp log price level

is short term interest rate

il long term interest rate

ld log price differential vs. USA

uis U.S. short term interest rate

uil U.S. long term interest rate

Details

total number of observations : 1768

observation : country

country : OECD

Source

Coakley J, Fuertes A, Smith R (2006). "Unobserved heterogeneity in panel time series models." *Computational Statistics & Data Analysis*, **50**(9), 2361–2380.

References

Coakley J, Fuertes A, Smith R (2006). "Unobserved heterogeneity in panel time series models." *Computational Statistics & Data Analysis*, **50**(9), 2361–2380.

Driscoll JC, Kraay AC (1998). "Consistent covariance matrix estimation with spatially dependent panel data." *Review of economics and statistics*, **80**(4), 549–560.

`pbgtest` *Breusch–Godfrey Test for Panel Models*

Description

Test of serial correlation for (the idiosyncratic component of) the errors in panel models.

Usage

```
pbgtest(x, ...)

## S3 method for class 'panelmodel'
pbgtest(x, order = NULL, type = c("Chisq", "F"), ...)

## S3 method for class 'formula'
pbgtest(
  x,
  order = NULL,
  type = c("Chisq", "F"),
  data,
  model = c("pooling", "random", "within"),
  ...
)
```

Arguments

`x`	an object of class `"panelmodel"` or of class `"formula"`,
`...`	further arguments (see `lmtest::bgtest()`).
`order`	an integer indicating the order of serial correlation to be tested for. `NULL` (default) uses the minimum number of observations over the time dimension (see also section **Details** below),
`type`	type of test statistic to be calculated; either `"Chisq"` (default) for the Chi-squared test statistic or `"F"` for the F test statistic,
`data`	only relevant for formula interface: data set for which the respective panel model (see `model`) is to be evaluated,

`model` only relevant for formula interface: compute test statistic for model `pooling` (default), `random`, or `within`. When `model` is used, the `data` argument needs to be passed as well,

Details

This Lagrange multiplier test uses the auxiliary model on (quasi-)demeaned data taken from a model of class `plm` which may be a `pooling` (default for formula interface), `random` or `within` model. It performs a Breusch–Godfrey test (using `bgtest` from package **lmtest** on the residuals of the (quasi-)demeaned model, which should be serially uncorrelated under the null of no serial correlation in idiosyncratic errors, as illustrated in Wooldridge (2010). The function takes the demeaned data, estimates the model and calls `bgtest`.

Unlike most other tests for serial correlation in panels, this one allows to choose the order of correlation to test for.

Value

An object of class `"htest"`.

Note

The argument `order` defaults to the minimum number of observations over the time dimension, while for `lmtest::bgtest` it defaults to 1.

Author(s)

Giovanni Millo

References

Breusch TS (1978). "Testing for autocorrelation in dynamic linear models." *Australian Economic Papers*, **17**(31), 334–355.

Godfrey LG (1978). "Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables." *Econometrica*, **46**(6), 1293–1301.

Wooldridge JM (2002). *Econometric Analysis of Cross–Section and Panel Data*. MIT Press.

Wooldridge JM (2010). *Econometric Analysis of Cross–Section and Panel Data*, 2nd edition. MIT Press.

Wooldridge JM (2013). *Introductory Econometrics: a modern approach*, 5th edition. South-Western (Cengage Learning). Sec. 12.2, pp. 421–422.

See Also

For the original test in package **lmtest** see `lmtest::bgtest()`. See `pdwtest()` for the analogous panel Durbin–Watson test. See `pbltest()`, `pbsytest()`, `pwartest()` and `pwfdtest()` for other serial correlation tests for panel models.

Examples

```
data("Grunfeld", package = "plm")
g <- plm(inv ~ value + capital, data = Grunfeld, model = "random")

# panelmodel interface
pbgtest(g)
pbgtest(g, order = 4)

# formula interface
pbgtest(inv ~ value + capital, data = Grunfeld, model = "random")

# F test statistic (instead of default type="Chisq")
pbgtest(g, type="F")
pbgtest(inv ~ value + capital, data = Grunfeld, model = "random", type = "F")
```

---

pbltest *Baltagi and Li Serial Dependence Test For Random Effects Models*

Description

Baltagi and Li (1995)'s Lagrange multiplier test for AR(1) or MA(1) idiosyncratic errors in panel models with random effects.

Usage

```
pbltest(x, ...)

## S3 method for class 'formula'
pbltest(x, data, alternative = c("twosided", "onesided"), index = NULL, ...)

## S3 method for class 'plm'
pbltest(x, alternative = c("twosided", "onesided"), ...)
```

Arguments

x	a model formula or an estimated random–effects model of class plm ,
...	further arguments.
data	for the formula interface only: a data.frame,
alternative	one of "twosided", "onesided". Selects either $H_A : \rho \neq 0$ or $H_A : \rho = 0$ (i.e., the Normal or the Chi-squared version of the test),
index	the index of the data.frame,

Details

This is a Lagrange multiplier test for the null of no serial correlation, against the alternative of either an AR(1) or a MA(1) process, in the idiosyncratic component of the error term in a random effects panel model (as the analytical expression of the test turns out to be the same under both alternatives, (see Baltagi and Li 1995 and Baltagi and Li 1997). The `alternative` argument, defaulting to `twosided`, allows testing for positive serial correlation only, if set to `onesided`.

Value

An object of class `"htest"`.

Author(s)

Giovanni Millo

References

Baltagi B, Li Q (1995). "Testing AR(1) Against MA(1) Disturbances in an Error Component Model." *Journal of Econometrics*, **68**, 133–151.

Baltagi B, Li Q (1997). "Monte Carlo Results on Pure and Pretest Estimators of an Error Components Model With Autocorrelated Disturbances." *Annales d'Economie et de Statistique*, **48**, 69–82.

See Also

`pdwtest()`, `pbnftest()`, `pbgtest()`, `pbsytest()`, `pwartest()` and `pwfdtest()` for other serial correlation tests for panel models.

Examples

```
data("Grunfeld", package = "plm")

# formula interface
pbltest(inv ~ value + capital, data = Grunfeld)

# plm interface
re_mod <- plm(inv ~ value + capital, data = Grunfeld, model = "random")
pbltest(re_mod)
pbltest(re_mod, alternative = "onesided")
```

---

pbnftest *Modified BNF–Durbin–Watson Test and Baltagi–Wu's LBI Test for Panel Models*

Description

Tests for AR(1) disturbances in panel models.

Usage

```
pbnftest(x, ...)

## S3 method for class 'panelmodel'
pbnftest(x, test = c("bnf", "lbi"), ...)

## S3 method for class 'formula'
pbnftest(
  x,
  data,
  test = c("bnf", "lbi"),
  model = c("pooling", "within", "random"),
  ...
)
```

Arguments

x	an object of class "panelmodel" or of class "formula",
...	only relevant for formula interface: further arguments to specify the model to test (arguments passed on to plm()), e.g., effect.
test	a character indicating the test to be performed, either "bnf" or "lbi" for the (modified) BNF statistic or Baltagi–Wu's LBI statistic, respectively,
data	a data.frame (only relevant for formula interface),
model	a character indicating on which type of model the test shall be performed ("pooling", "within", "random", only relevant for formula interface),

Details

The default, test = "bnf", gives the (modified) BNF statistic, the generalised Durbin-Watson statistic for panels. For balanced and consecutive panels, the reference is Bhargava/Franzini/Narendranathan (1982). The modified BNF is given for unbalanced and/or non-consecutive panels (d1 in formula 16 of Baltagi and Wu (1999)).

test = "lbi" yields Baltagi–Wu's LBI statistic (Baltagi and Wu 1999), the locally best invariant test which is based on the modified BNF statistic.

No specific variants of these tests are available for random effect models. As the within estimator is consistent also under the random effects assumptions, the test for random effect models is performed by taking the within residuals.

No p-values are given for the statistics as their distribution is quite difficult. Bhargava et al. (1982) supply tabulated bounds for p = 0.05 for the balanced case and consecutive case.

For large N, (Bhargava et al. 1982) suggest it is sufficient to check whether the BNF statistic is < 2 to test against positive serial correlation.

Value

An object of class "htest".

Author(s)

Kevin Tappe

References

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Baltagi BH, Wu PX (1999). "Unequally Spaced Panel Data Regressions with AR(1) Disturbances." *Econometric Theory*, **15**(6), 814–823. ISSN 02664666, 14694360.

Bhargava A, Franzini L, Narendranathan W (1982). "Serial Correlation and the Fixed Effects Model." *The Review of Economic Studies*, **49**(4), 533–549. ISSN 00346527, 1467937X.

See Also

`pdwtest()` for the original Durbin–Watson test using (quasi-)demeaned residuals of the panel model without taking the panel structure into account. `pbltest()`, `pbsytest()`, `pwartest()` and `pwfdtest()` for other serial correlation tests for panel models.

Examples

```
data("Grunfeld", package = "plm")

# formula interface, replicate Baltagi/Wu (1999), table 1, test case A:
data_A <- Grunfeld[!Grunfeld[["year"]] %in% c("1943", "1944"), ]
pbnftest(inv ~ value + capital, data = data_A, model = "within")
pbnftest(inv ~ value + capital, data = data_A, test = "lbi", model = "within")

# replicate Baltagi (2013), p. 101, table 5.1:
re <- plm(inv ~ value + capital, data = Grunfeld, model = "random")
pbnftest(re)
pbnftest(re, test = "lbi")
```

`pbsytest` *Bera, Sosa-Escudero and Yoon Locally–Robust Lagrange Multiplier Tests for Panel Models and Joint Test by Baltagi and Li*

Description

Test for residual serial correlation (or individual random effects) locally robust vs. individual random effects (serial correlation) for panel models and joint test of serial correlation and the random effect specification by Baltagi and Li.

Usage

```
pbsytest(x, ...)

## S3 method for class 'formula'
pbsytest(
  x,
  data,
  ...,
  test = c("ar", "re", "j"),
  re.normal = if (test == "re") TRUE else NULL
)

## S3 method for class 'panelmodel'
pbsytest(
  x,
  test = c("ar", "re", "j"),
  re.normal = if (test == "re") TRUE else NULL,
  ...
)
```

Arguments

x	an object of class formula or of class panelmodel,
...	further arguments.
data	a data.frame,
test	a character string indicating which test to perform: first–order serial correlation ("ar"), random effects ("re") or joint test for either of them ("j"),
re.normal	logical, only relevant for test = "re": TRUE (default) computes the one-sided "re" test, FALSE the two-sided test (see also Details); not relevant for other values of test and, thus, should be NULL,

Details

These Lagrange multiplier tests are robust vs. local misspecification of the alternative hypothesis, i.e., they test the null of serially uncorrelated residuals against AR(1) residuals in a pooling model, allowing for local departures from the assumption of no random effects; or they test the null of no random effects allowing for local departures from the assumption of no serial correlation in residuals. They use only the residuals of the pooled OLS model and correct for local misspecification as outlined in Bera et al. (2001).

For test = "re", the default (re.normal = TRUE) is to compute a one-sided test which is expected to lead to a more powerful test (asymptotically N(0,1) distributed). Setting re.normal = FALSE gives the two-sided test (asymptotically chi-squared(2) distributed). Argument re.normal is irrelevant for all other values of test.

The joint test of serial correlation and the random effect specification (test = "j") is due to Baltagi and Li (1991) (also mentioned in Baltagi and Li (1995), pp. 135–136) and is added for convenience under this same function.

The unbalanced version of all tests are derived in Sosa-Escudero and Bera (2008). The functions implemented are suitable for balanced as well as unbalanced panel data sets.

A concise treatment of the statistics for only balanced panels is given in Baltagi (2013), p. 108.

Here is an overview of how the various values of the `test` argument relate to the literature:

- `test = "ar"`:
  - $RS*_{\rho}$ in Bera et al. (2001), p. 9 (balanced)
  - $LM*_{\rho}$ in Baltagi (2013), p. 108 (balanced)
  - $RS*_{\lambda}$ in Sosa-Escudero/Bera (2008), p. 73 (unbalanced)
- `test = "re", re.normal = TRUE` (default) (one-sided test, asymptotically N(0,1) distributed):
  - $RSO*_{\mu}$ in Bera et al. (2001), p. 11 (balanced)
  - $RSO*_{\mu}$ in Sosa-Escudero/Bera (2008), p. 75 (unbalanced)
- `test = "re", re.normal = FALSE` (two-sided test, asymptotically chi-squared(2) distributed):
  - $RS*_{\mu}$ in Bera et al. (2001), p. 7 (balanced)
  - $LM*_{\mu}$ in Baltagi (2013), p. 108 (balanced)
  - $RS*_{\mu}$ in Sosa-Escudero/Bera (2008), p. 73 (unbalanced)
- `test = "j"`:
  - $RS_{\mu\rho}$ in Bera et al. (2001), p. 10 (balanced)
  - LM in Baltagi/Li (2001), p. 279 (balanced)
  - LM_1 in Baltagi and Li (1995), pp. 135–136 (balanced)
  - $LM1$ in Baltagi (2013), p. 108 (balanced)
  - $RS_{\lambda\rho}$ in Sosa-Escudero/Bera (2008), p. 74 (unbalanced)

Value

An object of class `"htest"`.

Author(s)

Giovanni Millo (initial implementation) & Kevin Tappe (extension to unbalanced panels)

References

Bera AK, Sosa–Escudero W, Yoon M (2001). "Tests for the Error Component Model in the Presence of Local Misspecification." *Journal of Econometrics*, **101**, 1–23.

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Baltagi B, Li Q (1991). "A Joint Test for Serial Correlation and Random Individual Effects." *Statistics and Probability Letters*, **11**, 277–280.

Baltagi B, Li Q (1995). "Testing AR(1) Against MA(1) Disturbances in an Error Component Model." *Journal of Econometrics*, **68**, 133–151.

Sosa-Escudero W, Bera AK (2008). "Tests for Unbalanced Error-Components Models under Local Misspecification." *The Stata Journal*, **8**(1), 68-78. doi:10.1177/1536867X0800800105, https://doi.org/10.1177/1536867X080

See Also

plmtest() for individual and/or time random effects tests based on a correctly specified model; pbltest(), pbgtest() and pdwtest() for serial correlation tests in random effects models.

Examples

```
## Bera et. al (2001), p. 13, table 1 use
## a subset of the original Grunfeld
## data which contains three errors -> construct this subset:
data("Grunfeld", package = "plm")
Grunsubset <- rbind(Grunfeld[1:80, ], Grunfeld[141:160, ])
Grunsubset[Grunsubset$firm == 2 & Grunsubset$year %in% c(1940, 1952), ][["inv"]] <- c(261.6, 645.2)
Grunsubset[Grunsubset$firm == 2 & Grunsubset$year == 1946, ][["capital"]] <- 232.6

## default is AR testing (formula interface)
pbsytest(inv ~ value + capital, data = Grunsubset, index = c("firm", "year"))
pbsytest(inv ~ value + capital, data = Grunsubset, index = c("firm", "year"), test = "re")
pbsytest(inv ~ value + capital, data = Grunsubset, index = c("firm", "year"),
  test = "re", re.normal = FALSE)
pbsytest(inv ~ value + capital, data = Grunsubset, index = c("firm", "year"), test = "j")

## plm interface
mod <- plm(inv ~ value + capital, data = Grunsubset, model = "pooling")
pbsytest(mod)
```

---

pcce *Common Correlated Effects estimators*

Description

Common Correlated Effects Mean Groups (CCEMG) and Pooled (CCEP) estimators for panel data with common factors (balanced or unbalanced)

Usage

```
pcce(
  formula,
  data,
  subset,
  na.action,
  model = c("mg", "p"),
  index = NULL,
  trend = FALSE,
  ...
)

## S3 method for class 'pcce'
```

```
summary(object, vcov = NULL, ...)

## S3 method for class 'summary.pcce'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)

## S3 method for class 'pcce'
residuals(object, type = c("defactored", "standard"), ...)

## S3 method for class 'pcce'
model.matrix(object, ...)

## S3 method for class 'pcce'
pmodel.response(object, ...)
```

Arguments

formula	a symbolic description of the model to be estimated,
data	a data.frame,
subset	see lm,
na.action	see lm,
model	one of "mg", "p", selects Mean Groups vs. Pooled CCE model,
index	the indexes, see pdata.frame(),
trend	logical specifying whether an individual-specific trend has to be included,
...	further arguments.
object, x	an object of class "pcce",
vcov	a variance-covariance matrix furnished by the user or a function to calculate one,
digits	digits,
width	the maximum length of the lines in the print output,
type	one of "defactored" or "standard",

Details

pcce is a function for the estimation of linear panel models by the Common Correlated Effects Mean Groups or Pooled estimator, consistent under the hypothesis of unobserved common factors and idiosyncratic factor loadings. The CCE estimator works by augmenting the model by cross-sectional averages of the dependent variable and regressors in order to account for the common factors, and adding individual intercepts and possibly trends.

Value

An object of class `c("pcce", "panelmodel")` containing:

`coefficients`	the vector of coefficients,
`residuals`	the vector of (defactored) residuals,
`stdres`	the vector of (raw) residuals,
`tr.model`	the transformed data after projection on H,
`fitted.values`	the vector of fitted values,
`vcov`	the covariance matrix of the coefficients,
`df.residual`	degrees of freedom of the residuals,
`model`	a data.frame containing the variables used for the estimation,
`call`	the call,
`indcoef`	the matrix of individual coefficients from separate time series regressions,
`r.squared`	numeric, the R squared.

Author(s)

Giovanni Millo

References

Kapetanios G, Pesaran MH, Yamagata T (2011). "Panels with non-stationary multifactor error structures." *Journal of Econometrics*, **160**(2), 326–348.

Holly S, Pesaran MH, Yamagata T (2010). "A spatio-temporal model of house prices in the USA." *Journal of Econometrics*, **158**(1), 160–173.

Examples

```
data("Produc", package = "plm")
ccepmod <- pcce(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc, model="p")
summary(ccepmod)
summary(ccepmod, vcov = vcovHC) # use argument vcov for robust std. errors

ccemgmod <- pcce(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc, model="mg")
summary(ccemgmod)
```

pcdtest *Tests of cross-section dependence for panel models*

Description

Pesaran's CD or Breusch–Pagan's LM (local or global) tests for cross sectional dependence in panel models

Usage

```
pcdtest(x, ...)

## S3 method for class 'formula'
pcdtest(
  x,
  data,
  index = NULL,
  model = NULL,
  test = c("cd", "sclm", "bcsclm", "lm", "rho", "absrho"),
  w = NULL,
  ...
)

## S3 method for class 'panelmodel'
pcdtest(
  x,
  test = c("cd", "sclm", "bcsclm", "lm", "rho", "absrho"),
  w = NULL,
  ...
)

## S3 method for class 'pseries'
pcdtest(
  x,
  test = c("cd", "sclm", "bcsclm", "lm", "rho", "absrho"),
  w = NULL,
  ...
)
```

Arguments

x	an object of class `formula`, `panelmodel`, or `pseries` (depending on the respective interface) describing the model to be tested,
...	further arguments to be passed on for model estimation to `plm`, such as `effect` or `random.method`.
data	a `data.frame`,

index — an optional numerical index, if NULL, the first two columns of the data.frame provided in argument data are assumed to be the index variables; for further details see pdata.frame(),

model — an optional character string indicating which type of model to estimate; if left to NULL, the original heterogeneous specification of Pesaran is used,

test — the type of test statistic to be returned. One of

- "cd" for Pesaran's CD statistic,
- "lm" for Breusch and Pagan's original LM statistic,
- "sclm" for the scaled version of Breusch and Pagan's LM statistic,
- "bcsclm" for the bias-corrected scaled version of Breusch and Pagan's LM statistic,
- "rho" for the average correlation coefficient,
- "absrho" for the average absolute correlation coefficient,

w — either NULL (default) for the global tests or – for the local versions of the statistics – a n x n matrix describing proximity between individuals, with $w_ij = a$ where a is any number such that as.logical(a)==TRUE, if i, j are neighbours, 0 or any number b such that as.logical(b)==FALSE elsewhere. Only the lower triangular part (without diagonal) of w after coercing by as.logical() is evaluated for neighbouring information (but w can be symmetric). See also **Details** and **Examples**,

Details

These tests are originally meant to use the residuals of separate estimation of one time–series regression for each cross-sectional unit in order to check for cross–sectional dependence (model = NULL). If a different model specification (model = "within", "random", ...) is assumed consistent, one can resort to its residuals for testing (which is common, e.g., when the time dimension's length is insufficient for estimating the heterogeneous model).

If the time dimension is insufficient and model = NULL, the function defaults to estimation of a within model and issues a warning. The main argument of this function may be either a model of class panelmodel or a formula and data frame; in the second case, unless model is set to NULL, all usual parameters relative to the estimation of a plm model may be passed on. The test is compatible with any consistent panelmodel for the data at hand, with any specification of effect (except for test = "bcsclm" which requires a within model with either individual or two-ways effect). E.g., specifying effect = "time" or effect = "twoways" allows to test for residual cross-sectional dependence after the introduction of time fixed effects to account for common shocks.

A **local** version of either test can be computed by supplying a proximity matrix (elements coercible to logical) with argument w which provides information on whether any pair of individuals are neighbours or not. If w is supplied, only neighbouring pairs will be used in computing the test; else, w will default to NULL and all observations will be used. The matrix need not be binary, so commonly used "row–standardized" matrices can be employed as well. nb objects from **spdep** must instead be transformed into matrices by **spdep**'s function nb2mat before using.

The methods implemented are suitable also for unbalanced panels.

Pesaran's CD test (test="cd"), Breusch and Pagan's LM test (test="lm"), and its scaled version (test="sclm") are all described in Pesaran (2004) (and complemented by Pesaran (2005)). The

bias-corrected scaled test (`test="bcsclm"`) is due to (Baltagi et al. 2012) and only valid for within models including the individual effect (it's unbalanced version uses max(Tij) for T) in the bias-correction term). Breusch and Pagan (1980) is the original source for the LM test.

