Package 'panglm'

November 3, 2026

Type Package

Title Generalized Linear Models for Panel Data

Version 1.7.0

Description Provides generalized linear models for panel data, including pooled, fixed-effects, and random-effects estimators for continuous, binary, count, ordinal, positive (gamma), and bounded (beta) outcomes, with the fixed-effects ordered logit of Baetschmann, Staub and Winkelmann (2015) <doi:10.1111/rssa.12090> and beta regression (Ferrari and Cribari-Neto 2004) <doi:10.1080/0266476042000214501>. Also offers correlated random effects (Mundlak 1978) <doi:10.2307/1913646>, dynamic models with the Wooldridge (2005) <doi:10.1002/jae.770> initial conditions solution, observation weights, average partial effects, regression-spline curves and linearity tests, difference-in-differences and event studies, and two-part models for costs (Duan et al. 1983) <doi:10.1080/07350015.1983.10509330>, including correlated random intercepts (Van Ourti 2004) <doi:10.1002/hec.920>. Estimators and tests follow standard panel-data references, including Hausman (1978) <doi:10.2307/1913827>, Chamberlain (1980) <doi:10.2307/2297110>, Allison and Waterman (2002) <doi:10.1111/1467-9531.00117>, and Croissant and Millo (2008) <doi:10.18637/jss.v027.i02>. Core numerical routines are implemented using 'Rcpp', 'RcppArmadillo', and 'RcppParallel'.

License MIT + file LICENSE

URL https://elbadisyimad.com/panglm/,
https://CRAN.R-project.org/package=panglm,
https://github.com/ielbadisy/panglm

BugReports https://github.com/ielbadisy/panglm/issues

Encoding UTF-8

Depends R (>= 3.6.0)

Imports Rcpp, grDevices, graphics, stats

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Suggests testthat (>= 3.0.0), plm, pglm, fixest, flexmix, sandwich, survival, MASS, AER, censReg, glmmTMB, marginaleffects, lme4, nlme, GLMMadaptive, ordinal, betareg, splines, pscl, sampleSelection, quantreg, generics, broom, modelsummary, did, etwfe, knitr, rmarkdown

Config/testthat/edition 3

SystemRequirements GNU make

RoxygenNote 7.3.3

VignetteBuilder knitr

LazyData true

LazyDataCompression xz

NeedsCompilation yes

Author Imad El Badisy [aut, cre]

Maintainer Imad El Badisy <elbadisyimad@gmail.com>

Repository CRAN

Date/Publication 2026-11-03 22:50:02 UTC

Contents

attgt . . . 4
cigar . . . 8
coefplot . . . 10
confint.panglm . . . 12
confint.twopart . . . 13
copd . . . 14
did . . . 16
docvisits . . . 18
effectcurve . . . 22
etwfe . . . 24
eventplot . . . 28
eventstudy . . . 30
eventstudy-methods . . . 32
fitted.panglm . . . 33
fixef . . . 34
glance.panglm . . . 36
healthspend . . . 37
hurdle-methods . . . 41
jointtest . . . 42
lineartest . . . 43
logLik.panglm . . . 44
medicaid . . . 45
meps . . . 49
model.matrix.panglm . . . 53
pandispersiontest . . . 55

panelplot . . . 56
panfmm . . . 57
panfmmselect . . . 61
panglm . . . 62
panglm-families . . . 67
pangmm . . . 69
panhausman . . . 73
panhurdle . . . 75
paniv . . . 77
panqr . . . 80
panselect . . . 84
partialeffects.panordered . . . 87
plot.panfmm . . . 89
plot.panglm . . . 90
plot.panglm_ape . . . 93
plot.panglm_attgt . . . 94
plot.panglm_effectcurve . . . 97
plot.panglm_etwfe . . . 98
plot.panglm_eventstudy . . . 101
plot.panglm_hurdle . . . 102
plot.pangmm . . . 103
plot.panqr . . . 105
plot.twopart . . . 106
predict.panfmm . . . 107
predict.panglm . . . 109
predict.pangmm . . . 110
predict.paniv . . . 111
predict.panordered . . . 112
predict.panselect . . . 113
predict.twopart . . . 115
residuals.panglm . . . 116
summary.panfmm . . . 117
summary.panglm . . . 117
summary.panglm_attgt . . . 118
summary.panglm_etwfe . . . 120
summary.pangmm . . . 121
summary.paniv . . . 122
summary.panqr . . . 123
summary.panselect . . . 123
tidy.panglm . . . 124
tidy.panglm_hurdle . . . 125
treatmentplot . . . 128
trendplot . . . 130
twopart . . . 132
vcov.panfmm . . . 134
vcov.panglm . . . 135
vcov.pangmm . . . 136
vcov.paniv . . . 137

vcov.panqr 138
vcov.panselect 139

Index **140**

`attgt` *Group-time average treatment effects for staggered adoption*

Description

The Callaway and Sant'Anna (2021) estimator of the group-time average treatment effects on the treated `ATT(g, t)`: the effect at period `t` for the cohort of individuals first treated at period `g`. Each `ATT(g, t)` is a two-group, two-period comparison of the change in the outcome between a base period and period `t` for cohort `g` against a clean control group (never-treated individuals, or individuals not yet treated by period `t`), so that already-treated individuals are never used as controls. This avoids the negative weights of the two-way fixed-effects regression of `did()` under staggered adoption with heterogeneous effects. The `ATT(g, t)` are then aggregated into an overall effect, an event study (effect by time since treatment), effects by cohort or by calendar period with `summary.panglm_attgt()`.

Usage

```
attgt(
  formula,
  data,
  index,
  treat,
  control = c("never", "notyet"),
  method = c("dr", "ipw", "reg"),
  base = c("varying", "universal"),
  anticipation = 0,
  level = 0.95,
  weights = NULL,
  cluster = NULL,
  bstrap = FALSE,
  cband = FALSE,
  biters = 1000,
  multiplier = c("rademacher", "mammen"),
  seed = NULL
)
```

Arguments

`formula`	`outcome ~ 1`, or `outcome ~ covariates` for conditional parallel trends (covariates are taken in the base period)
`data`	a data.frame in long format
`index`	`c(individual, time)`; the time index must be numeric

`treat`	name of the column holding each individual's first treated period (in the units of the time index); `NA`, `0` or `Inf` for units that are never treated
`control`	comparison group: `"never"` (never-treated individuals) or `"notyet"` (never treated or not yet treated, nor anticipating treatment, at both periods compared)
`method`	`"dr"` (doubly robust), `"ipw"` or `"reg"`; only matters with covariates
`base`	base period for pre-treatment `ATT(g, t)`: `"varying"` or `"universal"`
`anticipation`	number of periods before `g` in which treatment can already affect the outcome (the base period moves back accordingly)
`level`	confidence level of the reported intervals
`weights`	optional sampling weights: the name of a column of `data` or a numeric vector with one value per row of `data`. They must be non-negative and constant within each individual (rows with a missing weight are dropped). The weights enter the propensity score, the outcome regression and the means of every `ATT(g, t)`, and the cohort shares used by the aggregations, as `weightsname` in `did::att_gt()`.
`cluster`	optional name of a column of `data` (or a vector with one value per row) defining clusters of individuals, constant within each individual (for example the region of a province). Standard errors, the pre-trend test and the bootstrap are then clustered at that level instead of the individual level.
`bstrap`	logical: multiplier bootstrap standard errors (`FALSE` by default: analytic standard errors)
`cband`	logical: uniform confidence bands from the bootstrap simultaneous critical value (implies `bstrap = TRUE`)
`biters`	number of bootstrap draws
`multiplier`	bootstrap multipliers: `"rademacher"` (`-1` or `1` with equal probabilities, as in current versions of `did`) or `"mammen"` (the two-point distribution of Mammen 1993)
`seed`	optional integer seed for the bootstrap draws. When given, the draws are reproducible and the global random number generator is left as it was; otherwise a seed is drawn from it. Either way the seed is stored in the fit and reused by `summary.panglm_attgt()`.

Value

an object of class `"panglm_attgt"`: a list with `estimates` (a data frame with `group`, `time`, `event` (`time - group`), `estimate`, `se`, `lower`, `upper` and `post`), `inf` (the matrix of influence functions, one row per individual and one column per `ATT(g, t)`), `units` (individual identifiers, cohorts, normalized weights and clusters), `pretrend` (the Wald test of the pre-treatment `ATT(g, t)`, or `NULL`), `n` (individuals), `groups`, `periods`, `boot` (with the bootstrap: the settings, the analytic standard errors `se_analytic` and the simultaneous critical value `crit`; `NULL` otherwise) and the settings. With `cband = TRUE`, `estimates` also has the uniform band `band_lower`, `band_upper`. Methods: `print`, `summary`, `coef`, `vcov`, `nobs`, `plot`, `tidy` and `glance`.

Estimators

With `y ~ 1` (no covariates) every `ATT(g, t)` is the difference between the mean outcome change of cohort `g` and that of the control group, whatever `method`. With covariates, parallel trends is

assumed to hold conditionally on them; their values in the base period (the earlier of the two periods compared) are used, and the comparison is made by

- `method = "dr"`: the doubly robust estimator of Sant'Anna and Zhao (2020): a logit propensity score for belonging to cohort `g` and a linear regression of the outcome change among controls; consistent if either model is correct;
- `method = "ipw"`: normalized inverse probability weighting with the logit propensity score;
- `method = "reg"`: outcome regression (the linear regression of the outcome change fitted on the controls, averaged over the treated).

Controls with a propensity score of 0.995 or more are trimmed, as in the `DRDID` package. The panel is balanced first (individuals not observed in every period are dropped), individuals already treated in the first period are dropped, and cohorts first treated after the last period are treated as never treated. When no individual is never treated, the last cohort to be treated serves as the never-treated group and the periods from its treatment date onwards are dropped (the convention of `did::att_gt()`).

With `base = "varying"` (the default) pre-treatment `ATT(g, t)` compare period `t` with period `t - 1`, which gives placebo estimates of the change in trends; with `base = "universal"` every `ATT(g, t)` is measured against the last period before treatment, `g - 1 - anticipation`, whose own `ATT` is zero. Post-treatment effects are the same either way.

Inference

Standard errors are analytic and come from the influence function of each estimator, including the estimation of the propensity score and outcome regression and, for the aggregations, of the cohort shares used as weights; they are robust to serial correlation within individuals. They match `did::att_gt(bstrap = FALSE)` and `did::aggte()`. With `cluster`, the influence functions are summed within clusters (which must nest individuals) before computing the standard errors, as `clustervars` in `did`. The joint Wald test of all pre-treatment `ATT(g, t)` is reported as a test of parallel pre-trends.

With `bstrap = TRUE`, the standard errors come instead from the multiplier bootstrap of `did`: each bootstrap draw multiplies the (cluster sums of the) influence functions by independent Rademacher or Mammen weights, and the standard error is the interquartile range of the draws divided by that of the standard normal, which is robust to a few extreme draws. With `cband = TRUE` the bootstrap also gives the simultaneous critical value, the `level` quantile of the largest studentized deviation across all `ATT(g, t)` (or all levels of an aggregation), and the uniform confidence bands `band_lower`, `band_upper` cover all effects jointly with probability `level`. The aggregations of `summary.panglm_attgt()` reuse the same multiplier draws (they are generated from `seed`), so that they are coherent with the `ATT(g, t)`. Pointwise intervals (`lower`, `upper`) use the normal critical value with the bootstrap standard errors.

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

References

Callaway, B. and Sant'Anna, P. H. C. (2021). Difference-in-differences with multiple time periods. *Journal of Econometrics*, 225, 200-230. doi:10.1016/j.jeconom.2020.12.001

Sant'Anna, P. H. C. and Zhao, J. (2020). Doubly robust difference-in-differences estimators. *Journal of Econometrics*, 219, 101-122. doi:10.1016/j.jeconom.2020.06.003

Mammen, E. (1993). Bootstrap and wild bootstrap for high dimensional linear models. *The Annals of Statistics*, 21, 255-285. doi:10.1214/aos/1176349025

See Also

`summary.panglm_attgt()` for the aggregations, `etwfe()` for the extended two-way fixed-effects alternative (also for count and binary outcomes), `did()` and `eventstudy()` for the TWFE versions.

Other policy evaluation: `did()`, `etwfe()`, `eventplot()`, `eventstudy()`, `panelplot()`, `treatmentplot()`, `trendplot()`

Examples

```
# A health insurance scheme (for example the AMO extension in Morocco)
# is rolled out region by region: provinces of region A join in 2016,
# region B in 2018, region C in 2020, and region D not before 2022.
# Outcome: outpatient visits per 1,000 inhabitants.
set.seed(1)
d <- expand.grid(province = 1:120, year = 2012:2021)
region <- rep(c(2016, 2018, 2020, NA), each = 30)
d$adoption <- region[d$province]
d$urban <- rep(runif(120), 10)[d$province]
a <- rnorm(120)
e <- d$year - d$adoption
d$visits <- 50 + 5 * a[d$province] + 0.5 * (d$year - 2012) * (1 + d$urban) +
  ifelse(!is.na(e) & e >= 0, 2 + 0.5 * e, 0) + rnorm(nrow(d))
fit <- attgt(visits ~ 1, data = d, index = c("province", "year"),
             treat = "adoption")
fit
summary(fit)                     # overall ATT
summary(fit, type = "dynamic")   # event study
summary(fit, type = "group")
plot(fit)
# conditional parallel trends (trends differ by urbanization)
fitx <- attgt(visits ~ urban, data = d, index = c("province", "year"),
              treat = "adoption", control = "notyet", method = "dr")
summary(fitx, type = "dynamic")
# multiplier bootstrap, clustered by region, with uniform bands
d$region <- ceiling(d$province / 10)
fitb <- attgt(visits ~ 1, data = d, index = c("province", "year"),
              treat = "adoption", cluster = "region", cband = TRUE,
              biters = 499, seed = 1)
plot(fitb)
```

`cigar` *Cigarette demand in US states, 1963-1992 (Baltagi and Levin)*

Description

The classic real public panel of cigarette consumption, prices and income for 46 US states (45 states plus the District of Columbia; Alaska, Colorado, Hawaii, North Carolina and Oregon are not included) observed every year from 1963 to 1992 (30 years), used by Baltagi and Levin (1986, 1992) and in Baltagi's textbook to estimate a dynamic demand equation for an addictive good. The panel is balanced with no missing values. The first nine columns are the original file; the last four are the log variables of the textbook equation ln C(it) = a + b1 ln C(i,t-1) + b2 ln P(it) + b3 ln Pn(it) + b4 ln Y(it) + u(it). The build script, with the validation against the published estimates, is `data-raw/cigar.R` in the package's GitHub repository.

Usage

```
cigar
```

Format

A data frame with 1,380 rows (46 states x 30 years) and 13 columns. The panel index pair is (`state`, `year`).

state factor, two-letter US state abbreviation (46 levels, `DC` for the District of Columbia); the panel individual index

year integer calendar year, 1963-1992; the panel time index

price numeric, average retail price per pack of cigarettes, cents, nominal (taxes included)

pop numeric, state population, thousands

pop16 numeric, population of smoking age, thousands (labelled "population above the age of 16" in the data archive; the textbook describes smoking age as 14 and older)

cpi numeric, consumer price index, 1983 = 100

ndi numeric, per capita disposable income, dollars, nominal

sales numeric, cigarette sales, packs per head of total population

pimin numeric, minimum price per pack in adjoining states, cents, nominal (proxy for casual smuggling across state borders)

lnc numeric, log real per capita consumption by persons of smoking age, `log(sales * pop / pop16)`

lnp numeric, log real price, `log(price / cpi * 100)` (1983 cents)

lnpn numeric, log real minimum neighbouring-state price, `log(pimin / cpi * 100)`

lny numeric, log real per capita disposable income, `log(ndi / cpi * 100)` (1983 dollars)

Teaching use

- How habit forming is smoking, and what are the short- and long-run price elasticities? Dynamic demand `lnc ~ lag(lnc, 1) + lnp + lnpn + lny`: pooled OLS (`panglm(model = "pooling")`) overstates persistence because the state effect sits in the lagged outcome; two-way within (`model = "within", effect = "twoways"`) understates it (Nickell bias, of order 1 / T); Arellano-Bond difference GMM and Blundell-Bond system GMM (`pangmm()` with `transformation = "d"` or `"ld"`) are consistent for fixed T. A credible GMM estimate of the lag coefficient lies between the within (0.83) and pooled (0.97) values. The long-run elasticity is b2 / (1 - b1).
- How much do the GMM results depend on the instrument set? With N = 46 states the full Arellano-Bond instrument matrix has 437 columns; compare `collapse = TRUE` and `maxlag`, and read the Hansen test (it is uninformative, p close to 1, when instruments proliferate) and the AR(2) test.
- Does cross-border bootlegging matter? The coefficient of the neighbouring-state price `lnpn` in the static and dynamic models.
- Static demand: fixed versus random state effects for `lnc ~ lnp + lnpn + lny`, `panglm()` with `model = "within"` and `"random"`, and `panhausman()`. Here the classical contrast V_fe - V_re is not positive definite (the classical statistic is negative), so use the regression-based `method = "mundlak"` version, which rejects random effects (F = 7.17 on 3 and 45 df).
- Initial conditions: the Wooldridge dynamic correlated random effects model (`panglm(dynamic = TRUE)`), here a contrast with GMM because the state-effect variance is estimated at zero once the initial outcome is conditioned on.

Source

File `Cigar.txt` of the data archive of Baltagi, Econometric Analysis of Panel Data (Wiley companion site), https://www.wiley.com/legacy/wileychi/baltagi/supp/Cigar.txt, listed at https://www.wiley.com/legacy/wileychi/baltagi/datasets.html, downloaded 2026-11-01. The original variables are identical to the copies distributed under GPL (>= 2) in the CRAN packages **plm** (`plm::Cigar`) and **Ecdat** (`Ecdat::Cigar`), which trace the data to the same archive; this package redistributes it on the same terms. The underlying series are public statistics (Tobacco Institute, The Tax Burden on Tobacco; US Census population estimates; consumer price index and personal income series). Cite Baltagi and Levin (1992) and Baltagi (2005) when using the data.

References

Baltagi, B. H. and Levin, D. (1986). Estimating dynamic demand for cigarettes using panel data: the effects of bootlegging, taxation and advertising reconsidered. Review of Economics and Statistics, 68(1), 148-155.

Baltagi, B. H. and Levin, D. (1992). Cigarette taxation: raising revenues and reducing consumption. Structural Change and Economic Dynamics, 3(2), 321-335.

Baltagi, B. H., Griffin, J. M. and Xiong, W. (2000). To pool or not to pool: homogeneous versus heterogeneous estimators applied to cigarette demand. Review of Economics and Statistics, 82(1), 117-126.

Baltagi, B. H. (2005). Econometric Analysis of Panel Data, 3rd edition. Wiley. Section 8.8, Table 8.1 (Pooled estimation results, cigarette demand equation 1963-92) and Table 8.2 (Arellano-Bond estimates).

Arellano, M. and Bond, S. (1991). Some tests of specification for panel data: Monte Carlo evidence and an application to employment equations. Review of Economic Studies, 58(2), 277-297.

Nickell, S. (1981). Biases in dynamic models with fixed effects. Econometrica, 49(6), 1417-1426.

Examples

```
data(cigar)
ix <- c("state", "year")
f <- lnc ~ lag(lnc, 1) + lnp + lnpn + lny

## Baltagi (2005), Table 8.1: OLS, two-way within and one-step
## Arellano-Bond GMM
ols <- panglm(f, data = cigar, index = ix, model = "pooling")
fe  <- panglm(f, data = cigar, index = ix, model = "within",
              effect = "twoways")
ab1 <- pangmm(lnc ~ lag(lnc, 1) + lnp + lnpn + lny | lag(lnc, 2:99),
              data = cigar, index = ix, transformation = "d",
              model = "onestep")
round(rbind(OLS = coef(ols)[-1], Within = coef(fe),
            "GMM one-step" = coef(ab1)[1:4]), 3)
## published: 0.97 -0.090 0.024 -0.03 / 0.83 -0.299 0.034 0.10 /
##            0.84 -0.377 -0.016 0.14

## long-run price elasticity, b2 / (1 - b1)
lr <- function(b) b[["lnp"]] / (1 - b[[1]])
c(within = lr(coef(fe)), gmm = lr(coef(ab1)))

## collapsed instruments: system GMM, two-step, Windmeijer errors
sys <- pangmm(lnc ~ lag(lnc, 1) + lnp + lnpn + lny | lag(lnc, 2:99),
              data = cigar, index = ix, transformation = "ld",
              model = "twostep", collapse = TRUE, maxlag = 6)
summary(sys)

## static demand: fixed vs random state effects (regression-based
## Mundlak version of the Hausman test, robust to serial correlation)
sfe <- panglm(lnc ~ lnp + lnpn + lny, data = cigar, index = ix,
              model = "within")
panhausman(sfe, method = "mundlak", data = cigar)
```

`coefplot` *Compare coefficient estimates of several fitted models*

Description

Dot-and-whisker plot of the coefficients of one or more fitted models, terms on the vertical axis and Wald confidence intervals as horizontal segments. Several models are drawn side by side for each term (dodged vertically, with distinct plotting symbols and colours and a legend), so that, for example, pooled, fixed-effects, random-effects and correlated random effects fits of the same specification can be compared in one figure. Every term present in at least one model is shown; a model that does not contain a term has no mark on that row. The left margin is widened to the longest term label so that no label is clipped.

Usage

```
coefplot(
  ...,
  parm = NULL,
  level = 0.95,
  labels = NULL,
  main = "Coefficient estimates",
  xlab = "Estimate",
  col = NULL,
  pch = NULL,
  ref = 0
)
```

Arguments

`...`	fitted models, or a single unclassed list of fitted models. Argument names are used as legend labels.
`parm`	names or positions (in the order the terms first appear across the models) of the terms to plot; by default every term except `"(Intercept)"`
`level`	confidence level of the intervals
`labels`	legend labels of the models; by default the argument names, or the deparsed arguments
`main`	plot title
`xlab`	horizontal-axis label
`col`	colours of the models, recycled
`pch`	plotting symbols of the models, recycled
`ref`	position of the dashed reference line; `NULL` for none

Details

Any fitted object with `coef()` and `vcov()` methods can be used, e.g. `panglm()`, `paniv()`, `pangmm()`, `panfmm()` or `twopart()` fits, and also `stats::glm()` fits. When `vcov()` fails or has no entry for a term, only the estimate is drawn.

Value

the models, invisibly, as a named list

See Also

Other panglm methods: `confint.panglm()`, `effectcurve()`, `fitted.panglm()`, `fixef()`, `glance.panglm()`, `logLik.panglm()`, `model.matrix.panglm()`, `partialeffects.panordered()`, `plot.panglm()`, `plot.panglm_ape()`, `plot.panglm_effectcurve()`, `predict.panglm()`, `predict.panordered()`, `residuals.panglm()`, `summary.panglm()`, `tidy.panglm()`, `vcov.panglm()`

Examples

```
data(copd)
f <- exacerbations ~ crp + treatment + age
pool <- panglm(f, data = copd, index = c("id", "visit"),
               model = "pooling", family = "poisson")
re <- panglm(f, data = copd, index = c("id", "visit"),
             model = "random", family = "poisson")
cre <- panglm(f, data = copd, index = c("id", "visit"),
              model = "random", family = "poisson", cre = TRUE)
coefplot(Pooled = pool, RE = re, CRE = cre)
```

`confint.panglm` *Confidence intervals for panglm coefficients*

Description

Wald-type intervals `estimate +/- z * se`, using whichever `vcov()` the object currently carries (classical by default; HC1/cluster if the fit or a prior `vcov.panglm()` call set it).

Usage

```
## S3 method for class 'panglm'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	a `"panglm"` fit
`parm`	which parameters (names or indices); defaults to all
`level`	confidence level
`...`	unused

Value

a numeric matrix with one row per requested parameter and two columns giving the lower and upper confidence limits, named after the quantiles implied by `level` (e.g. `"2.5 %"` and `"97.5 %"`). Row names are the parameter names.

See Also

Other panglm methods: `coefplot()`, `effectcurve()`, `fitted.panglm()`, `fixef()`, `glance.panglm()`, `logLik.panglm()`, `model.matrix.panglm()`, `partialeffects.panordered()`, `plot.panglm()`, `plot.panglm_ape()`, `plot.panglm_effectcurve()`, `predict.panglm()`, `predict.panordered()`, `residuals.panglm()`, `summary.panglm()`, `tidy.panglm()`, `vcov.panglm()`

Examples

```
data(copd)
fit <- panglm(fev1 ~ crp, data = copd, index = c("id", "visit"),
              model = "within", family = "gaussian")
confint(fit)
```

`confint.twopart` *Confidence intervals for a two-part model*

Description

Wald intervals for the coefficients of both parts of a `twopart()` fit, named as in `coef()` (`"binary:..."`, `"positive:..."`), from `vcov()` (block diagonal for independent parts, the joint covariance for `correlated = TRUE`).

Usage

```
## S3 method for class 'twopart'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	a `"twopart"` fit
`parm`	names or positions of the coefficients; all by default
`level`	confidence level
`...`	unused

Value

a two-column matrix

See Also

Other twopart methods: `plot.twopart()`, `predict.twopart()`

Examples

```
data(copd)
copd$cost <- ifelse(copd$exacerbations > 0, exp(6 + 0.1 * copd$crp), 0)
fit <- twopart(cost ~ treatment + crp, data = copd, index = c("id", "visit"))
confint(fit)
```

copd *Synthetic COPD follow-up panel with costs and a staggered programme*

Description

A simulated longitudinal (panel) dataset of chronic obstructive pulmonary disease (COPD) patients followed over five clinic visits, used throughout this package's examples and vignettes. It is entirely synthetic (not real patient data), generated to have a realistic panel structure (time-invariant baseline covariates plus individual-level frailties or random intercepts). Because every data-generating value is known (see Details), it is the dataset on which students can check that an estimator recovers the truth before using it on the real panels shipped with the package (docvisits, meps, medicaid, healthspend, cigar). The generating code is in `data-raw/copd.R` in the package's GitHub repository.

Usage

```
copd
```

Format

A data frame with 300 rows (60 patients x 5 visits) and 16 columns; the panel index is `c("id", "visit")`:

id integer patient identifier (1-60), the panel individual index

visit integer visit number (1-5), the panel time index

treatment integer treatment arm, time-invariant (0 = control, 1 = active)

age numeric patient age in whole years at baseline, time-invariant

smoker integer current-smoker indicator, time-invariant (0/1)

bmi numeric body mass index (kg/m^2) at baseline, one decimal, time-invariant; not used to generate the outcomes

crp numeric C-reactive protein (mg/L) measured at each visit, one decimal and at least 0.1; the one time-varying covariate, needed for fixed-effects ("within") fits to have something left to estimate after demeaning

fev1 numeric forced expiratory volume in 1 second (litres), two decimals and at least 0.5; a continuous lung-function outcome, one value per visit

exacerbations integer number of COPD exacerbations since the previous visit; a count outcome, one value per visit

hospitalized integer indicator (0/1) of a hospital admission in the interval since the previous visit; a binary outcome, one value per visit

clinic integer clinic (1-6) treating the patient, time-invariant; 10 patients per clinic

adopt integer visit at which the patient's clinic adopted the care-management programme (3 or 4), 0 if the clinic never adopted; the `treat` column for `attgt()`, `etwfe()` and `eventstudy()`

programme integer indicator (0/1) that the programme is in place at this visit (`adopt > 0` and `visit >= adopt`)

cost numeric outpatient plus inpatient cost (euros, two decimals) since the previous visit; 28.7 percent zeros and a right-skewed positive part

adherence numeric proportion of prescribed inhaler doses taken since the previous visit, in (0, 1), three decimals

symptoms ordered factor of symptom severity at the visit: none < mild < moderate < severe

Details

The first ten columns (`id` to `hospitalized`) are unchanged from earlier releases. The last six (`clinic` to `symptoms`) add health economics outcomes and were generated afterwards in a separate random number stream, so the clinical columns did not change.

Data-generating values (patient random intercepts are normal with mean zero and are independent of the covariates; `visit` enters linearly).

- `exacerbations`: Poisson with mean `f_i exp(0.4 - 0.35 treatment + 0.02 (age - 65) + 0.3 smoker + 0.05 crp)` with a gamma frailty `f_i` of mean 1 and variance 0.25.
- `hospitalized`: logit with index `-1.5 - 0.6 treatment + 0.03 (age - 65) + 0.5 smoker + 0.08 crp + r_i`, `sd(r_i) = 0.4`.
- `fev1`: `3.2 - 0.02 (age - 65) - 0.25 smoker + 0.15 treatment - 0.03 crp + r_i + e`, `sd(e) = 0.2` (same `r_i`).
- `cost`, any cost: probit with index `-0.6 + 0.4 exacerbations + 1.5 hospitalized + 0.3 smoker - 0.2 treatment` `sd(a_i) = 0.5`.
- `cost`, positive part: lognormal, 'log(cost) = 5.8 + 0.1 exacerbations
  - 0.7 hospitalized + 0.25 smoker - 0.2 treatment + 0.04 crp + 0.05 visit
  - 0.5 programme + c_k + b_i + e, with sd(b_i) = 0.4, corr(a_i, b_i) = 0.5, sd(e) = 0.4and `clinic effectsc_k = (0.20, 0.10, 0.15, 0.00, -0.10, -0.15)for clinics 1 to 6. The programme therefo` - exp(-0.5)', about 39 percent, without changing the probability of any cost. Clinics 1 and 2 adopt at visit 3, clinics 3 and 4 at visit 4, clinics 5 and 6 never; adopting clinics have higher baseline costs (selection on levels) but parallel trends hold. Because both potential costs were simulated from the same draws, the sample average effect on the treated rows is known: about -406.0 euros per visit interval (100 treated patient-visits).
- `adherence`: beta with precision 25 and logit mean `0.8 + 0.6 treatment - 0.5 smoker - 0.08 visit + v_i`, `sd(v_i) = 0.3`.
- `symptoms`: ordered logit with latent index `0.5 smoker - 0.6 treatment + 0.25 crp + w_i`, `sd(w_i) = 0.8`, and cut points 0.5, 2.0 and 3.5.

Random-effects (conditional) estimators target these values; pooled nonlinear fits estimate population-averaged coefficients, which are attenuated for the binary, ordered and beta outcomes. With only 60 patients the estimates are noisy: `data-raw/copd.R` also refits the correctly specified models on a 6,000-patient panel drawn from the same design and checks that they land close to these values.

Teaching use

- Lung function and inflammation (`fev1` on `crp`): pooled, within, between and random effects, CRE and `panhausman()`.
- Exacerbation counts: `family = "poisson"` or `"negbin"`, `panhurdle()`, `panfmm()`.

- Hospital admission: `family = "binomial"` or `"probit"`, with `dynamic = TRUE` for state dependence.
- What drives outpatient and inpatient cost? `twopart()` (probit plus lognormal or gamma, `correlated = TRUE`), `family = "gamma"` on positive costs, `family = "tobit"`, `panqr()` for cost quantiles.
- Did the care-management programme lower costs? `attgt()`, `etwfe()`, `eventstudy()` with `treat = "adopt"`, against the known effect above.
- Adherence (`family = "beta"`) and symptom severity (`family = "ologit"`, within = fixed-effects ordered logit).

Source

Simulated; see `data-raw/copd.R` in the package's GitHub repository.

Examples

```
data(copd)
str(copd)
mean(copd$cost == 0)
# two-part model of cost, programme effect on the positive part
# (clinic dummies absorb the clinic cost levels that differ between
# adopting and never-adopting clinics)
tp <- twopart(cost ~ exacerbations + hospitalized + smoker + treatment +
                crp + visit + programme + factor(clinic),
              binary_formula = ~ exacerbations + hospitalized + smoker +
                treatment + crp,
              data = copd, index = c("id", "visit"), positive = "lognormal")
summary(tp)
# staggered adoption: group-time effects of the programme on cost
att <- attgt(cost ~ 1, data = copd, index = c("id", "visit"),
             treat = "adopt", control = "notyet")
summary(att)
```

---

did *Two-way fixed-effects difference-in-differences*

Description

Adds the indicator `treated = 1(t >= first treated period)` to the model and fits it with individual and time fixed effects (two-way fixed effects, TWFE). With a single treatment date (the canonical two-group, two-period design and its multi-period extension) the coefficient on `treated` is the average treatment effect on the treated under parallel trends. With staggered adoption and effects that vary over time or across cohorts the TWFE coefficient is a weighted average that can put negative weight on some effects (Goodman-Bacon 2021; de Chaisemartin and D'Haultfoeuille 2020): use `eventstudy()` to look at the dynamics and a heterogeneity-robust estimator (`attgt()` for Callaway and Sant'Anna 2021, or `etwfe()` for the extended TWFE of Wooldridge 2021, 2023) when adoption is staggered.

Usage

```
did(formula, data, index, treat, family = "gaussian", vcov = "cluster", ...)
```

Arguments

formula	outcome and control variables, e.g. `y ~ age` or `y ~ 1`; the treatment indicator is added
data	a data.frame in long format
index	`c(individual, time)`; the time index must be numeric
treat	name of the column holding each individual's first treated period (in the units of the time index); `NA`, `0` or `Inf` for units that are never treated
family	`"gaussian"`, `"poisson"`, `"negbin"` or `"binomial"` (conditional logit, logit link only; see Details)
vcov	covariance type, cluster-robust by individual by default
...	further arguments passed to `panglm()`

Details

The family can be Gaussian (effect on the mean), Poisson or negative binomial (effect on the log rate: `exp(coef)` is a ratio, and Poisson FE DiD needs only the conditional mean to be correct, Wooldridge 2023), or binomial for a binary outcome. The binomial model is the two-way conditional (Chamberlain) logit of `panglm()`: the individual effects are conditioned out exactly and the time effects enter as dummies inside the conditional likelihood, as in `survival::clogit(y ~ treated + factor(time) + strata(id))`. Its coefficient `treated` is then a log odds ratio, conditional on the individual effect (`exp(coef)` is the odds ratio), parallel trends is assumed on the log-odds scale, and individuals whose outcome never changes carry no information and are dropped. It is not an effect on the probability scale: for average effects on the treated in probability points use `etwfe()` with `family = "binomial"`. Controls that do not vary within individuals are absorbed by the individual effects and dropped.

Value

a `panglm()` fit whose coefficient `treated` is the DiD estimate (a log odds ratio for `family = "binomial"`), with a component `did` (number of treated and never-treated individuals, treatment cohorts). Its `call` is the `did()` call. All `panglm()` methods apply.

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

References

Goodman-Bacon, A. (2021). Difference-in-differences with variation in treatment timing. *Journal of Econometrics*, 225, 254-277. doi:10.1016/j.jeconom.2021.03.014

de Chaisemartin, C. and D'Haultfoeuille, X. (2020). Two-way fixed effects estimators with heterogeneous treatment effects. *American Economic Review*, 110, 2964-2996. doi:10.1257/aer.20181169

Wooldridge, J. M. (2023). Simple approaches to nonlinear difference-in-differences with panel data. *The Econometrics Journal*, 26, C31-C66. doi:10.1093/ectj/utad016

See Also

`summary.panglm()` and the other `panglm()` methods for the fitted model.

Other policy evaluation: `attgt()`, `etwfe()`, `eventplot()`, `eventstudy()`, `panelplot()`, `treatmentplot()`, `trendplot()`

Examples

```
set.seed(1)
d <- expand.grid(id = 1:200, year = 2015:2022)
d$first <- ifelse(d$id <= 100, 2019, NA)        # policy in 2019 for half
a <- rnorm(200)
d$visits <- 3 + a[d$id] + 0.1 * (d$year - 2015) +
  1.5 * (!is.na(d$first) & d$year >= d$first) + rnorm(nrow(d))
fit <- did(visits ~ 1, data = d, index = c("id", "year"), treat = "first")
summary(fit)
# binary outcome: conditional logit, the coefficient is a log odds ratio
d$admitted <- rbinom(nrow(d), 1, plogis(-1 + a[d$id] + 0.05 * (d$year - 2015) +
  0.8 * (!is.na(d$first) & d$year >= d$first)))
fitb <- did(admitted ~ 1, data = d, index = c("id", "year"), treat = "first",
            family = "binomial")
exp(coef(fitb))                                  # odds ratio
```

---

docvisits — *German health-care demand panel (doctor and hospital visits, GSOEP)*

Description

A real, public, unbalanced panel of 7,293 adults (aged 25 to 64) from the German Socio-Economic Panel (GSOEP), observed in up to seven waves (1984 to 1988, 1991 and 1994), 27,326 person-years in total. It is the data set of Riphahn, Wambach and Million (2003), who studied doctor and hospital visits in relation to public and add-on private health insurance, and it is the "German health care data" used in Greene's Econometric Analysis and his discrete choice lecture notes. The outcomes cover the main types of health-econometrics panel models: a count with many zeros and a long right tail (`docvis`, `hospvis`), binary use indicators (`doctor`, `hospital`) and an 11-point ordinal self-assessment (`hsat`). The unit is the person and the time index is the calendar year; the panel index is `c("id", "year")`. The build script, including the checks against published numbers, is `data-raw/docvisits.R` in the package's GitHub repository.

Usage

```
docvisits
```

Format

A data frame with 27326 rows (7,293 persons, 1 to 7 waves each) and 21 columns. Panel index: `id` (individual) and `year` (time).

id integer person identifier (1 to 7293), the panel individual index

year integer calendar year of the interview (1984, 1985, 1986, 1987, 1988, 1991, 1994), the panel time index

docvis integer number of doctor visits in the last three months (0 to 121); count outcome

hospvis integer number of hospital visits in the last calendar year (0 to 51); count outcome

doctor integer indicator (0/1) of at least one doctor visit, `docvis > 0`; binary outcome

hospital integer indicator (0/1) of at least one hospital visit, `hospvis > 0`; binary outcome

hsat integer health satisfaction, 0 (low) to 10 (high); ordinal outcome

female integer indicator, 1 = female, 0 = male; time-invariant

age integer age in years (25 to 64)

educ numeric years of schooling (7 to 18); constant within person for almost everyone

married integer indicator, 1 = married

hhkids integer indicator, 1 = children under 16 in the household

hhninc numeric household nominal monthly net income, in 1000 German marks (DM); 4 person-years report 0

working integer indicator, 1 = employed

bluecollar integer indicator, 1 = blue-collar employee

whitecollar integer indicator, 1 = white-collar employee

self integer indicator, 1 = self-employed

civil integer indicator, 1 = civil servant (Beamter)

handicap integer indicator, 1 = officially handicapped (1987 values recoded, see Description)

public integer indicator, 1 = insured in the public (statutory) health insurance

addon integer indicator, 1 = holds add-on (supplementary private) health insurance

Details

Cleaning relative to the original file: the 40 imputed non-integer values of `hsat` (between 6.5 and 7) are set to 7, as in Greene (2008); the handicap indicator, which the data authors report as swapped (0 and 1) in 1987, is recoded for that year; the degree of handicap (erroneously imputed in 1987) and five overlapping schooling-degree dummies are dropped; income is rescaled to 1000 DM. Missing values of `hsat`, `educ` and the handicap indicator were imputed by the data authors with gender-specific sample means (so a few `educ` values are not whole or half years). Note that the waves are not equally spaced (gaps after 1988 and 1991): lags and `dynamic = TRUE` fits only use consecutive years, so dynamic examples restrict the data to 1984 to 1988.

Teaching use

- Is income or add-on insurance associated with more doctor visits once stable individual differences are removed? Count models: `panglm(docvis ~ ..., family = "poisson", model = "within")` (conditional fixed-effects Poisson) against `model = "random"` or `family = "negbin"`, compared with the classical `panhausman()` test or a correlated random effects fit (`cre = TRUE`, Mundlak).
- Is a doctor visit this year partly caused by a visit last year (state dependence) or only by persistent health? Binary model with Wooldridge initial conditions on the consecutive waves 1984 to 1988: `panglm(doctor ~ ..., family = "probit", model = "random", dynamic = TRUE)`; `partialeffects()` for the effect on the probability.
- Which life events change health satisfaction within a person? Ordered outcome: `panglm(hsat ~ ..., family = "ologit", model = "within")` (blow-up-and-cluster fixed-effects ordered logit) against `model = "pooling"` or `model = "random"`.
- Do covariates act on the decision to see a doctor or on how often one goes? Fixed-effects hurdle count model `panhurdle(docvis ~ ..., family = "poisson")` (time-varying regressors only: time-invariant ones such as `female` are dropped with a warning).
- Are there latent "low user" and "high user" classes, as in Deb and Trivedi? Finite mixture `panfmm(docvis ~ ..., k = 2, family = "negbin")`, with partial pooling by person.
- Effects of time-invariant characteristics (gender, schooling) when time-varying regressors may be correlated with the individual effect: `paniv(..., model = "hausmantaylor")`, or `cre = TRUE` in `panglm()`. (`educ` varies slightly within a few persons because of the imputation, so `paniv()` treats it as time-varying.)

Source

Journal of Applied Econometrics Data Archive, Riphahn, Wambach and Million (2003), file `rwm-data.zip`: http://qed.econ.queensu.ca/jae/2003-v18.4/riphahn-wambach-million/ (now at https://journaldata.zbw.eu/, DOI 10.15456/jae.2022314.1312212303). The original survey is the German Socio-Economic Panel (SOEP, DIW Berlin). The same 27,326 x 25 file is distributed on CRAN under the GPL (>= 2) as `Health` in package Rchoice (citing Riphahn et al. 2003), and subsets are in package COUNT (GPL-2) as `rwm`, `rwm5yr` and `rwm1984`; the build script checks that the JAE file and the Rchoice copy are identical. The JAE readme states that "the distributors of the data require that the data are only used for the purpose of scientific studies": use these data for teaching and research only, and cite Riphahn, Wambach and Million (2003).

References

Riphahn, R. T., Wambach, A. and Million, A. (2003). Incentive effects in the demand for health care: a bivariate panel count data estimation. *Journal of Applied Econometrics*, 18(4), 387-405. doi:10.1002/jae.680

Greene, W. H. (2018). *Econometric Analysis*, 8th ed. Pearson (uses these German health care data in several examples).

Greene, W. H. (2014). Discrete choice modeling, Part 3: panel data binary choice models. Lecture notes, Stern School of Business, New York University (pooled and random-effects probit of `doctor`, reproduced by the build script).

Greene, W. H. (2008). Modeling ordered choices. Working paper, Stern School of Business, New York University (Table 3.1, ordered logit of collapsed health satisfaction in 1988).

Hilbe, J. M. (2011). *Negative Binomial Regression*, 2nd ed. Cambridge University Press.

Examples

```
data(docvisits)
str(docvisits)
table(docvisits$year)
# sample means by gender
aggregate(cbind(docvis, hospvis, doctor, addon, public, hsat) ~ female,
          data = docvisits, FUN = mean)

idx <- c("id", "year")

# Count outcome: fixed-effects (conditional) Poisson versus random
# effects, with a Hausman test
fe <- panglm(docvis ~ age + hhninc + hhkids + married + working + addon,
             data = docvisits, index = idx, model = "within",
             family = "poisson")
re <- panglm(docvis ~ age + hhninc + hhkids + married + working + addon,
             data = docvisits, index = idx, model = "random",
             family = "poisson")
summary(fe)
panhausman(fe, re)

# Ordinal outcome: fixed-effects ordered logit (blow-up and cluster)
# of health satisfaction
fo <- panglm(hsat ~ age + hhninc + hhkids + married + working + handicap,
             data = docvisits, index = idx, model = "within",
             family = "ologit")
summary(fo)

# Binary outcome with state dependence, consecutive waves 1984-1988
early <- subset(docvisits, year <= 1988)
dp <- panglm(doctor ~ age + hhninc + hhkids + married + working,
             data = early, index = idx, model = "pooling",
             family = "probit", dynamic = TRUE, vcov = "cluster")
coef(dp)[c("lag_doctor", "initial_doctor")]


# Dynamic random-effects probit (Wooldridge initial conditions)
dr <- panglm(doctor ~ age + hhninc + hhkids + married + working,
             data = early, index = idx, model = "random",
             family = "probit", dynamic = TRUE, R = 8)
summary(dr)

# Latent "low user" / "high user" classes (Deb-Trivedi style),
# women 1984-1988
women <- subset(docvisits, female == 1 & year <= 1988)
lc <- panfmm(docvis ~ age + educ + hhninc + hhkids + working + addon,
             data = women, index = idx, k = 2, family = "negbin",
```

```
             nstart = 3, seed = 1)
summary(lc)

# Hausman-Taylor: effect of the time-invariant female indicator when
# income may be correlated with the individual effect
ht <- paniv(log(1 + docvis) ~ age + hhninc + working + female |
              age + working + female,
            data = docvisits, index = idx, model = "hausmantaylor")
summary(ht)

# Replicates Greene's random-effects probit (DOCTOR on AGE, EDUC,
# HHNINC; log L = -16290.72); slow (about 1.5 minutes) on 27,326 rows:
# panglm(doctor ~ age + educ + I(hhninc / 10), data = docvisits,
#        index = idx, model = "random", family = "probit")
```

`effectcurve` *Estimated effect curve of a regressor*

Description

Three views of how the expected outcome depends on one regressor, each with delta-method pointwise confidence limits from the covariance of the fit. Use `plot()` to draw them.

Usage

```
effectcurve(
  object,
  variable,
  values = NULL,
  ref = NULL,
  vcov = NULL,
  level = 0.95,
  scale = c("link", "response", "slope"),
  by = NULL
)
```

Arguments

`object`	a `"panglm"` fit
`variable`	name of the regressor
`values`	values at which to evaluate the curve; default: 100 points between the 1st and 99th percentile of the regressor in the fit, or every level of a discrete regressor
`ref`	reference value of the link-scale curve; default: the median of the regressor
`vcov`	`NULL` (the covariance stored on the fit) or a type passed to `vcov.panglm()`
`level`	confidence level of the pointwise band
`scale`	`"link"`, `"response"` or `"slope"`, see Details

by — optional name of a discrete variable (factor, character, logical, or at most 10 distinct values) giving one curve per level; response and slope scales only

Details

- scale = "link" (default): for a regressor entered through a regression spline (splines::ns(), splines::bs()), a polynomial or linearly, the change in the linear predictor when the regressor moves from the reference value ref to each of values. Centering at a reference is what makes the curve comparable across models: in fixed-effects fits the level is absorbed by the individual effects. The scale is the link scale: a difference in mean for the identity link, a log rate ratio for Poisson and negative binomial models, a log odds ratio for the logit link.
- scale = "response": the average adjusted prediction (partial dependence), that is the mean outcome when the regressor is set to each value for every observation of the fitted data, all other regressors at their observed values. The mean is the one used by partialeffects(): the GLM mean for pooled fits and the population-averaged mean for random-effects fits, with the individual means of correlated random effects fits held at their fitted values. For a factor, logical or 0/1 regressor the curve has one point per level. Not available for fixed-effects (within) fits, where the level of the mean is not identified.
- scale = "slope": the average marginal effect d E[y] / dx when the regressor is set to each value for every observation, the derivative of the response-scale curve (a central difference in the regressor). Averaged over the sample at the observed values it would be the APE of partialeffects().

On the response and slope scales the regressor may enter interactions, and by gives one curve per level of a second variable, set to that level for every observation. Standard errors treat variance components, dispersion and fixed-effect intercepts as fixed, as in partialeffects().

Value

a data frame of class "panglm_effectcurve" with columns value, estimate, se, lower and upper (and by when by is given). The attributes record the variable, scale, reference value and the percentiles of the observed regressor used for the rug.