The test on a `pseries` is the same as a test on a pooled regression model of that variable on a constant, i.e., `pcdtest(some_pseries)` is equivalent to `pcdtest(plm(some_var ~ 1, data = some_pdata.frame, model = "`
and also equivalent to `pcdtest(some_var ~ 1, data = some_data)`, where `some_var` is the variable name in the data which corresponds to `some_pseries`.

Value

An object of class `"htest"`.

References

Baltagi BH, Feng Q, Kao C (2012). "A Lagrange Multiplier test for cross-sectional dependence in a fixed effects panel data model." *Journal of Econometrics*, **170**(1), 164 - 177. ISSN 0304-4076, https://doi.org/10.1016/j.jeconom.2012.04.004.

Breusch TS, Pagan AR (1980). "The Lagrange Multiplier Test and Its Applications to Model Specification in Econometrics." *Review of Economic Studies*, **47**, 239–253.

Pesaran MH (2004). "General Diagnostic Tests for Cross Section Dependence in Panels." CESifo Working Paper Series, 1229.

Pesaran MH (2015). "Testing Weak Cross-Sectional Dependence in Large Panels." *Econometric Reviews*, **34**(6-10), 1089-1117. doi:10.1080/07474938.2014.956623, https://doi.org/10.1080/07474938.2014.956623.

Examples

```
data("Grunfeld", package = "plm")
## test on heterogeneous model (separate time series regressions)
pcdtest(inv ~ value + capital, data = Grunfeld,
        index = c("firm", "year"))

## test on two-way fixed effects homogeneous model
pcdtest(inv ~ value + capital, data = Grunfeld, model = "within",
        effect = "twoways", index = c("firm", "year"))

## test on panelmodel object
g <- plm(inv ~ value + capital, data = Grunfeld, index = c("firm", "year"))
pcdtest(g)

## scaled LM test
pcdtest(g, test = "sclm")

## test on pseries
pGrunfeld <- pdata.frame(Grunfeld)
pcdtest(pGrunfeld$value)

## local test
## define neighbours for individual 2: 1, 3, 4, 5 in lower triangular matrix
w <- matrix(0, ncol= 10, nrow=10)
w[2,1] <- w[3,2] <- w[4,2] <- w[5,2] <- 1
```

```
pcdtest(g, w = w)
```

---

pdata.frame *pdata.frame: a data.frame for panel data*

Description

An object of class 'pdata.frame' is a data.frame with an index attribute that describes its individual and time dimensions.

Usage

```
pdata.frame(
  x,
  index = NULL,
  drop.index = FALSE,
  row.names = TRUE,
  stringsAsFactors = FALSE,
  replace.non.finite = FALSE,
  drop.NA.series = FALSE,
  drop.const.series = FALSE,
  drop.unused.levels = FALSE,
  ...
)

## S3 replacement method for class 'pdata.frame'
x$name <- value

## S3 method for class 'pdata.frame'
x[i, j, drop]

## S3 method for class 'pdata.frame'
x[[y]]

## S3 method for class 'pdata.frame'
x$y

## S3 method for class 'pdata.frame'
print(x, ...)

## S3 method for class 'pdata.frame'
as.list(x, keep.attributes = FALSE, ...)

## S3 method for class 'pdata.frame'
as.data.frame(
  x,
```

```
  row.names = NULL,
  optional = FALSE,
  keep.attributes = TRUE,
  ...
)
```

Arguments

x	a data.frame for the pdata.frame function and a pdata.frame for the methods,
index	this argument indicates the individual and time indexes. See **Details**,
drop.index	logical, indicates whether the indexes are to be excluded from the resulting pdata.frame,
row.names	NULL or logical, indicates whether "fancy" row names (combination of individual index and time index) are to be added to the returned (p)data.frame (NULL and FALSE have the same meaning for pdata.frame; for as.data.frame.pdata.frame see Details),
stringsAsFactors	logical, indicating whether character vectors are to be converted to factors,
replace.non.finite	logical, indicating whether values for which is.finite() yields TRUE are to be replaced by NA values, except for character variables (defaults to FALSE),
drop.NA.series	logical, indicating whether all-NA columns are to be removed from the pdata.frame (defaults to FALSE),
drop.const.series	logical, indicating whether constant columns are to be removed from the pdata.frame (defaults to FALSE),
drop.unused.levels	logical, indicating whether unused levels of factors are to be dropped (defaults to FALSE) (unused levels are always dropped from variables serving to construct the index variables),
...	further arguments passed on to internal usage of data.frame.
name	the name of the data.frame,
value	the name of the variable to include,
i	see Extract(),
j	see Extract(),
drop	see Extract(),
y	one of the columns of the data.frame,
keep.attributes	logical, only for as.list and as.data.frame methods, indicating whether the elements of the returned list/columns of the data.frame should have the pdata.frame's attributes added (default: FALSE for as.list, TRUE for as.data.frame),
optional	see as.data.frame(),

Details

The index argument indicates the dimensions of the panel. It can be:

- a vector of two character strings which contains the names of the individual and of the time indexes,
- a character string which is the name of the individual index variable. In this case, the time index is created automatically and a new variable called "time" is added, assuming consecutive and ascending time periods in the order of the original data,
- an integer, the number of individuals. In this case, the data need to be a balanced panel and be organized as a stacked time series (successive blocks of individuals, each block being a time series for the respective individual) assuming consecutive and ascending time periods in the order of the original data. Two new variables are added: "id" and "time" which contain the individual and the time indexes.

The "[[" and "$" extract a series from the pdata.frame. The "index" attribute is then added to the series and a class attribute "pseries" is added. The "[" method behaves as for data.frame, except that the extraction is also applied to the index attribute. A safe way to extract the index attribute is to use the function index() for 'pdata.frames' (and other objects).

as.data.frame removes the index attribute from the pdata.frame and adds it to each column. For its argument row.names set to FALSE row names are an integer series, TRUE gives "fancy" row names; if a character (with length of the resulting data frame), the row names will be the character's elements.

as.list behaves by default identical to base::as.list.data.frame() which means it drops the attributes specific to a pdata.frame; if a list of pseries is wanted, the attribute keep.attributes can to be set to TRUE. This also makes lapply work as expected on a pdata.frame (see also **Examples**).

Value

a pdata.frame object: this is a data.frame with an index attribute which is a data.frame with two variables, the individual and the time indexes, both being factors. The resulting pdata.frame is sorted by the individual index, then by the time index.

Author(s)

Yves Croissant

See Also

index() to extract the index variables from a 'pdata.frame' (and other objects), pdim() to check the dimensions of a 'pdata.frame' (and other objects), pvar() to check for each variable if it varies cross-sectionally and over time. To check if the time periods are consecutive per individual, see is.pconsecutive().

Examples

```
# Gasoline contains two variables which are individual and time
# indexes
data("Gasoline", package = "plm")
Gas <- pdata.frame(Gasoline, index = c("country", "year"), drop.index = TRUE)
```

```
# Hedonic is an unbalanced panel, townid is the individual index
data("Hedonic", package = "plm")
Hed <- pdata.frame(Hedonic, index = "townid", row.names = FALSE)

# In case of balanced panel, it is sufficient to give number of
# individuals data set 'Wages' is organized as a stacked time
# series
data("Wages", package = "plm")
Wag <- pdata.frame(Wages, 595)

# lapply on a pdata.frame by making it a list of pseries first
lapply(as.list(Wag[ , c("ed", "lwage")], keep.attributes = TRUE), lag)
```

---

pdim *Check for the Dimensions of the Panel*

Description

This function checks the number of individuals and time observations in the panel and whether it is balanced or not.

Usage

```
pdim(x, ...)

## Default S3 method:
pdim(x, y, ...)

## S3 method for class 'data.frame'
pdim(x, index = NULL, ...)

## S3 method for class 'pdata.frame'
pdim(x, ...)

## S3 method for class 'pseries'
pdim(x, ...)

## S3 method for class 'pggls'
pdim(x, ...)

## S3 method for class 'pcce'
pdim(x, ...)

## S3 method for class 'pmg'
pdim(x, ...)
```

```
## S3 method for class 'pgmm'
pdim(x, ...)

## S3 method for class 'panelmodel'
pdim(x, ...)

## S3 method for class 'pdim'
print(x, ...)
```

Arguments

x	a data.frame, a pdata.frame, a pseries, a panelmodel, or a pgmm object,
...	further arguments.
y	a vector,
index	see pdata.frame(),

Details

pdim is called by the estimation functions and can be also used stand-alone.

Value

An object of class pdim containing the following elements:

nT	a list containing n, the number of individuals, T, the number of time observations, N the total number of observations,
Tint	a list containing two vectors (of type integer): Ti gives the number of observations for each individual and nt gives the number of individuals observed for each period,
balanced	a logical value: TRUE for a balanced panel, FALSE for an unbalanced panel,
panel.names	a list of character vectors: id.names contains the names of each individual and time.names contains the names of each period.

Note

Calling pdim on an estimated panelmodel object and on the corresponding (p)data.frame used for this estimation does not necessarily yield the same result. When called on an estimated panelmodel, the number of observations (individual, time) actually used for model estimation are taken into account. When called on a (p)data.frame, the rows in the (p)data.frame are considered, disregarding any NA values in the dependent or independent variable(s) which would be dropped during model estimation.

Author(s)

Yves Croissant

See Also

is.pbalanced() to just determine balancedness of data (slightly faster than pdim),
punbalancedness() for measures of unbalancedness,
nobs(), pdata.frame(),
pvar() to check for each variable if it varies cross-sectionally and over time.

Examples

```
# There are 595 individuals
data("Wages", package = "plm")
pdim(Wages, 595)

# Gasoline contains two variables which are individual and time
# indexes and are the first two variables
data("Gasoline", package="plm")
pdim(Gasoline)

# Hedonic is an unbalanced panel, townid is the individual index
data("Hedonic", package = "plm")
pdim(Hedonic, "townid")

# An example of the panelmodel method
data("Produc", package = "plm")
z <- plm(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp,data=Produc,
         model="random", subset = gsp > 5000)
pdim(z)
```

---

pdwtest *Durbin–Watson Test for Panel Models*

Description

Test of serial correlation for (the idiosyncratic component of) the errors in panel models.

Usage

```
pdwtest(x, ...)

## S3 method for class 'panelmodel'
pdwtest(x, ...)

## S3 method for class 'formula'
pdwtest(x, data, ...)
```

Arguments

x	an object of class "panelmodel" or of class "formula",
...	further arguments to be passed on to dwtest, e.g., alternative, see lmtest::dwtest() for further details.
data	a data.frame,

Details

This Durbin–Watson test uses the auxiliary model on (quasi-)demeaned data taken from a model of class plm which may be a pooling (the default), random or within model. It performs a Durbin–Watson test (using dwtest from package **lmtest** on the residuals of the (quasi-)demeaned model, which should be serially uncorrelated under the null of no serial correlation in idiosyncratic errors. The function takes the demeaned data, estimates the model and calls dwtest. Thus, this test does not take the panel structure of the residuals into consideration; it shall not be confused with the generalized Durbin-Watson test for panels in pbnftest.

Value

An object of class "htest".

Author(s)

Giovanni Millo

References

Durbin J, Watson GS (1950). "Testing for Serial Correlation in Least Squares Regression: I." *Biometrika*, **37**(3/4), 409–428. ISSN 00063444.

Durbin J, Watson GS (1951). "Testing for serial correlation in least squares regression. II." *Biometrika*, **38**(1-2), 159-178. ISSN 0006-3444, doi:10.1093/biomet/38.12.159.

Durbin J, Watson GS (1971). "Testing for Serial Correlation in Least Squares Regression. III." *Biometrika*, **58**(1), 1–19. ISSN 00063444.

Wooldridge JM (2002). *Econometric Analysis of Cross–Section and Panel Data*. MIT Press.

Wooldridge JM (2010). *Econometric Analysis of Cross–Section and Panel Data*, 2nd edition. MIT Press.

See Also

lmtest::dwtest() for the Durbin–Watson test in **lmtest**, pbgtest() for the analogous Breusch–Godfrey test for panel models, lmtest::bgtest() for the Breusch–Godfrey test for serial correlation in the linear model. pbltest(), pbsytest(), pwartest() and pwfdtest() for other serial correlation tests for panel models.

For the Durbin-Watson test generalized to panel data models see pbnftest().

Examples

```
data("Grunfeld", package = "plm")
g <- plm(inv ~ value + capital, data = Grunfeld, model="random")
pdwtest(g)
pdwtest(g, alternative="two.sided")
## formula interface
pdwtest(inv ~ value + capital, data=Grunfeld, model="random")
```

`pFtest` *F Test for Individual and/or Time Effects*

Description

Test of individual and/or time effects based on the comparison of the `within` and the `pooling` model.

Usage

```
pFtest(x, ...)

## S3 method for class 'formula'
pFtest(x, data, ...)

## S3 method for class 'plm'
pFtest(x, z, ...)
```

Arguments

`x`	an object of class `"plm"` or of class `"formula"`,
`...`	further arguments.
`data`	a `data.frame`,
`z`	an object of class `"plm"`,

Details

For the `plm` method, the argument of this function is two `plm` objects, the first being a within model, the second a pooling model. The effects tested are either individual, time or twoways, depending on the effects introduced in the within model.

Value

An object of class `"htest"`.

Author(s)

Yves Croissant

See Also

plmtest() for Lagrange multiplier tests of individuals and/or time effects.

Examples

```
data("Grunfeld", package="plm")
gp <- plm(inv ~ value + capital, data = Grunfeld, model = "pooling")
gi <- plm(inv ~ value + capital, data = Grunfeld,
          effect = "individual", model = "within")
gt <- plm(inv ~ value + capital, data = Grunfeld,
          effect = "time", model = "within")
gd <- plm(inv ~ value + capital, data = Grunfeld,
          effect = "twoways", model = "within")
pFtest(gi, gp)
pFtest(gt, gp)
pFtest(gd, gp)
pFtest(inv ~ value + capital, data = Grunfeld, effect = "twoways")
```

---

pggls *General FGLS Estimators*

Description

General FGLS estimators for panel data (balanced or unbalanced)

Usage

```
pggls(
  formula,
  data,
  subset,
  na.action,
  effect = c("individual", "time"),
  model = c("within", "pooling", "fd"),
  index = NULL,
  ...
)

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

## S3 method for class 'summary.pggls'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
```

```
)

## S3 method for class 'pggls'
residuals(object, ...)
```

Arguments

formula	a symbolic description of the model to be estimated,
data	a data.frame,
subset	see lm(),
na.action	see lm(),
effect	the effects introduced in the model, one of "individual" or "time",
model	one of "within", "pooling", "fd",
index	the indexes, see pdata.frame(),
...	further arguments.
object, x	an object of class pggls,
digits	digits,
width	the maximum length of the lines in the print output,

Details

pggls is a function for the estimation of linear panel models by general feasible generalized least squares, either with or without fixed effects. General FGLS is based on a two-step estimation process: first a model is estimated by OLS (model = "pooling"), fixed effects (model = "within") or first differences (model = "fd"), then its residuals are used to estimate an error covariance matrix for use in a feasible-GLS analysis. This framework allows the error covariance structure inside every group (if effect = "individual", else symmetric) of observations to be fully unrestricted and is therefore robust against any type of intragroup heteroskedasticity and serial correlation. Conversely, this structure is assumed identical across groups and thus general FGLS estimation is inefficient under groupwise heteroskedasticity. Note also that this method requires estimation of $T(T+1)/2$ variance parameters, thus efficiency requires N » T (if effect = "individual", else the opposite).

If model = "within" (the default) then a FEGLS (fixed effects GLS, see Wooldridge, Ch. 10.5) is estimated; if model = "fd" a FDGLS (first-difference GLS). Setting model = "pooling" produces an unrestricted FGLS model (see ibid.) (model = "random" does the same, but using this value is deprecated and included only for retro–compatibility reasons).

Value

An object of class c("pggls","panelmodel") containing:

coefficients	the vector of coefficients,
residuals	the vector of residuals,
fitted.values	the vector of fitted values,
vcov	the covariance matrix of the coefficients,
df.residual	degrees of freedom of the residuals,

`model`	a data.frame containing the variables used for the estimation,
`call`	the call,
`sigma`	the estimated intragroup (or cross-sectional, if `effect = "time"`) covariance of errors,

Author(s)

Giovanni Millo

References

Im KS, Ahn SC, Schmidt P, Wooldridge JM (1999). "Efficient estimation of panel data models with strictly exogenous explanatory variables." *Journal of Econometrics*, **93**(1), 177 - 201. ISSN 0304-4076, https://doi.org/10.1016/S0304-4076(99)00008-1.

Kiefer NM (1980). "Estimation of fixed effect models for time series of cross-sections with arbitrary intertemporal covariance." *Journal of Econometrics*, **14**(2), 195–202.

Wooldridge JM (2002). *Econometric Analysis of Cross–Section and Panel Data*. MIT Press.

Wooldridge JM (2010). *Econometric Analysis of Cross–Section and Panel Data*, 2nd edition. MIT Press.

Examples

```
data("Produc", package = "plm")
zz_wi <- pggls(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
               data = Produc, model = "within")
summary(zz_wi)

zz_pool <- pggls(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
                 data = Produc, model = "pooling")
summary(zz_pool)

zz_fd <- pggls(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
               data = Produc, model = "fd")
summary(zz_fd)
```

---

pgmm *Generalized Method of Moments (GMM) Estimation for Panel Data*

Description

Generalized method of moments estimation for static or dynamic models with panel data.

Usage

```
pgmm(
  formula,
  data,
  subset,
  na.action,
  effect = c("twoways", "individual"),
  model = c("onestep", "twosteps"),
  collapse = FALSE,
  lost.ts = NULL,
  transformation = c("d", "ld"),
  fsm = switch(transformation, d = "G", ld = "full"),
  index = NULL,
  ...
)

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

## S3 method for class 'pgmm'
summary(object, robust = TRUE, time.dummies = FALSE, ...)

## S3 method for class 'summary.pgmm'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)
```

Arguments

formula	a symbolic description for the model to be estimated. The preferred interface is now to indicate a multi–part formula, the first two parts describing the covariates and the GMM instruments and, if any, the third part the 'normal' instruments,
data	a data.frame (neither factors nor character vectors will be accepted in data.frame),
subset	see lm(),
na.action	see lm(),
effect	the effects introduced in the model, one of "twoways" (the default) or "individual",
model	one of "onestep" (the default) or "twosteps",
collapse	if TRUE, the GMM instruments are collapsed (default is FALSE),
lost.ts	the number of lost time series: if NULL, this is automatically computed. Otherwise, it can be defined by the user as a numeric vector of length 1 or 2. The first element is the number of lost time series in the model in difference, the second one in the model in level. If the second element is missing, it is set to the first one minus one,

transformation	the kind of transformation to apply to the model: either "d" (the default value) for the "difference GMM" model or "ld" for the "system GMM" model,
fsm	character of length 1 to specify type of weighting matrix for the first step /the "onestep" estimator: one of "I" (identity matrix) or "G" ($= D'D$ where D is the first–difference operator), if transformation="d", one of "GI" or "full" if transformation="ld",
index	the indexes,
...	further arguments.
object, x	an object of class "pgmm",
robust	for pgmm's summary method: if TRUE (default), robust inference is performed in the summary, for a two-steps model with the small-sample correction by Windmeijer (2005),
time.dummies	for pgmm's summary method: if TRUE, the estimated coefficients of time dummies are present in the table of coefficients; default is FALSE, thus time dummies are dropped in summary's coefficient table (argument is only meaningful if there are time dummies in the model, i.e., only for effect = "twoways"),
digits	digits,
width	the maximum length of the lines in the print output.

Details

pgmm estimates a model for panel data with a generalized method of moments (GMM) estimator. The description of the model to estimate is provided with a multi–part formula which is (or which is coerced to) a Formula object. The first right–hand side part describes the covariates. The second one, which is mandatory, describes the GMM instruments. The third one, which is optional, describes the 'normal' instruments. By default, all the variables of the model which are not used as GMM instruments are used as normal instruments with the same lag structure as the one specified in the model.

y~lag(y, 1:2)+lag(x1, 0:1)+lag(x2, 0:2) | lag(y, 2:99) is similar to

y~lag(y, 1:2)+lag(x1, 0:1)+lag(x2, 0:2) | lag(y, 2:99) | lag(x1, 0:1)+lag(x2, 0:2)

and indicates that all lags from 2 of y are used as GMM instruments.

transformation indicates how the model should be transformed for the estimation. "d" gives the "difference GMM" model (see Arellano and Bond 1991), "ld" the "system GMM" model (see Blundell and Bond 1998).

pgmm is an attempt to adapt GMM estimators available within the DPD library for GAUSS (see Arellano and Bond 1998) and Ox (see Arellano and Bond 2012) and within the xtabond2 library for Stata (see Roodman 2009).

Value

An object of class c("pgmm","panelmodel"), which has the following elements:

coefficients	the vector (or the list for fixed effects) of coefficients,
residuals	the list of residuals for each individual,
vcov	the covariance matrix of the coefficients,

`fitted.values`	the vector of fitted values,
`df.residual`	degrees of freedom of the residuals,
`model`	a list containing the variables used for the estimation for each individual,
`W`	a list containing the instruments for each individual (a matrix per list element) (two lists in case of system GMM,
`A1`	the weighting matrix for the one–step estimator,
`A2`	the weighting matrix for the two–steps estimator,
`call`	the call.

It has `print`, `summary` and `print.summary` methods.

Author(s)

Yves Croissant

References

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

Arellano M, Bond S (1998). "Dynamic panel data estimation using DPD98 for GAUSS: a guide for users." unpublished, https://ifs.org.uk/publications/dpd-gauss.

Arellano M, Bond S (2012). "Panel data estimation using DPD for Ox." unpublished, https://www.doornik.com/download/oxmetrics7/Ox_Packages/dpd.pdf.

Blundell R, Bond S (1998). "Initial Conditions and Moment Restrictions in Dynamic Panel Data Models." *Journal of Econometrics*, **87**, 115–143.

Roodman D (2009). "How to do xtabond2: An introduction to difference and system GMM in Stata." *The Stata Journal*, **9**, 86-136. https://www.stata-journal.com/article.html?article=st0159.

Windmeijer F (2005). "A Finite Sample Correction for the Variance of Linear Efficient Two–Steps GMM Estimators." *Journal of Econometrics*, **126**, 25–51.

See Also

`sargan()` for the Hansen–Sargan test and `mtest()` for Arellano–Bond's test of serial correlation. vcovHC.pgmm for the robust inference.

Examples

```
data("EmplUK", package = "plm")

# Arellano/Bond 1991, Table 4, column (a1) (has robust SEs)
ab.a1 <- pgmm(log(emp) ~ lag(log(emp), 1:2) + lag(log(wage), 0:1)
             + lag(log(capital), 0:2) + lag(log(output), 0:2) | lag(log(emp), 2:99),
```

```
             data = EmplUK, effect = "twoways", model = "onestep")
summary(ab.a1, robust = TRUE)

# Arellano/Bond 1991, Table 4, column (a2) (has non-robust SEs)
ab.a2 <- pgmm(log(emp) ~ lag(log(emp), 1:2) + lag(log(wage), 0:1)
              + lag(log(capital), 0:2) + lag(log(output), 0:2) | lag(log(emp), 2:99),
              data = EmplUK, effect = "twoways", model = "twosteps")
summary(ab.a2, robust = FALSE)

# Arellano and Bond (1991), table 4 col. b / # Windmeijer (2005), table 2, std. errc
ab.b <- pgmm(log(emp) ~ lag(log(emp), 1:2) + lag(log(wage), 0:1)
             + log(capital) + lag(log(output), 0:1) | lag(log(emp), 2:99),
             data = EmplUK, effect = "twoways", model = "twosteps")
summary(ab.b, robust = FALSE) # Arellano/Bond
summary(ab.b, robust = TRUE)  # Windmeijer

## Blundell and Bond (1998) table 4 (cf. DPD for OX p. 12 col. 4)
bb.4 <- pgmm(log(emp) ~ lag(log(emp), 1)+ lag(log(wage), 0:1) +
           lag(log(capital), 0:1) | lag(log(emp), 2:99) +
           lag(log(wage), 2:99) + lag(log(capital), 2:99),
           data = EmplUK, effect = "twoways", model = "onestep",
           transformation = "ld")
summary(bb.4, robust = TRUE)

## Not run:
## Same with the old formula or dynformula interface
## Arellano and Bond (1991), table 4, col. b
ab.b <- pgmm(log(emp) ~ log(wage) + log(capital) + log(output),
            lag.form = list(2,1,0,1), data = EmplUK,
            effect = "twoways", model = "twosteps",
            gmm.inst = ~log(emp), lag.gmm = list(c(2,99)))
summary(ab.b, robust = FALSE)

## Blundell and Bond (1998) table 4 (cf. DPD for OX p. 12 col. 4)
bb.4 <- pgmm(dynformula(log(emp) ~ log(wage) + log(capital), list(1,1,1)),
            data = EmplUK, effect = "twoways", model = "onestep",
            gmm.inst = ~log(emp) + log(wage) + log(capital),
            lag.gmm = c(2,99), transformation = "ld")
summary(bb.4, robust = TRUE)

## End(Not run)
```

---

pgrangertest *Panel Granger (Non-)Causality Test (Dumitrescu/Hurlin (2012))*

Description

Test for Granger (non-)causality in panel data.

Usage

```
pgrangertest(
  formula,
  data,
  test = c("Ztilde", "Zbar", "Wbar"),
  order = 1L,
  index = NULL
)
```

Arguments

formula	a formula object to describe the direction of the hypothesized Granger causation,
data	a pdata.frame or a data.frame,
test	a character to request the statistic to be returned, either "Ztilde" (default),or "Zbar", alternatively, set to "Wbar" for an intermediate statistic (see Details),
order	integer(s) giving the number of lags to include in the test's auxiliary regressions, the length of order must be either 1 (same lag order for all individuals) or equal to the number of individuals (to specify a lag order per individual),
index	only relevant if data is data.frame and not a pdata.frame; if NULL, the first two columns of the data.frame are assumed to be the index variables, for further details see pdata.frame().

Details

The panel Granger (non-)causality test is a combination of Granger tests (Granger 1969) performed per individual. The test is developed by Dumitrescu and Hurlin (2012), a shorter exposition is given in Lopez and Weber (2017).

The formula formula describes the direction of the (panel) Granger causation where y ~ x means "x (panel) Granger causes y".

By setting argument test to either "Ztilde" (default) or "Zbar", two different statistics can be requested. "Ztilde" gives the standardised statistic recommended by Dumitrescu/Hurlin (2012) for fixed T samples. If set to "Wbar", the intermediate Wbar statistic (average of individual Granger chi-square statistics) is given which is used to derive the other two.

The Zbar statistic is not suitable for unbalanced panels. For the Wbar statistic, no p-value is available.

The implementation uses lmtest::grangertest() from package **lmtest** to perform the individual Granger tests.

Value

An object of class c("pgrangertest", "htest"). Besides the usual elements of a htest object, it contains the data frame indgranger which carries the Granger test statistics per individual along the associated p-values, degrees of freedom, and the specified lag order.

Author(s)

Kevin Tappe

References

Dumitrescu E, Hurlin C (2012). "Testing for Granger non-causality in heterogeneous panels." *Economic Modelling*, **29**(4), 1450 - 1460. ISSN 0264-9993, https://doi.org/10.1016/j.econmod.2012.02.014.

Granger CWJ (1969). "Investigating Causal Relations by Econometric Models and Cross-spectral Methods." *Econometrica*, **37**(3), 424–438. ISSN 00129682, 14680262.

Lopez L, Weber S (2017). "Testing for Granger causality in panel data." *Stata Journal*, **17**(4), 972-984. https://www.stata-journal.com/article.html?article=st0507.

See Also

`lmtest::grangertest()` for the original (non-panel) Granger causality test in **lmtest**.

Examples

```
## not meaningful, just to demonstrate usage
## H0: 'value' does not Granger cause 'inv' for all invididuals

data("Grunfeld", package = "plm")
pgrangertest(inv ~ value, data = Grunfeld)
pgrangertest(inv ~ value, data = Grunfeld, order = 2L)
pgrangertest(inv ~ value, data = Grunfeld, order = 2L, test = "Zbar")

# varying lag order (last individual lag order 3, others lag order 2)
(pgrt <- pgrangertest(inv ~ value, data = Grunfeld, order = c(rep(2L, 9), 3L)))
# chisq statistics per individual
pgrt$indgranger
```

---

phansitest *Simes Test for unit roots in panel data*

Description

Simes' test of intersection of individual hypothesis tests (Simes (1986)) applied to panel unit root tests as suggested by Hanck (2013).

Usage

```
phansitest(object, alpha = 0.05)

## S3 method for class 'phansitest'
print(x, cutoff = 10L, ...)
```

Arguments

object	either a numeric containing p-values of individual unit root test results (does not need to be sorted) or a suitable purtest object (as produced by purtest() for a test which gives p-values of the individuals (Hadri's test in purtest is not suitable)),
alpha	numeric, the pre-specified significance level (defaults to 0.05),
x	an object of class c("phansitest", "list") as produced by phansitest to be printed,
cutoff	integer, cutoff value for printing of enumeration of individuals with rejected individual H0, for print method only,
...	further arguments (currently not used).

Details

Simes' approach to testing is combining p-values from single hypothesis tests with a global (intersected) hypothesis. Hanck (2013) mentions it can be applied to any panel unit root test which yields a p-value for each individual series. The test is robust versus general patterns of cross-sectional dependence.

Further, this approach allows to discriminate between individuals for which the individual H0 (unit root present for individual series) is rejected/is not rejected by Hommel's procedure (Hommel (1988)) for family-wise error rate control (FWER) at a pre-specified significance level α via argument alpha (defaulting to 0.05), i.e., it controls for the multiplicity in testing.

The function phansitest takes as main input object either a plain numeric containing p-values of individual tests or a purtest object which holds a suitable pre-computed panel unit root test (one that produces p-values per individual series).

The function's return value (see section Value) is a list with detailed evaluation of the applied Simes test.

The associated print method prints a verbal evaluation.

Value

For phansitest, an object of class c("phansitest", "list") which is a list with the elements:

- id: integer, the identifier of the individual (integer sequence referring to position in input),
- name: character, name of the input's individual (if it has a name, otherwise "1", "2", "3", ...),
- p: numeric, p-values as input (either the numeric or extracted from the purtest object),
- p.hommel: numeric, p-values after Hommel's transformation,
- rejected: logical, indicating for which individual the individual null hypothesis is rejected (TRUE)/non-rejected (FALSE) (after controlling for multiplicity),
- rejected.no: integer, giving the total number of rejected individual series,
- alpha: numeric, the input alpha.

Author(s)

Kevin Tappe

References

Hanck C (2013). "An Intersection Test for Panel Unit Roots." *Econometric Reviews*, **32**, 183-203.

Hommel G (1988). "A stage wise rejective multiple test procedure based on a modified Bonferroni test." *Biometrika*, **75**, 383-386.

Simes RJ (1986). "An improved Bonferroni procedure for multiple tests of significance." *Biometrika*, **73**, 751-754.

See Also

purtest(), cipstest()

Examples

```
### input is numeric (p-values)
#### example from Hanck (2013), Table 11 (left side)
pvals <- c(0.0001,0.0001,0.0001,0.0001,0.0001,0.0001,0.0050,0.0050,0.0050,
           0.0050,0.0175,0.0175,0.0200,0.0250,0.0400,0.0500,0.0575,0.2375,0.2475)

countries <- c("Argentina","Sweden","Norway","Mexico","Italy","Finland","France",
               "Germany","Belgium","U.K.","Brazil","Australia","Netherlands",
               "Portugal","Canada", "Spain","Denmark","Switzerland","Japan")
names(pvals) <- countries

h <- phansitest(pvals)
print(h)              # (explicitly) prints test's evaluation
print(h, cutoff = 3L) # print only first 3 rejected ids
h$rejected # logical indicating the individuals with rejected individual H0


### input is a (suitable) purtest object
data("Grunfeld", package = "plm")
y <- data.frame(split(Grunfeld$inv, Grunfeld$firm))
obj <- purtest(y, pmax = 4, exo = "intercept", test = "madwu")

phansitest(obj)
```

---

pht *Hausman–Taylor Estimator for Panel Data*

Description

The Hausman–Taylor estimator is an instrumental variable estimator without external instruments (function deprecated).

Usage

```
pht(
  formula,
  data,
  subset,
  na.action,
  model = c("ht", "am", "bms"),
  index = NULL,
  ...
)

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

## S3 method for class 'summary.pht'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  subset = NULL,
  ...
)
```

Arguments

formula	a symbolic description for the model to be estimated,
data	a data.frame,
subset	see lm() for "plm", a character or numeric vector indicating a subset of the table of coefficient to be printed for "print.summary.plm",
na.action	see lm(),
model	one of "ht" for Hausman–Taylor, "am" for Amemiya–MaCurdy and "bms" for Breusch–Mizon–Schmidt,
index	the indexes,
...	further arguments.
object, x	an object of class "plm",
digits	digits,
width	the maximum length of the lines in the print output,

Details

pht estimates panels models using the Hausman–Taylor estimator, Amemiya–MaCurdy estimator, or Breusch–Mizon–Schmidt estimator, depending on the argument model. The model is specified as a two–part formula, the second part containing the exogenous variables.

Value

An object of class `c("pht", "plm", "panelmodel")`.

A `"pht"` object contains the same elements as `plm` object, with a further argument called `varlist` which describes the typology of the variables. It has `summary` and `print.summary` methods.

Note

The function `pht` is deprecated. Please use function `plm` to estimate Taylor–Hausman models like this with a three-part formula as shown in the example:
`plm(<formula>, random.method = "ht", model = "random", inst.method = "baltagi")`.
The Amemiya–MaCurdy estimator and the Breusch–Mizon–Schmidt estimator is computed likewise with `plm`.

Author(s)

Yves Croissant

References

(Amemiya and MaCurdy 1986)

(Baltagi 2013)

(Breusch et al. 1989)

(Hausman and Taylor 1981)

Examples

```
## replicates Baltagi (2005, 2013), table 7.4; Baltagi (2021), table 7.5
## preferred way with plm()
data("Wages", package = "plm")
ht <- plm(lwage ~ wks + south + smsa + married + exp + I(exp ^ 2) +
              bluecol + ind + union + sex + black + ed |
              bluecol + south + smsa + ind + sex + black |
              wks + married + union + exp + I(exp ^ 2),
          data = Wages, index = 595,
          random.method = "ht", model = "random", inst.method = "baltagi")
summary(ht)

am <- plm(lwage ~ wks + south + smsa + married + exp + I(exp ^ 2) +
              bluecol + ind + union + sex + black + ed |
              bluecol + south + smsa + ind + sex + black |
              wks + married + union + exp + I(exp ^ 2),
          data = Wages, index = 595,
          random.method = "ht", model = "random", inst.method = "am")
summary(am)

## deprecated way with pht() for HT
#ht <- pht(lwage ~ wks + south + smsa + married + exp + I(exp^2) +
#          bluecol + ind + union + sex + black + ed |
#          sex + black + bluecol + south + smsa + ind,
```

```
#          data = Wages, model = "ht", index = 595)
#summary(ht)
# deprecated way with pht() for AM
#am <- pht(lwage ~ wks + south + smsa + married + exp + I(exp^2) +
#          bluecol + ind + union + sex + black + ed |
#          sex + black + bluecol + south + smsa + ind,
#          data = Wages, model = "am", index = 595)
#summary(am)
```

---

phtest *Hausman Test for Panel Models*

Description

Specification test for panel models.

Usage

```
phtest(x, ...)

## S3 method for class 'formula'
phtest(
  x,
  data,
  model = c("within", "random"),
  effect = c("individual", "time", "twoways"),
  method = c("chisq", "aux"),
  index = NULL,
  vcov = NULL,
  ...
)

## S3 method for class 'panelmodel'
phtest(x, x2, ...)
```

Arguments

x	an object of class "panelmodel" or "formula",
...	further arguments to be passed on (currently none).
data	a data.frame,
model	a character vector containing the names of two models (length(model) must be 2),
effect	a character specifying the effect to be introduced to both models, one of "individual", "time", or "twoways" (only for formula method),

method	one of "chisq" or "aux",
index	an optional vector of index variables,
vcov	an optional covariance function,
x2	an object of class "panelmodel" (only for panelmodel method/interface),

Details

The Hausman test (sometimes also called Durbin–Wu–Hausman test) is based on the difference of the vectors of coefficients of two different models. The panelmodel method computes the original version of the test based on a quadratic form (Hausman 1978). The formula method, if method = "chisq" (default), computes the original version of the test based on a quadratic form; if method ="aux" then the auxiliary-regression-based version as in Wooldridge (2010), Sec.10.7.3. Only the latter can be robustified by specifying a robust covariance estimator as a function through the argument vcov (see **Examples**).

The effect argument is only relevant for the formula method/interface and is then applied to both models. For the panelmodel method/interface, the test is run with the effects of the already estimated models.

The equivalent tests in the **one-way** case using a between model (either "within vs. between" or "random vs. between") (see Hausman and Taylor 1981 or Baltagi 2013 Sec.4.3) can also be performed by phtest, but only for test = "chisq", not for the regression-based test. NB: These equivalent tests using the between model do not extend to the two-ways case. There are, however, some other equivalent tests, (see Kang 1985 or Baltagi 2013 Sec.4.3.7), but those are unsupported by phtest.

Value

An object of class "htest".

Author(s)

Yves Croissant, Giovanni Millo

References

Hausman JA (1978). "Specification Tests in Econometrics." *Econometrica*, **46**, 1251–1271.

Hausman JA, Taylor WE (1981). "Panel Data and Unobservable Individual Effects." *Econometrica*, **49**, 1377–1398.

Kang S (1985). "A note on the equivalence of specification tests in the two-factor multivariate variance components model." *Journal of Econometrics*, **28**(2), 193 - 203. ISSN 0304-4076, https://doi.org/10.1016/0304-4076(85)90119-8.

Wooldridge JM (2010). *Econometric Analysis of Cross–Section and Panel Data*, 2nd edition. MIT Press.

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Examples

```
data("Gasoline", package = "plm")
form <- lgaspcar ~ lincomep + lrpmg + lcarpcap
wi <- plm(form, data = Gasoline, model = "within")
re <- plm(form, data = Gasoline, model = "random")
phtest(wi, re)
phtest(form, data = Gasoline)
phtest(form, data = Gasoline, effect = "time")

# Regression-based Hausman test
phtest(form, data = Gasoline, method = "aux")

# robust Hausman test with vcov supplied as a function and
# with additional parameters
phtest(form, data = Gasoline, method = "aux", vcov = vcovHC)
phtest(form, data = Gasoline, method = "aux",
  vcov = function(x) vcovHC(x, method="white2", type="HC3"))
```

---

piest *Chamberlain estimator and test for fixed effects*

Description

General estimator useful for testing the within specification

Usage

```
piest(formula, data, subset, na.action, index = NULL, robust = TRUE, ...)

## S3 method for class 'piest'
print(x, ...)

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

## S3 method for class 'summary.piest'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  subset = NULL,
  ...
)
```

Arguments

formula	a symbolic description for the model to be estimated,
data	a data.frame,
subset	see lm(),
na.action	see lm(),
index	the indexes,
robust	logical, if FALSE, the error is assumed to be spherical, if TRUE, a robust estimation of the covariance matrix is computed,
...	further arguments.
object, x	an object of class "piest" and of class "summary.piest" for the print method of summary for piest objects,
digits	number of digits for printed output,
width	the maximum length of the lines in the printed output,

Details

The Chamberlain method consists in using the covariates of all the periods as regressors. It allows to test the within specification.

Value

An object of class "piest".

Author(s)

Yves Croissant

References

Chamberlain G (1982). "Multivariate regression models for panel data." *Journal of Econometrics*, **18**, 5–46.

See Also

aneweytest()

Examples

```
data("RiceFarms", package = "plm")
pirice <- piest(log(goutput) ~ log(seed) + log(totlabor) + log(size), RiceFarms, index = "id")
summary(pirice)
```

pldv *Panel estimators for limited dependent variables*

Description

Fixed and random effects estimators for truncated or censored limited dependent variable

Usage

```
pldv(
  formula,
  data,
  subset,
  weights,
  na.action,
  model = c("fd", "random", "pooling"),
  index = NULL,
  R = 20,
  start = NULL,
  lower = 0,
  upper = +Inf,
  objfun = c("lsq", "lad"),
  sample = c("cens", "trunc"),
  ...
)
```

Arguments

formula	a symbolic description for the model to be estimated,
data	a data.frame,
subset	see lm,
weights	see lm,
na.action	see lm,
model	one of "fd", "random", or "pooling",
index	the indexes, see pdata.frame(),
R	the number of points for the gaussian quadrature,
start	a vector of starting values,
lower	the lower bound for the censored/truncated dependent variable,
upper	the upper bound for the censored/truncated dependent variable,
objfun	the objective function for the fixed effect model (model = "fd", irrelevant for other values of the model argument): one of "lsq" for least squares (minimise sum of squares of the residuals) and "lad" for least absolute deviations (minimise sum of absolute values of the residuals),
sample	"cens" for a censored (tobit-like) sample, "trunc" for a truncated sample,
...	further arguments.

Details

pldv computes two kinds of models: a LSQ/LAD estimator for the first-difference model (model = "fd") and a maximum likelihood estimator with an assumed normal distribution for the individual effects (model = "random" or "pooling").

For maximum-likelihood estimations, pldv uses internally function maxLik::maxLik() (from package **maxLik**).

Value

For model = "fd", an object of class c("plm", "panelmodel"), for model = "random" and model = "pooling" an object of class c("maxLik", "maxim").

Author(s)

Yves Croissant

References

Honoré BE (1992). "Trimmed LAD and least squares estimation of truncated and censored regression models with fixed effects." *Econometrica*, **60**(3).

Examples

```
## as these examples take a bit of time, do not run them automatically
## Not run:
data("Donors", package = "pder")
library("plm")
pDonors <- pdata.frame(Donors, index = "id")

# replicate Landry/Lange/List/Price/Rupp (2010), online appendix, table 5a, models A and B
modA <- pldv(donation ~ treatment +  prcontr, data = pDonors,
            model = "random", method = "bfgs")
summary(modA)
modB <- pldv(donation ~ treatment * prcontr - prcontr, data = pDonors,
            model = "random", method = "bfgs")
summary(modB)

## End(Not run)
```

---

plm *Panel Data Estimators*

Description

Linear models for panel data estimated using the lm function on transformed data.