Validation

The package tests compare the response and slope curves of pooled fits with marginaleffects::avg_predictions() and marginaleffects::avg_slopes() on counterfactual grids.

See Also

Other panglm methods: coefplot(), confint.panglm(), fitted.panglm(), fixef(), glance.panglm(), logLik.panglm(), model.matrix.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_ape(), plot.panglm_effectcurve(), predict.panglm(), predict.panordered(), residuals.panglm(), summary.panglm(), tidy.panglm(), vcov.panglm()

Examples

```
data(copd)
fit <- panglm(fev1 ~ splines::ns(crp, 3), data = copd,
              index = c("id", "visit"), model = "within", vcov = "cluster")
```

```
curve_crp <- effectcurve(fit, "crp")
head(curve_crp)
plot(curve_crp)

pois <- panglm(exacerbations ~ splines::ns(crp, 3) + factor(smoker),
               data = copd, index = c("id", "visit"), model = "random",
               family = "poisson", cre = TRUE, vcov = "cluster")
plot(effectcurve(pois, "crp", scale = "response", by = "smoker"))
plot(effectcurve(pois, "crp", scale = "slope"))
plot(effectcurve(pois, "smoker", scale = "response"))
```

`etwfe` *Extended two-way fixed-effects difference-in-differences*

Description

The extended two-way fixed-effects (ETWFE) estimator of Wooldridge (2021, 2023) for staggered adoption, with linear, Poisson or logit outcome models. The model interacts the treatment with every cohort by period cell, so that each cohort can have its own effect in each period: with `g` the first treated period of the cohort and `t` the period,

$$E[y_{it}] = F(\alpha + \gamma_g + \lambda_t + \sum_{(g,t)} \tau_{gt} D_{gt,it} + x_{it}'\beta + \ldots)$$

where `D_gt` indicates cohort `g` in period `t`, `gamma_g` and `lambda_t` are cohort and period effects and `F` is the identity, the exponential or the logistic function. Because the cell effects are unrestricted, the estimator does not suffer from the negative weights of the single-coefficient TWFE regression of `did()`; in the linear case with `control = "notyet"` it is numerically equivalent to an imputation estimator. For nonlinear models, parallel trends is imposed on the index (for example on the log of the mean for Poisson), which suits counts and binary outcomes.

Usage

```
etwfe(
  formula,
  data,
  index,
  treat,
  family = c("gaussian", "poisson", "binomial", "logit"),
  control = c("notyet", "never"),
  vcov = c("cluster", "classical"),
  level = 0.95,
  weights = NULL,
  xvar = NULL,
  ivar = FALSE
)
```

Arguments

formula	outcome ~ 1, or outcome ~ controls
data	a data.frame in long format
index	c(individual, time); the time index must be numeric. The individual identifies the clusters for the standard errors.
treat	name of the column holding each individual's first treated period; NA, 0 or Inf for units that are never treated
family	"gaussian" (linear), "poisson" (counts, log link) or "binomial" (binary, logit link; "logit" is accepted)
control	"notyet" or "never"; see Specification
vcov	"cluster" (by individual, the default) or "classical". The cluster covariance uses the factor G / (G - 1) * (n - 1) / (n - k) with k counting all coefficients, cohort and period dummies included (with ivar = TRUE, the individual effects, nested in the clusters, count as one), as fixest does.
level	confidence level of the reported intervals
weights	optional sampling weights: the name of a column of data or a non-negative numeric vector with one value per row (rows with a missing weight are dropped). They weight the quasi-likelihood (as weights = ~w in etwfe::etwfe() and fixest) and the averages of the treatment effects over the treated observations in summary.panglm_etwfe() (as wts in etwfe::emfx()). The centring of the controls at their cohort means (and of xvar at its cell means) is unweighted, as in etwfe. Unlike in attgt(), the weights need not be constant within individuals: the model is a weighted pooled (quasi-)likelihood over the rows, so observation-level weights are well defined, and the cluster-robust covariance by individual remains valid (as in fixest). Use weights constant within individuals when they are sampling weights of the individuals, so that the estimates are comparable with attgt().
xvar	optional name of one numeric or logical column of data: a covariate by which the treatment effects may vary (see Heterogeneity by a covariate)
ivar	logical; TRUE uses individual (the first element of index) instead of cohort fixed effects, for family = "gaussian" only (see Specification)

Details

The average treatment effects on the treated are then computed on the response scale as marginal effects: for each treated observation, the difference between the predicted mean with its cell effect and without it, averaged over the treated observations (all of them, by cohort, by calendar period or by time since treatment; see summary.panglm_etwfe()). Standard errors use the delta method with analytic gradients.

Value

an object of class "panglm_etwfe": a list with coefficients, vcov, cells (a data frame of the cohort by period cells with their coefficient on the index scale), family, control, reference (the reference cohort), xvar, ivar, nobs, n_groups, the design (X, y, rows) and the call. Methods: print, summary (the ATTs), coef, vcov, nobs, plot, tidy and glance.

Specification

Cohort and period effects enter as dummies (cohort effects rather than individual effects: for linear models on balanced panels without time-varying controls the two give the same cell effects, and for nonlinear models the pooled quasi-likelihood with cohort dummies is the estimator of Wooldridge 2023). With `control = "notyet"` (the default) the cell dummies cover the post-treatment cells only, so all untreated observations, of never-treated and not-yet-treated cohorts, identify the counterfactual. With `control = "never"` every cell of each treated cohort gets a dummy except the period just before treatment, so only never-treated individuals identify the period effects and the pre-treatment cells give placebo estimates. The reference cohort is the never-treated group; without never-treated individuals and with `control = "notyet"` the last cohort to be treated is the reference and the periods from its treatment date onwards are dropped. Individuals treated in the first period are dropped.

Control variables in `formula` enter with their main effect, their interactions with the cohort and period dummies, and interactions of the cell dummies with the control variables centred at their cohort means, as in Wooldridge (2021) and the `etwfe` package, so that the cell effects can vary with the controls.

With `ivar = TRUE` (linear model only) individual fixed effects replace the cohort effects, as `ivar` in `etwfe::etwfe()`: they are swept out by within-individual demeaning (weighted when there are weights). On a balanced panel without time-varying controls the cell effects and ATTs are the same as with cohort effects. They differ with time-varying controls (the individual effects then also absorb the part of the individual heterogeneity that is correlated with the controls, as the Mundlak means of Wooldridge 2021 would) and on unbalanced panels; the individual effects also absorb time-invariant controls. Nonlinear models keep the cohort effects (with few periods, individual effects in a logit model suffer from the incidental parameters problem; the pooled Poisson and logit quasi-likelihoods with cohort effects are the estimators of Wooldridge 2023), and `ivar = TRUE` is an error there, whereas `etwfe::etwfe()` ignores `ivar` for nonlinear families.

Heterogeneity by a covariate

`xvar` names a covariate by which the treatment effects may vary, as `xvar` in `etwfe::etwfe()` (typically a time-invariant indicator, such as sex, rural residence or a baseline risk group). The covariate is centred at its cohort by period cell means and enters through its interactions with the period dummies (except the first period) and with every cell dummy, so each cohort by period effect becomes a linear function of the covariate, with the cell coefficient as the effect at the cell mean. This is the specification of the `etwfe` package; the level of the covariate itself is not a regressor, so add it to `formula` as a control to adjust for it as well. `summary.panglm_etwfe()` then reports the ATTs separately for each value of the covariate (`byxvar = TRUE`, the default for such a fit), as `etwfe::emfx(by_xvar = TRUE)`; with a continuous covariate there is one row per distinct value, so summarise a binary or discrete version of it.

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

References

Wooldridge, J. M. (2021). Two-way fixed effects, the two-way Mundlak regression, and difference-in-differences estimators. Working paper. doi:10.2139/ssrn.3906345

Wooldridge, J. M. (2023). Simple approaches to nonlinear difference-in-differences with panel data. *The Econometrics Journal*, 26, C31-C66. doi:10.1093/ectj/utad016

See Also

`attgt()` for the Callaway and Sant'Anna estimator, `did()` and `eventstudy()` for the TWFE versions.

Other policy evaluation: `attgt()`, `did()`, `eventplot()`, `eventstudy()`, `panelplot()`, `treatmentplot()`, `trendplot()`

Examples

```
# Staggered rollout of a health insurance scheme by region (for example
# RAMED, then AMO, in Morocco): regions join in 2016, 2018 or 2020, the
# last region not before 2022. Outcome: number of hospital admissions
# per district and year (a count).
set.seed(1)
d <- expand.grid(district = 1:160, year = 2012:2021)
d$adoption <- rep(c(2016, 2018, 2020, NA), each = 40)[d$district]
d$poverty <- rep(runif(160), 10)[d$district]
a <- rnorm(160, sd = 0.3)
treated <- !is.na(d$adoption) & d$year >= d$adoption
d$admissions <- rpois(nrow(d), exp(2 + a[d$district] + 0.03 * (d$year - 2012) +
                                   0.2 * treated))
fit <- etwfe(admissions ~ 1, data = d, index = c("district", "year"),
             treat = "adoption", family = "poisson")
fit                                  # overall ATT, admissions per district
summary(fit, type = "event")         # by time since adoption
summary(fit, type = "group")
plot(fit)
# linear model with a control variable
d$visits <- 50 + 5 * a[d$district] + 0.5 * (d$year - 2012) + 2 * treated +
  rnorm(nrow(d))
summary(etwfe(visits ~ poverty, data = d, index = c("district", "year"),
              treat = "adoption"), type = "calendar")
# individual instead of cohort fixed effects (linear model)
summary(etwfe(visits ~ poverty, data = d, index = c("district", "year"),
              treat = "adoption", ivar = TRUE), type = "group")
# effects by a covariate: poor versus other districts
d$poor <- as.numeric(d$poverty > 0.5)
fx <- etwfe(admissions ~ 1, data = d, index = c("district", "year"),
            treat = "adoption", family = "poisson", xvar = "poor")
summary(fx)                          # ATT for poor = 0 and poor = 1
summary(fx, type = "event", byxvar = FALSE)
```

`eventplot` *Compare event studies on one plot*

Description

Draws the event-time effects of several fits on one axis, offset horizontally, with pointwise confidence intervals and a legend: for example the two-way fixed-effects event study of `eventstudy()`, the Callaway and Sant'Anna estimator of `attgt()` and the extended two-way fixed-effects estimator of `etwfe()`. Uniform bands are not drawn here; use the `plot()` method of a single fit for them.

Usage

```
eventplot(
  ...,
  labels = NULL,
  min_e = -Inf,
  max_e = Inf,
  col = c("black", "#D55E00", "#0072B2", "#009E73", "#CC79A7"),
  pch = c(19, 17, 15, 18, 25),
  offset = 0.2,
  legend = TRUE,
  xlab = "Event time",
  ylab = "Estimate",
  main = NULL,
  ylim = NULL,
  style = c("classic", "points", "line"),
  ref = NULL,
  onset = NULL,
  onset_label = NA,
  caps = NULL
)
```

Arguments

`...`	fits from `eventstudy()`, `attgt()` or `etwfe()` (or the event-time summaries of the last two). Names of the arguments are used as labels.
`labels`	optional labels, one per fit (default: the argument names, else the estimator)
`min_e`, `max_e`	event window to plot
`col`, `pch`	colours and (filled) plotting symbols, recycled over the fits; the hollow version of each symbol is used before treatment
`offset`	horizontal distance between the fits, in event-time units
`legend`	logical: add a legend
`xlab`, `ylab`, `main`, `ylim`	axis labels, title and y-axis limits

style	"classic" (default), "points" or "line" (see Styles); for the event-time, cohort and calendar plots
ref	event time of the reference period, marked when the estimate of a fit is exactly zero there; the default (NULL) uses the omitted period of each fit (-1, or the ref of an eventstudy() fit)
onset	position of the dashed treatment-onset line on the event time axis: NULL gives 0 for style = "classic" and -0.5 (between the last pre-treatment and the first treated period) otherwise
onset_label	text written above the onset line; the default (NA) is "Intervention" for style = "classic" and none otherwise, NULL suppresses it
caps	logical: draw caps on the confidence-interval whiskers; the default (NULL) draws them for style = "classic" only

Details

The styles are those of plot.panglm_attgt(). With style = "classic" (the default) the points are filled, the whiskers capped, each reference period is a hollow symbol at zero and the dashed onset line is at event time 0, labelled "Intervention". With style = "points" the points are hollow before treatment and filled after, and the onset line is at -0.5. With style = "line" each fit is a coloured line with its pointwise intervals as a semi-transparent band (the fits are not offset).

Value

a data frame of the plotted estimates (model, event, estimate, lower, upper), invisibly

See Also

Other policy evaluation: attgt(), did(), etwfe(), eventstudy(), panelplot(), treatmentplot(), trendplot()

Examples

```
set.seed(2)
d <- expand.grid(id = 1:150, t = 1:8)
d$first <- rep(c(4, 6, NA), each = 50)[d$id]
d$y <- rnorm(150)[d$id] + 0.2 * d$t +
  ifelse(!is.na(d$first) & d$t >= d$first, 1, 0) + rnorm(nrow(d))
es <- eventstudy(y ~ 1, d, index = c("id", "t"), treat = "first",
                 leads = 3, lags = 3)
cs <- attgt(y ~ 1, d, index = c("id", "t"), treat = "first")
ew <- etwfe(y ~ 1, d, index = c("id", "t"), treat = "first", control = "never")
eventplot(TWFE = es, CS = cs, ETWFE = ew, min_e = -3, max_e = 3)
eventplot(TWFE = es, ETWFE = ew, min_e = -3, max_e = 3, style = "points")
eventplot(TWFE = es, CS = cs, min_e = -3, max_e = 3, style = "line")
```

eventstudy *Event-study regression with a pre-trend test*

Description

Replaces the single treatment indicator of `did()` by indicators of event time `e = t - first treated period`, from `-leads` to `lags`, with `e = ref` (by default the period before treatment) as the omitted reference, in a two-way fixed-effects model. Event times beyond the window are binned into the end points (`bin = TRUE`) or dropped from the sample (`bin = FALSE`). The lead coefficients estimate differences in trends before treatment; their joint Wald test is the usual check of parallel pre-trends (Freyaldenhoven et al. 2019). The lag coefficients trace the effect over time since treatment.

Usage

```
eventstudy(
  formula,
  data,
  index,
  treat,
  leads = 4,
  lags = 4,
  ref = -1,
  bin = TRUE,
  family = "gaussian",
  vcov = "cluster",
  level = 0.95,
  ...
)
```

Arguments

`formula`	outcome and control variables, e.g. `y ~ age` or `y ~ 1`; the treatment indicator is added
`data`	a data.frame in long format
`index`	`c(individual, time)`; the time index must be numeric
`treat`	name of the column holding each individual's first treated period (in the units of the time index); `NA`, `0` or `Inf` for units that are never treated
`leads, lags`	number of pre- and post-treatment event times in the window (event times `-leads, ..., lags`)
`ref`	omitted reference event time, default `-1`
`bin`	logical; bin event times outside the window into the end points (`TRUE`) or drop those observations (`FALSE`)
`family`	`"gaussian"`, `"poisson"`, `"negbin"` or `"binomial"` (conditional logit, logit link only; see Details)
`vcov`	covariance type, cluster-robust by individual by default
`level`	confidence level of the reported intervals
`...`	further arguments passed to `panglm()`

Details

Identification of all event-time coefficients needs never-treated units or a window narrower than the observed event times; without them a second coefficient is dropped (with a warning). With staggered adoption and heterogeneous effects the TWFE event-study coefficients are contaminated by effects from other periods (Sun and Abraham 2021); use a heterogeneity-robust estimator such as `attgt()` or `etwfe()` there.

With `family = "binomial"` the model is the two-way conditional logit (see `did()`) and the event-time coefficients are log odds ratios relative to the reference event time.

Value

an object of class `"panglm_eventstudy"`: a list with `fit` (the `panglm()` fit), `estimates` (a data frame with `event`, `estimate`, `se`, `lower`, `upper`; the reference row is zero) and `pretrend` (the joint test of the leads, from `jointtest()`) and `call` (the `eventstudy()` call, also stored as the `call` of `fit`). Methods: `print`, `summary`, `coef`, `vcov`, `confint`, `logLik`, `nobs`, `plot`, `tidy` and `glance`.

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

References

Freyaldenhoven, S., Hansen, C. and Shapiro, J. M. (2019). Pre-event trends in the panel event-study design. *American Economic Review*, 109, 3307-3338. doi:10.1257/aer.20180609

Sun, L. and Abraham, S. (2021). Estimating dynamic treatment effects in event studies with heterogeneous treatment effects. *Journal of Econometrics*, 225, 175-199. doi:10.1016/j.jeconom.2020.09.006

See Also

eventstudy-methods and `plot.panglm_eventstudy()` for the fitted model, tidy-methods for `tidy()` and `glance()`.

Other policy evaluation: `attgt()`, `did()`, `etwfe()`, `eventplot()`, `panelplot()`, `treatmentplot()`, `trendplot()`

Examples

```
set.seed(1)
d <- expand.grid(id = 1:200, year = 2015:2022)
d$first <- ifelse(d$id <= 100, 2019, NA)
a <- rnorm(200)
e <- d$year - d$first
d$visits <- 3 + a[d$id] + 0.1 * (d$year - 2015) +
  ifelse(!is.na(e) & e >= 0, 0.5 + 0.4 * e, 0) + rnorm(nrow(d))
es <- eventstudy(visits ~ 1, data = d, index = c("id", "year"),
                 treat = "first", leads = 4, lags = 3)
es
plot(es)
```

`eventstudy-methods` *Methods for event-study fits*

Description

`vcov()`, `nobs()`, `confint()` and `summary()` for the fits of `eventstudy()`. The coefficients are labelled by event time, as in `coef()`. The reference period enters as a coefficient fixed at zero, with zero variance and covariance and a degenerate interval at zero; an event time that the data do not identify has missing values throughout. Intervals and tests are Wald (normal) based, from the variance of the underlying two-way fixed-effects fit (`object$fit`).

Usage

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

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

## S3 method for class 'panglm_eventstudy'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'panglm_eventstudy'
summary(object, level = object$level, ...)

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

## S3 method for class 'panglm_eventstudy'
logLik(object, ...)
```

Arguments

`object`	a `"panglm_eventstudy"` fit from `eventstudy()`
`...`	unused
`parm`	event times (as numbers or names) or positions of the coefficients; all by default
`level`	confidence level
`x`	a `"summary.panglm_eventstudy"` object
`digits`	significant digits to print
`signif.stars`	passed to `stats::printCoefmat()`

Value

`vcov()`: a square matrix with one row per event time; `nobs()`: the number of observations of the fit; `confint()`: a two-column matrix; `summary()`: an object of class `"summary.panglm_eventstudy"` with the coefficient table `coefficients` (estimate, standard error, z value, p-value), the intervals `conf.int`, the pre-trend test `pretrend` and the fit description.

See Also

Other eventstudy methods: plot.panglm_eventstudy()

Examples

```
set.seed(1)
d <- expand.grid(id = 1:200, year = 2015:2022)
d$first <- ifelse(d$id <= 100, 2019, NA)
d$visits <- 3 + rnorm(200)[d$id] + 0.1 * (d$year - 2015) +
  ifelse(!is.na(d$first) & d$year >= d$first, 0.5, 0) + rnorm(nrow(d))
es <- eventstudy(visits ~ 1, data = d, index = c("id", "year"),
                 treat = "first", leads = 3, lags = 2)
summary(es)
confint(es)
vcov(es)
```

---

fitted.panglm *Extract fitted values from a panglm model*

Description

Values are returned in the row order of the data supplied to panglm(), even though estimation internally sorts observations by panel index. Fixed-effects binomial values are conditional inclusion probabilities, conditional on each individual's observed number of successes.

Usage

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

Arguments

object	a fitted "panglm" object
...	unused

Value

a numeric vector of fitted values, one per row of the original data supplied to panglm() (in that row order), or NA at rows dropped for missingness, following the model's na.action.

See Also

Other panglm methods: coefplot(), confint.panglm(), effectcurve(), fixef(), glance.panglm(), logLik.panglm(), model.matrix.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_ape(), plot.panglm_effectcurve(), predict.panglm(), predict.panordered(), residuals.panglm(), summary.panglm(), tidy.panglm(), vcov.panglm()

Examples

```
data(copd)
fit <- panglm(exacerbations ~ crp + treatment, data = copd,
              index = c("id", "visit"), model = "random", family = "poisson")
head(fitted(fit))
```

`fixef` *Individual effects of a panel model*

Description

`fixef()` returns the estimated individual effects `alpha_i` of a fixed-effects (`model = "within"`) fit, and `ranef()` the empirical Bayes predictions of the random effects of a `model = "random"` fit.

Usage

```
fixef(object, ...)

ranef(object, ...)