Usage

```
plm(
  formula,
  data,
  subset,
  weights,
  na.action,
  effect = c("individual", "time", "twoways", "nested"),
  model = c("within", "random", "ht", "between", "pooling", "fd"),
  random.method = NULL,
  random.models = NULL,
  random.dfcor = NULL,
  inst.method = c("bvk", "baltagi", "am", "bms"),
  restrict.matrix = NULL,
  restrict.rhs = NULL,
  index = NULL,
  ...
)

## S3 method for class 'plm.list'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)

## S3 method for class 'panelmodel'
terms(x, ...)

## S3 method for class 'panelmodel'
vcov(object, ...)

## S3 method for class 'panelmodel'
fitted(object, ...)

## S3 method for class 'panelmodel'
residuals(object, ...)

## S3 method for class 'panelmodel'
df.residual(object, ...)

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

## S3 method for class 'panelmodel'
print(
  x,
```

```
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)

## S3 method for class 'panelmodel'
update(object, formula., ..., evaluate = TRUE)

## S3 method for class 'panelmodel'
deviance(object, model = NULL, ...)

## S3 method for class 'plm'
formula(x, ...)

## S3 method for class 'plm'
plot(
  x,
  dx = 0.2,
  N = NULL,
  seed = 1,
  within = TRUE,
  pooling = TRUE,
  between = FALSE,
  random = FALSE,
  ...
)

## S3 method for class 'plm'
residuals(object, model = NULL, effect = NULL, ...)

## S3 method for class 'plm'
fitted(object, model = NULL, effect = NULL, ...)
```

Arguments

formula	a symbolic description for the model to be estimated,
data	a data.frame,
subset	see stats::lm(),
weights	see stats::lm(),
na.action	see stats::lm(); currently, not fully supported,
effect	the effects introduced in the model, one of "individual", "time", "twoways", or "nested",
model	one of "pooling", "within", "between", "random" "fd", or "ht",
random.method	method of estimation for the variance components in the random effects model, one of "swar" (default), "amemiya", "walhus", "nerlove"; for Hausman-Taylor estimation set to "ht" (see Details and Examples),

random.models	an alternative to the previous argument, the models used to compute the variance components estimations are indicated,
random.dfcor	a numeric vector of length 2 indicating which degree of freedom should be used,
inst.method	the instrumental variable transformation: one of "bvk", "baltagi", "am", or "bms" (see also Details),
restrict.matrix	a matrix which defines linear restrictions on the coefficients,
restrict.rhs	the right hand side vector of the linear restrictions on the coefficients,
index	the indexes,
...	further arguments.
x, object	an object of class "plm",
digits	number of digits for printed output,
width	the maximum length of the lines in the printed output,
formula.	a new formula for the update method,
evaluate	a boolean for the update method, if TRUE the updated model is returned, if FALSE the call is returned,
dx	the half–length of the individual lines for the plot method (relative to x range),
N	the number of individual to plot,
seed	the seed which will lead to individual selection,
within	if TRUE, the within model is plotted,
pooling	if TRUE, the pooling model is plotted,
between	if TRUE, the between model is plotted,
random	if TRUE, the random effect model is plotted,

Details

plm is a general function for the estimation of linear panel models. It supports the following estimation methods: pooled OLS (model = "pooling"), fixed effects ("within"), random effects ("random"), first–differences ("fd"), and between ("between"). It supports unbalanced panels and two–way effects (although not with all methods).

For random effects models, four estimators of the transformation parameter are available by setting random.method to one of "swar" (Swamy and Arora 1972) (default), "amemiya" (Amemiya 1971), "walhus" (Wallace and Hussain 1969), or "nerlove" (Nerlove 1971) (see below for Hausman-Taylor instrumental variable case).

The nested random effect model ((Baltagi et al. 2001)) is estimated by setting model = "random" and effect = "nested", requiring the data to be indexed by a third index in which the "individual" dimension is nested (see section *Examples* and the vignette "Estimation of error components models with the plm function".)

For first–difference models, the intercept is maintained (which from a specification viewpoint amounts to allowing for a trend in the levels model). The user can exclude it from the estimated specification the usual way by adding "-1" to the model formula.

Instrumental variables estimation is obtained using two–part formulas, the second part indicating the instrumental variables used. This can be a complete list of instrumental variables or an update of the first part. If, for example, the model is `y ~ x1 + x2 + x3`, with `x1` and `x2` endogenous and `z1` and `z2` external instruments, the model can be estimated with:

- `formula = y~x1+x2+x3 | x3+z1+z2`,
- `formula = y~x1+x2+x3 | . -x1-x2+z1+z2`.

If an instrument variable estimation is requested, argument `inst.method` selects the instrument variable transformation method:

- `"bvk"` (default) for Balestra and Varadharajan–Krishnakumar (1987),
- `"baltagi"` for Baltagi (1981),
- `"am"` for Amemiya and MaCurdy (1986),
- `"bms"` for Breusch et al. (1989).

The Hausman–Taylor estimator (Hausman and Taylor 1981) is computed with arguments `random.method = "ht"`, `model = "random"`, `inst.method = "baltagi"` (the other way with only `model = "ht"` is deprecated).

See also the vignettes for introductions to model estimations (and more) with examples.

Value

An object of class `"plm"`.

A `"plm"` object has the following elements :

`coefficients`	the vector of coefficients,
`vcov`	the variance–covariance matrix of the coefficients,
`residuals`	the vector of residuals (these are the residuals of the (quasi-)demeaned model),
`weights`	(only for weighted estimations) weights as specified,
`df.residual`	degrees of freedom of the residuals,
`formula`	an object of class `"Formula"` describing the model,
`model`	the model frame as a `"pdata.frame"` containing the variables used for estimation: the response is in first column followed by the other variables, the individual and time indexes are in the 'index' attribute of `model`,
`ercomp`	an object of class `"ercomp"` providing the estimation of the components of the errors (for random effects models only),
`aliased`	named logical vector indicating any aliased coefficients which are silently dropped by `plm` due to linearly dependent terms (see also `detect.lindep()`),
`call`	the call.

It has `print`, `summary` and `print.summary` methods. The `summary` method creates an object of class `"summary.plm"` that extends the object it is run on with information about (inter alia) F statistic and (adjusted) R-squared of model, standard errors, t–values, and p–values of coefficients, (if supplied) the furnished vcov, see `summary.plm()` for further details.

Author(s)

Yves Croissant

References

Amemiya T (1971). "The Estimation of the Variances in a Variance–Components Model." *International Economic Review*, **12**, 1–13.

Amemiya T, MaCurdy TE (1986). "Instrumental-Variable Estimation of an Error-Components Model." *Econometrica*, **54**(4), 869-80.

Balestra P, Varadharajan–Krishnakumar J (1987). "Full Information Estimations of a System of Simultaneous Equations With Error Components." *Econometric Theory*, **3**, 223–246.

Baltagi BH (1981). "Simultaneous Equations With Error Components." *Journal of Econometrics*, **17**, 21–49.

Baltagi BH, Song SH, Jung BC (2001). "The unbalanced nested error component regression model." *Journal of Econometrics*, **101**, 357-381.

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Breusch TS, Mizon GE, Schmidt P (1989). "Efficient Estimation Using Panel Data." *Econometrica*, **57**(3), 695-700.

Hausman JA, Taylor WE (1981). "Panel Data and Unobservable Individual Effects." *Econometrica*, **49**, 1377–1398.

Nerlove M (1971). "Further Evidence on the Estimation of Dynamic Economic Relations from a Time–Series of Cross–Sections." *Econometrica*, **39**, 359–382.

Swamy PAVB, Arora SS (1972). "The Exact Finite Sample Properties of the Estimators of Coefficients in the Error Components Regression Models." *Econometrica*, **40**, 261–275.

Wallace TD, Hussain A (1969). "The Use of Error Components Models in Combining Cross Section With Time Series Data." *Econometrica*, **37**(1), 55–72.

See Also

`summary.plm()` for further details about the associated summary method and the "summary.plm" object both of which provide some model tests and tests of coefficients. `fixef()` to compute the fixed effects for "within" models (=fixed effects models). `predict.plm()` for predicted values.

Examples

```
data("Produc", package = "plm")
zz <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
          data = Produc, index = c("state","year"))
summary(zz)

# replicates some results from Baltagi (2013), table 3.1
data("Grunfeld", package = "plm")
p <- plm(inv ~ value + capital,
         data = Grunfeld, model = "pooling")

wi <- plm(inv ~ value + capital,
```

```
          data = Grunfeld, model = "within", effect = "twoways")

swar <- plm(inv ~ value + capital,
            data = Grunfeld, model = "random", effect = "twoways")

amemiya <- plm(inv ~ value + capital,
               data = Grunfeld, model = "random", random.method = "amemiya",
               effect = "twoways")

walhus <- plm(inv ~ value + capital,
              data = Grunfeld, model = "random", random.method = "walhus",
              effect = "twoways")

# summary and summary with a furnished vcov (passed as matrix,
# as function, and as function with additional argument)
summary(wi)
summary(wi, vcov = vcovHC(wi))
summary(wi, vcov = vcovHC)
summary(wi, vcov = function(x) vcovHC(x, method = "white2"))


## nested random effect model
# replicate Baltagi/Song/Jung (2001), p. 378 (table 6), columns SA, WH
# == Baltagi (2013), pp. 204-205
data("Produc", package = "plm")
pProduc <- pdata.frame(Produc, index = c("state", "year", "region"))
form <- log(gsp) ~ log(pc) + log(emp) + log(hwy) + log(water) + log(util) + unemp
summary(plm(form, data = pProduc, model = "random", effect = "nested"))
summary(plm(form, data = pProduc, model = "random", effect = "nested",
            random.method = "walhus"))

## Instrumental variable estimations
# replicate Baltagi (2013/2021), p. 133/162, table 7.1
data("Crime", package = "plm")
FE2SLS <- plm(lcrmrte ~ lprbarr + lpolpc + lprbconv + lprbpris + lavgsen +
                ldensity + lwcon + lwtuc + lwtrd + lwfir + lwser + lwmfg + lwfed +
                lwsta + lwloc + lpctymle + lpctmin + region + smsa + factor(year)
              | . - lprbarr - lpolpc + ltaxpc + lmix,
              data = Crime, model = "within")
G2SLS <- update(FE2SLS, model = "random", inst.method = "bvk")
EC2SLS <- update(G2SLS, model = "random", inst.method = "baltagi")

## Hausman-Taylor estimator and Amemiya-MaCurdy estimator
# replicate Baltagi (2005, 2013), table 7.4; Baltagi (2021), table 7.5
data("Wages", package = "plm")
ht <- plm(lwage ~ wks + south + smsa + married + exp + I(exp ^ 2) +
              bluecol + ind + union + sex + black + ed |
              bluecol + south + smsa + ind + sex + black |
              wks + married + union + exp + I(exp ^ 2),
          data = Wages, index = 595,
          random.method = "ht", model = "random", inst.method = "baltagi")
summary(ht)
```

```
am <- plm(lwage ~ wks + south + smsa + married + exp + I(exp ^ 2) +
              bluecol + ind + union + sex + black + ed |
              bluecol + south + smsa + ind + sex + black |
              wks + married + union + exp + I(exp ^ 2),
          data = Wages, index = 595,
          random.method = "ht", model = "random", inst.method = "am")
summary(am)
```

---

plm-deprecated *Deprecated functions of plm*

Description

dynformula, pht, plm.data, and pvcovHC are deprecated functions which could be removed from **plm** in a near future.

Usage

```
pvcovHC(x, ...)

plm.data(x, indexes = NULL)

dynformula(formula, lag.form = NULL, diff.form = NULL, log.form = NULL)

## S3 method for class 'dynformula'
formula(x, ...)

## S3 method for class 'dynformula'
print(x, ...)
```

Arguments

x	an object of class "plm",
...	further arguments.
indexes	a vector (of length one or two) indicating the (individual and time) indexes (see Details);
formula	a formula,
lag.form	a list containing the lag structure of each variable in the formula,
diff.form	a vector (or a list) of logical values indicating whether variables should be differenced,
log.form	a vector (or a list) of logical values indicating whether variables should be in logarithms.
data	a data.frame,

Details

dynformula was used to construct a dynamic formula which was the first argument of pgmm. pgmm uses now multi-part formulas.

pht estimates the Hausman-Taylor model, which can now be estimated using the more general plm function.

plm.data is replaced by pdata.frame.

pvcovHC is replaced by vcovHC.

detect_lin_dep was renamed to detect.lindep.

---

plm.fast *Option to Switch On/Off Fast Data Transformations*

Description

A significant speed up can be gained by using fast (panel) data transformation functions from package collapse. An additional significant speed up for the two-way fixed effects case can be achieved if package fixest or lfe is installed (package collapse needs to be installed for the fast mode in any case).

Details

By default, this speed up is enabled. Option plm.fast can be used to enable/disable the speed up. The option is evaluated prior to execution of supported transformations (see below), so option("plm.fast" = TRUE) enables the speed up while option("plm.fast" = FALSE) disables the speed up.

To have it always switched off, put options("plm.fast" = FALSE) in your .Rprofile file.

See **Examples** for how to use the option and for a benchmarking example.

For long, package plm used base R implementations and R-based code. The package collapse provides fast data transformation functions written in C/C++, among them some especially suitable for panel data. Having package collapse installed is a requirement for the speed up, so this package is a hard dependency for package plm.

Availability of packages fixest and lfe is checked for once when package plm is attached and the additional speed up for the two-way fixed effect case is enabled automatically (fixest wins over lfe), given one of the packages is detected and options("plm.fast" = TRUE) (default) is set. If so, the packages' fast algorithms to partial out fixed effects are used (fixest::demean (via collapse::fhdwithin), lfe::demeanlist). Both packages are 'Suggests' dependencies.

Users might experience neglectable numerical differences between enabled and disabled fast mode and base R implementation, depending on the platform and the additional packages installed.

Currently, these basic functions benefit from the speed-up, used as building blocks in most model estimation functions, e.g., in plm (more functions are under investigation):

- between,
- Between,
- Sum,

- Within,
- lag, lead, and diff,
- pseriesfy,
- pdiff (internal function).

Examples

```
## Not run:
### A benchmark of plm without and with speed-up
library("plm")
library("collapse")
library("microbenchmark")
rm(list = ls())
data("wlddev", package = "collapse")
form <- LIFEEX ~ PCGDP + GINI

# produce big data set (taken from collapse's vignette)
wlddevsmall <- get_vars(wlddev, c("iso3c","year","OECD","PCGDP","LIFEEX","GINI","ODA"))
wlddevsmall$iso3c <- as.character(wlddevsmall$iso3c)
data <- replicate(100, wlddevsmall, simplify = FALSE)
rm(wlddevsmall)
uniquify <- function(x, i) {
  x$iso3c <- paste0(x$iso3c, i)
  x
}
data <- unlist2d(Map(uniquify, data, as.list(1:100)), idcols = FALSE)
data <- pdata.frame(data, index = c("iso3c", "year"))
pdim(data) # Balanced Panel: n = 21600, T = 59, N = 1274400 // but many NAs
# data <- na.omit(data)
# pdim(data) # Unbalanced Panel: n = 13300, T = 1-31, N = 93900

times <- 1 # no. of repetitions for benchmark - this takes quite long!

onewayFE <- microbenchmark(
 {options("plm.fast" = FALSE); plm(form, data = data, model = "within")},
 {options("plm.fast" = TRUE);  plm(form, data = data, model = "within")},
  times = times)

summary(onewayFE, unit = "relative")

## two-ways FE benchmark requires pkg fixest and lfe
## (End-users shall only set option plm.fast. Option plm.fast.pkg.FE.tw shall
##  _not_ be set by the end-user, it is determined automatically when pkg plm
## is attached; however, it needs to be set explicitly in this example for the
## benchmark.)
if(requireNamespace("fixest", quietly = TRUE) &&
   requireNamespace("lfe", quietly = TRUE)) {

twowayFE <-  microbenchmark(
 {options("plm.fast" = FALSE);
    plm(form, data = data, model = "within", effect = "twoways")},
 {options("plm.fast" = TRUE, "plm.fast.pkg.FE.tw" = "collapse");
```

```
    plm(form, data = data, model = "within", effect = "twoways")},
 {options("plm.fast" = TRUE, "plm.fast.pkg.FE.tw" = "fixest");
    plm(form, data = data, model = "within", effect = "twoways")},
 {options("plm.fast" = TRUE, "plm.fast.pkg.FE.tw" = "lfe");
    plm(form, data = data, model = "within", effect = "twoways")},
  times = times)

summary(twowayFE, unit = "relative")
}

onewayRE <- microbenchmark(
 {options("plm.fast" = FALSE); plm(form, data = data, model = "random")},
 {options("plm.fast" = TRUE);  plm(form, data = data, model = "random")},
  times = times)

summary(onewayRE, unit = "relative")

twowayRE <-  microbenchmark(
 {options("plm.fast" = FALSE); plm(form, data = data, model = "random", effect = "twoways")},
 {options("plm.fast" = TRUE);  plm(form, data = data, model = "random", effect = "twoways")},
  times = times)

summary(twowayRE, unit = "relative")

## End(Not run)
```

---

plmtest *Lagrange FF Multiplier Tests for Panel Models*

Description

Test of individual and/or time effects for panel models.

Usage

```
plmtest(x, ...)

## S3 method for class 'plm'
plmtest(
  x,
  effect = c("individual", "time", "twoways"),
  type = c("honda", "bp", "ghm", "kw"),
  ...
)

## S3 method for class 'formula'
plmtest(
  x,
  data,
```

```
  ...,
  effect = c("individual", "time", "twoways"),
  type = c("honda", "bp", "ghm", "kw")
)
```

Arguments

x	an object of class "plm" or a formula of class "formula",
...	further arguments passed to plmtest.
effect	a character string indicating which effects are tested: individual effects ("individual"), time effects ("time") or both ("twoways"),
type	a character string indicating the test to be performed: • "honda" (default) for Honda (1985), • "bp" for Breusch and Pagan (1980), • "kw" for King and Wu (1997), or • "ghm" for Gourieroux et al. (1982) for unbalanced panel data sets, the respective unbalanced version of the tests are computed,
data	a data.frame,

Details

These Lagrange multiplier tests use only the residuals of the pooling model. The first argument of this function may be either a pooling model of class plm or an object of class formula describing the model. For input within (fixed effects) or random effects models, the corresponding pooling model is calculated internally first as the tests are based on the residuals of the pooling model.

The "bp" test for unbalanced panels was derived in Baltagi and Li (1990) (1990), the "kw" test for unbalanced panels in Baltagi et al. (1998).

The "ghm" test and the "kw" test were extended to two-way effects in Baltagi et al. (1992).

For a concise overview of all these statistics see Baltagi (2003), Sec. 4.2, pp. 68–76 (for balanced panels) and Sec. 9.5, pp. 200–203 (for unbalanced panels).

Value

An object of class "htest".

Note

For the King-Wu statistics ("kw"), the oneway statistics ("individual" and "time") coincide with the respective Honda statistics ("honda"); twoway statistics of "kw" and "honda" differ.

Author(s)

Yves Croissant (initial implementation), Kevin Tappe (generalization to unbalanced panels)

References

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Baltagi BH, Li Q (1990). "A Lagrange multiplier test for the error components model with incomplete panels." *Econometric Reviews*, **9**, 103–107.

Baltagi BH, Chang YJ, Li Q (1992). "Monte Carlo results on several new and existing tests for the error components model." *Journal of Econometrics*, **54**, 95–120.

Baltagi B, Chang YA, Li Q (1998). "Testing for random individual and time effects using unbalanced panel data." *Advances in econometrics*, **13**, 1-20.

Breusch TS, Pagan AR (1980). "The Lagrange Multiplier Test and Its Applications to Model Specification in Econometrics." *Review of Economic Studies*, **47**, 239–253.

Gourieroux C, Holly A, Monfort A (1982). "Likelihood Ratio Test, Wald Test, and Kuhn–Tucker Test in Linear Models With Inequality Constraints on the Regression Parameters." *Econometrica*, **50**, 63–80.

Honda Y (1985). "Testing the Error Components Model With Non–Normal Disturbances." *Review of Economic Studies*, **52**, 681–690.

King ML, Wu PX (1997). "Locally Optimal One–Sided Tests for Multiparameter Hypothese." *Econometric Reviews*, **33**, 523–529.

See Also

`pFtest()` for individual and/or time effects tests based on the within model.

Examples

```
data("Grunfeld", package = "plm")
g <- plm(inv ~ value + capital, data = Grunfeld, model = "pooling")
plmtest(g)
plmtest(g, effect="time")
plmtest(inv ~ value + capital, data = Grunfeld, type = "honda")
plmtest(inv ~ value + capital, data = Grunfeld, type = "bp")
plmtest(inv ~ value + capital, data = Grunfeld, type = "bp",  effect = "twoways")
plmtest(inv ~ value + capital, data = Grunfeld, type = "ghm", effect = "twoways")
plmtest(inv ~ value + capital, data = Grunfeld, type = "kw",  effect = "twoways")

Grunfeld_unbal <- Grunfeld[1:(nrow(Grunfeld)-1), ] # create an unbalanced panel data set
g_unbal <- plm(inv ~ value + capital, data = Grunfeld_unbal, model = "pooling")
plmtest(g_unbal) # unbalanced version of test is indicated in output
```

---

pmg *Mean Groups (MG), Demeaned MG and CCE MG estimators*

Description

Mean Groups (MG), Demeaned MG (DMG) and Common Correlated Effects MG (CCEMG) estimators for heterogeneous panel models, possibly with common factors (CCEMG)

Usage

```
pmg(
  formula,
  data,
  subset,
  na.action,
  model = c("mg", "cmg", "dmg"),
  index = NULL,
  trend = FALSE,
  ...
)

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

## S3 method for class 'summary.pmg'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)

## S3 method for class 'pmg'
residuals(object, ...)
```

Arguments

formula	a symbolic description of the model to be estimated,
data	a data.frame,
subset	see lm(),
na.action	see lm(),
model	one of "mg", "cmg", or "dmg",
index	the indexes, see pdata.frame(),
trend	logical specifying whether an individual-specific trend has to be included,
...	further arguments.
object, x	an object of class pmg,
digits	digits,
width	the maximum length of the lines in the print output,

Details

pmg is a function for the estimation of linear panel models with heterogeneous coefficients by various Mean Groups estimators. Setting argument model = "mg" specifies the standard Mean Groups estimator, based on the average of individual time series regressions. If model = "dmg" the data

are demeaned cross-sectionally, which is believed to reduce the influence of common factors (and is akin to what is done in homogeneous panels when `model = "within"` and `effect = "time"`). Lastly, if `model = "cmg"` the CCEMG estimator is employed which is consistent under the hypothesis of unobserved common factors and idiosyncratic factor loadings; it works by augmenting the model by cross-sectional averages of the dependent variable and regressors in order to account for the common factors, and adding individual intercepts and possibly trends.

Value

An object of class `c("pmg", "panelmodel")` containing:

`coefficients`	the vector of coefficients,
`residuals`	the vector of residuals,
`fitted.values`	the vector of fitted values,
`vcov`	the covariance matrix of the coefficients,
`df.residual`	degrees of freedom of the residuals,
`model`	a data.frame containing the variables used for the estimation,
`r.squared`	numeric, the R squared,
`call`	the call,
`indcoef`	the matrix of individual coefficients from separate time series regressions.

Author(s)

Giovanni Millo

References

Pesaran MH (2006). "Estimation and inference in large heterogeneous panels with a multifactor error structure." *Econometrica*, **74**(4), 967–1012.

Examples

```
data("Produc", package = "plm")
## Mean Groups estimator
mgmod <- pmg(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc)
summary(mgmod)

## demeaned Mean Groups
dmgmod <- pmg(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
             data = Produc, model = "dmg")
summary(dmgmod)

## Common Correlated Effects Mean Groups
ccemgmod <- pmg(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
                data = Produc, model = "cmg")
summary(ccemgmod)
```

`pmodel.response` *A function to extract the model.response*

Description

pmodel.response has several methods to conveniently extract the response of several objects.

Usage

```
pmodel.response(object, ...)

## S3 method for class 'plm'
pmodel.response(object, ...)

## S3 method for class 'data.frame'
pmodel.response(object, ...)

## S3 method for class 'formula'
pmodel.response(object, data, ...)
```

Arguments

`object`	an object of class `"plm"`, or a formula of class `"Formula"`,
`...`	further arguments.
`data`	a `data.frame`

Details

The model response is extracted from a `pdata.frame` (where the response must reside in the first column; this is the case for a model frame), a `Formula + data` (being a model frame) or a `plm` object, and the transformation specified by `effect` and `model` is applied to it.
Constructing the model frame first ensures proper `NA` handling and the response being placed in the first column, see also **Examples** for usage.

Value

A pseries except if model responses' of a `"between"` or `"fd"` model as these models "compress" the data (the number of observations used in estimation is smaller than the original data due to the specific transformation). A numeric is returned for the `"between"` and `"fd"` model.

Author(s)

Yves Croissant

See Also

`plm`'s `model.matrix()` for (transformed) model matrix and the corresponding `model.frame()` method to construct a model frame.

Examples

```
# First, make a pdata.frame
data("Grunfeld", package = "plm")
pGrunfeld <- pdata.frame(Grunfeld)

# then make a model frame from a Formula and a pdata.frame
form <- inv ~ value + capital
mf <- model.frame(pGrunfeld, form)

# retrieve (transformed) response directly from model frame
resp_mf <- pmodel.response(mf, model = "within", effect = "individual")

# retrieve (transformed) response from a plm object, i.e., an estimated model
fe_model <- plm(form, data = pGrunfeld, model = "within")
pmodel.response(fe_model)

# same as constructed before
all.equal(resp_mf, pmodel.response(fe_model), check.attributes = FALSE) # TRUE
```

---

pooltest *Test of Poolability*

Description

A Chow test for the poolability of the data.

Usage

```
pooltest(x, ...)

## S3 method for class 'plm'
pooltest(x, z, ...)

## S3 method for class 'formula'
pooltest(x, data, ...)
```

Arguments

x	an object of class "plm" for the plm method; an object of class "formula" for the formula interface,
...	further arguments passed to plm.
z	an object of class "pvcm" obtained with model="within",
data	a data.frame,

Details

pooltest is a *F* test of stability (or Chow test) for the coefficients of a panel model. For argument x, the estimated plm object should be a "pooling" model or a "within" model (the default); intercepts are assumed to be identical in the first case and different in the second case.

Value

An object of class "htest".

Author(s)

Yves Croissant

Examples

```
data("Gasoline", package = "plm")
form <- lgaspcar ~ lincomep + lrpmg + lcarpcap
gasw <- plm(form, data = Gasoline, model = "within")
gasp <- plm(form, data = Gasoline, model = "pooling")
gasnp <- pvcm(form, data = Gasoline, model = "within")
pooltest(gasw, gasnp)
pooltest(gasp, gasnp)

pooltest(form, data = Gasoline, effect = "individual", model = "within")
pooltest(form, data = Gasoline, effect = "individual", model = "pooling")
```

---

predict.plm *Model Prediction for plm Objects*

Description

Predicted values of response based on plm models.

Usage

```
## S3 method for class 'plm'
predict(
  object,
  newdata = NULL,
  na.fill = !inherits(newdata, "pdata.frame"),
  ...
)
```

Arguments

object	An object of class "plm",
newdata	An optional pdata.frame in which to look for variables to be used for prediction. If NULL, the fitted values are returned. For fixed effects models, supplying a pdata.frame is recommended.
na.fill	A logical, only relevant if object is a pdata.frame, indicating whether for any supplied out-of-sample indexes (individual, time, combination of both), the missing fixed effect estimate is filled with the weighted mean of the model's present fixed effect estimates or not.
...	further arguments.

Details

predictcalculates predicted values by evaluating the regression function of a plm model for newdata or, if newdata = NULL, it returns the fitted values the plm model.

The fixed effects (within) model is somewhat special in prediction as it has fixed effects estimated per individual, time period (one-way) or both (two-ways model) which should to be respected when predicting values relating to these fixed effects in the model: To do so, it is recommended to supply a pdata.frame (and not a plain data.frame) in newdata as it describes the relationship between the data supplied to the individual. and/or time periods. In case the newdata´'s pdata.frame has out-of-sample data (data contains individuals and/or time periods not contained in the original model), it is not clear how values are to be predicted and the result will contain NA values for these out-of-sample data. Argument na.fill can be set to TRUE to apply the original model's weighted mean of fixed effects for the out-of-sample data to derive a prediction.

If a plain data.frame is given in newdata for a fixed effects model, the weighted mean is used for all fixed effects as newdata for prediction as a plain data.frame cannot describe any relation to individuals/time periods (na.fill is automatically set to TRUE and the function warns).

See also **Examples**.

Value

A numeric (or a pseries if newdata is a pdata.frame) carrying the predicted values with length equal to the number of rows as the data supplied in newdata and with names the row names of newdata or, if newdata = NULL, the original model's fitted values given in object.

Examples

```
library(plm)
data("Grunfeld", package = "plm")

# fit a fixed effect model
fit.fe <- plm(inv ~ value + capital, data = Grunfeld, model = "within")

# generate 55 new observations of three firms used for prediction:
#  * firm 1 with years 1935:1964 (has out-of-sample years 1955:1964),
#  * firm 2 with years 1935:1949 (all in sample),
#  * firm 11 with years 1935:1944 (firm 11 is out-of-sample)
set.seed(42L)
```

```
new.value2   <- runif(55, min = min(Grunfeld$value),   max = max(Grunfeld$value))
new.capital2 <- runif(55, min = min(Grunfeld$capital), max = max(Grunfeld$capital))

newdata <- data.frame(firm = c(rep(1, 30), rep(2, 15), rep(11, 10)),
                      year = c(1935:(1935+29), 1935:(1935+14), 1935:(1935+9)),
                      value = new.value2, capital = new.capital2)
# make pdata.frame
newdata.p <- pdata.frame(newdata, index = c("firm", "year"))

## predict from fixed effect model with new data as pdata.frame
predict(fit.fe, newdata = newdata.p)

## set na.fill = TRUE to have the weighted mean used to for fixed effects -> no NA values
predict(fit.fe, newdata = newdata.p, na.fill = TRUE)

## predict with plain data.frame from fixed effect model: uses mean fixed effects
## for prediction and, thus, yields different result with a warning
predict(fit.fe, newdata = newdata)
```

`Produc` *US States Production*

Description

A panel of 48 observations from 1970 to 1986

Format

A data frame containing :

state the state

year the year

region the region

pcap public capital stock

hwy highway and streets

water water and sewer facilities

util other public buildings and structures

pc private capital stock

gsp gross state product

emp labor input measured by the employment in non–agricultural payrolls

unemp state unemployment rate

Details

total number of observations : 816

observation : regional

country : United States

Source

Online complements to Baltagi (2001):

https://www.wiley.com/legacy/wileychi/baltagi/

Online complements to Baltagi (2013):

https://bcs.wiley.com/he-bcs/Books?action=resource&bcsId=4338&itemId=1118672321&resourceId=13452

References

Baltagi BH (2001). *Econometric Analysis of Panel Data*, 3rd edition. John Wiley and Sons ltd.

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Baltagi BH, Pinnoi N (1995). "Public capital stock and state productivity growth: further evidence from an error components model." *Empirical Economics*, **20**, 351-359.

Munnell A (1990). "Why Has Productivity Growth Declined? Productivity and Public Investment." *New England Economic Review*, 3–22.

`pseries` *panel series*

Description

A class for panel series for which several useful computations and data transformations are available.

Usage

```
## S3 method for class 'pseries'
print(x, ...)

## S3 method for class 'pseries'
as.matrix(x, idbyrow = TRUE, ...)

## S3 method for class 'pseries'
plot(
  x,
  plot = c("lattice", "superposed"),
  scale = FALSE,
  transparency = TRUE,
```

```
  col = "blue",
  lwd = 1,
  ...
)

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

## S3 method for class 'summary.pseries'
plot(x, ...)

## S3 method for class 'summary.pseries'
print(x, ...)

Sum(x, ...)

## Default S3 method:
Sum(x, effect, ...)

## S3 method for class 'pseries'
Sum(x, effect = c("individual", "time", "group"), ...)

## S3 method for class 'matrix'
Sum(x, effect, ...)

Between(x, ...)

## Default S3 method:
Between(x, effect, ...)

## S3 method for class 'pseries'
Between(x, effect = c("individual", "time", "group"), ...)

## S3 method for class 'matrix'
Between(x, effect, ...)

between(x, ...)

## Default S3 method:
between(x, effect, ...)

## S3 method for class 'pseries'
between(x, effect = c("individual", "time", "group"), ...)

## S3 method for class 'matrix'
between(x, effect, ...)

Within(x, ...)
```

```
## Default S3 method:
Within(x, effect, ...)

## S3 method for class 'pseries'
Within(x, effect = c("individual", "time", "group", "twoways"), ...)

## S3 method for class 'matrix'
Within(x, effect, ...)
```

Arguments

x, object	a pseries or a matrix; or a summary.pseries object,
...	further arguments, e. g., na.rm = TRUE for transformation functions like between, see **Details** and **Examples**.
idbyrow	if TRUE in the as.matrix method, the lines of the matrix are the individuals,
plot, scale, transparency, col, lwd	plot arguments,
effect	for the pseries methods: character string indicating the "individual", "time", or "group" effect, for Within "twoways" additionally; for non-pseries methods, effect is a factor specifying the dimension ("twoways" is not possible),

Details

The functions between, Between, Within, and Sum perform specific data transformations, i. e., the between, within, and sum transformation, respectively.

between returns a vector/matrix containing the individual means (over time) with the length of the vector equal to the number of individuals (if effect = "individual" (default); if effect = "time", it returns the time means (over individuals)). Between duplicates the values and returns a vector/matrix which length/number of rows is the number of total observations. Within returns a vector/matrix containing the values in deviation from the individual means (if effect = "individual", from time means if effect = "time"), the so called demeaned data. Sum returns a vector/matrix with sum per individual (over time) or the sum per time period (over individuals) with effect = "individual" or effect = "time", respectively, and has length/ number of rows of the total observations (like Between).

For between, Between, Within, and Sum in presence of NA values it can be useful to supply na.rm = TRUE as an additional argument to keep as many observations as possible in the resulting transformation. na.rm is passed on to the mean()/sum() function used by these transformations (i.e., it does not remove NAs prior to any processing!), see also **Examples**.

Value

All these functions return an object of class pseries or a matrix, except:
between, which returns a numeric vector or a matrix; as.matrix, which returns a matrix.

Author(s)

Yves Croissant

See Also

is.pseries() to check if an object is a pseries. For more functions on class 'pseries' see lag(), lead(), diff() for lagging values, leading values (negative lags) and differencing.

Examples

```
# First, create a pdata.frame
data("EmplUK", package = "plm")
Em <- pdata.frame(EmplUK)

# Then extract a series, which becomes additionally a pseries
z <- Em$output
class(z)

# obtain the matrix representation
as.matrix(z)

# compute the between and within transformations
between(z)
Within(z)

# Between and Sum replicate the values for each time observation
Between(z)
Sum(z)

# between, Between, Within, and Sum transformations on other dimension
between(z, effect = "time")
Between(z, effect = "time")
Within(z, effect = "time")
Sum(z, effect = "time")

# NA treatment for between, Between, Within, and Sum
z2 <- z
z2[length(z2)] <- NA # set last value to NA
between(z2, na.rm = TRUE) # non-NA value for last individual
Between(z2, na.rm = TRUE) # only the NA observation is lost
Within(z2, na.rm = TRUE)  # only the NA observation is lost
Sum(z2, na.rm = TRUE)     # only the NA observation is lost

sum(is.na(Between(z2))) # 9 observations lost due to one NA value
sum(is.na(Between(z2, na.rm = TRUE))) # only the NA observation is lost
sum(is.na(Within(z2))) # 9 observations lost due to one NA value
sum(is.na(Within(z2, na.rm = TRUE))) # only the NA observation is lost
sum(is.na(Sum(z2))) # 9 observations lost due to one NA value
sum(is.na(Sum(z2, na.rm = TRUE))) # only the NA observation is lost
```

---

pseriesfy *Turn all columns of a pdata.frame into class pseries.*

Description

This function takes a pdata.frame and turns all of its columns into objects of class pseries.

Usage

```
pseriesfy(x, ...)
```

Arguments

x	an object of class "pdata.frame",
...	further arguments (currently not used).

Details

Background: Initially created pdata.frames have as columns the pure/basic class (e.g., numeric, factor, character). When extracting a column from such a pdata.frame, the extracted column is turned into a pseries.

At times, it can be convenient to apply data transformation operations on such a pseriesfy-ed pdata.frame, see Examples.

Value

A pdata.frame like the input pdata.frame but with all columns turned into pseries.

See Also

pdata.frame(), as.list()

Examples

```
library("plm")
data("Grunfeld", package = "plm")
pGrun <- pdata.frame(Grunfeld[ , 1:4], drop.index = TRUE)
pGrun2 <- pseriesfy(pGrun) # pseriesfy-ed pdata.frame

# compare classes of columns
lapply(pGrun,  class)
lapply(pGrun2, class)

# When using with()
with(pGrun,  lag(value)) # dispatches to base R's lag()
with(pGrun2, lag(value)) # dispatches to plm's lag() respect. panel structure

# When lapply()-ing
lapply(pGrun,  lag) # dispatches to base R's lag()
lapply(pGrun2, lag) # dispatches to plm's lag() respect. panel structure

# as.list(., keep.attributes = TRUE) on a non-pseriesfy-ed
# pdata.frame is similar and dispatches to plm's lag
lapply(as.list(pGrun, keep.attributes = TRUE), lag)
```

punbalancedness *Measures for Unbalancedness of Panel Data*

Description

This function reports unbalancedness measures for panel data as defined in Ahrens and Pincus (1981) and Baltagi et al. (2001).

Usage

```
punbalancedness(x, ...)

## S3 method for class 'pdata.frame'
punbalancedness(x, ...)

## S3 method for class 'data.frame'
punbalancedness(x, index = NULL, ...)

## S3 method for class 'panelmodel'
punbalancedness(x, ...)
```

Arguments

x	a panelmodel, a data.frame, or a pdata.frame object,
...	further arguments.
index	only relevant for data.frame interface, for details see pdata.frame(),

Details

punbalancedness returns measures for the unbalancedness of a panel data set.

- For two-dimensional data:
  The two measures of Ahrens and Pincus (1981) are calculated, called "gamma" (γ) and "nu" (ν).

If the panel data are balanced, both measures equal 1. The more "unbalanced" the panel data, the lower the measures (but > 0). The upper and lower bounds as given in Ahrens and Pincus (1981) are:
$0 < \gamma, \nu \leq 1$, and for ν more precisely $\frac{1}{n} < \nu \leq 1$, with n being the number of individuals (as in pdim(x)nTn).

- For nested panel data (meaning including a grouping variable):
  The extension of the above measures by Baltagi et al. (2001), p. 368, are calculated:

  - c1: measure of subgroup (individual) unbalancedness,
  - c2: measure of time unbalancedness,
  - c3: measure of group unbalancedness due to each group size.

Values are 1 if the data are balanced and become smaller as the data become more unbalanced.

An application of the measure "gamma" is found in e. g. Baltagi et al. (2001), pp. 488-491, and Baltagi and Chang (1994), pp. 78–87, where it is used to measure the unbalancedness of various unbalanced data sets used for Monte Carlo simulation studies. Measures c1, c2, c3 are used for similar purposes in Baltagi et al. (2001).

In the two-dimensional case, `punbalancedness` uses output of `pdim()` to calculate the two unbalancedness measures, so inputs to `punbalancedness` can be whatever `pdim` works on. `pdim` returns detailed information about the number of individuals and time observations (see `pdim()`).

Value

A named numeric containing either two or three entries, depending on the panel structure inputted:

- For the two-dimensional panel structure, the entries are called `gamma` and `nu`,
- For a nested panel structure, the entries are called `c1`, `c2`, `c3`.

Note

Calling `punbalancedness` on an estimated `panelmodel` object and on the corresponding `(p)data.frame` used for this estimation does not necessarily yield the same result (true also for `pdim`). When called on an estimated `panelmodel`, the number of observations (individual, time) actually used for model estimation are taken into account. When called on a `(p)data.frame`, the rows in the `(p)data.frame` are considered, disregarding any `NA` values in the dependent or independent variable(s) which would be dropped during model estimation.

Author(s)

Kevin Tappe

References

Ahrens H, Pincus R (1981). "On Two Measures of Unbalancedness in a One-Way Model and Their Relation to Efficiency." *Biometrical Journal*, **23**(3), 227-235. doi:10.1002/bimj.4710230302.

Baltagi BH, Chang YJ (1994). "Incomplete panels: a comparative study of alternative estimators for the unbalanced one-way error component regression model." *Journal of Econometrics*, **62**, 67-89.

Baltagi BH, Song SH, Jung BC (2001). "The unbalanced nested error component regression model." *Journal of Econometrics*, **101**, 357-381.

Baltagi BH, Song SH, Jung BC (2002). "A comparative study of alternative estimators for the unbalanced two-way error component regression model." *The Econometrics Journal*, **5**(2), 480–493. ISSN 13684221, 1368423X.

See Also

`nobs()`, `pdim()`, `pdata.frame()`

Examples

```
# Grunfeld is a balanced panel, Hedonic is an unbalanced panel
data(list=c("Grunfeld", "Hedonic"), package="plm")

# Grunfeld has individual and time index in first two columns
punbalancedness(Grunfeld) # c(1,1) indicates balanced panel
pdim(Grunfeld)$balanced   # TRUE

# Hedonic has individual index in column "townid" (in last column)
punbalancedness(Hedonic, index="townid") # c(0.472, 0.519)
pdim(Hedonic, index="townid")$balanced   # FALSE

# punbalancedness on estimated models
plm_mod_pool <- plm(inv ~ value + capital, data = Grunfeld)
punbalancedness(plm_mod_pool)

plm_mod_fe <- plm(inv ~ value + capital, data = Grunfeld[1:99, ], model = "within")
punbalancedness(plm_mod_fe)

# replicate results for panel data design no. 1 in Ahrens/Pincus (1981), p. 234
ind_d1  <- c(1,1,1,2,2,2,3,3,3,3,3,4,4,4,4,4,4,5,5,5,5,5,5,5)
time_d1 <- c(1,2,3,1,2,3,1,2,3,4,5,1,2,3,4,5,6,7,1,2,3,4,5,6,7)
df_d1 <- data.frame(individual = ind_d1, time = time_d1)
punbalancedness(df_d1) # c(0.868, 0.887)

# example for a nested panel structure with a third index variable
# specifying a group (states are grouped by region) and without grouping
data("Produc", package = "plm")
punbalancedness(Produc, index = c("state", "year", "region"))
punbalancedness(Produc, index = c("state", "year"))
```

`purtest` *Unit root tests for panel data*

Description

`purtest` implements several testing procedures that have been proposed to test unit root hypotheses with panel data.

Usage

```
purtest(
  object,
  data = NULL,
  index = NULL,
  test = c("levinlin", "ips", "madwu", "Pm", "invnormal", "logit", "hadri"),
  exo = c("none", "intercept", "trend"),
  lags = c("SIC", "AIC", "Hall"),
```

```
  pmax = 10,
  Hcons = TRUE,
  q = NULL,
  dfcor = FALSE,
  fixedT = TRUE,
  ips.stat = NULL,
  ...
)

## S3 method for class 'purtest'
print(x, ...)

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

## S3 method for class 'summary.purtest'
print(x, ...)
```

Arguments

object, x	Either a "data.frame" or a matrix containing the time series (individuals as columns), a "pseries" object, a formula; a "purtest" object for the print and summary methods,
data	a "data.frame" or a "pdata.frame" object (required for formula interface, see Details and Examples),
index	the indexes,
test	the test to be computed: one of "levinlin" for Levin et al. (2002), "ips" for Im et al. (2003), "madwu" for Maddala and Wu (1999), "Pm" , "invnormal", or "logit" for various tests as in Choi (2001), or "hadri" for Hadri (2000), see Details,
exo	the exogenous variables to introduce in the augmented Dickey–Fuller (ADF) regressions, one of: no exogenous variables ("none"), individual intercepts ("intercept"), or individual intercepts and trends ("trend"), but see Details,
lags	the number of lags to be used for the augmented Dickey-Fuller regressions: either a single value integer (the number of lags for all time series), a vector of integers (one for each time series), or a character string for an automatic computation of the number of lags, based on the AIC ("AIC"), the SIC ("SIC"), or on the method by Hall (1994) ("Hall"); argument is irrelevant for test = "hadri",
pmax	maximum number of lags (irrelevant for test = "hadri"),
Hcons	logical, only relevant for test = "hadri", indicating whether the heteroskedasticity-consistent test of Hadri (2000) should be computed,
q	the bandwidth for the estimation of the long-run variance (only relevant for test = "levinlin", the default (q = NULL) gives the value as suggested by the authors as round(3.21 * T^(1/3))),
dfcor	logical, indicating whether the standard deviation of the regressions is to be computed using a degrees-of-freedom correction,

fixedT	logical, indicating whether the individual ADF regressions are to be computed using the same number of observations (irrelevant for test = "hadri"),
ips.stat	NULL or character of length 1 to request a specific IPS statistic, one of "Wtbar" (also default if ips.stat = NULL), "Ztbar", "tbar",
...	further arguments (can set argument p.approx to be passed on to non-exported function padf to either "MacKinnon1994" or "MacKinnon1996" to force a specific method for p-value approximation, the latter only being possible if package 'urca' is installed).