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

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

## S3 method for class 'panglm'
ranef(object, type = c("mean", "mode"), ...)
```

Arguments

`object`	a fitted `panglm()` model (or a `panhurdle()` fit for `fixef()`)
`...`	unused
`type`	for `ranef()`, `"mean"` (posterior mean, default) or `"mode"` (posterior mode)

Value

`fixef()`: a named numeric vector of individual effects (names are the individual identifiers), or for two-way fits a list with components `individual` and `time`. `ranef()`: a matrix with one row per individual (row names are the identifiers) and one column per random effect, with attribute `"type"`.

Fixed effects

- Gaussian within fits: `alpha_i = ybar_i - xbar_i' beta`, the intercepts of the least-squares dummy-variable regression. They are not normalized: `fixest::fixef()` gives the same values for a model without an intercept.

- Poisson within fits: the conditional (Poisson fixed-effects) estimator does not estimate `alpha_i`, but the individual intercept that maximizes the unconditional likelihood at the conditional `beta` has the closed form `alpha_i = log(sum_t y_it / sum_t exp(x_it' beta))`, the values reported by `fixest::fepois()`. Individuals with no positive outcome (`alpha_i = -Inf`) are omitted, as in fixest.
- Negative binomial and gamma within fits: the dummy-variable intercepts of the joint maximum likelihood fit. The negative binomial fit drops the individuals whose outcome is always zero.
- Two-way fits (Gaussian, Poisson, negative binomial): a list with the `individual` and `time` effects, normalized so that the time effects average zero.
- The conditional logit (`family = "binomial"`, one-way or two-way) and the fixed-effects ordered logit condition the individual effects out of the likelihood and do not estimate them: `fixef()` stops with an error. The individual intercepts are not consistently estimable with a fixed number of periods anyway (incidental parameters).
- `panhurdle()` fits: the dummy-variable intercepts of the zero-truncated count part (individuals with at least one positive count); the binary part is a conditional logit.

Random effects

`ranef()` returns the posterior mean (`type = "mean"`, default) or the posterior mode (`type = "mode"`) of each individual's random effects given the individual's outcomes, at the estimated parameters (empirical Bayes).

- Gaussian random intercept: the BLUP `sigma_mu^2 T_i / (sigma_mu^2 T_i + sigma_e^2) * mean_t(y_it - x_it'` with the variance components of the fit (Swamy-Arora, or maximum likelihood for weighted fits). Mean and mode coincide.
- Gaussian random slopes: the BLUPs stored by the fit.
- Binomial, gamma, beta, Tobit and ordered random intercepts, which are normal on the link scale: the posterior mean is computed by adaptive Gauss-Hermite quadrature with the `R` nodes of the fit, centred at the posterior mode and scaled by the curvature there, the rule used for the likelihood. With `R = 1` (Laplace) the mean is the mode.
- Poisson and negative binomial random intercepts (Hausman, Hall and Griliches), whose mixing distribution is not normal: the effect is reported on the log scale of the multiplicative effect on the mean. For the Poisson-gamma model `nu_i ~ Gamma(d, d)` and the posterior is `Gamma(d + sum_t y_it, d + sum_t exp(x_it' beta))`, so the posterior mean of `log(nu_i)` is `digamma(d + Y_i) - log(d + L_i)` and its mode `log((d + Y_i) / (d + L_i))`. For the beta-mixed negative binomial model the effect is `log(delta_i)`, with `delta_i / (1 + delta_i)` distributed as `Beta(b, a)`, giving `digamma(b + Y_i) - digamma(a + L_i)` and `log((b + Y_i) / (a + L_i))`.
- Random slopes of the Poisson, negative binomial, gamma and binomial families: posterior modes are those stored by the fit; posterior means use the adaptive product rule of the fit (`R^q` nodes).

Empirical Bayes predictions are shrunk towards zero, the more so the fewer observations an individual has: their spread is smaller than the estimated standard deviation of the random effect (for the Gaussian model, `var(b_i) = sigma_mu^2 * shrinkage_i`), so their histogram is narrower than the assumed normal density (see `plot.panglm()`, `which = "effects"`).

Generics

The generics `fixef()` and `ranef()` are also defined by nlme, whose generics lme4 and fixest re-export. panglm exports its own generics so that it does not depend on nlme, and when nlme is loaded it also registers its methods for nlme's generics: whichever package is attached last, `fixef(fit)` and `ranef(fit)` dispatch to the panglm methods.

See Also

`plot.panglm()` with `which = "effects"` to plot them.

Other panglm methods: `coefplot()`, `confint.panglm()`, `effectcurve()`, `fitted.panglm()`, `glance.panglm()`, `logLik.panglm()`, `model.matrix.panglm()`, `partialeffects.panordered()`, `plot.panglm()`, `plot.panglm_ape()`, `plot.panglm_effectcurve()`, `predict.panglm()`, `predict.panordered()`, `residuals.panglm()`, `summary.panglm()`, `tidy.panglm()`, `vcov.panglm()`

Examples

```
data(copd)
fe <- panglm(fev1 ~ crp, data = copd, index = c("id", "visit"),
             model = "within")
head(fixef(fe))

re <- panglm(fev1 ~ crp + treatment, data = copd,
             index = c("id", "visit"), model = "random")
head(ranef(re))

hosp <- panglm(hospitalized ~ crp + treatment, data = copd,
               index = c("id", "visit"), model = "random",
               family = "binomial")
head(ranef(hosp))
```

`glance.panglm` *One-row model summary of a panglm fit*

Description

Implements the `generics::glance()` generic.

Usage

```
## S3 method for class 'panglm'
glance(x, ...)
```

Arguments

`x`	a "panglm" fit
`...`	unused

Details

`r.squared` is `1 - RSS / TSS` on the scale of `fitted()` and the outcome. For Gaussian fixed-effects fits the fitted values include the absorbed individual (and, with `effect = "twoways"`, time) effects, so `r.squared` is the R-squared of the dummy-variable regression, `fixest::r2(type = "r2")`, not the within R-squared reported by `plm`. `adj.r.squared` is `1 - (1 - R^2) (n - 1) / df`, with `df` the residual degrees of freedom of the fit, which count the absorbed effects, as `fixest::r2(type = "ar2")`. For non-Gaussian families both are Gaussian-style pseudo-R-squared measures, not likelihood based.

Value

a one-row `data.frame` including log-likelihood, AIC, BIC, the likelihood parameter count when a likelihood is available, R-squared and adjusted R-squared (see Details), and `theta`, the NB2 dispersion parameter of pooled, fixed-effects and random-slope negative binomial fits (`NA` otherwise)

See Also

tidy-methods for the other model classes.

Other panglm methods: `coefplot()`, `confint.panglm()`, `effectcurve()`, `fitted.panglm()`, `fixef()`, `logLik.panglm()`, `model.matrix.panglm()`, `partialeffects.panordered()`, `plot.panglm()`, `plot.panglm_ape()`, `plot.panglm_effectcurve()`, `predict.panglm()`, `predict.panordered()`, `residuals.panglm()`, `summary.panglm()`, `tidy.panglm()`, `vcov.panglm()`

Examples

```
data(copd)
fit <- panglm(fev1 ~ treatment + age + smoker + crp, data = copd,
              index = c("id", "visit"), model = "pooling", family = "gaussian")
if (requireNamespace("generics", quietly = TRUE)) {
  generics::glance(fit)
}
```

---

healthspend — *Health expenditure, income and population health: a country-year panel*

Description

A real public panel of 193 countries observed every year from 2000 to 2022 (23 years), assembled from the World Bank World Development Indicators (WDI). It links current health expenditure (from the WHO Global Health Expenditure Database, as redistributed in WDI), its financing mix (out-of-pocket and domestic government shares), real income, population ageing, life expectancy and under-5 mortality. The panel is balanced in its rows (every country has all 23 years) but unbalanced in its values: some country-years are missing (`NA`) for some indicators, so fitted models use an unbalanced panel. Countries with no health expenditure data in any year (mostly small territories) and all regional and income aggregates are excluded. The build script, with the download calls and the validation checks, is `data-raw/healthspend.R` in the package's GitHub repository.

Usage

```
healthspend
```

Format

A data frame with 4,439 rows (193 countries x 23 years) and 15 columns. The panel index pair is (`iso3c`, `year`).

iso3c factor, ISO 3166-1 alpha-3 country code (193 levels); the panel individual index

year integer calendar year, 2000-2022; the panel time index

country character, World Bank country name

region factor, World Bank region (7 levels: East Asia & Pacific; Europe & Central Asia; Latin America & Caribbean; Middle East, North Africa, Afghanistan & Pakistan; North America; South Asia; Sub-Saharan Africa); time invariant

income factor, World Bank income group with levels `low`, `lowermiddle`, `uppermiddle`, `high`; the classification current at download (2026-11-01), so it is time invariant here and does not track past reclassifications

chepc numeric, current health expenditure (CHE) per capita, PPP, current international dollars (WDI `SH.XPD.CHEX.PP.CD`), 2 decimals

chereal numeric, real CHE per capita in constant 2021 international dollars, derived as `chegdp / 100 * gdppc`; use this (not `chepc`, which is in current prices) together with `gdppc` for income elasticities

chegdp numeric, CHE as a percentage of GDP (`SH.XPD.CHEX.GD.ZS`)

oop numeric, out-of-pocket expenditure as a percentage of CHE (`SH.XPD.OOPC.CH.ZS`), 0-100; divide by 100 for a proportion

gghe numeric, domestic general government health expenditure as a percentage of CHE (`SH.XPD.GHED.CH.ZS`), 0-100

gdppc numeric, GDP per capita, PPP, constant 2021 international dollars (`NY.GDP.PCAP.PP.KD`), 1 decimal

lifeexp numeric, life expectancy at birth, total, years (`SP.DYN.LE00.IN`); period estimates include crisis years, e.g. 14.7 for the Central African Republic in 2009

u5mr numeric, under-5 mortality rate per 1,000 live births (`SH.DYN.MORT`)

pop integer, total population (`SP.POP.TOTL`)

pop65 numeric, population aged 65 and above, percentage of total (`SP.POP.65UP.TO.ZS`)

Missing values: `chepc` 92, `chereal` 217, `chegdp`, `oop` and `gghe` 91, `gdppc` 174, `u5mr` 23; `lifeexp`, `pop` and `pop65` are complete.

Teaching use

The panel is suited to the classic macro health-economics questions. Take logs first, e.g. `hs <- transform(healthspend, lche = log(chereal), lgdp = log(gdppc))`.

- Is health care a luxury good? Income elasticity of real health spending, `lche ~ lgdp`: pooled vs between vs within vs random effects (`panglm()` with `model = "pooling"`, `"between"`, `"within"`, `"random"`), correlated random effects (`cre = TRUE`) and `panhausman()` (classical and Mundlak). Year effects (`effect = "twoways"`) remove common shocks and lower the within elasticity from about 1.15 to about 0.84.
- How persistent is health spending, and what is the long-run income elasticity? Dynamic model with a lagged dependent variable, `pangmm()` (difference GMM `"d"` and system GMM `"ld"`, with `collapse = TRUE` to keep the instrument count below the number of countries); long-run elasticity = b / (1 - rho).
- What do time-invariant characteristics add once income is allowed to be correlated with the country effect? Hausman-Taylor with income endogenous and population ageing plus region exogenous: `paniv(..., model = "hausmantaylor")`.
- Does out-of-pocket financing fall as countries grow richer? The out-of-pocket share `oop / 100` is a proportion in (0, 1): `panglm(family = "beta")`, pooled with clustered standard errors or random effects with `cre = TRUE`.
- Does the income elasticity differ across countries or across the spending distribution? Random slopes (`random = ~ lgdp`, with `lgdp` centred) and fixed-effects quantile regression (`panqr()`, `"mmqr"` or `"canay"`).
- Does spending buy health? Life expectancy or under-5 mortality on `lche` and `lgdp` with two-way fixed effects (descriptive: spending is endogenous to health needs).

Source

World Bank, World Development Indicators, API v2 (https://api.worldbank.org/v2/country/all/indicator/SH.XPD.CHEX.PP.CD?date=2000:2023&format=json&per_page=20000 and the same call for the other indicator codes; country metadata from https://api.worldbank.org/v2/country?format=json&per_page=400), WDI release last updated 2026-10-08, downloaded 2026-11-01. Health expenditure series: World Health Organization, Global Health Expenditure Database (update of 12 December 2025), via WDI. Life expectancy and population: UN World Population Prospects; under-5 mortality: UN Inter-agency Group for Child Mortality Estimation; GDP: World Bank International Comparison Program; all via WDI.

Licence: Creative Commons Attribution 4.0 (CC BY 4.0). The WDI catalog entry (https://datatopics.worldbank.org/world-development-indicators/) states "This dataset is licensed under Creative Commons Attribution 4.0", and the metadata of each of the nine series used carries `License_Type` "CC BY-4.0". Attribution: "World Bank, World Development Indicators (CC BY 4.0)". Changes from the source: country subset, renamed columns, rounding, and the derived column `chereal`.

References

World Bank (2026). World Development Indicators. Washington, DC: The World Bank. https://datatopics.worldbank.org/world-development-indicators/

World Health Organization (2025). Global Health Expenditure Database. https://apps.who.int/nha/database

Getzen, T. E. (2000). Health care is an individual necessity and a national luxury: applying multilevel decision models to the analysis of health care expenditures. Journal of Health Economics, 19(2), 259-270.

Baltagi, B. H. and Moscone, F. (2010). Health care expenditure and income in the OECD reconsidered: evidence from panel data. Economic Modelling, 27(4), 804-811.

Hausman, J. A. and Taylor, W. E. (1981). Panel data and unobservable individual effects. Econometrica, 49(6), 1377-1398.

Examples

```
data(healthspend)
hs <- transform(healthspend, lche = log(chereal), lgdp = log(gdppc))
ix <- c("iso3c", "year")

## income elasticity of real health spending
fe <- panglm(lche ~ lgdp, data = hs, index = ix, model = "within")
re <- panglm(lche ~ lgdp, data = hs, index = ix, model = "random")
fe2 <- panglm(lche ~ lgdp, data = hs, index = ix, model = "within",
              effect = "twoways", vcov = "cluster")
cbind(within = coef(fe), random = coef(re)[-1], twoways = coef(fe2))
panhausman(fe, re)
panhausman(fe, method = "mundlak", data = hs)

## persistence and long-run elasticity: system GMM, collapsed instruments
gs <- pangmm(lche ~ lag(lche, 1) + lgdp | lag(lche, 2:99) + lag(lgdp, 2:99),
             data = hs, index = ix, transformation = "ld",
             model = "twostep", collapse = TRUE)
summary(gs)
b <- coef(gs)
b["lgdp"] / (1 - b["lag(lche, 1)"])   # long-run income elasticity

## Hausman-Taylor: income endogenous; ageing and region exogenous
ht <- paniv(lche ~ lgdp + pop65 + region | pop65 + region,
            data = hs, index = ix, model = "hausmantaylor")
coef(ht)

## out-of-pocket share, a proportion: beta regression
hs$oopshare <- hs$oop / 100
bp <- panglm(oopshare ~ lgdp + pop65, data = hs, index = ix,
             model = "pooling", family = "beta", vcov = "cluster")
coef(bp)

## quantile heterogeneity of the elasticity
coef(panqr(lche ~ lgdp, data = hs, index = ix, tau = c(0.1, 0.5, 0.9),
           model = "mmqr"))


## random-effects beta with Mundlak terms, every fifth year (slower)
hs5 <- subset(hs, year %% 5 == 0)
br <- panglm(oopshare ~ lgdp + pop65, data = hs5, index = ix,
             model = "random", family = "beta", cre = TRUE,
             vcov = "cluster", R = 7)
summary(br)

## country-specific income slopes (centre the regressor first)
```

```
hs$lgdpc <- hs$lgdp - 9
rs <- panglm(lche ~ lgdpc, data = hs, index = ix, model = "random",
             random = ~ lgdpc, R = 7)
summary(rs)
```

---

hurdle-methods *Coefficient methods for the hurdle model*

Description

coef(), vcov(), confint(), nobs() and summary() for hurdle fits (panhurdle()). The coefficients of the zero part (conditional logit of 1(y > 0)) and of the count part (zero-truncated count model on the positive observations) are named "zero:..." and "count:...". The two parts are estimated from separate factors of the hurdle likelihood, so vcov() is block diagonal. No logLik() method is provided: the zero part maximises a conditional likelihood and the count part an unconditional likelihood with individual intercepts, so their sum is not the log-likelihood of one model.

Usage

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

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

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

## S3 method for class 'panglm_hurdle'
confint(object, parm, level = 0.95, ...)

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

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

Arguments

object	a "panglm_hurdle" fit
...	unused
parm	names or positions of the coefficients; all by default
level	confidence level
x	a "summary.panglm_hurdle" object
digits	significant digits to print

Value

coef(): a named vector; vcov(): a matrix; confint(): a two-column matrix; nobs(): the number of observations; summary(): an object of class "summary.panglm_hurdle" with one coefficient table per part (zero, count: estimate, standard error, z value and p-value, from the inverse Hessian of each part) and the fit information.

See Also

Other panhurdle methods: plot.panglm_hurdle()

Examples

```
data(copd)
fit <- panhurdle(exacerbations ~ crp, data = copd, index = c("id", "visit"))
coef(fit)
confint(fit)
summary(fit)
```

---

jointtest *Joint Wald test of coefficients*

Description

Tests H0: R beta = r with the statistic (R b - r)' (R V R')^(-1) (R b - r), referred to a chi-squared distribution with nrow(R) degrees of freedom. The coefficients are chosen by name (terms), by a regular expression (pattern) or through an explicit restriction matrix R. Typical uses are the joint test of the spline columns of a regressor (does x matter at all), of the Mundlak means of a correlated random effects fit, or of the leads of an event study.

Usage

```
jointtest(object, terms = NULL, pattern = NULL, R = NULL, r = 0, vcov = NULL)
```

Arguments

object	a "panglm" fit
terms	character vector of coefficient names
pattern	regular expression selecting coefficient names (used when terms and R are NULL)
R	optional restriction matrix with one column per coefficient
r	right-hand side of the restrictions (recycled), default 0
vcov	NULL (the covariance stored on the fit) or a type passed to vcov.panglm(), e.g. "cluster"

Value

an object of class "panglm_jointtest": a list with statistic, df, p.value, terms and vcov_type

Validation

Checked against a hand computation of the Wald statistic, in both the `terms` and the `R` matrix forms, in the package tests.

See Also

Other tests: `lineartest()`, `pandispersiontest()`, `panhausman()`

Examples

```
data(copd)
fit <- panglm(fev1 ~ splines::ns(crp, 3), data = copd,
              index = c("id", "visit"), model = "within", vcov = "cluster")
jointtest(fit, pattern = "crp")
```

---

lineartest *Test whether the effect of a regressor is linear*

Description

For a regressor entered through a spline or polynomial basis whose span contains the linear function (true for `splines::ns()`, `splines::bs()` and `poly()`), linearity means the basis coefficients are proportional to the coefficients `a` that reproduce `x` itself (`x = c + B a`). The test is the joint Wald test of the `q - 1` restrictions that the coefficient vector is orthogonal to the complement of `a`, so it accepts any covariance type, including cluster-robust ones. For a test of whether the regressor matters at all, use `jointtest()` on its columns.

Usage

```
lineartest(object, variable, vcov = NULL)
```

Arguments

`object`	a `"panglm"` fit
`variable`	name of the regressor
`vcov`	`NULL` (the covariance stored on the fit) or a type passed to `vcov.panglm()`

Value

an object of class `"panglm_jointtest"`, as returned by `jointtest()`.

Validation

The package tests check that the test rejects a nonlinear curve and keeps its size under linearity, and the fixed-effects likelihood-ratio statistic against the Wald statistic.

See Also

Other tests: jointtest(), pandispersiontest(), panhausman()

Examples

```
data(copd)
fit <- panglm(fev1 ~ splines::ns(crp, 3), data = copd,
              index = c("id", "visit"), model = "within", vcov = "cluster")
lineartest(fit, "crp")
```

`logLik.panglm` *Extract the model log-likelihood*

Description

The `df` attribute counts every parameter associated with the reported likelihood, including fitted shape, dispersion, variance, and explicit fixed-effect parameters. For conditional Poisson and conditional logistic likelihoods, only parameters remaining in the conditional likelihood are counted. Models estimated without a likelihood return `NA`.

Usage

```
## S3 method for class 'panglm'
logLik(object, ...)
```

Arguments

`object`	a fitted `"panglm"` object
`...`	unused

Value

an object of class `"logLik"`

See Also

Other panglm methods: coefplot(), confint.panglm(), effectcurve(), fitted.panglm(), fixef(), glance.panglm(), model.matrix.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_ape(), plot.panglm_effectcurve(), predict.panglm(), predict.panordered(), residuals.panglm(), summary.panglm(), tidy.panglm(), vcov.panglm()

Examples

```
data(copd)
fit <- panglm(exacerbations ~ crp, data = copd,
              index = c("id", "visit"), model = "pooling",
              family = "poisson")
logLik(fit)
AIC(fit)
BIC(fit)
```

`medicaid` *Health insurance coverage and the ACA Medicaid expansion, US states 2009-2019*

Description

A real public state-year panel of health insurance coverage for the 50 US states and the District of Columbia, 2009 to 2019, built from the American Community Survey (ACS) 1-year estimates of the US Census Bureau, with the date on which each state implemented the Affordable Care Act (ACA) Medicaid expansion to adults with incomes up to 138 percent of the federal poverty level. Adoption was staggered (2014, 2015, 2016, 2017, 2019, with ten states never adopting and several adopting after 2019), which makes it the standard teaching example for staggered difference-in-differences. The panel unit is the state and the time unit is the calendar year; the panel is balanced (51 x 11).

Usage

```
medicaid
```

Format

A data frame with 561 rows (51 states x 11 years) and 20 columns. The panel index pair is `c("state", "year")`.

state factor, state name (51 levels, District of Columbia included); the panel individual index

year integer calendar year of the ACS estimate (2009-2019); the panel time index

abb factor, two-letter postal abbreviation

fips integer state FIPS code

region factor, Census region (Northeast, Midwest, South, West)

pop integer civilian noninstitutionalized population (all ages), the universe of the health insurance tables (B27001)

uninsured numeric percent of the civilian noninstitutionalized population without health insurance coverage, all ages (B27001)

adultpop integer adults aged 18 to 64 (2009-2016) or 19 to 64 (2017-2019) for whom poverty status is determined (B27016); the ACS moved its age break from 18 to 19 in 2017, for all states at once

uninsadult numeric percent of `adultpop` without health insurance coverage

medicaidadult numeric percent of `adultpop` with Medicaid or other means-tested public coverage (alone or with other coverage)

povpop integer adults (same ages) with family income below 100 percent of the poverty threshold

povbase integer `povpop` of the state in 2013, the last year before the main expansion wave: a time-invariant weight, as needed by `attgt()` (weights constant within individuals)

uninspov numeric percent of `povpop` without health insurance coverage; the low-income outcome available in every year

medicaidpov numeric percent of `povpop` with Medicaid or other means-tested public coverage

lowpop integer adults (same ages) with family income below 138 percent of the poverty threshold, the ACA eligibility cut; `NA` in 2009-2010, when the ACS tables cut at 100 and 150 percent

uninslow numeric percent of `lowpop` without health insurance coverage; `NA` in 2009-2010

adopt integer expansion year: the first calendar year coded as treated, `NA` for the 10 states that had not adopted by October 2026. Convention of Miller, Johnson and Wherry (2021): the year of implementation if the expansion took effect before July 1, otherwise the following year (so Michigan, April 2014, is 2014; New Hampshire, August 2014, is 2015; Louisiana, July 2016, is 2017). Values after 2019 (Idaho and Utah 2020, Nebraska 2021, Missouri and Oklahoma 2022, North Carolina and South Dakota 2024) are kept: in this panel those states are never treated, and `attgt()` treats cohorts first treated after the last period as never treated

expdate Date the expansion took effect (KFF); `NA` if not adopted

expanded integer 0/1 indicator that `year >= adopt`

prior factor coverage of low-income adults before 2014: `"full"` for DE, DC, MA, NY and VT, which already covered adults up to the poverty line or more (Miller, Johnson and Wherry 2021, who drop them); `"early"` for CA, CT, MN, NJ and WA, which made limited early expansions in 2010-2012 under the ACA option or waivers (Sommers et al. 2013); `"none"` otherwise. A standard robustness check drops `prior != "none"`. Wisconsin is coded as a non-adopter but has covered adults up to 100 percent of poverty since 2014 through a waiver

Details

The ACS starts asking about health insurance in 2008, but the 2008 Summary File has no table by ratio of income to poverty nor a Medicaid table, so the panel starts in 2009. It stops in 2019 because the Census Bureau did not release standard 2020 ACS 1-year estimates (only experimental ones), after the pandemic disrupted data collection. The rates are computed from the published counts of tables B27001 and B27016 (state level), so they reproduce the published percentages; see `data-raw/medicaid.R` in the package's GitHub repository for the build and its validation checks.

Teaching use

- Did the expansion reduce uninsurance among low-income adults? Staggered DiD with `attgt()` (`treat = "adopt"`, never-treated or not-yet-treated controls), aggregated by cohort and event time with `summary.panglm_attgt()`. Weights must be constant within a state: use the base-year population `povbase` (`povpop` in 2013).
- How large is the bias of the single-coefficient two-way fixed effects regression? Compare `did()` (or `panglm()` with `model = "within"`, `effect = "twoways"` and the `expanded` dummy) with `attgt()` and `etwfe()`: the effect grows with time since expansion, so early cohorts serve as controls for later ones.

- Are pre-trends parallel, and how does the effect build up? `eventstudy()` with leads and lags, and `summary(type = "dynamic")` of `attgt()`. Louisiana (July 2016, coded 2017) is partly treated in 2016: compare `anticipation = 1`.
- Inference with 51 clusters: the TWFE coefficient from `panglm()` with `model = "within"` and `effect = "twoways"` has a classical standard error less than half the cluster-robust one (`vcov = "cluster"`), the default of `did()`.
- Did Medicaid enrolment (`medicaidpov`, `medicaidadult`) rise, and is the fall in uninsurance among all ages (`uninsured`) smaller, as expected from a policy aimed at low-income adults? Same estimators with other outcomes (a "first stage" and a dilution check).
- Robustness: drop the states with prior coverage (`prior != "none"`), restrict to the ACA income cut (`uninslow`, 2011-2019), or use population weights versus none. Weighted fixed-effects fits are not available in `panglm()`; a weighted TWFE is `model = "pooling"` with state and year dummies.

Source

US Census Bureau, American Community Survey 1-year estimates, 2009-2019, detailed tables B27001 (health insurance coverage status by sex by age) and B27016 (health insurance coverage status and type by ratio of income to poverty level by age), Summary File, https://www2.census.gov/programs-surveys/acs/summary_file/. Works of the US federal government are in the public domain (17 U.S.C. 105); the Census Bureau states "You do not need permission to use American Community Survey data" (https://www.census.gov/help/topics/faq.how-do-i-cite-or-source-acs-data.html). Attribution: "Source: U.S. Census Bureau, American Community Survey 1-year estimates." This product is not endorsed or certified by the Census Bureau. Expansion dates: KFF, Status of State Action on the Medicaid Expansion Decision (State Health Facts), status as of October 2026, https://www.kff.org/affordable-care-act/state-indicator/state-activity-around-expanding-medicaid-u

References

Miller, S., Johnson, N. and Wherry, L. R. (2021). Medicaid and mortality: new evidence from linked survey and administrative data. *Quarterly Journal of Economics*, 136(3), 1783-1829. doi:10.1093/qje/qjab004

Courtemanche, C., Marton, J., Ukert, B., Yelowitz, A. and Zapata, D. (2017). Early impacts of the Affordable Care Act on health insurance coverage in Medicaid expansion and non-expansion states. *Journal of Policy Analysis and Management*, 36(1), 178-210. doi:10.1002/pam.21961

Sommers, B. D., Arntson, E., Kenney, G. M. and Epstein, A. M. (2013). Lessons from early Medicaid expansions under health reform: interviews with Medicaid officials. *Medicare and Medicaid Research Review*, 3(4). doi:10.5600/mmrr.003.04.a02

Smith, J. C. and Medalia, C. (2015). *Health Insurance Coverage in the United States: 2014*. Current Population Reports P60-253, US Census Bureau.

Callaway, B. and Sant'Anna, P. H. C. (2021). Difference-in-differences with multiple time periods. *Journal of Econometrics*, 225(2), 200-230. doi:10.1016/j.jeconom.2020.12.001

Goodman-Bacon, A. (2021). Difference-in-differences with variation in treatment timing. *Journal of Econometrics*, 225(2), 254-277. doi:10.1016/j.jeconom.2021.03.014

Examples

```
data(medicaid)
str(medicaid)
# adoption cohorts (first treated year; NA = not adopted by 2026)
table(unique(medicaid[, c("state", "adopt")])$adopt, useNA = "ifany")
ix <- c("state", "year")

# Callaway-Sant'Anna ATT(g, t) for the uninsured rate of adults below
# the poverty line, never-treated controls, states weighted equally
cs <- attgt(uninspov ~ 1, data = medicaid, index = ix, treat = "adopt",
            control = "never")
summary(cs)
summary(cs, type = "dynamic")   # leads test pre-trends, lags the build-up

# weighted by the pre-expansion (2013) low-income adult population,
# with not-yet-treated controls
cs2 <- attgt(uninspov ~ 1, data = medicaid, index = ix, treat = "adopt",
             control = "notyet", weights = "povbase")
summary(cs2)

# extended TWFE and the single-coefficient TWFE for comparison (the
# latter can be biased under staggered adoption with dynamic effects)
ew <- etwfe(uninspov ~ 1, data = medicaid, index = ix, treat = "adopt")
summary(ew)
tw <- did(uninspov ~ 1, data = medicaid, index = ix, treat = "adopt")
coef(tw)["treated"]

# event-study regression with a joint pre-trend test
eventstudy(uninspov ~ 1, data = medicaid, index = ix, treat = "adopt",
           leads = 4, lags = 5)

# first stage: Medicaid coverage of adults below the poverty line
summary(attgt(medicaidpov ~ 1, data = medicaid, index = ix,
              treat = "adopt", control = "never"))


# the same TWFE coefficient with classical and cluster-robust SEs
fe <- panglm(uninspov ~ expanded, data = medicaid, index = ix,
             model = "within", effect = "twoways")
fecl <- update(fe, vcov = "cluster")
cbind(classical = sqrt(diag(vcov(fe))), cluster = sqrt(diag(vcov(fecl))))

# weighted TWFE: a pooled fit with state and year dummies
tww <- panglm(uninspov ~ expanded + state + factor(year), data = medicaid,
              index = ix, model = "pooling", weights = "povbase",
              vcov = "cluster")
coef(summary(tww))["expanded", ]

# robustness: drop the states with pre-2014 coverage of these adults
summary(attgt(uninspov ~ 1, data = subset(medicaid, prior == "none"),
              index = ix, treat = "adopt", control = "never"))
```

`meps` *Health care expenditure and use of US adults, MEPS panel 23 (2018-2019)*

Description

A real public two-year person panel from the US Medical Expenditure Panel Survey (MEPS) Household Component, longitudinal file HC-217 (panel 23). The unit is a person, the time index is the calendar year (2018, 2019): each of 3,000 adults contributes exactly two rows. Expenditure outcomes have the classic health-economics shape (a mass of zeros plus a long right tail), utilisation outcomes are overdispersed counts with many zeros, and insurance, income and chronic conditions vary over time within person, which makes the data a natural testbed for two-part, hurdle, tobit, gamma, quantile, selection and correlated random effects (Mundlak) panel models.

Usage

```
meps
```

Format

A data frame with 6,000 rows (3,000 persons x 2 years) and 20 columns. The panel index pair is `c("id", "year")`.

id integer person identifier (1-3000), the panel individual index; a new sequential id, not the MEPS `DUPERSID`

year integer calendar year (2018 or 2019), the panel time index (MEPS suffix Y1, Y2)

totexp numeric total health care expenditure in the year, US dollars, all payers (`TOTEXPY1`, `TOTEXPY2`); 14 percent zeros

oopexp numeric out-of-pocket (self or family) expenditure, US dollars (`TOTSLFY1`, `TOTSLFY2`); never larger than `totexp`

anyexp integer indicator (0/1) of any health care expenditure in the year, `as.integer(totexp > 0)`

officevis integer number of office-based provider visits (`OBTOTVY1`, `OBTOTVY2`)

ervis integer number of emergency room visits (`ERTOTY1`, `ERTOTY2`)

ipstays integer number of hospital inpatient discharges (`IPDISY1`, `IPDISY2`)

insurance factor health insurance coverage in the year (`INSCOVY1`, `INSCOVY2`): `private` (any private coverage), `public` (public coverage only, e.g. Medicare, Medicaid), `uninsured` (uninsured all year); reference level `private`

age integer age in years at the end of the year (or last age in scope), top-coded at 85 (`AGEY1X`, `AGEY2X`). Age rises by one year for (almost) everyone, so within person it is collinear with the year: use it in pooled and random effects models but not as a time-varying regressor in within or Mundlak (`cre = TRUE`) fits

sex factor `male`, `female` (`SEX`), time-invariant

race factor race and ethnicity (RACETHX): hispanic, white (non-Hispanic white only), black (non-Hispanic black only), asian (non-Hispanic Asian only), other (non-Hispanic other or multiple race); time-invariant

region factor Census region at the end of the year (REGIONY1, REGIONY2): northeast, midwest, south, west

povcat factor family income as a percent of the federal poverty line, in categories (POVCATY1, POVCATY2): poor (below 100 percent, including negative income), nearpoor (100 to below 125), low (125 to below 200), middle (200 to below 400), high (400 or more)

povlev numeric family income as a percent of the poverty line, one decimal (POVLEVY1, POVLEVY2); can be negative

health ordered factor self-rated (perceived) health status, excellent < verygood < good < fair < poor, from the interview round closing the year (2018: round 3 RTHLTH3; 2019: round 5 RTHLTH5), as in the MEPS annual files; 4 values NA (refused, don't know or not ascertained)

nchronic integer number (0-8) of eight MEPS priority conditions ever diagnosed as reported in that year: high blood pressure, coronary heart disease, stroke, emphysema, high cholesterol, diabetes, arthritis, asthma (HIBPDXYt, CHDDXYt, STRKDXYt, EMPHDXYt, CHOLDXYt, DIABDXYt_M18, ARTHDXYt, ASTHDXYt); refused, don't know and inapplicable answers count as not diagnosed. The longitudinal file carries the same "ever diagnosed" answers in both years, so nchronic is constant within person: a baseline morbidity measure, dropped by within (fixed effects) estimators

longwt numeric MEPS panel 23 longitudinal person weight (LONGWT), constant within person

varstr integer variance estimation stratum (VARSTR)

varpsu integer variance estimation primary sampling unit within stratum (VARPSU)

Details

The extract keeps persons with data for all five interview rounds (ALL5RDS = 1, in scope for the whole of 2018 and 2019) who were aged 18 or older in 2018, and draws a stratified random sample of 3,000 of the 10,084 eligible persons (strata: 2018 age 18-34, 35-49, 50-64, 65+; proportional allocation; set.seed(20261101)). The original person identifier DUPERSID is replaced by a new sequential id so that rows cannot be matched back to the public file or linked to other sources. The build script, with its validation against the codebook and against official AHRQ estimates, is data-raw/meps.R in the package's GitHub repository.

Because the sample is drawn with equal probability within age strata, longwt remains approximately valid for weighted descriptive statistics of the adult population (the 2018 and 2019 weighted means of totexp in the sample are within 2 percent of those of all eligible adults in the full file), but the design variables varstr and varpsu only describe the original MEPS design: design-based standard errors computed from this subsample are approximate. The panglm examples below are unweighted model-based fits, as usual for teaching.

Teaching use

All outcomes are observed for every person in both years, so the data are a balanced short panel (T = 2): within and Mundlak (CRE) estimators are identified only from people whose insurance or income changed between 2018 and 2019 (223 insurance changes, 1,114 income-category changes), and age and nchronic should enter only pooled or random effects fits.

- **Does insurance status change health spending, and through which margin?** `totexp` is zero for 14 percent of person-years and highly skewed: `twopart()` (probit + gamma, `model = "pooling"` or `"random"`) with `partialeffects()` for the dollar effect of being uninsured; compare with `panglm(family = "tobit")` on `log1p(totexp)`.
- **Is the cross-sectional insurance gap causal or selection?** `panglm()` `family = "gamma"` on positive spending, `model = "pooling"` or `"random"` versus `model = "within"` (identified from people who gained or lost coverage).
- **Who uses any care?** `anyexp` with `family = "probit"`, `model = "random"`, `cre = TRUE` (Mundlak test of the random effects assumption) and `partialeffects()`.
- **Utilisation counts.** `officevis` (overdispersed, 24 percent zeros) or `ervis`, `ipstays` (mostly zero) with `family = "negbin"` or `"poisson"`, `model = "random"` or `"within"`; `panhurdle()` for the zero versus positive split with fixed effects.
- **Distributional effects.** `panqr()` (`model = "canay"` or `"pooling"`) on `log1p(totexp)` or `log(oopexp + 1)` at several quantiles; `model = "mmqr"` needs T >= 3 and does not apply here.
- **Selection on spending.** `panselect()` with `anyexp` as the selection equation and log spending as the outcome (the alternative to the two-part model); self-rated `health` as an ordered outcome with `family = "oprobit"`, `model = "random"`.

Source

Agency for Healthcare Research and Quality (AHRQ), Medical Expenditure Panel Survey, Household Component, public use file HC-217: MEPS Panel 23 Longitudinal Data File (released December 2021). Rockville, MD: AHRQ. https://meps.ahrq.gov/mepsweb/data_stats/download_data_files_detail.jsp?cboPufNumber=HC-217. Data: https://meps.ahrq.gov/mepsweb/data_files/pufs/h217/h217dta.zip.

MEPS is produced by AHRQ, an agency of the US federal government; the public use files carry no copyright notice. AHRQ "requests that users cite AHRQ and the Medical Expenditure Panel Survey as the data source in any publications or research based upon these data." The files are released under the following Data Use Agreement (section A of the HC-217 documentation, https://meps.ahrq.gov/mepsweb/data_stats/download_data/pufs/h217/h217doc.shtml), which binds every user of this extract as well:

"Individual identifiers have been removed from the micro-data contained in these files. Nevertheless, under sections 308 (d) and 903 (c) of the Public Health Service Act (42 U.S.C. 242m and 42 U.S.C. 299 a-1), data collected by the Agency for Healthcare Research and Quality (AHRQ) and/or the National Center for Health Statistics (NCHS) may not be used for any purpose other than for the purpose for which they were supplied; any effort to determine the identity of any reported cases is prohibited by law. Therefore in accordance with the above referenced Federal Statute, it is understood that: (1) No one is to use the data in this data set in any way except for statistical reporting and analysis; and (2) If the identity of any person or establishment should be discovered inadvertently, then (a) no use will be made of this knowledge, (b) the Director Office of Management AHRQ will be advised of this incident, (c) the information that would identify any individual or establishment will be safeguarded or destroyed, as requested by AHRQ, and (d) no one else will be informed of the discovered identity; and (3) No one will attempt to link this data set with individually identifiable records from any data sets other than the Medical Expenditure Panel Survey or the National Health Interview Survey. By using these data you signify your agreement to comply with the above stated statutorily based requirements with the knowledge that deliberately making a false statement in any matter within the jurisdiction of any department or agency of the Federal Government violates Title

18 part 1 Chapter 47 Section 1001 and is punishable by a fine of up to $10,000 or up to 5 years in prison."

Validation (in `data-raw/meps.R`): on the full HC-217 file the unweighted codebook frequencies of `INSCOVY1`, `RACETHX`, `POVCATY2`, `TOTEXPY1` and `OBTOTVY2` are reproduced exactly, and `LONGWT` sums to 305.7 million over the 13,044 `ALL5RDS = 1` persons as in the documentation. The `LONGWT`-weighted mean total expenditure per person is 5,676 dollars in 2018 and 6,249 dollars in 2019, against 6,063 and 6,252 dollars from the annual files in AHRQ Statistical Briefs #533 and #540; the share with any expense is 87.1 and 84.4 percent against about 87 and 86 percent. The 2018 gap (-6 percent) is expected: the longitudinal weight covers only people in scope for the whole two years, which excludes 2018 decedents, whose end-of-life spending is high.

References

Mitchell, E. M. (2021). Concentration of healthcare expenditures and selected characteristics of persons with high expenses, U.S. civilian noninstitutionalized population, 2018. Statistical Brief #533. Agency for Healthcare Research and Quality, Rockville, MD. https://meps.ahrq.gov/data_files/publications/st533/stat533.shtml

Mitchell, E. M. (2022). Concentration of healthcare expenditures and selected characteristics of persons with high expenses, U.S. civilian noninstitutionalized population, 2019. Statistical Brief #540. Agency for Healthcare Research and Quality, Rockville, MD. https://meps.ahrq.gov/data_files/publications/st540/stat540.shtml

Cohen, J. W., Cohen, S. B. and Banthin, J. S. (2009). The Medical Expenditure Panel Survey: a national information resource to support healthcare cost research and inform policy and practice. *Medical Care*, 47(7 Suppl 1), S44-S50. doi:10.1097/MLR.0b013e3181a23e3a

Deb, P., Norton, E. C. and Manning, W. G. (2017). *Health Econometrics Using Stata*. College Station, TX: Stata Press. (Two-part, generalized linear and count models for MEPS expenditure and use data.)

Duan, N., Manning, W. G., Morris, C. N. and Newhouse, J. P. (1983). A comparison of alternative models for the demand for medical care. *Journal of Business and Economic Statistics*, 1, 115-126. doi:10.1080/07350015.1983.10509330

Examples

```
data(meps)
str(meps)
# survey-weighted mean spending per adult and share with any expense
sapply(split(meps, meps$year), function(d)
  c(mean = weighted.mean(d$totexp, d$longwt),
    anyexp = weighted.mean(d$anyexp, d$longwt)))

# two-part model: P(any spending) and E(spending | spending > 0)
tp <- twopart(totexp ~ insurance + I(age / 10) + sex + nchronic + povcat,
              data = meps, index = c("id", "year"), positive = "gamma")
summary(tp)
partialeffects(tp, variables = "insurance")   # dollars per person-year

# positive spending: pooled versus within (coverage switchers)
pos <- subset(meps, totexp > 0)
fe <- panglm(totexp ~ insurance + povcat, data = pos,
```

```
             index = c("id", "year"), family = "gamma", model = "within")
summary(fe)

# office-based visits: random effects negative binomial
nb <- panglm(officevis ~ insurance + I(age / 10) + sex + nchronic,
             data = meps, index = c("id", "year"), family = "negbin",
             model = "random")
summary(nb)


# any use: correlated random effects probit with Mundlak test
cre <- panglm(anyexp ~ insurance + povcat + sex + nchronic, data = meps,
              index = c("id", "year"), family = "probit",
              model = "random", cre = TRUE)
summary(cre)

# censored log spending: tobit
summary(panglm(log1p(totexp) ~ insurance + I(age / 10) + sex + nchronic,
               data = meps, index = c("id", "year"), family = "tobit",
               model = "pooling"))

# quantiles of log spending with individual effects (the many tied
# values of log1p(totexp) make quantreg warn that the solution may be
# nonunique)
qr <- suppressWarnings(panqr(log1p(totexp) ~ insurance + povlev,
  data = meps, index = c("id", "year"), tau = c(0.25, 0.5, 0.75),
  model = "canay", boot = 50, seed = 1))
summary(qr)

# selection correction for log spending observed only if positive
meps$logexp <- ifelse(meps$totexp > 0, log(meps$totexp), NA)
summary(panselect(anyexp ~ insurance + povcat, logexp ~ insurance,
                  data = meps, index = c("id", "year"),
                  model = "within", boot = 49, seed = 1))

# self-rated health: random effects ordered probit
summary(panglm(health ~ nchronic + I(age / 10) + povcat, data = meps,
               index = c("id", "year"), family = "oprobit",
               model = "random"))
```

`model.matrix.panglm` *Design matrix of a panglm fit*

Description

Returns the model matrix used for the fit (`data = NULL`) or rebuilt for new data with the stored terms, factor levels, contrasts and spline knots (`splines::ns()`, `splines::bs()` and `poly()` keep the basis of the fit). Its columns match `coef(object)`. With a regression spline, `model.matrix(fit,`

grid) %*% coef(fit) traces the fitted curve, and the covariance vcov(fit) gives pointwise confidence bands; for fixed-effects fits the curve is identified up to a constant, so center it at a reference value.

Usage

```
## S3 method for class 'panglm'
model.matrix(object, data = NULL, cre_means = NULL, ...)
```

Arguments

object	a "panglm" fit
data	optional new data containing the regressors
cre_means	for correlated random effects fits with new data: a matrix of the mean_* columns, one row per row of data; when NULL they are recomputed as individual means within data, which then needs the individual index
...	unused

Value

a numeric matrix with one column per coefficient

See Also

Other panglm methods: coefplot(), confint.panglm(), effectcurve(), fitted.panglm(), fixef(), glance.panglm(), logLik.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_ape(), plot.panglm_effectcurve(), predict.panglm(), predict.panordered(), residuals.panglm(), summary.panglm(), tidy.panglm(), vcov.panglm()

Examples

```
data(copd)
fit <- panglm(fev1 ~ splines::ns(crp, 3), data = copd,
              index = c("id", "visit"), model = "within")
grid <- data.frame(crp = seq(min(copd$crp), max(copd$crp), length.out = 5))
X <- model.matrix(fit, grid)
X0 <- model.matrix(fit, data.frame(crp = median(copd$crp)))
Xc <- sweep(X, 2, X0)            # centered at the median CRP
cbind(grid, effect = drop(Xc %*% coef(fit)),
      se = sqrt(rowSums((Xc %*% vcov(fit)) * Xc)))
```

`pandispersiontest` *Pearson chi-squared/df overdispersion test*

Description

A quick diagnostic for count models: the Pearson goodness-of-fit statistic `sum((y - mu)^2 / mu)` divided by its residual degrees of freedom should be close to 1 under a correctly-specified Poisson model; values well above 1 indicate overdispersion (evidence for a negative binomial specification instead). The diagnostic is available for pooled and fixed-effects Poisson or negative binomial models. It is not applied to random-effects models because their marginal variance includes the fitted mixing distribution and is not equal to the observation-level Poisson variance used by this statistic.

Usage

```
pandispersiontest(object)
```

Arguments

`object`	a `"panglm"` fit with `family` `"poisson"` or `"negbin"`

Value

a list of class `"panglm_dispersiontest"` with the Pearson statistic, its degrees of freedom, the ratio (statistic/df), a one-sided p-value against `chisq(df)`, and `n_excluded`: the number of observations dropped from the statistic because their fitted value was exactly zero or unavailable (e.g. rows in a structurally-zero panel individual screened out of a fixed-effects negbin fit)

Validation

No reference implementation exists for panel fits; the package tests check that it flags simulated overdispersed panels and not equidispersed ones.

See Also

`negbinfamily()` for the negative binomial alternative.

Other tests: `jointtest()`, `lineartest()`, `panhausman()`

Examples

```
data(copd)
fit <- panglm(exacerbations ~ treatment + age + smoker + crp, data = copd,
              index = c("id", "visit"), model = "pooling", family = "poisson")
pandispersiontest(fit)
```

panelplot *Plot a panel as a unit by period heat map*

Description

Draws one row per individual and one column per period. With `value` the cells are coloured by that variable: a sequential palette with a colour bar for a numeric variable, one colour per level for a factor, character or logical variable. Without `value` the plot shows which periods each individual is observed in, so the gaps of an unbalanced panel stand out. Periods in which an individual is not observed, and missing values of `value`, are drawn in neutral grey.

Usage

```
panelplot(
  data,
  index,
  value = NULL,
  treat = NULL,
  labels = NULL,
  col = NULL,
  zlim = NULL,
  legend = TRUE,
  xlab = "Period",
  ylab = NULL,
  main = NULL,
  ...
)
```

Arguments

`data`	a data frame in long format
`index`	`c(individual, time)`: column names; the time must be numeric when `treat` is given
`value`	optional name of the column that colours the cells; `NULL` (the default) shows observed versus missing periods
`treat`	optional name of the column with the first treated period, as in `attgt()` (`0`, `NA` or `Inf` for never-treated units)
`labels`	logical: show the individual labels on the left axis. The default shows them for at most 60 individuals.
`col`	colours: for a numeric `value`, the palette (default: a sequential blue palette); for a discrete `value`, one colour per level; without `value`, the colour of the observed cells
`zlim`	limits of the colour scale for a numeric `value` (default: its range)
`legend`	logical: add a legend (or colour bar) below the plot
`xlab, ylab, main`	axis labels and title (default `ylab`: none)
`...`	passed to `graphics::image()`

Details

With `treat` the individuals are sorted by treatment cohort as in `treatmentplot()` (earliest adopters at the top, never-treated units at the bottom), the cohorts are separated by white lines and labelled on the right with their sizes, and a black edge marks the first treated period of each cohort. Without `treat` the individuals are in the order of their identifiers. The unit labels are shown for at most 60 individuals by default; with many individuals the tiles are drawn as a raster.

Value

a data frame with one row per individual in plotting order (`id`, `cohort` when `treat` is given, `n_periods` observed and `n_missing` missing values of `value`), invisibly

See Also

`treatmentplot()` for the treatment timing alone, `trendplot()` for the mean outcome paths

Other policy evaluation: `attgt()`, `did()`, `etwfe()`, `eventplot()`, `eventstudy()`, `treatmentplot()`, `trendplot()`

Examples

```
set.seed(1)
d <- expand.grid(id = 1:30, year = 2010:2019)
d$first <- rep(c(2013, 2015, 0), each = 10)[d$id]
d$y <- rnorm(30)[d$id] + 0.1 * (d$year - 2010) + rnorm(nrow(d), sd = 0.3)
d$y[sample(nrow(d), 15)] <- NA
# observed and missing periods of an unbalanced panel
u <- d[-sample(nrow(d), 40), ]
panelplot(u, index = c("id", "year"))
# outcome values, sorted by treatment cohort
panelplot(d, index = c("id", "year"), value = "y", treat = "first")
# a discrete variable
d$high <- d$y > 0.5
panelplot(d, index = c("id", "year"), value = "high")
```

`panfmm` *Latent class (finite mixture) models for panel data*

Description

Finite mixture regression in which every individual belongs to one of `k` latent classes for all of its observations, the panel latent class model of Deb and Trivedi (1997, 2002) and Bago d'Uva (2006). Within class `c` the outcome follows a GLM with class-specific coefficients `beta_c` (and dispersion), and the individual likelihood is `L_i = sum_c pi_ic prod_t f_c(y_it | x_it)`. Class probabilities are constant (`pi_c`) or depend on individual-level concomitant variables through a multinomial logit. In health economics the classes typically separate a large group of low users of care from a smaller group of high users with a different response to income, insurance or health status.

Usage