Details

All these tests except "hadri" are based on the estimation of augmented Dickey-Fuller (ADF) regressions for each time series. A statistic is then computed using the t-statistics associated with the lagged variable. The Hadri residual-based LM statistic is the cross-sectional average of the individual KPSS statistics Kwiatkowski et al. (1992), standardized by their asymptotic mean and standard deviation.

Several Fisher-type tests that combine p-values from tests based on ADF regressions per individual are available:

- "madwu" is the inverse chi-squared test Maddala and Wu (1999), also called P test by Choi (2001).
- "Pm" is the modified P test proposed by Choi (2001) for large N,
- "invnormal" is the inverse normal test by Choi (2001), and
- "logit" is the logit test by Choi (2001).

The individual p-values for the Fisher-type tests are approximated as described in MacKinnon (1996) if the package **urca** (Pfaff (2008)) is available, otherwise as described in MacKinnon (1994).

For the test statistic tbar of the test of Im/Pesaran/Shin (2003) (ips.stat = "tbar"), no p-value is given but 1%, 5%, and 10% critical values are interpolated from paper's tabulated values via inverse distance weighting (printed and contained in the returned value's element statistic$ips.tbar.crit).

Hadri's test, the test of Levin/Lin/Chu, and the tbar statistic of Im/Pesaran/Shin are not applicable to unbalanced panels; the tbar statistic is not applicable when lags > 0 is given.

The exogenous instruments of the tests (where applicable) can be specified in several ways, depending on how the data is handed over to the function:

- For the formula/data interface (if data is a data.frame, an additional index argument should be specified); the formula should be of the form: y ~ 0, y ~ 1, or y ~ trend for a test with no exogenous variables, with an intercept, or with individual intercepts and time trend, respectively. The exo argument is ignored in this case.
- For the data.frame, matrix, and pseries interfaces: in these cases, the exogenous variables are specified using the exo argument.

With the associated summary and print methods, additional information can be extracted/displayed (see also Value).

Value

For purtest: An object of class "purtest": a list with the elements named:

- "statistic" (a "htest" object),
- "call",
- "args",
- "idres" (containing results from the individual regressions),
- "adjval" (containing the simulated means and variances needed to compute the statistic, for test = "levinlin" and "ips", otherwise NULL),
- "sigma2" (short-run and long-run variance for test = "levinlin", otherwise NULL).

Author(s)

Yves Croissant and for "Pm", "invnormal", and "logit" Kevin Tappe

References

Choi I (2001). "Unit root tests for panel data." *Journal of International Money and Finance*, **20**(2), 249 - 272. ISSN 0261-5606, https://doi.org/10.1016/S0261-5606(00)00048-6.

Hadri K (2000). "Testing for stationarity in heterogeneous panel data." *The Econometrics Journal*, **3**(2), 148–161. ISSN 13684221, 1368423X.

Hall A (1994). "Testing for a unit root in time series with pretest data-based model selection." *Journal of Business & Economic Statistics*, **12**(4), 461–470.

Im KS, Pesaran MH, Shin Y (2003). "Testing for unit roots in heterogenous panels." *Journal of Econometrics*, **115(1)**, 53-74.

Kwiatkowski D, Phillips PC, Schmidt P, Shin Y (1992). "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?" *Journal of Econometrics*, **54**(1), 159 - 178. ISSN 0304-4076, https://doi.org/10.1016/0304-4076(92)90104-Y.

Levin A, Lin CF, Chu CSJ (2002). "Unit root tests in panel data : asymptotic and finite-sample properties." *Journal of Econometrics*, **108**, 1-24.

MacKinnon JG (1994). "Approximate Asymptotic Distribution Functions for Unit-Root and Cointegration Tests." *Journal of Business & Economic Statistics*, **12**(2), 167–176. ISSN 07350015.

MacKinnon JG (1996). "Numerical Distribution Functions for Unit Root and Cointegration Tests." *Journal of Applied Econometrics*, **11**(6), 601–618. ISSN 08837252.

Maddala GS, Wu S (1999). "A comparative study of unit root tests with panel data and a new simple test." *Oxford Bulletin of Economics and Statistics*, **61**, 631-52.

Pfaff B (2008). *Analysis of Integrated and Cointegrated Time Series with R*, Second edition. Springer, New York. ISBN 0-387-27960-1, https://CRAN.r-project.org/package=urca.

See Also

cipstest(), phansitest()

Examples

```
data("Grunfeld", package = "plm")
y <- data.frame(split(Grunfeld$inv, Grunfeld$firm)) # individuals in columns

purtest(y, pmax = 4, exo = "intercept", test = "madwu")

## same via pseries interface
pGrunfeld <- pdata.frame(Grunfeld, index = c("firm", "year"))
purtest(pGrunfeld$inv, pmax = 4, exo = "intercept", test = "madwu")

## same via formula interface
purtest(inv ~ 1, data = Grunfeld, index = c("firm", "year"), pmax = 4, test = "madwu")
```

---

pvar *Check for Cross-Sectional and Time Variation*

Description

This function checks for each variable of a panel if it varies cross-sectionally and over time.

Usage

```
pvar(x, ...)

## S3 method for class 'matrix'
pvar(x, index = NULL, ...)

## S3 method for class 'data.frame'
pvar(x, index = NULL, ...)

## S3 method for class 'pdata.frame'
pvar(x, ...)

## S3 method for class 'pseries'
pvar(x, ...)

## S3 method for class 'pvar'
print(x, ...)
```

Arguments

x	a (p)data.frame or a matrix,
...	further arguments.
index	see pdata.frame(),

Details

For (p)data.frame and matrix interface: All-NA columns are removed prior to calculation of variation due to coercing to pdata.frame first.

Value

An object of class pvar containing the following elements:

- `id.variation` a logical vector with TRUE values if the variable has individual variation, FALSE if not,
- `time.variation` a logical vector with TRUE values if the variable has time variation, FALSE if not,
- `id.variation_anyNA` a logical vector with TRUE values if the variable has at least one individual-time combination with all NA values in the individual dimension for at least one time period, FALSE if not,
- `time.variation_anyNA` a logical vector with TRUE values if the variable has at least one individual-time combination with all NA values in the time dimension for at least one individual, FALSE if not.

Note

pvar can be time consuming for "big" panels. As a fast alternative `collapse::varying()` from package **collapse** could be used.

Author(s)

Yves Croissant

See Also

`pdim()` to check the dimensions of a 'pdata.frame' (and other objects),

Examples

```
# Gasoline contains two variables which are individual and time
# indexes and are the first two variables
data("Gasoline", package = "plm")
pvar(Gasoline)

# Hedonic is an unbalanced panel, townid is the individual index;
# the drop.index argument is passed to pdata.frame
data("Hedonic", package = "plm")
pvar(Hedonic, "townid", drop.index = TRUE)

# same using pdata.frame
Hed <- pdata.frame(Hedonic, "townid", drop.index = TRUE)
pvar(Hed)
```

```
# Gasoline with pvar's matrix interface
Gasoline_mat <- as.matrix(Gasoline)
pvar(Gasoline_mat)
pvar(Gasoline_mat, index=c("country", "year"))
```

---

pvcm *Variable Coefficients Models for Panel Data*

Description

Estimators for random and fixed effects models with variable coefficients.

Usage

```
pvcm(
  formula,
  data,
  subset,
  na.action,
  effect = c("individual", "time"),
  model = c("within", "random"),
  index = NULL,
  ...
)

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

## S3 method for class 'summary.pvcm'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)
```

Arguments

formula	a symbolic description for the model to be estimated,
data	a data.frame,
subset	see lm,
na.action	see lm,
effect	the effects introduced in the model: one of "individual", "time",
model	one of "within", "random",
index	the indexes, see pdata.frame(),

...	further arguments.
object, x	an object of class "pvcm",
digits	digits,
width	the maximum length of the lines in the print output,

Details

pvcm estimates variable coefficients models. Individual or time effects are introduced, respectively, if effect = "individual" (default) or effect = "time".

Coefficients are assumed to be fixed if model = "within", i.e., separate pooled OLS models are estimated per individual (effect = "individual") or per time period (effect = "time"). Coefficients are assumed to be random if model = "random" and the model by Swamy (1970) is estimated; it is a generalized least squares model which uses the results of the OLS models estimated per individual/time dimension (coefficient estimates are weighted averages of the single OLS estimates with weights inversely proportional to the variance-covariance matrices). The corresponding unbiased single coefficients, variance-covariance matrices, and standard errors of the random coefficients model are available in the returned object (see *Value*).

A test for parameter stability (homogeneous coefficients) of the random coefficients model is printed in the model's summary and is available in the returned object (see *Value*).

pvcm objects have print, summary and print.summary methods.

Value

An object of class c("pvcm", "panelmodel"), which has the following elements:

coefficients	the vector (numeric) of coefficients (or data frame for fixed effects),
residuals	the vector (numeric) of residuals,
fitted.values	the vector of fitted values,
vcov	the covariance matrix of the coefficients (a list for fixed effects model (model = "within")),
df.residual	degrees of freedom of the residuals,
model	a data frame containing the variables used for the estimation,
call	the call,
args	the arguments of the call,

random coefficients model only (model = "random"):

Delta	the estimation of the covariance matrix of the coefficients,
single.coefs	matrix of unbiased coefficients of single estimations,
single.vcov	list of variance-covariance matrices for single.coefs,
single.std.error	matrix of standard errors of single.coefs,
chisq.test	htest object: parameter stability test (homogeneous coefficients),

separate OLS estimations only (model = "within"):

std.error	a data frame containing standard errors for all coefficients for each single regression.

Author(s)

Yves Croissant, Kevin Tappe

References

Swamy PAVB (1970). "Efficient Inference in a Random Coefficient Regression Model." *Econometrica*, **38**, 311–323.

Swamy PAVB (1971). *Statistical Inference in Random Coefficient Regression Models*. Springer.

Greene WH (2018). *Econometric Analysis*, 8th edition. Prentice Hall.

Poi BP (2003). "From the help desk: Swamy's random-coefficients model." *Stata Journal*, **3**(3), 302–308.

Kleiber C, Zeileis A (2010). "The Grunfeld Data at 50." *German Economic Review*, **11**, 404-417. https://doi.org/10.1111/j.1468-0475.2010.00513.x.

Examples

```
data("Produc", package = "plm")
zw <- pvcm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc, model = "within")
zr <- pvcm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc, model = "random")

## replicate Greene (2018), p. 452, table 11.22/(2012), p. 419, table 11.14
summary(pvcm(log(gsp) ~ log(pc) + log(hwy) + log(water) + log(util) + log(emp) + unemp,
             data = Produc, model = "random"))

## replicate Poi (2003) (need data adjustment, remaining tiny diffs are due
## Poi's data set having more digits, not justified by the original Grunfeld data)
data(Grunfeld) # need firm = 1, 4, 3, 8, 2
Gr.Poi.2003 <- Grunfeld[c(1:20, 61:80, 41:60, 141:160, 21:40), ]
Gr.Poi.2003$firm <- rep(1:5, each = 20)
Gr.Poi.2003[c(86, 98), "inv"] <- c(261.6, 645.2)
Gr.Poi.2003[c(92), "capital"] <- c(232.6)

mod.poi <- pvcm(inv ~ value + capital, data = Gr.Poi.2003, model = "random")
summary(mod.poi)
print(mod.poi$single.coefs)
print(mod.poi$single.std.err)

## Not run:
# replicate Swamy (1971), p. 166, table 5.2
data(Grunfeld, package = "AER") # 11 firm Grunfeld data needed from package AER
gw <- pvcm(invest ~ value + capital, data = Grunfeld, index = c("firm", "year"))
# close replication of Swamy (1970), (7.4) [remaining diffs likely due to less
# precise numerical methods in the 1970, as supposed in Kleiber/Zeileis (2010), p. 9]
gr <- pvcm(invest ~ value + capital, data = Grunfeld, index = c("firm", "year"), model = "random")

## End(Not run)
```

pwaldtest *Wald-style Chi-square Test and F Test*

Description

Wald-style Chi-square test and F test of slope coefficients being zero jointly, including robust versions of the tests.

Usage

```
pwaldtest(x, ...)

## S3 method for class 'plm'
pwaldtest(
  x,
  test = c("Chisq", "F"),
  vcov = NULL,
  df2adj = (test == "F" && !is.null(vcov) && missing(.df2)),
  .df1,
  .df2,
  ...
)

## S3 method for class 'pvcm'
pwaldtest(x, ...)

## S3 method for class 'pgmm'
pwaldtest(x, param = c("coef", "time", "all"), vcov = NULL, ...)
```

Arguments

x	an estimated model of which the coefficients should be tested (usually of class "plm"/"pvcm"/"pgmm")‘,
...	further arguments (currently none).
test	a character, indicating the test to be performed, may be either "Chisq" or "F" for the Wald-style Chi-square test or F test, respectively,
vcov	NULL by default; a matrix giving a variance–covariance matrix or a function which computes such; if supplied (non NULL), the test is carried out using the variance–covariance matrix indicated resulting in a robust test,
df2adj	logical, only relevant for test = "F", indicating whether the adjustment for clustered standard errors for the second degrees of freedom parameter should be performed (see **Details**, also for further requirements regarding the variance–covariance matrix in vcov for the adjustment to be performed),
.df1	a numeric, used if one wants to overwrite the first degrees of freedom parameter in the performed test (usually not used),

`.df2`	a numeric, used if one wants to overwrite the second degrees of freedom parameter for the F test (usually not used),
`param`	(for pgmm method only): select the parameters to be tested: `"coef"`, `"time"`, or ‘"all"‘.

Details

`pwaldtest` can be used stand–alone with a plm object, a pvcm object, and a pgmm object (for pvcm objects only the 'random' type is valid and no further arguments are processed; for pgmm objects only arguments `param` and `vcov` are valid). It is also used in `summary.plm()` to produce the F statistic and the Chi-square statistic for the joint test of coefficients and in `summary.pgmm()`.

`pwaldtest` performs the test if the slope coefficients of a panel regression are jointly zero. It does not perform general purpose Wald-style tests (for those, see `lmtest::waldtest()` (from package **lmtest**) or `car::linearHypothesis()` (from package **car**)).

If a user specified variance-covariance matrix/function is given in argument `vcov`, the robust version of the tests are carried out. In that case, if the F test is requested (`test = "F"`) and no overwriting of the second degrees of freedom parameter is given (by supplying argument (`.df2`)), the adjustment of the second degrees of freedom parameter is performed by default. The second degrees of freedom parameter is adjusted to be the number of unique elements of the cluster variable - 1, e. g., the number of individuals minus 1. For the degrees of freedom adjustment of the F test in general, see e. g. Cameron and Miller (2015), section VII; (Andreß et al. 2013), pp. 126, footnote 4.

The degrees of freedom adjustment requires the vcov object supplied or created by a supplied function to carry an attribute called "cluster" with a known clustering described as a character (for now this could be either `"group"` or `"time"`). The vcovXX functions of the package **plm** provide such an attribute for their returned variance–covariance matrices. No adjustment is done for unknown descriptions given in the attribute "cluster" or when the attribute "cluster" is not present. Robust vcov objects/functions from package **clubSandwich** work as inputs to `pwaldtest`'s F test because a they are translated internally to match the needs described above.

Value

An object of class `"htest"`, except for pvcm's within model for which a data.frame with results of the Wald chi-square tests and F tests per regression is returned.

Author(s)

Yves Croissant (initial implementation) and Kevin Tappe (extensions: vcov argument and F test's df2 adjustment)

References

Wooldridge JM (2010). *Econometric Analysis of Cross–Section and Panel Data*, 2nd edition. MIT Press.

Andreß H, Golsch K, Schmidt-Catran A (2013). *Applied Panel Data Analysis for Economic and Social Surveys*. Springer. doi:10.1007/9783642329142.

Cameron AC, Miller DL (2015). "A Practitioner's Guide to Cluster-Robust Inference." *Journal of Human Resources*, **50**(2), 317-372. https://ideas.repec.org/a/uwp/jhriss/v50y2015i2p317-372.html.

See Also

vcovHC() for an example of the vcovXX functions, a robust estimation for the variance–covariance matrix; summary.plm()

Examples

```
data("Grunfeld", package = "plm")
mod_fe <- plm(inv ~ value + capital, data = Grunfeld, model = "within")
mod_re <- plm(inv ~ value + capital, data = Grunfeld, model = "random")
pwaldtest(mod_fe, test = "F")
pwaldtest(mod_re, test = "Chisq")

# with robust vcov (matrix, function)
pwaldtest(mod_fe, vcov = vcovHC(mod_fe))
pwaldtest(mod_fe, vcov = function(x) vcovHC(x, type = "HC3"))

pwaldtest(mod_fe, vcov = vcovHC(mod_fe), df2adj = FALSE) # w/o df2 adjustment

# example without attribute "cluster" in the vcov
vcov_mat <- vcovHC(mod_fe)
attr(vcov_mat, "cluster") <- NULL  # remove attribute
pwaldtest(mod_fe, vcov = vcov_mat) # no df2 adjustment performed
```

---

pwartest *Wooldridge Test for AR(1) Errors in FE Panel Models*

Description

Test of serial correlation for (the idiosyncratic component of) the errors in fixed–effects panel models.

Usage

```
pwartest(x, ...)

## S3 method for class 'formula'
pwartest(x, data, ...)

## S3 method for class 'panelmodel'
pwartest(x, ...)
```

Arguments

x	an object of class formula or of class panelmodel,
...	further arguments to be passed on to vcovHC (see Details and Examples).
data	a data.frame,

Details

As Wooldridge (2010), Sec. 10.5.4 observes, under the null of no serial correlation in the errors, the residuals of a FE model must be negatively serially correlated, with $cor(\hat{u}_{it}, \hat{u}_{is}) = -1/(T-1)$ for each t, s. He suggests basing a test for this null hypothesis on a pooled regression of FE residuals on their first lag: $\hat{u}_{i,t} = \alpha + \delta \hat{u}_{i,t-1} + \eta_{i,t}$. Rejecting the restriction $\delta = -1/(T-1)$ makes us conclude against the original null of no serial correlation.

`pwartest` estimates the `within` model and retrieves residuals, then estimates an AR(1) `pooling` model on them. The test statistic is obtained by applying a F test to the latter model to test the above restriction on δ, setting the covariance matrix to `vcovHC` with the option `method="arellano"` to control for serial correlation.

Unlike the `pbgtest()` and `pdwtest()`, this test does not rely on large–T asymptotics and has therefore good properties in "short" panels. Furthermore, it is robust to general heteroskedasticity.

Value

An object of class `"htest"`.

Author(s)

Giovanni Millo

References

Wooldridge JM (2002). *Econometric Analysis of Cross–Section and Panel Data*. MIT Press.

Wooldridge JM (2010). *Econometric Analysis of Cross–Section and Panel Data*, 2nd edition. MIT Press.

See Also

`pwfdtest()`, `pdwtest()`, `pbgtest()`, `pbltest()`, `pbsytest()`.

Examples

```
data("EmplUK", package = "plm")
pwartest(log(emp) ~ log(wage) + log(capital), data = EmplUK)

# pass argument 'type' to vcovHC used in test
pwartest(log(emp) ~ log(wage) + log(capital), data = EmplUK, type = "HC3")
```

`pwfdtest` *Wooldridge first–difference–based test for AR(1) errors in levels or first–differenced panel models*

Description

First–differencing–based test of serial correlation for (the idiosyncratic component of) the errors in either levels or first–differenced panel models.

Usage

```
pwfdtest(x, ...)

## S3 method for class 'formula'
pwfdtest(x, data, ..., h0 = c("fd", "fe"))

## S3 method for class 'panelmodel'
pwfdtest(x, ..., h0 = c("fd", "fe"))
```

Arguments

`x`	an object of class `formula` or a `"fd"`-model (plm object),
`...`	further arguments to be passed on to `vcovHC` (see Details and Examples).
`data`	a `data.frame`,
`h0`	the null hypothesis: one of `"fd"`, `"fe"`,

Details

As Wooldridge (2010), Sec. 10.6.3 observes, if the idiosyncratic errors in the model in levels are uncorrelated (which we label hypothesis `"fe"`), then the errors of the model in first differences (FD) must be serially correlated with $cor(\hat{e}_{it}, \hat{e}_{is}) = -0.5$ for each t, s. If on the contrary the levels model's errors are a random walk, then there must be no serial correlation in the FD errors (hypothesis `"fd"`). Both the fixed effects (FE) and the first–differenced (FD) estimators remain consistent under either assumption, but the relative efficiency changes: FE is more efficient under `"fe"`, FD under `"fd"`.

Wooldridge (ibid.) suggests basing a test for either hypothesis on a pooled regression of FD residuals on their first lag: $\hat{e}_{i,t} = \alpha + \rho \hat{e}_{i,t-1} + \eta_{i,t}$. Rejecting the restriction $\rho = -0.5$ makes us conclude against the null of no serial correlation in errors of the levels equation (`"fe"`). The null hypothesis of no serial correlation in differenced errors (`"fd"`) is tested in a similar way, but based on the zero restriction on ρ ($\rho = 0$). Rejecting `"fe"` favours the use of the first–differences estimator and the contrary, although it is possible that both be rejected.

`pwfdtest` estimates the `fd` model (or takes an `fd` model as input for the panelmodel interface) and retrieves its residuals, then estimates an AR(1) `pooling` model on them. The test statistic is obtained by applying a F test to the latter model to test the relevant restriction on ρ, setting the covariance matrix to `vcovHC` with the option `method="arellano"` to control for serial correlation.

Unlike the pbgtest and pdwtest, this test does not rely on large–T asymptotics and has therefore good properties in "short" panels. Furthermore, it is robust to general heteroskedasticity. The "fe" version can be used to test for error autocorrelation regardless of whether the maintained specification has fixed or random effects (see Drukker 2003).

Value

An object of class "htest".

Author(s)

Giovanni Millo

References

Drukker DM (2003). "Testing for Serial Correlation in Linear Panel–Data Models." *The Stata Journal*, **3**(2), 168-177.