```
panfmm(
  formula,
  data,
  index,
  k = 2,
  family = "poisson",
  concomitant = NULL,
  nstart = 10,
  seed = NULL,
  vcov = c("classical", "cluster"),
  maxit = 1000,
  tol = 1e-06,
  na.action = stats::na.omit,
  weights = NULL,
  offset = NULL
)
```

Arguments

formula	a model formula `y ~ x1 + x2` for the class-specific regressions (the same regressors in every class). It may contain `offset()` terms, e.g. `offset(log(exposure))` for counts over different exposure periods; the offset enters the linear predictor of every class with coefficient 1. `lag(x, k)`, `lead(x, k)` and `diff(x)` are taken within each individual on the grid of sorted distinct time values (see "Lags and differences" in `panglm()`; needs `index = c(id, time)`); the first waves of each individual get `NA` and leave the sample, and `lag(x, 0:2)` gives several lags
data	a data.frame in long format
index	length-1 or length-2 character vector giving the individual (and time) columns; class membership is constant within individual. Use a row identifier for cross-section data
k	number of latent classes
family	`"poisson"`, `"negbin"` (NB2), `"gaussian"`, `"binomial"` (logit) or `"probit"` (binomial with probit link), or the corresponding panglm family spec, e.g. `binomialfamily("probit")`
concomitant	optional one-sided formula `~ z1 + z2` of individual-level (time-invariant) variables for the class probabilities
nstart	number of EM starts (the first deterministic, the others random)
seed	optional random seed for the random starts; the global random number stream is restored afterwards
vcov	`"classical"` (inverse negative Hessian) or `"cluster"` (sandwich clustered by individual); see `vcov.panfmm()`
maxit	maximum number of EM iterations per start
tol	convergence tolerance on the relative change of the EM log-likelihood
na.action	missing-data action passed to `stats::model.frame()`

`weights` optional nonnegative individual (sampling) weights: the name of a column of `data` or a numeric vector with one value per row of `data`, constant within individuals; see the Weights section

`offset` optional numeric offset (the name of a column of `data` or a vector with one value per row of `data`), added to the linear predictor of every class (in addition to any `offset()` terms of `formula`), as in `stats::glm()`

Value

an object of class `"panfmm"`, a list with components `coefficients` (all parameters, named `class<c>:<term>` and `prior<c>:<term>`), `vcov`, `vcov_type`, `classcoef` (a matrix of class-specific regression coefficients, one column per class), `sigma` (Gaussian) or `theta` (NB2) by class, `prior` (the class probabilities of each individual, one row per individual; with no concomitant variables all rows are equal), `prior_mean` (their averages), `posterior` (posterior class probabilities, one row per individual, rownames the individual identifiers), `class` (the modal posterior class), `classmean` (the mean outcome of each class, used for the label order), `loglik`, `df`, `starts` (log-likelihood of each EM start), `nobs`, `n_groups`, `weights` (the individual weights, `NULL` without weights), `offset` (the total offset of each observation in panel order, `NULL` without offsets), `k`, `family`, `call` and `formula`. Methods: `print`, `summary`, `coef`, `vcov`, `confint`, `predict`, `logLik` (hence `AIC` and `BIC`, with `n` the number of observations), `nobs`, `tidy` and `glance`.

Estimation

The EM algorithm is run from `nstart` starting partitions of the individuals: the first splits them by the quantiles of their mean outcome, the others are random. The E step computes the posterior class probabilities `tau_ic` of each individual from the whole history `y_i1, ..., y_iT`; the M step fits one weighted GLM per class (weights `tau_ic` repeated over the individual's observations, iteratively reweighted least squares warm-started at the previous iteration's coefficients; NB2 by IRLS and a Newton step for `theta`) and the class probabilities (sample means of `tau_ic`, or a weighted multinomial logit on the concomitant variables). The run with the largest log-likelihood is then polished by BFGS and Newton steps on the observed-data log-likelihood `sum_i log L_i`, whose numerical Hessian (from analytic scores) gives the classical covariance `solve(-H)`. The cluster covariance is the sandwich `B (sum_g s_g s_g') B` with `B = solve(-H)`, the individual scores `s_i` summed within clusters (the individual by default) and the factor `G / (G - 1)`, the convention used in the rest of the package for likelihoods whose units are individuals.

Weights

Class membership is a property of the individual, so `weights` are individual (sampling) weights: they must be constant within individuals (one weight per individual, repeated over its rows) and `panfmm()` stops otherwise. The fit maximizes the weighted pseudo log-likelihood `sum_i w_i log L_i`, the convention of flexmix (whose integer weights must also be identical within groups). In the EM algorithm the M step weights become `w_i tau_ic`, and the class probabilities are weighted means of the posteriors. The classical covariance is `solve(-H_w)` from the weighted Hessian, which is correct for frequency weights; for sampling weights use `vcov = "cluster"`, the sandwich `B (sum_g S_g S_g') B` with the weighted individual scores `S_i = w_i s_i` summed within clusters. Individuals with zero weight do not contribute. With weights the reported `loglik` (hence AIC and BIC) is the weighted log-likelihood, and BIC keeps `n` the number of observations (flexmix uses the sum of the weights, which differs for sampling weights scaled to population totals).

Ancillary parameters are estimated on the log scale: `log(sigma)` for the Gaussian family and `log(theta)` for NB2 (variance `mu + mu^2 / theta`, i.e. `alpha = 1 / theta`). The class probabilities are parameterized by the multinomial logit coefficients of classes `2, ..., k` against class 1 (an intercept only when there are no concomitant variables), and these are the parameters reported in `coef()` and `vcov()`; the implied probabilities are in `fit$prior`.

Label switching

Mixture likelihoods are invariant to permutations of the class labels. `panfmm()` orders the classes by ascending mean outcome: the mean over all observations of the class-specific fitted mean `mu_c(x_it)`, so that class 1 is the low-use class and class `k` the high-use class. The multinomial logit of the class probabilities uses class 1 as reference.

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

References

Deb, P. and Trivedi, P. K. (1997). Demand for medical care by the elderly: a finite mixture approach. *Journal of Applied Econometrics*, 12, 313-336.

Deb, P. and Trivedi, P. K. (2002). The structure of demand for health care: latent class versus two-part models. *Journal of Health Economics*, 21, 601-625. doi:10.1016/S01676296(02)000085

Bago d'Uva, T. (2006). Latent class models for utilisation of health care. *Health Economics*, 15, 329-343. doi:10.1002/hec.1112

Gruen, B. and Leisch, F. (2008). FlexMix version 2: finite mixtures with concomitant variables and varying and constant parameters. *Journal of Statistical Software*, 28(4). doi:10.18637/jss.v028.i04

See Also

`summary.panfmm()`, `vcov.panfmm()` and `predict.panfmm()` for the fitted model, tidy-methods for `tidy()` and `glance()`.

Other panel models: `panfmmselect()`, `panglm()`, `pangmm()`, `panhurdle()`, `paniv()`, `panqr()`, `panselect()`, `twopart()`

Examples

```
# Doctor visits of 400 patients over 4 years: 70% low users and 30%
# high users with a stronger effect of chronic illness
set.seed(1)
N <- 400; TT <- 4
d <- data.frame(id = rep(seq_len(N), each = TT), year = rep(seq_len(TT), N))
high <- rep(rbinom(N, 1, 0.3), each = TT)
d$chronic <- rbinom(N * TT, 1, 0.4)
d$age <- rep(round(runif(N, 40, 80)), each = TT)
mu <- exp(ifelse(high == 1, 1.5 + 0.6 * d$chronic, 0 + 0.3 * d$chronic) +
          0.01 * (d$age - 60))
d$visits <- rpois(N * TT, mu)
```

```
fit <- panfmm(visits ~ chronic + age, data = d, index = c("id", "year"),
              k = 2, family = "poisson", nstart = 3, seed = 1)
summary(fit)
head(fit$posterior)
table(predict(fit, type = "class"))
```

`panfmmselect` *Compare latent class panel models with different numbers of classes*

Description

Fits `panfmm()` for each number of classes in `k` and tabulates the log-likelihood, number of parameters, AIC and BIC (with `n` the number of observations), the usual criteria for choosing the number of classes (Deb and Trivedi 2002 prefer BIC). Mixtures with too many classes may fail to converge or return near-empty classes; such fits are reported with `NA`.

Usage

```
panfmmselect(
  formula,
  data,
  index,
  k = 1:3,
  family = "poisson",
  concomitant = NULL,
  ...
)
```

Arguments

`formula`, `data`, `index`, `family`, `concomitant`, `...`	passed to `panfmm()`
`k`	integer vector of numbers of classes

Value

a data.frame with columns `k`, `logLik`, `df`, `AIC`, `BIC` and `smallest` (the smallest mean class probability), with the fits in the attribute `"fits"`

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

See Also

Other panel models: `panfmm()`, `panglm()`, `pangmm()`, `panhurdle()`, `paniv()`, `panqr()`, `panselect()`, `twopart()`

Examples

```
set.seed(3)
d <- data.frame(id = rep(1:150, each = 3), t = rep(1:3, 150), x = rnorm(450))
d$y <- rpois(450, exp(ifelse(rep(rbinom(150, 1, 0.4), each = 3) == 1, 1.5, 0) + 0.3 * d$x))
panfmmselect(y ~ x, data = d, index = c("id", "t"), k = 1:2, nstart = 2, seed = 1)

# up to three classes (a few seconds)
panfmmselect(y ~ x, data = d, index = c("id", "t"), k = 1:3, nstart = 2, seed = 1)
```

---

panglm *Fit a generalized linear model for panel data*

Description

Fast pooled, fixed-effects ("within") and random-effects panel GLMs, backed by an Rcpp/RcppArmadillo/RcppParallel numerical core. See the package vignette for the estimators currently implemented per family.

Usage

```
panglm(
  formula,
  data,
  index,
  model = c("pooling", "within", "random", "between"),
  family = "gaussian",
  weights = NULL,
  offset = NULL,
  na.action = stats::na.omit,
  effect = c("individual", "twoways"),
  vcov = c("classical", "HC1", "cluster"),
  cre = dynamic,
  dynamic = FALSE,
  random = ~1,
  R = 21,
  maxit = 100,
  tol = 1e-10
)
```

Arguments

formula	a model formula, e.g. `y ~ x1 + x2`. `lag(x, k)`, `lead(x, k)` and `diff(x)` are taken within each individual on the grid of sorted distinct time values (see "Lags and differences" in `panglm()`; needs `index = c(id, time)`); the first waves of each individual get `NA` and leave the sample, and `lag(x, 0:2)` gives several lags
data	a data.frame containing the variables in `formula` and `index`

index — length-1 or length-2 character vector identifying the individual (and optionally time) columns, e.g. c("id", "time")

model — one of "pooling", "within", "random", "between". The between estimator (family = "gaussian" only) is least squares on the individual means of the outcome and the regressors, one row per individual, as in plm(model = "between"); it uses only the variation between individuals and is the cross-sectional counterpart of the within estimator. Its robust covariances ("HC1", "cluster") are the same, since each individual is one observation of the means regression.

family — one of "gaussian", "poisson", "binomial", "probit", "negbin", "gamma", "ologit", "oprobit", "beta", "tobit", or a family spec from gaussianfamily() / poissonfamily() / binomialfamily() / negbinfamily() / gammafamily() / orderedfamily() / betafamily() / tobitfamily(). Base R family objects with a matching distribution and link are also accepted and mapped onto these: stats::gaussian(), stats::poisson(), stats::binomial("logit"), stats::binomial("probit" and stats::Gamma("log"); other stats families stop with an error naming the panglm constructor to use.

weights — optional nonnegative observation (prior) weights: the name of a column of data or a numeric vector with one value per row of data, e.g. inverse probability weights for attrition. Nontrivial weights are supported for model = "pooling" (all families); the fit then maximizes the weighted (pseudo) log-likelihood and robust covariances use the weighted scores. For model = "random" the weights must be constant within individuals (one weight per individual, e.g. a baseline sampling weight) and the fit maximizes the weighted sum of the individual integrated log-likelihoods; the Gaussian model is then estimated by maximum likelihood instead of Swamy-Arora FGLS. Use vcov = "cluster" for design-based standard errors. Weighted fixed-effects fits are rejected.

offset — optional model offset (the name of a column of data or a numeric vector with one value per row of data). Nonzero offsets, including offsets specified in formula, are rejected explicitly rather than ignored.

na.action — missing-data action passed to stats::model.frame()

effect — one of "individual" or "twoways" (individual + time fixed effects). "twoways" is implemented for model = "within" only: family = "gaussian" (iterative alternating-projections demeaning, iterated until it converges to tolerance, with a warning if the limit of 10,000 sweeps is reached first), "poisson" and "negbin" (both via outer-IRLS / inner-weighted-FWL, the algorithm behind fixest::feglm()), and "binomial", the two-way conditional (Chamberlain) logit: the individual effects are conditioned out exactly and the time effects are estimated as time dummies inside the conditional likelihood, as in survival::clogit(y ~ x + factor(time) + strata(id), method = "exact"). For the two-way logit coef() holds the slopes only; the time effects (first time level as reference, NA when not identified) and their standard errors are in fit$time_effects and fit$time_effects_se, and the classical and robust covariances of the slopes account for their estimation.

vcov — one of "classical", "HC1" (heteroskedasticity-robust) or "cluster" (cluster-robust, clustered by the panel individual by default). For model = "random" only "classical" and "cluster" are available; the cluster covariance uses the

individual scores of the integrated likelihood, so clusters must contain whole individuals. See vcov.panglm() to recompute a different type after fitting.

cre — logical. If TRUE, fit a correlated random effects (Mundlak) model: the individual means of every time-varying regressor are added to the design matrix, named mean_<column>. Available for model = "pooling" and model = "random" and every family. The slopes of the time-varying regressors then have a within interpretation, and the joint Wald test of the mean terms is a Hausman-type (Mundlak) test, reported by summary.panglm() with the covariance stored on the fit: it is robust to heteroskedasticity and within-individual correlation only when that covariance is (vcov = "cluster", or "HC1" for heteroskedasticity alone); with the default classical covariance it assumes the model's error structure. Defaults to dynamic.

dynamic — logical. If TRUE, fit a dynamic model with the Wooldridge (2005) solution to the initial conditions problem: the outcome of the previous wave (lag_<y>) and the outcome of the first observed wave (initial_<y>) are added to the regressors, the first wave is dropped, and the individual means of the time-varying regressors are added (cre = TRUE by default; the lag is not averaged). The second element of index must be a numeric time variable; a lag is only defined when the previous wave t - 1 is observed, other rows are dropped. Factor and logical outcomes (ordered and binary models) give factor or logical lags, so their partial effects are discrete changes. With model = "random" the coefficient of the lag measures true state dependence net of unobserved heterogeneity; partialeffects() gives it on the probability or mean scale. Available for model = "pooling" and model = "random".

random — one-sided formula for the random effects of model = "random". The default ~ 1 is the random intercept. A formula with variables, e.g. ~ time, adds correlated random slopes: g(E[y_it | b_i]) = x_it' beta + z_it' b_i with z_it = (1, time) and b_i ~ N(0, Sigma), Sigma unstructured (Cholesky parameterized), the model of lme4::lmer(y ~ x + (1 + time | id)). The slope variables usually also appear in formula (their average slope). Available for family = "gaussian" (exact maximum likelihood, any number of slopes), and for "poisson", "negbin" (NB2, the model of glmmTMB::nbinom2(), with theta stored as negbin_theta), "gamma" (log link, the model of glmmTMB with Gamma(link = "log"), shape stored as shape) and "binomial" (logit or probit), with any number of slopes, by adaptive Gauss-Hermite product quadrature with R nodes per dimension, R^q nodes for q random effects; R = 1 is the Laplace approximation of lme4::glmer() and glmmTMB. With random slopes the random effects are always normal on the link scale, so for "poisson" and "negbin" the multiplicative effect exp(z_it' b_i) is lognormal. These are different models from the random-intercept fits of random = ~ 1: the random-intercept "poisson" model uses gamma mixing (the conjugate model of Hausman, Hall and Griliches, whose likelihood is in closed form, as Stata's xtpoisson, re) and the random-intercept "negbin" model is the beta-mixed negative binomial of Hausman, Hall and Griliches (Stata's xtnbreg, re), whereas with random slopes "negbin" is NB2 with a lognormal effect, as glmmTMB::nbinom2(). Adding a slope whose variance is estimated at zero therefore does not reproduce the random = ~ 1 fit for these two families: the slope coefficients keep their conditional (rate ratio) meaning, but the intercept (E[exp(b_i)] = exp(sigma^2 /

2) under lognormal mixing against 1 under gamma mixing), the variance parameters and the log-likelihood differ. For "binomial" and "gamma" the random-intercept models are already normal on the link scale and are the zero-slope special case. The fit stores varcomp (the covariance matrix Sigma), ranef (posterior modes, the BLUPs in the Gaussian model, one row per individual) and, for the Gaussian model, the residual variance sigma_v2. Weights, dynamic = TRUE and the other families are not supported with random slopes; cre = TRUE is. Response predictions and partialeffects() are population averages over the random effects.

R — number of adaptive Gauss-Hermite quadrature nodes for the random-effects models whose likelihood has no closed form: binomial (logit, probit), gamma, ordered, beta and Tobit. The default R = 21 reproduces lme4::glmer(nAGQ = 21) and ordinal::clmm(nAGQ = 21) to about 1e-5 in the coefficients and the log-likelihood (for a probit model with a large random-intercept spread, say sigma > 1.5, raise R to 41 or more for that accuracy). The nodes are centred at the mode of each individual's integrand and scaled by its curvature, so R = 1 is the Laplace approximation (with the observed curvature, except for the beta model, which uses the expected one). R is ignored by the closed-form random-intercept Gaussian, Poisson and negative binomial models. With random slopes, R is the number of nodes per dimension of the adaptive product rule; when it is not given it is 7 for one slope, 5 for two, 3 for three to five and 1 (Laplace) beyond, so that R^q stays moderate, and a warning is given when R^q exceeds 2000.

maxit — maximum IRLS/Newton iterations

tol — convergence tolerance

Value

an object of class "panglm" (with the additional class "panordered" for the ordered families): a list with coefficients, vcov, vcov_type, fitted.values, linear.predictors, loglik, the variance components of random-effects fits (sigma_mu2, sigma_v2, theta), the panel structure, nobs, n_groups, call and formula. Methods: print, summary, coef, vcov, confint, predict, fitted, residuals, logLik, nobs, model.matrix, plot, partialeffects(), effectcurve(), tidy and glance.

Lags and differences

Inside the formula, lag(x, k), lead(x, k) and diff(x, lag, differences) are taken within each individual, never across individuals. Shifts step through the sorted distinct values of the time index, so lag(x, 1) is the value at the previous observed wave. With gaps in the time index this differs from plm: for waves coded 2010, 2012, 2014, lag(x, 1) at 2012 is the 2010 value in panglm, while plm::lag() (which shifts by the numeric time value) gives NA. To obtain plm's behaviour, keep a row for every period, with NA outcomes for the missing waves. pangmm() uses the same rule. The first k waves of each individual get NA and leave the estimation sample. diff() follows base R's diff(x, lag, differences); plm's third argument of diff() is shift, not differences. The same rule applies in paniv(), panqr() and panfmm().

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

See Also

panglm-families for the outcome families, summary.panglm(), vcov.panglm(), predict.panglm() and partialeffects() for the fitted model, tidy-methods for tidy() and glance().

Other panel models: panfmm(), panfmmselect(), pangmm(), panhurdle(), paniv(), panqr(), panselect(), twopart()

Examples

```
data(copd)

# pooled OLS on lung function (ignores the panel structure)
fit_pool <- panglm(fev1 ~ treatment + age + smoker + crp, data = copd,
                   index = c("id", "visit"), model = "pooling", family = "gaussian")
summary(fit_pool)

# fixed-effects ("within") estimator: only the time-varying regressor
# (crp) survives demeaning; time-invariant covariates like treatment
# or age are not identified by a fixed-effects estimator
fit_fe <- panglm(fev1 ~ crp, data = copd,
                 index = c("id", "visit"), model = "within", family = "gaussian")
coef(fit_fe)

# random-effects (Swamy-Arora) estimator
fit_re <- panglm(fev1 ~ treatment + age + smoker + crp, data = copd,
                 index = c("id", "visit"), model = "random", family = "gaussian")
coef(fit_re)

# fixed-effects Poisson on exacerbation counts
fit_pois <- panglm(exacerbations ~ crp, data = copd,
                   index = c("id", "visit"), model = "within", family = "poisson")
coef(fit_pois)

# correlated random effects (Mundlak) probit for hospitalization
fit_cre <- panglm(hospitalized ~ treatment + age + crp, data = copd,
                  index = c("id", "visit"), model = "random",
                  family = "probit", cre = TRUE)
summary(fit_cre)

# patient trajectories: random intercept and random slope of FEV1
# decline per visit, correlated
fit_rs <- panglm(fev1 ~ visit + treatment + age + smoker, data = copd,
                 index = c("id", "visit"), model = "random",
                 random = ~ visit)
summary(fit_rs)
fit_rs$varcomp
```

```
# two-way conditional logit: smoking across survey waves, with year
# shocks (e.g. a tobacco tax) absorbed by the time effects
set.seed(1)
smk <- expand.grid(wave = 1:5, id = 1:200)
smk$price <- rnorm(nrow(smk)) + 0.2 * smk$wave
smk$smoking <- rbinom(nrow(smk), 1, plogis(rnorm(200)[smk$id] - 0.5 * smk$price +
                                           c(0, 0.2, -0.3, -0.6, -0.8)[smk$wave]))
fit_2w <- panglm(smoking ~ price, data = smk, index = c("id", "wave"),
                 model = "within", family = "binomial", effect = "twoways",
                 vcov = "cluster")
summary(fit_2w)
fit_2w$time_effects

# nonlinear effect through a natural spline: linearity test and curve
fit_ns <- panglm(fev1 ~ splines::ns(crp, 3), data = copd,
                 index = c("id", "visit"), model = "within", vcov = "cluster")
lineartest(fit_ns, "crp")
plot(effectcurve(fit_ns, "crp"))
```

`panglm-families` *Family specifications for panglm*

Description

Lightweight family objects used by `panglm()`, analogous in spirit to stats::family objects but keyed to the integer codes the C++ backend expects (see `src/families.h`).

Usage

```
gaussianfamily(link = "identity")

poissonfamily(link = "log")

binomialfamily(link = c("logit", "probit"))

negbinfamily(link = "log")

gammafamily(link = "log")

orderedfamily(link = c("logit", "probit"))

betafamily(link = c("logit", "probit"))

tobitfamily(left = 0, right = Inf)
```

Arguments

`link`	canonical link name
`left, right`	censoring limits of `tobitfamily()`

Details

`gammafamily()` is the gamma GLM with a log link, the usual choice for skewed positive outcomes such as health care costs: the variance is proportional to the squared mean and the coefficients act multiplicatively on the mean, with no retransformation needed.

`orderedfamily()` is the ordered logit or ordered probit model for ordinal outcomes such as self-assessed health: `P(y <= j) = F(c_j - x'beta - u_i)`. The response may be a factor, an ordered factor or integer-valued; categories follow the factor levels or the sorted values. The intercept is absorbed in the cutpoints `c_j`, which are reported after the slopes. `model = "pooling"` and `model = "random"` (Gaussian random intercept by adaptive Gauss-Hermite quadrature) are available for both links, `model = "within"` for the logit link only (the "blow-up and cluster" fixed-effects ordered logit of Baetschmann, Staub and Winkelmann 2015). The fit has class `c("panordered", "panglm")`; see `predict.panordered()`.

`betafamily()` is beta regression for continuous outcomes in the open interval (0, 1), such as utility index scores rescaled to (0, 1), proportions of days in hospital or adherence fractions: `y ~ Beta(mu * phi, (1 - mu) * phi)` with `g(mu) = x'beta`, so `E(y) = mu` and `Var(y) = mu (1 - mu) / (1 + phi)`. The precision `phi` is reported as `fit$precision`. `model = "pooling"` and `model = "random"` (Gaussian random intercept on the link scale, adaptive Gauss-Hermite quadrature) are available; there is no fixed-effects estimator. Outcomes equal to 0 or 1 must be transformed first, e.g. with `(y * (n - 1) + 0.5) / n` (Smithson and Verkuilen 2006).

`tobitfamily()` is the censored normal (Tobit) model for outcomes piled up at a floor or a ceiling, such as health scores at the ceiling of a scale (motor function scores, EQ-5D utility at 1) or spending at zero: the latent outcome is `y* = x'beta + u_i + e_it` with `e_it ~ N(0, sigma_e^2)`, and `y = left` when `y* <= left`, `y = right` when `y* >= right`, `y = y*` otherwise. Use `left = -Inf` or `right = Inf` for one-sided censoring. `model = "pooling"` and `model = "random"` (Gaussian random intercept `u_i ~ N(0, sigma_u^2)`, adaptive Gauss-Hermite quadrature) are available. There is no fixed-effects Tobit (incidental parameters); use `model = "random"` with `cre = TRUE`. `predict(type = "link")` returns the latent mean `x'beta` and `predict(type = "response")` the mean of the observed, censored outcome (averaged over the random intercept), which is the scale of `partialeffects()`. `fit$sigma_e` and `fit$sigma` (random intercept SD) are stored on the fit.

Value

a list with `family`, `link`, `family_id`, `link_id`

References

Baetschmann, G., Staub, K. E. and Winkelmann, R. (2015). Consistent estimation of the fixed effects ordered logit model. *Journal of the Royal Statistical Society A*, 178, 685-703. doi:10.1111/rssa.12090

Ferrari, S. and Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions. *Journal of Applied Statistics*, 31, 799-815. doi:10.1080/0266476042000214501

Smithson, M. and Verkuilen, J. (2006). A better lemon squeezer? Maximum-likelihood regression with beta-distributed dependent variables. *Psychological Methods*, 11, 54-71. doi:10.1037/1082-989X.11.1.54

Tobin, J. (1958). Estimation of relationships for limited dependent variables. *Econometrica*, 26, 24-36. doi:10.2307/1907382

See Also

panglm(), whose family argument takes these objects, and panglm-deprecated for the former names.

Examples

```
gaussianfamily()
poissonfamily()
binomialfamily("probit")
negbinfamily()
gammafamily()
orderedfamily("probit")
betafamily()
tobitfamily(left = 0)

data(copd)
fit <- panglm(hospitalized ~ crp, data = copd, index = c("id", "visit"),
              model = "within", family = binomialfamily("logit"))
coef(fit)
```

---

pangmm *Difference and system GMM for dynamic linear panel models*

Description

Arellano and Bond (1991) difference GMM and Blundell and Bond (1998) system GMM for linear panel models with a lagged dependent variable and an unobserved individual effect, `y_it = sum_j rho_j y_i,t-j + x_it'b`. Both one-step and two-step estimators are available, with robust one-step and Windmeijer (2005) corrected two-step standard errors, the Arellano-Bond tests for first- and second-order serial correlation of the differenced residuals and the Sargan and Hansen tests of the overidentifying restrictions.

Usage

```
pangmm(
  formula,
  data,
  index,
  transformation = c("d", "ld"),
  model = c("onestep", "twostep"),
  effect = c("twoways", "individual"),
  collapse = FALSE,
  maxlag = NULL,
  vcov = c("robust", "classical", "cluster")
)
```

Arguments

formula	a formula `y ~ regressors | gmm instruments` or `y ~ regressors | gmm instruments | standard ir` see the Formula section
data	a data.frame in long format
index	length-2 character vector giving the individual and time columns
transformation	`"d"` (difference GMM, Arellano-Bond) or `"ld"` (system GMM, Blundell-Bond)
model	`"onestep"` or `"twostep"`
effect	`"twoways"` (time dummies, the default) or `"individual"`
collapse	logical: collapse the GMM-style instruments to one column per lag (and, for system GMM, one column for the level equation). A named logical vector applies it to selected GMM variables, named as in the formula, e.g. `c("log(bmi)" = TRUE)`.
maxlag	optional positive integer: the largest lag of the GMM-style instruments, overriding larger upper lags in the formula (`lag(y, 2:99)` with `maxlag = 4` uses lags 2 to 4)
vcov	`"robust"` (one-step sandwich or Windmeijer-corrected two-step, both clustered by individual) or `"classical"`; `"cluster"` is accepted as a synonym of `"robust"`, the name used by the other estimators of the package. See `vcov.pangmm()` to switch after fitting

Value

an object of class `"pangmm"`, a list with components `coefficients` (the final estimates; time dummies are named by their period), `coefficients_onestep`, `vcov`, `vcov_type`, `vcov_list` (the classical and robust covariances), `residuals` (one numeric vector per individual, differenced rows first and, for system GMM, level rows after; rows not used are zero), `artest` (data frame of the Arellano-Bond tests), `sargan`, `hansen` and `wald` (lists with `statistic`, `df` and `p.value`; `wald` tests the joint significance of the coefficients other than time dummies and the constant), `sigma2` (`s2`), `n_instruments`, `nobs` (number of equations used), `nobs_diff`, `nobs_level`, `n_groups`, `n_periods`, `timedummies` (names of the time-effect coefficients), `transformation`, `model`, `effect`, `collapse`, `call` and `formula`. Methods: `print`, `summary`, `coef`, `vcov`, `confint`, `predict`, `residuals`, `nobs`, `tidy` and `glance`.

Formula

The formula follows `plm::pgmm()` and has two or three parts separated by `|`:

`y ~ lag(y, 1) + x + lag(x, 1) | lag(y, 2:99) | z`

- The first part lists the regressors. `lag(v, k)` is the lag of order `k` of `v` within the individual, `k` may be a vector (`lag(v, 0:2)` gives `v` and its first two lags) and `lag(v)` means `lag(v, 1)`. Any expression of the data can be lagged, e.g. `lag(log(bmi), 1)`. The intercept is not estimated: the individual effects are differenced out, and time effects or the system-GMM constant are added through `effect`.
- The second part lists the GMM-style instruments: `lag(v, a:b)` uses, for the differenced equation of period `t`, the levels `v_(t-a)`, ..., `v_(t-b)` that exist in the panel (`b = 99` means all available lags). For an endogenous variable `a = 2`, for a predetermined one `a = 1`.

- The optional third part lists standard (IV-style) instruments, with the same `lag()` syntax; each enters in first differences in the differenced equation (and in levels in the level equation of system GMM). Without a third part, every regressor whose variable does not appear among the GMM-style instruments is used as its own standard instrument, as in `plm::pgmm()`.

Lags are taken on the time grid formed by the sorted distinct values of the time index, so missing waves are respected and waves need not be equally spaced in calendar time (`plm` instead lags by the numeric value of the time index, so biennial waves coded as years have no lag one period back there); rows with a missing value in the response or a regressor are dropped from the moment conditions and missing instruments are set to zero, as in `plm` and Stata's `xtabond2`.

Estimators

With `transformation = "d"` (difference GMM) the model is estimated in first differences, with the GMM-style instruments of each period in their own columns (block-diagonal instrument matrix) or, with `collapse = TRUE`, one column per lag (Roodman 2009). The one-step weight matrix is `(sum_i Z_i' H Z_i)^-1` with `H` the tridiagonal matrix with 2 on the diagonal and -1 off it, the covariance of the differenced errors under homoskedasticity. With `transformation = "ld"` (system GMM) the level equations of the same periods are stacked under the differenced ones and instrumented by the first difference of each GMM-style variable lagged `a - 1` periods (`a` its smallest lag); the one-step weight uses the block matrix `H = [G, D; D', I]` of `plm::pgmm()` (`fsm = "full"`). The two-step estimator uses the weight `(sum_i Z_i' e_i e_i' Z_i)^-1` built from the one-step residuals.

With `effect = "twoways"` (the default) time dummies are included: in differences for difference GMM, and for system GMM a constant and time dummies in the level equation (which instrument themselves there). With `effect = "individual"` there are no time effects; system GMM then keeps a constant in the level equation (`plm::pgmm()` omits it).

Regressors that are not identified, because they are collinear with the time effects (for instance age in a balanced annual panel, whose first difference is constant) or with each other, are dropped with a warning, as in `panglm()`; they remain standard instruments when listed as such.

Inference

`vcov = "robust"` gives the one-step sandwich covariance or, for the two-step estimator, the Windmeijer (2005) finite-sample corrected covariance. `vcov = "classical"` gives, for the one-step estimator, `s2 (X'Z S1^-1 Z'X)^-1` with `S1 = sum_i Z_i' H Z_i` and `s2 = sum(de^2) / (2 (n_d - k))` estimated from the `n_d` differenced one-step residuals `de`; for the two-step estimator it gives the conventional (downward biased) `(X'Z S2^-1 Z'X)^-1` with `S2 = sum_i Z_i' e1_i e1_i' Z_i`. (The one-step `vcov()` of `plm::pgmm()` is `(X'Z S1^-1 Z'X)^-1 / N`, without the error variance.)

The Arellano-Bond statistic of order `m` tests for autocorrelation of order `m` in the differenced residuals; it is standard normal under the null of no autocorrelation. First-order autocorrelation is expected by construction; second-order autocorrelation indicates that `y_(t-2)` is not a valid instrument. It is computed with the robust covariance of the fit, as in `summary(plm::pgmm())`. The Sargan statistic `(Z'e1)' (sum Z_i' H Z_i)^-1 (Z'e1) / s2` uses the one-step residuals and is valid only under homoskedasticity; the Hansen J statistic `(Z'e2)' (sum Z_i' e1_i e1_i' Z_i)^-1 (Z'e2)` uses the two-step residuals and is robust. Both are chi-squared with `L - k` degrees of freedom (`L` instruments, `k` coefficients); `plm::sargan()` returns the Hansen statistic by default and `N s2` times the Sargan statistic with `weights = "onestep"`. Many instruments weaken the Hansen test (Roodman

2009); collapsing or limiting the lag depth keeps the instrument count well below the number of individuals. When the second-step matrix `S2` is singular (more instruments than individuals) a generalized inverse is used, as in `plm::pgmm()`; a two-step fit warns because its estimates depend on it, a one-step fit does not because only its Hansen statistic does.

Finite-sample bias

The estimators are consistent as the number of individuals grows, but in moderate samples they can be badly biased, and matching another package does not protect against it. Two mechanisms matter. Too many instruments overfit the endogenous regressors and pull the estimate toward pooled OLS, while making the Hansen test look too good (Roodman 2009). And when the variance of the individual effect is large relative to the idiosyncratic variance, the level moments of system GMM become weak and the estimate again drifts toward OLS (Bun and Windmeijer 2010). In a simulated BMI panel (true persistence 0.6, effect SD 1.5, error SD 1, 8 waves, 63 instruments) uncollapsed system GMM averaged 0.78 one-step and 0.72 two-step with 400 individuals, 0.65 and 0.61 with 2000 individuals; collapsing the instruments gave 0.62 already with 400. Practical rules: prefer the two-step estimator, use `collapse = TRUE` and a small `maxlag`, keep the instrument count well below the number of individuals (`pangmm()` warns when it is not), treat a Hansen p-value close to 1 as a warning sign rather than as support, and check that the coefficient of the lagged outcome lies between the fixed-effects (downward biased) and pooled OLS (upward biased) estimates (Bond 2002).

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

References

Bond, S. (2002). Dynamic panel data models: a guide to micro data methods and practice. *Portuguese Economic Journal*, 1, 141-162.

Bun, M. J. G. and Windmeijer, F. (2010). The weak instrument problem of the system GMM estimator in dynamic panel data models. *The Econometrics Journal*, 13, 95-126.

Arellano, M. and 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.

Blundell, R. and Bond, S. (1998). Initial conditions and moment restrictions in dynamic panel data models. *Journal of Econometrics*, 87, 115-143.

Windmeijer, F. (2005). A finite sample correction for the variance of linear efficient two-step GMM estimators. *Journal of Econometrics*, 126, 25-51.

Roodman, D. (2009). How to do xtabond2: An introduction to difference and system GMM in Stata. *Stata Journal*, 9, 86-136.

See Also

`summary.pangmm()`, `vcov.pangmm()` and `predict.pangmm()` for the fitted model, tidy-methods for `tidy()` and `glance()`.

Other panel models: `panfmm()`, `panfmmselect()`, `panglm()`, `panhurdle()`, `paniv()`, `panqr()`, `panselect()`, `twopart()`

Examples

```
# Habit persistence in smoking: cigarettes per day depend on last
# wave's consumption (state dependence) and on the price of tobacco,
# with a permanent individual propensity to smoke. The lagged outcome is
# correlated with that propensity, so pooled OLS overstates persistence
# and the within estimator understates it (Nickell bias); GMM is
# consistent. True values: persistence 0.5, price -0.3.
set.seed(1)
N <- 300; TT <- 7
d <- data.frame(id = rep(seq_len(N), each = TT), wave = rep(seq_len(TT), N))
propensity <- rnorm(N)
d$price <- rnorm(N * TT)
d$cigs <- NA
for (i in seq_len(N)) {
  r <- (i - 1) * TT + seq_len(TT)
  y <- propensity[i] / 0.5 + rnorm(1)
  for (t in seq_len(TT)) {
    y <- 0.5 * y - 0.3 * d$price[r[t]] + propensity[i] + rnorm(1)
    d$cigs[r[t]] <- y
  }
}

# Arellano-Bond difference GMM, two-step with Windmeijer standard errors
ab <- pangmm(cigs ~ lag(cigs, 1) + price | lag(cigs, 2:99), data = d,
             index = c("id", "wave"), model = "twostep")
summary(ab)

# Blundell-Bond system GMM with collapsed instruments and at most three
# lags of the outcome as instruments
bb <- pangmm(cigs ~ lag(cigs, 1) + price | lag(cigs, 2:99), data = d,
             index = c("id", "wave"), transformation = "ld",
             model = "twostep", collapse = TRUE, maxlag = 4)
summary(bb)

# Employment equation of Arellano and Bond (1991)
if (requireNamespace("plm", quietly = TRUE)) {
  data("EmplUK", package = "plm")
  emp <- pangmm(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, index = c("firm", "year"), model = "twostep")
  summary(emp)
}
```

---

panhausman *Hausman specification test (fixed vs. random effects)*

Description

Tests the null that the random-effects estimator is consistent (i.e. individual effects are uncorrelated with the regressors) against the alternative that only the fixed-effects estimator is consistent.

Usage

```
panhausman(
  fe,
  re = NULL,
  method = c("classical", "mundlak"),
  data = NULL,
  vcov = c("cluster", "HC1", "classical")
)
```

Arguments

fe	a "panglm" fit with model = "within"
re	a "panglm" fit with model = "random", same formula/data/family; required for method = "classical", ignored for "mundlak"
method	"classical" (default) or "mundlak"
data	for method = "mundlak": the data fe was fit on
vcov	for method = "mundlak": covariance of the pooled fit, one of "cluster" (default), "HC1", "classical"

Details

method = "classical" computes the Hausman (1978) statistic t(b_fe - b_re) %*% solve(V_fe - V_re) %*% (b_fe - b_re) over the coefficients common to both models (the random-effects intercept is dropped), with a chi-square reference. It always uses the model-based (classical) covariances, also when fe was fit with a robust vcov: the contrast variance V_fe - V_re relies on the random-effects estimator being efficient under the null, which holds only for model-based covariances and independent, homoskedastic idiosyncratic errors. In finite samples V_fe - V_re need not be positive definite and the statistic can then be negative. The signed value is returned, with p.value = NA and a warning (plm::phtest() reports its absolute value instead); the regression-based method = "mundlak" has no such problem.

method = "mundlak" is the regression-based (correlated random effects) version of the same test (Mundlak 1978; Wooldridge 2010, section 10.7.3), valid under heteroskedasticity and serial correlation. The pooled model y ~ x + xbar is fit with the individual means xbar of the time-varying regressors of fe, and the null coef(xbar) = 0 is tested with a Wald statistic using the vcov covariance. With vcov = "cluster" the reference is F(k, G - 1) (G individuals), otherwise chi-square(k). Gaussian family only.

Value

a list with class "panglm_hausman": statistic, parameter (df; two values for an F reference), p.value, method, distribution ("chisq" or "F"), pos_def ("classical" only: whether V_fe - V_re is positive definite), and the compared coefficient vectors (coef_fe, coef_re for "classical"; coef_mean for "mundlak")

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

See Also

panglm() with cre = TRUE, whose summary reports the Mundlak test.

Other tests: jointtest(), lineartest(), pandispersiontest()

Examples

```
data(copd)
fit_fe <- panglm(fev1 ~ crp, data = copd, index = c("id", "visit"),
                 model = "within", family = "gaussian")
fit_re <- panglm(fev1 ~ crp, data = copd, index = c("id", "visit"),
                 model = "random", family = "gaussian")
panhausman(fit_fe, fit_re)
panhausman(fit_fe, method = "mundlak", data = copd)
```

---

panhurdle *Fixed-effects hurdle model for panel count data*

Description

A hurdle decomposition for panels where a large fraction of groups report *structural* zeros (never observed to occur), rather than zeros arising from ordinary Poisson/NB sampling variation at a low mean, a pattern that shows up as extreme, family-invariant overdispersion under either family = "poisson" or family = "negbin" in panglm().

Usage

```
panhurdle(
  formula,
  data,
  index,
  family = c("poisson", "negbin"),
  maxit = 100,
  tol = 1e-10
)
```

Arguments

formula	a model formula, e.g. y ~ x1 + x2
data	a data.frame containing the variables in formula and index
index	length-2 character vector identifying the individual and time columns, e.g. c("id", "time")
family	count distribution for positive observations, either "poisson" or "negbin" (NB2 parameterization). It is stored on the fit as count_family.
maxit	maximum IRLS/Newton iterations
tol	convergence tolerance

Details

The model is fit in two independent parts, both using fixed effects, and reported together rather than folded into `panglm()`'s single-family dispatch, since they operate on different sample sizes (the count part drops the zeros):

1. **Zero vs. positive.** `1(y > 0)` fit via the exact conditional logistic regression already used for `panglm(..., family = "binomial")` (Chamberlain 1980): no new estimator, just a new outcome vector.
2. **Count given** `y > 0`**.** A zero-truncated Poisson or NB2 fixed-effects model (Allison-Waterman-style dummy-variable intercepts, the same unconditional approach used for FE-NB2) on the subsample where `y > 0`. The log-density is the ordinary count log-density minus `log(1 - P(Y = 0))`; see `pscl::hurdle()` for the reference form of the truncated-count likelihood.

Groups with no remaining within-group variation after truncation (a single surviving observation) contribute no information about the shared count-part slope, the same profiling fact that applies to the other fixed-effects estimators in this package; they are not dropped explicitly since a perfectly-fit singleton doesn't diverge, it simply carries zero weight. Groups whose positive counts are all equal to 1 are dropped from the count part with a message: their intercept has no finite estimate (the truncated likelihood approaches its maximum as the group mean goes to 0, whatever the slopes), so they carry no information either.

Both parts remove one effect per individual, so regressors that do not vary within individuals (sex, baseline covariates) are not identified: they are dropped with a warning, as in `panglm(model = "within")`. A regressor that varies only in individuals a part does not use (for example individuals with a single positive count, for the count part) is dropped from that part alone, with a warning.

The count part is fit by Newton-Raphson with the analytic gradient and Hessian of the joint likelihood. The group intercepts enter through a diagonal Hessian block that is eliminated at each step, so the cost is linear in the number of groups and the intercepts are never stored as dummy columns. Standard errors are the slope block of the inverse observed information of the joint likelihood (slopes, intercepts and, for NB2, `log(theta)`). When the NB2 shape parameter runs off to infinity (no overdispersion), it is stopped at `1e8` with a warning and the fit equals the zero-truncated Poisson one.

Value

an object of class `"panglm_hurdle"`: a list with `zero` (the `panglm()` binomial fit for the zero-vs-positive part) and `count` (the zero-truncated fixed-effects fit for the count part). Methods: `print`, `summary`, `coef`, `vcov`, `confint`, `nobs`, `plot`, `tidy` and `glance`.

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

See Also

hurdle-methods and `plot.panglm_hurdle()` for the fitted model, `pandispersiontest()` to check the count part, `twopart()` for continuous costs.

Other panel models: `panfmm()`, `panfmmselect()`, `panglm()`, `pangmm()`, `paniv()`, `panqr()`, `panselect()`, `twopart()`

Examples

```
data(copd)
fit <- panhurdle(exacerbations ~ crp, data = copd, index = c("id", "visit"))
fit
```

---

paniv *Instrumental-variable estimators for linear panel models*

Description

Two-stage least squares for linear panel models with endogenous regressors: fixed-effects 2SLS, random-effects generalized 2SLS (Balestra and Varadharajan-Krishnakumar) or error-components 2SLS (Baltagi), pooled 2SLS, and the Hausman and Taylor (1981) estimator, which also identifies the effects of time-invariant regressors.

Usage