Wooldridge JM (2002). *Econometric Analysis of Cross–Section and Panel Data*. MIT Press. Sec. 10.6.3, pp. 282–283.

Wooldridge JM (2010). *Econometric Analysis of Cross–Section and Panel Data*, 2nd edition. MIT Press. Sec. 10.6.3, pp. 319–320

See Also

pdwtest, pbgtest, pwartest,

Examples

```
data("EmplUK" , package = "plm")
pwfdtest(log(emp) ~ log(wage) + log(capital), data = EmplUK)
pwfdtest(log(emp) ~ log(wage) + log(capital), data = EmplUK, h0 = "fe")

# pass argument 'type' to vcovHC used in test
pwfdtest(log(emp) ~ log(wage) + log(capital), data = EmplUK, type = "HC3", h0 = "fe")


# same with panelmodel interface
mod <- plm(log(emp) ~ log(wage) + log(capital), data = EmplUK, model = "fd")
pwfdtest(mod)
pwfdtest(mod, h0 = "fe")
pwfdtest(mod, type = "HC3", h0 = "fe")
```

pwtest *Wooldridge's Test for Unobserved Effects in Panel Models*

Description

Semi-parametric test for the presence of (individual or time) unobserved effects in panel models.

Usage

```
pwtest(x, ...)

## S3 method for class 'formula'
pwtest(x, data, effect = c("individual", "time"), ...)

## S3 method for class 'panelmodel'
pwtest(x, effect = c("individual", "time"), ...)
```

Arguments

x	an object of class "formula", or an estimated model of class panelmodel,
...	further arguments passed to plm.
data	a data.frame,
effect	the effect to be tested for, one of "individual" (default) or "time",

Details

This semi-parametric test checks the null hypothesis of zero correlation between errors of the same group. Therefore, it has power both against individual effects and, more generally, any kind of serial correlation.

The test relies on large-N asymptotics. It is valid under error heteroskedasticity and departures from normality.

The above is valid if effect="individual", which is the most likely usage. If effect="time", symmetrically, the test relies on large-T asymptotics and has power against time effects and, more generally, against cross-sectional correlation.

If the panelmodel interface is used, the inputted model must be a pooling model.

Value

An object of class "htest".

Author(s)

Giovanni Millo

References

Wooldridge JM (2002). *Econometric Analysis of Cross–Section and Panel Data*. MIT Press.

Wooldridge JM (2010). *Econometric Analysis of Cross–Section and Panel Data*, 2nd edition. MIT Press.

See Also

pbltest(), pbgtest(), pdwtest(), pbsytest(), pwartest(), pwfdtest() for tests for serial correlation in panel models. plmtest() for tests for random effects.

Examples

```
data("Produc", package = "plm")
## formula interface
pwtest(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc)
pwtest(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc, effect = "time")

## panelmodel interface
# first, estimate a pooling model, than compute test statistics
form <- formula(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp)
pool_prodc <- plm(form, data = Produc, model = "pooling")
pwtest(pool_prodc) # == effect="individual"
pwtest(pool_prodc, effect="time")
```

---

r.squared *R squared and adjusted R squared for panel models*

Description

This function computes R squared or adjusted R squared for plm objects. It allows to define on which transformation of the data the (adjusted) R squared is to be computed and which method for calculation is used.

Usage

```
r.squared(object, model = NULL, type = c("cor", "rss", "ess"), dfcor = FALSE)
```

Arguments

object	an object of class "plm",
model	on which transformation of the data the R-squared is to be computed. If NULL, the transformation used to estimate the model is also used for the computation of R squared,
type	indicates method which is used to compute R squared. One of "rss" (residual sum of squares), "ess" (explained sum of squares), or "cor" (coefficient of correlation between the fitted values and the response),
dfcor	if TRUE, the adjusted R squared is computed.

Value

A numerical value. The R squared or adjusted R squared of the model estimated on the transformed data, e. g., for the within model the so called "within R squared".

See Also

`plm()` for estimation of various models; `summary.plm()` which makes use of `r.squared`.

Examples

```
data("Grunfeld", package = "plm")
p <- plm(inv ~ value + capital, data = Grunfeld, model = "pooling")
r.squared(p)
r.squared(p, dfcor = TRUE)
```

---

ranef.plm *Extract the Random Effects*

Description

Function to calculate the random effects from a `plm` object (random effects model).

Usage

```
## S3 method for class 'plm'
ranef(object, effect = NULL, ...)
```

Arguments

`object`	an object of class `"plm"`, needs to be a fitted random effects model,
`effect`	`NULL`, `"individual"`, or `"time"`, the effects to be extracted, see **Details**,
`...`	further arguments (currently not used).

Details

Function `ranef` calculates the random effects of a fitted random effects model. For one-way models, the effects of the estimated model are extracted (either individual or time effects). For two-way models, extracting the individual effects is the default (both, argument `effect = NULL` and `effect = "individual"` will give individual effects). Time effects can be extracted by setting `effect = "time"`.

Not all random effect model types are supported (yet?).

Value

A named numeric with the random effects per dimension (individual or time).

Author(s)

Kevin Tappe

See Also

`fixef()` to extract the fixed effects from a fixed effects model (within model).

Examples

```
data("Grunfeld", package = "plm")
m1 <- plm(inv ~ value + capital, data = Grunfeld, model = "random")
ranef(m1) # individual random effects

# compare to random effects by ML estimation via lme from package nlme
library(nlme)
m2 <- lme(inv ~ value + capital, random = ~1|firm, data = Grunfeld)
cbind("plm" = ranef(m1), "lme" = unname(ranef(m2)))

# two-ways RE model, calculate individual and time random effects
data("Cigar", package = "plm")
tw <- plm(sales ~ pop + price, data = Cigar, model = "random", effect = "twoways")
ranef(tw)                  # individual random effects
ranef(tw, effect = "time")  # time random effects
```

RiceFarms *Production of Rice in Indonesia*

Description

a panel of 171 observations

Format

A dataframe containing :

id the farm identifier

size the total area cultivated with rice, measured in hectares

status land status, on of `'owner'` (non sharecroppers, owner operators or leaseholders or both), `'share'` (sharecroppers), `'mixed'` (mixed of the two previous status)

varieties one of `'trad'` (traditional varieties), `'high'` (high yielding varieties) and `'mixed'` (mixed varieties)

bimas bIMAS is an intensification program; one of `'no'` (non-bimas farmer), `'yes'` (bimas farmer) or `'mixed'` (part but not all of farmer's land was registered to be in the bimas program)

seed seed in kilogram

urea urea in kilogram

phosphate phosphate in kilogram

pesticide pesticide cost in Rupiah

pseed price of seed in Rupiah per kg

purea price of urea in Rupiah per kg

pphosph price of phosphate in Rupiah per kg

hiredlabor hired labor in hours

famlabor family labor in hours

totlabor total labor (excluding harvest labor)

wage labor wage in Rupiah per hour

goutput gross output of rice in kg

noutput net output, gross output minus harvesting cost (paid in terms of rice)

price price of rough rice in Rupiah per kg

region one of `'wargabinangun'`, `'langan'`, `'gunungwangi'`, `'malausma'`, `'sukaambit'`, `'ciwangi'`

Details

number of observations : 1026

observation : farms

country : Indonesia

Source

Feng Q, Horrace WC (2012). "Alternative technical efficiency measures: Skew, bias and scale." *Journal of Applied Econometrics*, **27**(2), 253-268. doi:10.1002/jae.1190.

`sargan` *Hansen–Sargan Test of Overidentifying Restrictions*

Description

A test of overidentifying restrictions for models estimated by GMM.

Usage

```
sargan(object, ...)

## S3 method for class 'pgmm'
sargan(object, weights = c("twosteps", "onestep"), ...)
```

Arguments

`object`	an object of class `"pgmm"`,
`...`	further arguments (currently unused).
`weights`	the weighting matrix to be used for the computation of the test,

Details

The Hansen–Sargan test ("J test") calculates the quadratic form of the moment restrictions that is minimized while computing the GMM estimator. It follows asymptotically a chi-square distribution with number of degrees of freedom equal to the difference between the number of moment conditions and the number of coefficients.

Value

An object of class `"htest"`.

Author(s)

Yves Croissant

References

(Hansen 1982)

(Sargan 1958)

See Also

`pgmm()`

Examples

```
data("EmplUK", package = "plm")
ar <- pgmm(log(emp) ~ lag(log(emp), 1:2) + lag(log(wage), 0:1) +
           lag(log(capital), 0:2) + lag(log(output), 0:2) | lag(log(emp), 2:99),
           data = EmplUK, effect = "twoways", model = "twosteps")
sargan(ar)
```

Snmesp *Employment and Wages in Spain*

Description

A panel of 738 observations from 1983 to 1990

Format

A data frame containing:

firm firm index

year year

n log of employment

w log of wages

y log of real output

i log of intermediate inputs

k log of real capital stock

f real cash flow

Details

total number of observations: 5904

observation: firms

country: Spain

Source

Journal of Business Economics and Statistics data archive:
https://amstat.tandfonline.com/loi/ubes20/.

References

Alonso-Borrego C, Arellano M (1999). "Symmetrically Normalized Instrumental-Variable Estimation Using Panel Data." *Journal of Business and Economic Statistics*, **17**(1), 36-49.

`SumHes` *The Penn World Table, v. 5*

Description

A panel of 125 observations from 1960 to 1985

Format

A data frame containing :

year the year

country the country name (factor)

opec OPEC member?

com communist regime?

pop country's population (in thousands)

gdp real GDP per capita (in 1985 US dollars)

sr saving rate (in percent)

Details

total number of observations : 3250

observation : country

country : World

Source

Online supplements to Hayashi (2000).

http://fhayashi.fc2web.com/datasets.htm

References

Hayashi F (2000). *Econometrics*. Princeton University Press.

Summers R, Heston A (1991). "The Penn World Table (Mark 5): An Expanded Set of International Comparisons, 1950–1988." *The Quarterly Journal of Economics*, **106**, 327-68. doi:10.2307/2937941.

`summary.plm.list` *Summary for plm objects*

Description

The summary method for plm objects generates some more information about estimated plm models.

Usage

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

## S3 method for class 'summary.plm.list'
coef(object, eq = NULL, ...)

## S3 method for class 'summary.plm.list'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)

## S3 method for class 'plm'
summary(object, vcov = NULL, ...)

## S3 method for class 'summary.plm'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  subset = NULL,
  ...
)
```

Arguments

object	an object of class "plm",
...	further arguments.
eq	the selected equation for list objects
x	an object of class "summary.plm",
digits	number of digits for printed output,
width	the maximum length of the lines in the printed output,
vcov	a variance–covariance matrix furnished by the user or a function to calculate one (see **Examples**),
subset	a character or numeric vector indicating a subset of the table of coefficients to be printed for "print.summary.plm",

Details

The summary method for plm objects (summary.plm) creates an object of class c("summary.plm", "plm", "panelmodel") that extends the plm object it is run on with various information about the estimated model like (inferential) statistics, see **Value**. It has an associated print method (print.summary.plm).

Value

An object of class c("summary.plm", "plm", "panelmodel"). Some of its elements are carried over from the associated plm object and described there (plm()). The following elements are new or changed relative to the elements of a plm object:

fstatistic	'htest' object: joint test of significance of coefficients (F or Chi-square test) (robust statistic in case of supplied argument vcov, see pwaldtest() for details),
coefficients	a matrix with the estimated coefficients, standard errors, t–values, and p–values, if argument vcov was set to non-NULL the standard errors (and t– and p–values) in their respective robust variant,
vcov	the "regular" variance–covariance matrix of the coefficients (class "matrix"),
rvcov	only present if argument vcov was set to non-NULL: the furnished variance–covariance matrix of the coefficients (class "matrix"),
r.squared	a named numeric containing the R-squared ("rsq") and the adjusted R-squared ("adjrsq") of the model,
df	an integer vector with 3 components, (p, n-p, p*), where p is the number of estimated (non-aliased) coefficients of the model, n-p are the residual degrees of freedom (n being number of observations), and p* is the total number of coefficients (incl. any aliased ones).

Author(s)

Yves Croissant

See Also

plm() for estimation of various models; vcovHC() for an example of a robust estimation of variance–covariance matrix; r.squared() for the function to calculate R-squared; stats::print.power.htest() for some information about class "htest"; fixef() to compute the fixed effects for "within" (=fixed effects) models and within_intercept() for an "overall intercept" for such models; pwaldtest()

Examples

```
data("Produc", package = "plm")
zz <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
          data = Produc, index = c("state","year"))
summary(zz)

# summary with a furnished vcov, passed as matrix, as function, and
# as function with additional argument
data("Grunfeld", package = "plm")
wi <- plm(inv ~ value + capital,
          data = Grunfeld, model="within", effect = "individual")
summary(wi, vcov = vcovHC(wi))
summary(wi, vcov = vcovHC)
summary(wi, vcov = function(x) vcovHC(x, method = "white2"))

# extract F statistic
wi_summary <- summary(wi)
Fstat <- wi_summary[["fstatistic"]]

# extract estimates and p-values
est <- wi_summary[["coefficients"]][ , "Estimate"]
pval <- wi_summary[["coefficients"]][ , "Pr(>|t|)"]

# print summary only for coefficient "value"
print(wi_summary, subset = "value")
```

---

vcovBK *Beck and Katz Robust Covariance Matrix Estimators*

Description

Unconditional Robust covariance matrix estimators *a la Beck and Katz* for panel models (a.k.a. Panel Corrected Standard Errors (PCSE)).

Usage

```
vcovBK(x, ...)

## S3 method for class 'plm'
vcovBK(
  x,
```

```
  type = c("HC0", "sss", "HC1", "HC2", "HC3", "HC4"),
  cluster = c("group", "time"),
  diagonal = FALSE,
  ...
)
```

Arguments

x	an object of class "plm",
...	further arguments.
type	the weighting scheme used, one of "HC0", "sss", "HC1", "HC2", "HC3", "HC4", see Details,
cluster	one of "group", "time",
diagonal	a logical value specifying whether to force non-diagonal elements to zero,

Details

vcovBK is a function for estimating a robust covariance matrix of parameters for a panel model according to the Beck and Katz (1995) method, a.k.a. Panel Corrected Standard Errors (PCSE), which uses an unconditional estimate of the error covariance across time periods (groups) inside the standard formula for coefficient covariance. Observations may be clustered either by "group" to account for timewise heteroskedasticity and serial correlation or by "time" to account for cross-sectional heteroskedasticity and correlation. It must be borne in mind that the Beck and Katz formula is based on N- (T-) asymptotics and will not be appropriate elsewhere.

The diagonal logical argument can be used, if set to TRUE, to force to zero all non-diagonal elements in the estimated error covariances; this is appropriate if both serial and cross–sectional correlation are assumed out, and yields a timewise- (groupwise-) heteroskedasticity–consistent estimator.

Weighting schemes specified by type are analogous to those in sandwich::vcovHC() in package **sandwich** and are justified theoretically (although in the context of the standard linear model) by MacKinnon and White (1985) and Cribari–Neto (2004) (see Zeileis 2004).

The main use of vcovBK (and the other variance-covariance estimators provided in the package vcovHC, vcovNW, vcovDC, vcovSCC) is to pass it to plm's own functions like summary, pwaldtest, and phtest or together with testing functions from the lmtest and car packages. All of these typically allow passing the vcov or vcov. parameter either as a matrix or as a function, e.g., for Wald–type testing: argument vcov. to coeftest(), argument vcov to waldtest() and other methods in the **lmtest** package; and argument vcov. to linearHypothesis() in the **car** package (see the examples), see (see also Zeileis 2004), 4.1-2, and examples below.

Value

An object of class "matrix" containing the estimate of the covariance matrix of coefficients.

Author(s)

Giovanni Millo

References

Beck N, Katz JN (1995). "What to do (and not to do) with time-series cross-section data." *American Political Science Review*, **89**(03), 634–647.

Cribari–Neto F (2004). "Asymptotic Inference Under Heteroskedasticity of Unknown Form." *Computational Statistics & Data Analysis*, **45**, 215–233.

Greene WH (2003). *Econometric Analysis*, 5th edition. Prentice Hall.

MacKinnon JG, White H (1985). "Some Heteroskedasticity–Consistent Covariance Matrix Estimators With Improved Finite Sample Properties." *Journal of Econometrics*, **29**, 305–325.

Zeileis A (2004). "Econometric Computing With HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. https://www.jstatsoft.org/article/view/v011i10.

See Also

`sandwich::vcovHC()` from the **sandwich** package for weighting schemes (`type` argument).

Examples

```
data("Produc", package="plm")
zz <- plm(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp, data=Produc, model="random")
summary(zz, vcov = vcovBK)
summary(zz, vcov = function(x) vcovBK(x, type="HC1"))

## standard coefficient significance test
library(lmtest)
coeftest(zz)
## robust significance test, cluster by group
## (robust vs. serial correlation), default arguments
coeftest(zz, vcov.=vcovBK)
## idem with parameters, pass vcov as a function argument
coeftest(zz, vcov.=function(x) vcovBK(x, type="HC1"))
## idem, cluster by time period
## (robust vs. cross-sectional correlation)
coeftest(zz, vcov.=function(x) vcovBK(x, type="HC1", cluster="time"))
## idem with parameters, pass vcov as a matrix argument
coeftest(zz, vcov.=vcovBK(zz, type="HC1"))
## joint restriction test
waldtest(zz, update(zz, .~.-log(emp)-unemp), vcov=vcovBK)
## Not run:
## test of hyp.: 2*log(pc)=log(emp)
library(car)
linearHypothesis(zz, "2*log(pc)=log(emp)", vcov.=vcovBK)

## End(Not run)
```

vcovDC *Double-Clustering Robust Covariance Matrix Estimator*

Description

High-level convenience wrapper for double-clustering robust covariance matrix estimators *a la* Thompson (2011) and Cameron et al. (2011) for panel models.

Usage

```
vcovDC(x, ...)

## S3 method for class 'plm'
vcovDC(x, type = c("HC0", "sss", "HC1", "HC2", "HC3", "HC4"), ...)
```

Arguments

x	an object of class "plm" or "pcce"
...	further arguments
type	the weighting scheme used, one of "HC0", "sss", "HC1", "HC2", "HC3", "HC4", see Details,

Details

vcovDC is a function for estimating a robust covariance matrix of parameters for a panel model with errors clustering along both dimensions. The function is a convenience wrapper simply summing a group- and a time-clustered covariance matrix and subtracting a diagonal one *a la* White.

Weighting schemes specified by type are analogous to those in sandwich::vcovHC() in package **sandwich** and are justified theoretically (although in the context of the standard linear model) by MacKinnon and White (1985) and Cribari–Neto (2004) (see Zeileis 2004).

The main use of vcovDC (and the other variance-covariance estimators provided in the package vcovHC, vcovBK, vcovNW, vcovSCC) is to pass it to plm's own functions like summary, pwaldtest, and phtest or together with testing functions from the lmtest and car packages. All of these typically allow passing the vcov or vcov. parameter either as a matrix or as a function, e.g., for Wald–type testing: argument vcov. to coeftest(), argument vcov to waldtest() and other methods in the **lmtest** package; and argument vcov. to linearHypothesis() in the **car** package (see the examples), see (see also Zeileis 2004), 4.1-2, and examples below.

Value

An object of class "matrix" containing the estimate of the covariance matrix of coefficients.

Author(s)

Giovanni Millo

References

Cameron AC, Gelbach JB, Miller DL (2011). "Robust inference with multiway clustering." *Journal of Business & Economic Statistics*, **29**(2).

Cribari–Neto F (2004). "Asymptotic Inference Under Heteroskedasticity of Unknown Form." *Computational Statistics & Data Analysis*, **45**, 215–233.

MacKinnon JG, White H (1985). "Some Heteroskedasticity–Consistent Covariance Matrix Estimators With Improved Finite Sample Properties." *Journal of Econometrics*, **29**, 305–325.

Thompson SB (2011). "Simple formulas for standard errors that cluster by both firm and time." *Journal of Financial Economics*, **99**(1), 1–10.

Zeileis A (2004). "Econometric Computing With HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. https://www.jstatsoft.org/article/view/v011i10.

See Also

`sandwich::vcovHC()` from the **sandwich** package for weighting schemes (`type` argument).

Examples

```
data("Produc", package="plm")
zz <- plm(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp, data=Produc, model="pooling")
## as function input to plm's summary method (with and without additional arguments):
summary(zz, vcov = vcovDC)
summary(zz, vcov = function(x) vcovDC(x, type="HC1", maxlag=4))
## standard coefficient significance test
library(lmtest)
coeftest(zz)
## DC robust significance test, default
coeftest(zz, vcov.=vcovDC)
## idem with parameters, pass vcov as a function argument
coeftest(zz, vcov.=function(x) vcovDC(x, type="HC1", maxlag=4))
## joint restriction test
waldtest(zz, update(zz, .~.-log(emp)-unemp), vcov=vcovDC)
## Not run:
## test of hyp.: 2*log(pc)=log(emp)
library(car)
linearHypothesis(zz, "2*log(pc)=log(emp)", vcov.=vcovDC)

## End(Not run)
```

---

vcovG *Generic Lego building block for Robust Covariance Matrix Estimators*

Description

Generic Lego building block for robust covariance matrix estimators of the vcovXX kind for panel models.

Usage

```
vcovG(x, ...)

## S3 method for class 'plm'
vcovG(
  x,
  type = c("HC0", "sss", "HC1", "HC2", "HC3", "HC4"),
  cluster = c("group", "time"),
  l = 0,
  inner = c("cluster", "white", "diagavg"),
  ...
)

## S3 method for class 'pcce'
vcovG(
  x,
  type = c("HC0", "sss", "HC1", "HC2", "HC3", "HC4"),
  cluster = c("group", "time"),
  l = 0,
  inner = c("cluster", "white", "diagavg"),
  ...
)
```

Arguments

x	an object of class "plm" or "pcce"
...	further arguments
type	the weighting scheme used, one of "HC0", "sss", "HC1", "HC2", "HC3", "HC4",
cluster	one of "group", "time",
l	lagging order, defaulting to zero
inner	the function to be applied to the residuals inside the sandwich: one of "cluster" or "white" or "diagavg", or a user specified R function,

Details

vcovG is the generic building block for use by higher–level wrappers vcovHC(), vcovSCC(), vcovDC(), and vcovNW(). The main use of vcovG is to be used internally by the former, but it is made available in the user space for use in non–standard combinations. For more documentation, see see wrapper functions mentioned.

Value

An object of class "matrix" containing the estimate of the covariance matrix of coefficients.

Author(s)

Giovanni Millo

References

Millo G (2017). "Robust standard error estimators for panel models: A unifying approach." *Journal of Statistical Software*, **82**(3), 1–27.

See Also

vcovHC(), vcovSCC(), vcovDC(), vcovNW(), and vcovBK() albeit the latter does not make use of vcovG.

Examples

```
data("Produc", package="plm")
zz <- plm(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp, data=Produc,
model="pooling")
## reproduce Arellano's covariance matrix
vcovG(zz, cluster="group", inner="cluster", l=0)
## define custom covariance function
## (in this example, same as vcovHC)
myvcov <- function(x) vcovG(x, cluster="group", inner="cluster", l=0)
summary(zz, vcov = myvcov)
## use in coefficient significance test
library(lmtest)
## robust significance test
coeftest(zz, vcov. = myvcov)
```

---

vcovHC.plm *Robust Covariance Matrix Estimators*

Description

Robust covariance matrix estimators *a la White* for panel models.

Usage

```
## S3 method for class 'plm'
vcovHC(
  x,
  method = c("arellano", "white1", "white2"),
  type = c("HC0", "sss", "HC1", "HC2", "HC3", "HC4"),
  cluster = c("group", "time"),
  ...
)

## S3 method for class 'pcce'
vcovHC(
  x,
  method = c("arellano", "white1", "white2"),
```

```
  type = c("HC0", "sss", "HC1", "HC2", "HC3", "HC4"),
  cluster = c("group", "time"),
  ...
)

## S3 method for class 'pgmm'
vcovHC(x, ...)
```

Arguments

x	an object of class "plm" which should be the result of a random effects or a within model or a model of class "pgmm" or an object of class "pcce",
method	one of "arellano", "white1", "white2",
type	the weighting scheme used, one of "HC0", "sss", "HC1", "HC2", "HC3", "HC4", see Details,
cluster	one of "group", "time",
...	further arguments.

Details

vcovHC is a function for estimating a robust covariance matrix of parameters for a fixed effects or random effects panel model according to the White method (White 1980, 1984; Arellano 1987). Observations may be clustered by "group" ("time") to account for serial (cross-sectional) correlation.

All types assume no intragroup (serial) correlation between errors and allow for heteroskedasticity across groups (time periods). As for the error covariance matrix of every single group of observations, "white1" allows for general heteroskedasticity but no serial (cross–sectional) correlation; "white2" is "white1" restricted to a common variance inside every group (time period) (see Greene 2003, Sec. 13.7.1-2, Greene 2012, Sec. 11.6.1-2

and Wooldridge 2002, Sec. 10.7.2); "arellano" (see

ibid. and the original ref. Arellano 1987) allows a fully general structure w.r.t. heteroskedasticity and serial (cross–sectional) correlation.

Weighting schemes specified by type are analogous to those in sandwich::vcovHC() in package **sandwich** and are justified theoretically (although in the context of the standard linear model) by MacKinnon and White (1985) and Cribari–Neto (2004) (Zeileis 2004). type = "sss" employs the small sample correction as used by Stata.

The main use of vcovHC (and the other variance-covariance estimators provided in the package vcovBK, vcovNW, vcovDC, vcovSCC) is to pass it to plm's own functions like summary, pwaldtest, and phtest or together with testing functions from the lmtest and car packages. All of these typically allow passing the vcov or vcov. parameter either as a matrix or as a function, e.g., for Wald–type testing: argument vcov. to coeftest(), argument vcov to waldtest() and other methods in the **lmtest** package; and argument vcov. to linearHypothesis() in the **car** package (see the examples), see (see also Zeileis 2004), 4.1-2, and examples below.

A method for pgmm objects, vcovHC.pgmm, is also provided and gives the robust variance-covariances matrix, in case of a two-steps panel GMM model with the small-sample correction proposed by Windmeijer (2005).

Value

An object of class "matrix" containing the estimate of the asymptotic covariance matrix of coefficients.

Note

The function pvcovHC is deprecated. Use vcovHC for the same functionality.

Author(s)

Giovanni Millo & Yves Croissant

References

Arellano M (1987). "Computing Robust Standard Errors for Within-groups Estimators." *Oxford bulletin of Economics and Statistics*, **49**(4), 431–434.

Cribari–Neto F (2004). "Asymptotic Inference Under Heteroskedasticity of Unknown Form." *Computational Statistics & Data Analysis*, **45**, 215–233.

Greene WH (2003). *Econometric Analysis*, 5th edition. Prentice Hall.

Greene WH (2012). *Econometric Analysis*, 7th edition. Prentice Hall.

MacKinnon JG, White H (1985). "Some Heteroskedasticity–Consistent Covariance Matrix Estimators With Improved Finite Sample Properties." *Journal of Econometrics*, **29**, 305–325.

Windmeijer F (2005). "A Finite Sample Correction for the Variance of Linear Efficient Two–Steps GMM Estimators." *Journal of Econometrics*, **126**, 25–51.

White H (1984). *Asymptotic Theory for Econometricians*. New York: Academic press. chap. 6

White H (1980). "A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity." *Econometrica*, **48**(4), 817–838.

Wooldridge JM (2002). *Econometric Analysis of Cross–Section and Panel Data*. MIT Press.

Zeileis A (2004). "Econometric Computing With HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. https://www.jstatsoft.org/article/view/v011i10.

See Also

sandwich::vcovHC() from the **sandwich** package for weighting schemes (type argument).

Examples

```
data("Produc", package = "plm")
zz <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
          data = Produc, model = "random")
## as function input to plm's summary method (with and without additional arguments):
summary(zz, vcov = vcovHC)
summary(zz, vcov = function(x) vcovHC(x, method="arellano", type="HC1"))

## standard coefficient significance test
library(lmtest)
coeftest(zz)
```

```
## robust significance test, cluster by group
## (robust vs. serial correlation)
coeftest(zz, vcov.=vcovHC)
## idem with parameters, pass vcov as a function argument
coeftest(zz, vcov.=function(x) vcovHC(x, method="arellano", type="HC1"))
## idem, cluster by time period
## (robust vs. cross-sectional correlation)
coeftest(zz, vcov.=function(x) vcovHC(x, method="arellano",
 type="HC1", cluster="group"))
## idem with parameters, pass vcov as a matrix argument
coeftest(zz, vcov.=vcovHC(zz, method="arellano", type="HC1"))
## joint restriction test
waldtest(zz, update(zz, .~.-log(emp)-unemp), vcov=vcovHC)
## Not run:
## test of hyp.: 2*log(pc)=log(emp)
library(car)
linearHypothesis(zz, "2*log(pc)=log(emp)", vcov.=vcovHC)

## End(Not run)
## Robust inference for CCE models
data("Produc", package = "plm")
ccepmod <- pcce(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp, data = Produc, model="p")
summary(ccepmod, vcov = vcovHC)

## Robust inference for GMM models
data("EmplUK", package="plm")
ar <- pgmm(log(emp) ~ lag(log(emp), 1:2) + lag(log(wage), 0:1)
           + log(capital) + lag(log(capital), 2) + log(output)
           + lag(log(output),2) | lag(log(emp), 2:99),
            data = EmplUK, effect = "twoways", model = "twosteps")
rv <- vcovHC(ar)
mtest(ar, order = 2, vcov = rv)
```

---

vcovNW *Newey and West (1987) Robust Covariance Matrix Estimator*

Description

Nonparametric robust covariance matrix estimators *a la Newey and West* for panel models with serial correlation.

Usage

```
vcovNW(x, ...)

## S3 method for class 'plm'
vcovNW(
  x,
  type = c("HC0", "sss", "HC1", "HC2", "HC3", "HC4"),
  maxlag = NULL,
```

```
  wj = function(j, maxlag) 1 - j/(maxlag + 1),
  ...
)

## S3 method for class 'pcce'
vcovNW(
  x,
  type = c("HC0", "sss", "HC1", "HC2", "HC3", "HC4"),
  maxlag = NULL,
  wj = function(j, maxlag) 1 - j/(maxlag + 1),
  ...
)
```

Arguments

x	an object of class "plm" or "pcce"
...	further arguments
type	the weighting scheme used, one of "HC0", "sss", "HC1", "HC2", "HC3", "HC4", see Details,
maxlag	either NULL or a positive integer specifying the maximum lag order before truncation
wj	weighting function to be applied to lagged terms,

Details

vcovNW is a function for estimating a robust covariance matrix of parameters for a panel model according to the Newey and West (1987) method. The function works as a restriction of the Driscoll and Kraay (1998) covariance (see vcovSCC()) to no cross–sectional correlation.

Weighting schemes specified by type are analogous to those in sandwich::vcovHC() in package **sandwich** and are justified theoretically (although in the context of the standard linear model) by MacKinnon and White (1985) and Cribari–Neto (2004) (see Zeileis 2004).

The main use of vcovNW (and the other variance-covariance estimators provided in the package vcovHC, vcovBK, vcovDC, vcovSCC) is to pass it to plm's own functions like summary, pwaldtest, and phtest or together with testing functions from the lmtest and car packages. All of these typically allow passing the vcov or vcov. parameter either as a matrix or as a function, e.g., for Wald–type testing: argument vcov. to coeftest(), argument vcov to waldtest() and other methods in the **lmtest** package; and argument vcov. to linearHypothesis() in the **car** package (see the examples), see (see also Zeileis 2004), 4.1-2, and examples below.

Value

An object of class "matrix" containing the estimate of the covariance matrix of coefficients.

Author(s)

Giovanni Millo

References

Cribari–Neto F (2004). "Asymptotic Inference Under Heteroskedasticity of Unknown Form." *Computational Statistics & Data Analysis*, **45**, 215–233.

Driscoll JC, Kraay AC (1998). "Consistent covariance matrix estimation with spatially dependent panel data." *Review of economics and statistics*, **80**(4), 549–560.

MacKinnon JG, White H (1985). "Some Heteroskedasticity–Consistent Covariance Matrix Estimators With Improved Finite Sample Properties." *Journal of Econometrics*, **29**, 305–325.

Newey WK, West KD (1987). "A Simple, Positive Semi-definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix." *Econometrica*, **55**(3), 703–08.

Zeileis A (2004). "Econometric Computing With HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. https://www.jstatsoft.org/article/view/v011i10.

See Also

`sandwich::vcovHC()` from the **sandwich** package for weighting schemes (`type` argument).

Examples

```
data("Produc", package="plm")
zz <- plm(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp, data=Produc, model="pooling")
## as function input to plm's summary method (with and without additional arguments):
summary(zz, vcov = vcovNW)
summary(zz, vcov = function(x) vcovNW(x, method="arellano", type="HC1"))
## standard coefficient significance test
library(lmtest)
coeftest(zz)
## NW robust significance test, default
coeftest(zz, vcov.=vcovNW)
## idem with parameters, pass vcov as a function argument
coeftest(zz, vcov.=function(x) vcovNW(x, type="HC1", maxlag=4))
## joint restriction test
waldtest(zz, update(zz, .~.-log(emp)-unemp), vcov=vcovNW)
## Not run:
## test of hyp.: 2*log(pc)=log(emp)
library(car)
linearHypothesis(zz, "2*log(pc)=log(emp)", vcov.=vcovNW)

## End(Not run)
```

`vcovSCC` *Driscoll and Kraay (1998) Robust Covariance Matrix Estimator*

Description

Nonparametric robust covariance matrix estimators *a la Driscoll and Kraay* for panel models with cross-sectional *and* serial correlation.

Usage

```
vcovSCC(x, ...)

## S3 method for class 'plm'
vcovSCC(
  x,
  type = c("HC0", "sss", "HC1", "HC2", "HC3", "HC4"),
  cluster = "time",
  maxlag = NULL,
  inner = c("cluster", "white", "diagavg"),
  wj = function(j, maxlag) 1 - j/(maxlag + 1),
  ...
)

## S3 method for class 'pcce'
vcovSCC(
  x,
  type = c("HC0", "sss", "HC1", "HC2", "HC3", "HC4"),
  cluster = "time",
  maxlag = NULL,
  inner = c("cluster", "white", "diagavg"),
  wj = function(j, maxlag) 1 - j/(maxlag + 1),
  ...
)
```

Arguments

x	an object of class "plm" or "pcce"
...	further arguments
type	the weighting scheme used, one of "HC0", "sss", "HC1", "HC2", "HC3", "HC4", see Details,
cluster	switch for vcovG; set at "time" here,
maxlag	either NULL or a positive integer specifying the maximum lag order before truncation
inner	the function to be applied to the residuals inside the sandwich: "cluster" for SCC, "white" for Newey-West, ("diagavg" for compatibility reasons)
wj	weighting function to be applied to lagged terms,

Details

vcovSCC is a function for estimating a robust covariance matrix of parameters for a panel model according to the Driscoll and Kraay (1998) method, which is consistent with cross–sectional and serial correlation in a T-asymptotic setting and irrespective of the N dimension. The use with random effects models is undocumented.

Weighting schemes specified by type are analogous to those in sandwich::vcovHC() in package **sandwich** and are justified theoretically (although in the context of the standard linear model) by MacKinnon and White (1985) and Cribari–Neto (2004) (see Zeileis 2004)).

The main use of vcovSCC (and the other variance-covariance estimators provided in the package vcovHC, vcovBK, vcovNW, vcovDC) is to pass it to plm's own functions like summary, pwaldtest, and phtest or together with testing functions from the lmtest and car packages. All of these typically allow passing the vcov or vcov. parameter either as a matrix or as a function, e.g., for Wald–type testing: argument vcov. to coeftest(), argument vcov to waldtest() and other methods in the **lmtest** package; and argument vcov. to linearHypothesis() in the **car** package (see the examples), (see also Zeileis 2004), 4.1-2, and examples below.

Value

An object of class "matrix" containing the estimate of the covariance matrix of coefficients.

Author(s)

Giovanni Millo, partially ported from Daniel Hoechle's (2007) Stata code

References

Cribari–Neto F (2004). "Asymptotic Inference Under Heteroskedasticity of Unknown Form." *Computational Statistics & Data Analysis*, **45**, 215–233.

Driscoll JC, Kraay AC (1998). "Consistent covariance matrix estimation with spatially dependent panel data." *Review of economics and statistics*, **80**(4), 549–560.

Hoechle D (2007). "Robust standard errors for panel regressions with cross-sectional dependence." *Stata Journal*, **7**(3), 281-312. https://ideas.repec.org/a/tsj/stataj/v7y2007i3p281-312.html.

MacKinnon JG, White H (1985). "Some Heteroskedasticity–Consistent Covariance Matrix Estimators With Improved Finite Sample Properties." *Journal of Econometrics*, **29**, 305–325.

Zeileis A (2004). "Econometric Computing With HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. https://www.jstatsoft.org/article/view/v011i10.

See Also

sandwich::vcovHC() from the **sandwich** package for weighting schemes (type argument).

Examples

```
data("Produc", package="plm")
zz <- plm(log(gsp)~log(pcap)+log(pc)+log(emp)+unemp, data=Produc, model="pooling")
## as function input to plm's summary method (with and without additional arguments):
summary(zz, vcov = vcovSCC)
summary(zz, vcov = function(x) vcovSCC(x, method="arellano", type="HC1"))
## standard coefficient significance test
library(lmtest)
coeftest(zz)
## SCC robust significance test, default
coeftest(zz, vcov.=vcovSCC)
## idem with parameters, pass vcov as a function argument
coeftest(zz, vcov.=function(x) vcovSCC(x, type="HC1", maxlag=4))
## joint restriction test
```

```
waldtest(zz, update(zz, .~.-log(emp)-unemp), vcov=vcovSCC)
## Not run:
## test of hyp.: 2*log(pc)=log(emp)
library(car)
linearHypothesis(zz, "2*log(pc)=log(emp)", vcov.=vcovSCC)

## End(Not run)
```

`Wages` *Panel Data of Individual Wages*

Description

A panel of 595 individuals from 1976 to 1982, taken from the Panel Study of Income Dynamics (PSID).

The data are organized as a stacked time series/balanced panel, see **Examples** on how to convert to a `pdata.frame`.

Format

A data frame containing:

exp years of full-time work experience.

wks weeks worked.

bluecol blue collar?

ind works in a manufacturing industry?

south resides in the south?

smsa resides in a standard metropolitan statistical area?

married married?

sex a factor with levels `"male"` and `"female"`

union individual's wage set by a union contract?

ed years of education.

black is the individual black?

lwage logarithm of wage.

Details

total number of observations : 4165

observation : individuals

country : United States

Source

Online complements to Baltagi (2001):

https://www.wiley.com/legacy/wileychi/baltagi/

Online complements to Baltagi (2013):

https://bcs.wiley.com/he-bcs/Books?action=resource&bcsId=4338&itemId=1118672321&resourceId=13452

References

Baltagi BH (2001). *Econometric Analysis of Panel Data*, 3rd edition. John Wiley and Sons ltd.

Baltagi BH (2013). *Econometric Analysis of Panel Data*, 5th edition. John Wiley and Sons ltd.

Cornwell C, Rupert P (1988). "Efficient Estimation With Panel Data: an Empirical Comparison of Instrumental Variables Estimators." *Journal of Applied Econometrics*, **3**, 149–155.

Examples

```
# data set 'Wages' is organized as a stacked time series/balanced panel
data("Wages", package = "plm")
Wag <- pdata.frame(Wages, index=595)
```

`within_intercept` *Overall Intercept for Within Models Along its Standard Error*

Description

This function gives an overall intercept for within models and its accompanying standard error or a within model with the overall intercept

Usage

```
within_intercept(object, ...)

## S3 method for class 'plm'
within_intercept(object, vcov = NULL, return.model = FALSE, ...)
```

Arguments

`object`	object of class `plm` which must be a within model (fixed effects model),
`...`	further arguments (currently none).
`vcov`	if not `NULL` (default), a function to calculate a user defined variance–covariance matrix (function for robust vcov), only used if `return.model = FALSE`,
`return.model`	a logical to indicate whether only the overall intercept (`FALSE` is default) or a full model object (`TRUE`) is to be returned,

Details

The (somewhat artificial) intercept for within models (fixed effects models) was made popular by Stata of StataCorp (see Gould 2013), EViews of IHS, and gretl (see Cottrell and Lucchetti 2021, p. 200-201, listing 23.1), see for treatment in the literature, e.g., Greene (2012), Ch. 11.4.4, p. 364. It can be considered an overall intercept in the within model framework and is the weighted mean of fixed effects (see **Examples** for the relationship).

`within_intercept` estimates a new model which is computationally more demanding than just taking the weighted mean. However, with `within_intercept` one also gets the associated standard error and it is possible to get an overall intercept for two-way fixed effect models.

Users can set argument `vcov` to a function to calculate a specific (robust) variance–covariance matrix and get the respective (robust) standard error for the overall intercept, e.g., the function `vcovHC()`, see examples for usage. Note: The argument `vcov` must be a function, not a matrix, because the model to calculate the overall intercept for the within model is different from the within model itself.

If argument `return.model = TRUE` is set, the full model object is returned, while in the default case only the intercept is returned.

Value

Depending on argument `return.model`: If `FALSE` (default), a named `numeric` of length one: The overall intercept for the estimated within model along attribute "se" which contains the standard error for the intercept. If `return.model = TRUE`, the full model object, a within model with the overall intercept (NB: the model identifies itself as a pooling model, e.g., in summary()).

Author(s)

Kevin Tappe

References

Cottrell A, Lucchetti R (2021). "Gretl User's Guide." https://gretl.sourceforge.net/.

Gould W (2013). "How can there be an intercept in the fixed-effects model estimated by xtreg, fe?" https://www.stata.com/support/faqs/statistics/intercept-in-fixed-effects-model/.

Greene WH (2012). *Econometric Analysis*, 7th edition. Prentice Hall.

See Also

`fixef()` to extract the fixed effects of a within model.

Examples

```
data("Hedonic", package = "plm")
mod_fe <- plm(mv ~ age + crim, data = Hedonic, index = "townid")
overallint <- within_intercept(mod_fe)
attr(overallint, "se") # standard error

# overall intercept is the weighted mean of fixed effects in the
```

```
# one-way case
weighted.mean(fixef(mod_fe), pdim(mod_fe)$Tint$Ti)

### relationship of type="dmean", "level" and within_intercept
## one-way balanced case
data("Grunfeld", package = "plm")
gi <- plm(inv ~ value + capital, data = Grunfeld, model = "within")
fx_level <- fixef(gi, type = "level")
fx_dmean <- fixef(gi, type = "dmean")
overallint <- within_intercept(gi)
all.equal(overallint + fx_dmean, fx_level, check.attributes = FALSE) # TRUE
## two-ways unbalanced case
gtw_u <- plm(inv ~ value + capital, data = Grunfeld[-200, ], effect = "twoways")
int_tw_u <- within_intercept(gtw_u)
fx_dmean_tw_i_u <- fixef(gtw_u, type = "dmean", effect = "individual")[index(gtw_u)[[1L]]]
fx_dmean_tw_t_u <- fixef(gtw_u, type = "dmean", effect = "time")[index(gtw_u)[[2L]]]
fx_level_tw_u <- as.numeric(fixef(gtw_u, "twoways", "level"))
fx_level_tw_u2 <- int_tw_u + fx_dmean_tw_i_u + fx_dmean_tw_t_u
all.equal(fx_level_tw_u, fx_level_tw_u2, check.attributes = FALSE) # TRUE

## overall intercept with robust standard error
within_intercept(gi, vcov = function(x) vcovHC(x, method="arellano", type="HC0"))

## have a model returned
mod_fe_int <- within_intercept(gi, return.model = TRUE)
summary(mod_fe_int)
# replicates Stata's robust standard errors exactly as model is with intercept
summary(mod_fe_int, vcov = function(x) vcovHC(x, type = "sss"))
```

Index