```
paniv(
  formula,
  data,
  index,
  model = c("within", "random", "pooling", "hausmantaylor"),
  effect = c("individual", "twoways"),
  inst.method = c("bvk", "baltagi"),
  vcov = c("classical", "cluster"),
  na.action = stats::na.omit
)
```

Arguments

formula	a two-part formula `y ~ regressors | instruments`; see the Formula section. `lag(x, k)`, `lead(x, k)` and `diff(x)` are taken within each individual on the grid of sorted distinct time values (see "Lags and differences" in `panglm()`; needs `index = c(id, time)`); the first waves of each individual get `NA` and leave the sample, and `lag(x, 0:2)` gives several lags
data	a data.frame in long format
index	length-1 or length-2 character vector giving the individual (and time) columns
model	one of `"within"`, `"random"`, `"pooling"` or `"hausmantaylor"`
effect	`"individual"`, or `"twoways"` (only for `model = "within"`)
inst.method	instrument set for `model = "random"`: `"bvk"` (G2SLS, the default, as in `plm`) or `"baltagi"` (EC2SLS)
vcov	`"classical"` or `"cluster"` (cluster-robust by individual); see `vcov.paniv()` to recompute after fitting
na.action	missing-data action passed to `stats::model.frame()`

Value

an object of class "paniv", a list with components coefficients, vcov, vcov_type, residuals (structural residuals y - X b on the original scale, in the panel order), model, effect, inst.method, endogenous (names of the endogenous regressors), instruments (names of the excluded instruments, or for Hausman-Taylor the four variable groups in varlist), first_stage (a data frame with one row per endogenous regressor, or NULL), overid (the Sargan or Hansen test, or NULL when exactly identified), ercomp (variance components and theta for the random and Hausman-Taylor estimators), nobs, n_groups, df.residual, call and formula. Methods: print, summary, coef, vcov, confint, predict, nobs, tidy and glance.

Formula

The formula has two parts separated by |. For model = "pooling", "within" and "random" the second part lists the complete instrument set, as in plm::plm(): exogenous regressors are repeated on both sides and excluded instruments appear only on the right, so that y ~ x1 + x2 | z1 + x2 treats x1 as endogenous, instrumented by z1. Regressors absent from the second part are the endogenous ones.

For model = "hausmantaylor" the conventions of plm::pht() are used: the first part lists all regressors and the second part lists those that are exogenous (uncorrelated with the individual effect), e.g. y ~ x1 + x2 + z1 + z2 | x1 + z1 makes x2 and z2 endogenous. Whether a regressor is time varying or time invariant is detected from the data. Identification requires at least as many exogenous time-varying regressors as endogenous time-invariant ones.

Estimators

- "within": 2SLS on within-demeaned regressors and instruments (one-way or, with effect = "twoways", two-way demeaning). Time invariant columns are dropped. The classical covariance uses the degrees of freedom n - k - N (N individual effects), as in plm.
- "random": the variance components are estimated in the Swamy-Arora way from within and between 2SLS residuals (the unbalanced-panel degrees of freedom of plm are used, which reduce to the usual ones for balanced panels). The data are then quasi-demeaned with theta_i = 1 - sqrt(s2_nu / (s2_ With inst.method = "bvk" (the default, G2SLS) the instruments are quasi-demeaned as well; with inst.method = "baltagi" (EC2SLS) the instrument matrix is the within and between transforms of the instruments.
- "pooling": ordinary 2SLS on the stacked data.
- "hausmantaylor": the variance components follow plm::pht(): s2_nu is the within residual variance, and s2_1 comes from a 2SLS regression of the estimated individual effects on the time-invariant regressors, instrumented by the exogenous time-invariant and time-varying regressors. The GLS-transformed regression is then estimated by 2SLS with instruments: within transforms of all time-varying regressors, individual means of the exogenous time-varying regressors, the exogenous time-invariant regressors and a constant. For unbalanced panels s2_mu = (s2_1 - s2_nu) / T_h with T_h the harmonic mean of the T_i, as in plm::pht(). A negative s2_mu is set to zero (with a warning), which reduces the estimator to pooled 2SLS on the same instruments.

In every case the classical covariance is s2 inverse(Xhat'Xhat) with s2 the mean square of the structural residuals of the transformed regression. The cluster covariance is the sandwich B (sum_g u_g u_g') B with u_g = Xhat_g' e_g summed within the panel individual (or

the supplied cluster) and the factor `G / (G - 1) * (n - 1) / (n - k)`, the convention of the rest of the package and of `plm::vcovHC(type = "sss")`.

Diagnostics

For each endogenous regressor, the first-stage regression of its transformed values on the transformed instruments is compared with the regression on the transformed included exogenous regressors only. The result reports the partial R-squared of the excluded instruments and the F statistic for their joint significance (with the absorbed fixed effects counted in the residual degrees of freedom); for `vcov = "cluster"` a cluster-robust Wald F is reported as well. These first-stage statistics are not computed for `"hausmantaylor"`, whose instruments are built from the regressors themselves. When the model is overidentified, the Sargan statistic `n e'P_W e / e'e` (classical) or the cluster-robust Hansen J statistic of the two-step efficient GMM estimator (cluster) tests the validity of the overidentifying restrictions; it is chi-squared with `L - k` degrees of freedom under the null.

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

References

Baltagi, B. H. (1981). Simultaneous equations with error components. *Journal of Econometrics*, 17, 189-200.

Balestra, P. and Varadharajan-Krishnakumar, J. (1987). Full information estimations of a system of simultaneous equations with error component structure. *Econometric Theory*, 3, 223-246.

Hausman, J. A. and Taylor, W. E. (1981). Panel data and unobservable individual effects. *Econometrica*, 49, 1377-1398.

See Also

`summary.paniv()`, `vcov.paniv()` and `predict.paniv()` for the fitted model, tidy-methods for `tidy()` and `glance()`.

Other panel models: `panfmm()`, `panfmmselect()`, `panglm()`, `pangmm()`, `panhurdle()`, `panqr()`, `panselect()`, `twopart()`

Examples

```
# Simulated panel of 500 patients over 4 waves. Private insurance is
# endogenous for doctor visits (a time-varying health shock drives both)
# and is instrumented by an employer offer of insurance.
set.seed(1)
N <- 500; TT <- 4
d <- data.frame(id = rep(seq_len(N), each = TT), wave = rep(seq_len(TT), N))
alpha <- rep(rnorm(N), each = TT)       # unobserved frailty
s0 <- rep(rnorm(N), each = TT)          # family background
need <- rnorm(N * TT)                   # time-varying health shock
d$school <- round(12 + 2 * s0 + alpha)  # correlated with the frailty
d$female <- rep(rbinom(N, 1, 0.5), each = TT)
d$offer <- rbinom(N * TT, 1, plogis(s0))
```

```
d$income <- rnorm(N * TT) + 0.5 * alpha
d$insured <- as.numeric(d$offer + need + rnorm(N * TT) > 0.5)
d$visits <- 1 + 0.6 * d$insured + 0.2 * d$income + alpha + need + rnorm(N * TT)

# the within estimator is biased upwards; FE-2SLS is not (true value 0.6)
coef(panglm(visits ~ insured + income, data = d, index = c("id", "wave"),
            model = "within"))
fe2sls <- paniv(visits ~ insured + income | offer + income, data = d,
                index = c("id", "wave"), model = "within", vcov = "cluster")
summary(fe2sls)

# Hausman-Taylor: the effect of schooling (time invariant, correlated
# with the frailty) on health costs, identified by the exogenous
# time-varying employer offer; income is endogenous (true values 0.4,
# 0.2, 0.3, 0.1)
d$cost <- 2 + 0.4 * d$offer + 0.2 * d$income + 0.3 * d$female +
  0.1 * d$school + alpha + rnorm(N * TT)
ht <- paniv(cost ~ offer + income + female + school | offer + female,
            data = d, index = c("id", "wave"), model = "hausmantaylor")
summary(ht)
```

---

panqr *Quantile regression for panel data with individual effects*

Description

Fixed-effects quantile regression for linear panel models: the two-step estimator of Canay (2011) and the method of moments quantile regression (MM-QR) of Machado and Santos Silva (2019), the estimator of the Stata command `xtqreg`. A pooled quantile regression that ignores the individual effects is available as a baseline. Several quantiles can be estimated in one call.

Usage

```
panqr(
  formula,
  data,
  index,
  tau = 0.5,
  model = c("canay", "mmqr", "pooling"),
  vcov = NULL,
  boot = 200,
  na.action = stats::na.omit,
  seed = NULL,
  R,
  cluster = NULL
)
```

Arguments

formula	a formula y ~ regressors. lag(x, k), lead(x, k) and diff(x) are taken within each individual on the grid of sorted distinct time values (see "Lags and differences" in panglm(); needs index = c(id, time)); the first waves of each individual get NA and leave the sample, and lag(x, 0:2) gives several lags
data	a data.frame in long format
index	length-1 or length-2 character vector giving the individual (and time) columns
tau	numeric vector of quantiles, each strictly between 0 and 1
model	"canay", "mmqr" or "pooling"
vcov	"analytic" (cluster-robust), "iid" (MM-QR only), "bootstrap" or "none"; the default is "bootstrap" for "canay" and "analytic" otherwise
boot	number of bootstrap replications for vcov = "bootstrap"
na.action	missing-data action passed to stats::model.frame()
seed	optional random seed for the bootstrap; the global random number state is restored afterwards. With seed = NULL (the default) the draws come from the global random number stream.
R	deprecated; use boot
cluster	optional name of a column of data to cluster the analytic covariance on (vcov = "analytic" only, including the MM-QR location and scale covariances); the default clusters by individual

Value

an object of class "panqr", a list with components coefficients (a matrix with one column per quantile), vcov (a list of covariance matrices, one per quantile, or NULL for vcov = "none"), vcov_type, tau, model, boot (the bootstrap replicates, an array boot x k x length(tau), for vcov = "bootstrap"), alpha (one per individual: for "canay" the normalized effects subtracted from the outcome, for "mmqr" the location effects including the constant), for "mmqr" also location, scale (lists with coefficients and vcov), q (the quantiles of the standardized errors) and delta (the individual scale effects including the constant, so that alpha_i(tau) = alpha_i + delta_i q(tau)), cluster and n_clusters (for the analytic covariances), dropped (time-invariant regressors dropped), n_singletons (individuals observed once, dropped for "canay" and "mmqr"), nobs, n_groups, call and formula. Methods: print, summary, coef, vcov, confint, nobs, plot, tidy and glance.

Estimators

- "canay": the individual effects are estimated from the within (mean) regression, alpha_i = ybar_i - xbar_i'b - c with c = ybar - xbar'b the overall constant (the normalization of Stata's xtreg, fe), and the quantile regression of the adjusted outcome y - alpha_i on the regressors (with an intercept) is fitted by quantreg::rq.fit() with method = "br". The intercept is the tau-quantile of c + error. The model assumes that the individual effects are pure location shifts, and the estimator is consistent when both N and T grow.

- "mmqr": the location-scale model y_it = alpha_i + x_it'b + (delta_i + x_it'g) U_it. The location parameters b come from the within regression of y on x; with R its residuals, the scale parameters g come from the within regression of 2 R (I(R >= 0) - mean(I(R >= 0))), an estimate of |R| that is robust to the first step; q(tau) is the sample tau-quantile of the standardized residuals R / sigma_it, with sigma_it the fitted scale including the individual effects. The quantile coefficients are b(tau) = b + g q(tau). The scale variables are the regressors, as in xtqreg. Only slopes are reported: the intercept is absorbed in the individual effects alpha_i(tau) = alpha_i + delta_i q(tau).
- "pooling": quantreg::rq.fit() of y on the regressors, ignoring the panel structure (except for the clustered covariance).

Time-invariant regressors are not identified with individual effects and are dropped (with a warning) for "canay" and "mmqr", as are individuals observed only once (with a message).

Covariance

vcov = "bootstrap" resamples whole individuals with replacement (a cluster bootstrap) boot times, repeats every step of the estimator and returns the covariance of the replicates. vcov = "analytic" gives:

- "mmqr": a cluster-robust (by individual, or by cluster) sandwich built from the influence function of b(tau) = b + g q(tau). The influence of observation it adds the three steps of the estimator: the location within regression, Q^{-1} x~_it R_it; the scale within regression, q(tau) Q^{-1} x~_it sigma_it V_it; and the quantile q(tau), g s_i w_it / sum(s), with Q = sum x~ x~', s_i the scale of individual i at the regressor means and w_it = (tau - I(U_it <= q(tau))) / f(q(ta
These are summed within clusters into psi_g, and V = G / (G - 1) * sum_g psi_g psi_g' for G clusters. The density of the standardized error at q(tau) is the difference quotient 2h / (q(tau + h) - q(tau - h)) (as Stata's qreg, vce(iid)) with the Hall and Sheather bandwidth h. The scale influence uses the centered V = 2 U (I(U >= 0) - mean(I(U >= 0))) - 1 of the paper (also used by the Stata command mmqreg). Unlike the iid formula, it allows the errors of an individual to be dependent (serial correlation, or an individual component in U) and the errors to be heteroskedastic.

vcov = "iid" (MM-QR only) gives the asymptotic covariance of Machado and Santos Silva (2019) under independent observations, with the matrix layout of xtqreg (version 1.5) and the same density estimate and centered V; xtqreg 1.5 omits the - 1 in V, which inflates its standard errors away from the median. This was the analytic covariance of "mmqr" before panglm 1.6.1. It replaces observation-level products by their sample means and ignores the dependence within individuals; it can be far off with short panels (see the package's validation suite).

The other models with vcov = "analytic":

- "canay": a cluster-robust (by individual) sandwich built from the influence function of the two-step estimator, which corrects the quantile regression score (tau - I(e < 0)) x for the estimation of the individual effects and of the within slopes. The conditional density is estimated with the Gaussian kernel and bandwidth of quantreg::summary.rq(se = "ker"). This is a large-N, large-T approximation; Canay (2011) recommends the bootstrap. It treats the individual effects as fixed, so its intercept SE omits the sampling variation of the mean individual effect that the cluster bootstrap includes; the slope SEs of the two agree.
- "pooling": the same kernel sandwich without the first-step terms, clustered by individual.

The "canay" and "pooling" cluster sandwiches carry no finite-sample factor; the "mmqr" one carries G / (G - 1). For "mmqr" the location and scale coefficients are returned with the cluster-robust within covariance of the rest of the package (G / (G - 1) * (n - 1) / (n - k), as plm::vcovHC(type = "sss")).

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

References

Canay, I. A. (2011). A simple approach to quantile regression for panel data. *The Econometrics Journal*, 14(3), 368-386. doi:10.1111/j.1368423X.2011.00349.x

Machado, J. A. F. and Santos Silva, J. M. C. (2019). Quantiles via moments. *Journal of Econometrics*, 213(1), 145-173. doi:10.1016/j.jeconom.2019.04.009

See Also

summary.panqr(), vcov.panqr() and plot.panqr() for the fitted model, tidy-methods for tidy() and glance().

Other panel models: panfmm(), panfmmselect(), panglm(), pangmm(), panhurdle(), paniv(), panselect(), twopart()

Examples

```
# Simulated panel of 300 patients over 6 years. Annual health costs have
# a patient effect and a heavy upper tail that widens with chronic
# conditions: a location-scale model in which the effect of chronic
# conditions grows across the cost distribution.
set.seed(1)
N <- 300; TT <- 6
d <- data.frame(id = rep(seq_len(N), each = TT), year = rep(seq_len(TT), N))
alpha <- rep(rnorm(N), each = TT)
d$chronic <- rpois(N * TT, 1 + 0.5 * (alpha > 0))
d$age <- rep(sample(40:70, N, TRUE), each = TT) + d$year
d$cost <- 2 + alpha + 0.5 * d$chronic + 0.02 * d$age +
  (1 + 0.3 * d$chronic) * rexp(N * TT)

if (requireNamespace("quantreg", quietly = TRUE)) {
  # median and upper tail of costs, MM-QR with analytic standard errors
  mm <- panqr(cost ~ chronic + age, data = d, index = c("id", "year"),
              tau = c(0.5, 0.9), model = "mmqr")
  summary(mm)

  # Canay two-step estimator with a small cluster bootstrap
  cn <- panqr(cost ~ chronic + age, data = d, index = c("id", "year"),
              tau = 0.9, model = "canay", boot = 50, seed = 1)
  coef(cn)
}
```

`panselect` *Sample-selection correction for linear panel models*

Description

Two-step selection correction of Wooldridge (1995) for linear panel models in which the outcome is observed only for a selected subsample (attrition in a cohort, survival, response to a wave), when selection may depend on the idiosyncratic error of the outcome equation.

Usage

```
panselect(
  selection,
  outcome,
  data,
  index,
  model = c("within", "pooling"),
  mundlak = TRUE,
  lambda = c("period", "common"),
  boot = 199L,
  seed = NULL,
  na.action = stats::na.omit
)
```

Arguments

`selection`	a two-sided formula `s ~ z1 + x1 + ...` for the selection equation; `s` is a 0/1 (or logical) indicator equal to one when the outcome is observed
`outcome`	a two-sided formula `y ~ x1 + ...` for the outcome equation, fitted on the observations with `s == 1`
`data`	a data.frame in long format, with one row per individual and period whether or not the outcome is observed
`index`	length-2 character vector giving the individual and period columns
`model`	`"within"` (fixed effects in the outcome equation) or `"pooling"` (pooled OLS, with Mundlak means when `mundlak = TRUE`)
`mundlak`	logical; add the individual means of the time-varying selection regressors to the probits (and to the pooled outcome equation)
`lambda`	`"period"` (inverse Mills ratio interacted with period dummies) or `"common"` (a single inverse Mills ratio term)
`boot`	number of panel bootstrap replicates; `0` skips the bootstrap and makes the cluster-robust covariance the default
`seed`	optional random seed for the bootstrap; the global random number state is restored afterwards
`na.action`	missing-data action passed to `stats::model.frame()` for each formula

Value

an object of class "panselect", a list with components coefficients, vcov (the default covariance), vcov_type ("boot" or "cluster"), vcov_boot, vcov_cluster, vcov_classical, boot (the replicate estimates, the number of requested and failed replicates), probit (a list with the probit coefficients of each period, NULL for a period without probit), lambda_terms (names of the correction terms), selection_test (the cluster-robust and bootstrap Wald tests), imr (the inverse Mills ratio of every observation of the selection sample, in the row order of the data), residuals, nobs (selected observations used), nobs_selection, n_groups, n_groups_selection, df.residual, periods, model, mundlak, lambda, call. Methods: print, summary, coef, vcov, confint, predict, nobs, tidy and glance.

Estimator

Step 1. For each period t a probit of the selection indicator s_it on the selection regressors z_it and, with mundlak = TRUE, the individual means zbar_i of the time-varying selection regressors (the Mundlak version of Chamberlain's device) is fitted on all observations of that period (by iteratively reweighted least squares, as stats::glm(), with the convergence tolerance 1e-12). The inverse Mills ratio is lambda_it = dnorm(xb_it) / pnorm(xb_it) with xb_it the probit index. The means are taken over all observations of the individual in the selection sample, selected or not. A period in which every observation (or none) is selected has no probit; its inverse Mills ratio is set to zero and it gets no correction term (for instance the first wave of a cohort, before any attrition).

Step 2. The outcome equation is fitted on the selected observations, adding the inverse Mills ratio interacted with period dummies, lambda_it * d_t (lambda = "period", Wooldridge's specification, one coefficient per period with a probit) or a single term lambda_it (lambda = "common"). With model = "within" the individual effects are removed by the within transformation on the selected sample; with model = "pooling" the outcome equation is estimated by pooled OLS and, when mundlak = TRUE, includes the same Mundlak means zbar_i to account for the correlation between the individual effect and the regressors. Period effects are not added automatically; include them in outcome (e.g. + factor(wave)) when needed.

The selection regressors should contain the outcome regressors plus at least one excluded variable (an instrument that affects selection but not the outcome); without one the correction is identified only by the nonlinearity of the inverse Mills ratio, and a warning is given.

Inference

The coefficient standard errors default to a panel (cluster) bootstrap: whole individuals are resampled with replacement and both steps, including the Mundlak means and the per-period probits, are re-estimated in each replicate. The covariance is the sample covariance of the boot replicate estimates; replicates that fail are dropped and counted.

The cluster-robust covariance vcov(fit, type = "cluster") treats the inverse Mills ratios as known regressors. It ignores the estimation of the first step and is therefore not valid for the coefficients when there is selection bias. It uses the factor G / (G - 1) * (n - 1) / (n - k), as in plm::vcovHC(type = "sss").

The test of selection bias is the joint Wald test that all inverse Mills ratio coefficients are zero. Under this null hypothesis the first step does not affect the asymptotic distribution of the second-step estimator, so the cluster-robust Wald statistic is valid (Wooldridge 2010, section 19.9); it is

chi-squared with as many degrees of freedom as correction terms. The same statistic computed with the bootstrap covariance is reported as well.

Validation

No other R package implements this panel estimator. Each step, and the recovery of the true coefficients in simulations with selection on the idiosyncratic error, are checked in the package tests and in the validation suite; see the validation article on the package website.

References

Wooldridge, J. M. (1995). Selection corrections for panel data models under conditional mean independence assumptions. *Journal of Econometrics*, 68, 115-132.

Wooldridge, J. M. (2010). *Econometric Analysis of Cross Section and Panel Data*, 2nd edition, section 19.9. MIT Press.

See Also

`summary.panselect()`, `vcov.panselect()` and `predict.panselect()` for the fitted model, tidy-methods for `tidy()` and `glance()`.

Other panel models: `panfmm()`, `panfmmselect()`, `panglm()`, `pangmm()`, `panhurdle()`, `paniv()`, `panqr()`, `twopart()`

Examples

```
# Simulated cohort of 300 patients followed over 4 waves. A quality of
# life score is observed only for patients who answer the questionnaire;
# non-response depends on the same health shock that lowers the score,
# and on a reminder letter that does not affect the score itself.
set.seed(1)
N <- 300; TT <- 4
d <- data.frame(id = rep(seq_len(N), each = TT), wave = rep(seq_len(TT), N))
alpha <- rep(rnorm(N), each = TT)
d$treat <- rnorm(N * TT) + 0.5 * alpha
d$reminder <- rnorm(N * TT)
v <- rnorm(N * TT)
u <- 0.7 * v + rnorm(N * TT, sd = 0.7)
d$respond <- as.numeric(0.3 + 0.5 * d$treat + d$reminder + 0.3 * alpha + v > 0)
d$qol <- ifelse(d$respond == 1, 1 + 0.5 * d$treat + alpha + u, NA)

fit <- panselect(respond ~ treat + reminder, qol ~ treat, data = d,
                 index = c("id", "wave"), model = "within", boot = 49, seed = 2)
summary(fit)
```

`partialeffects.panordered` *Average partial effects*

Description

Average partial effects (APE, also called average marginal effects) of each regressor on the expected outcome, with delta-method standard errors. For a continuous regressor the APE is the sample average of the derivative of the mean with respect to that regressor; for a binary (0/1 or logical) regressor and for each non-reference level of a factor it is the sample average of the discrete change in the mean, all other regressors held at their observed values. Derivatives and changes are taken through the model formula, so interactions, polynomial terms and factor codings are handled automatically.

Usage

```
## S3 method for class 'panordered'
partialeffects(object, variables = NULL, level = 0.95, ...)

partialeffects(object, ...)

## S3 method for class 'panglm'
partialeffects(object, variables = NULL, level = 0.95, ...)

## S3 method for class 'twopart'
partialeffects(
  object,
  variables = NULL,
  level = 0.95,
  part = c("total", "binary", "positive"),
  ...
)
```

Arguments

`object`	a `"panglm"` or `"twopart"` fit
`variables`	character vector of variable names (as they appear in the data); defaults to every regressor in the model formula(s)
`level`	confidence level for the Wald intervals
`...`	unused
`part`	for two-part models: `"total"` (the unconditional mean), `"binary"` (the probability of a positive outcome) or `"positive"` (the mean conditional on a positive outcome)

Details

The mean used is the one returned by `predict(type = "response")`: the GLM mean for pooled models; the mean including the estimated individual (and time) effects for fixed-effects models; and the population-averaged mean (integrating over the random intercept, and the random slopes when `random` has slope variables) for random-effects models, which is the average structural function of Wooldridge (2005). For correlated random effects fits (`cre = TRUE`) the individual means are held fixed, so the APE of a time-varying regressor is the within-individual effect. For two-part models the mean is `P(y > 0) * E[y | y > 0]` (or one of its parts, see `part`).

Standard errors use the covariance stored on the fit (so a fit made with `vcov = "cluster"` gives cluster-robust APE standard errors) and treat variance components, dispersion and fixed-effect intercepts as fixed. Exact conditional logit fits are not supported, because their individual intercepts are conditioned out and the mean is not identified.

Value

a data frame of class `"panglm_ape"` with columns `term`, `contrast`, `estimate`, `std.error`, `statistic`, `p.value`, `conf.low` and `conf.high`, one row per continuous or binary regressor and one row per non-reference factor level.

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

References

Wooldridge, J. M. (2005). Unobserved heterogeneity and estimation of average partial effects. In *Identification and Inference for Econometric Models*, Cambridge University Press.

See Also

`plot.panglm_ape()` for a dot-and-whisker chart, and `effectcurve()` for the effect of a regressor across its range on the link, response or slope scale.

Other panglm methods: `coefplot()`, `confint.panglm()`, `effectcurve()`, `fitted.panglm()`, `fixef()`, `glance.panglm()`, `logLik.panglm()`, `model.matrix.panglm()`, `plot.panglm()`, `plot.panglm_ape()`, `plot.panglm_effectcurve()`, `predict.panglm()`, `predict.panordered()`, `residuals.panglm()`, `summary.panglm()`, `tidy.panglm()`, `vcov.panglm()`

Examples

```
data(copd)
fit <- panglm(hospitalized ~ treatment + age + crp, data = copd,
              index = c("id", "visit"), model = "random",
              family = "probit", cre = TRUE)
ape <- partialeffects(fit)
ape
plot(ape)
```

plot.panfmm *Plot a latent class (finite mixture) model*

Description

`which = "coefficients"` compares the coefficients of the latent classes in one dot-and-whisker panel (the classes dodged within each term, as in `coefplot()`); the coefficients of the class-membership (concomitant) model are not shown. `which = "posterior"` is a classification-quality diagnostic: the histogram of each individual's largest posterior class probability. Mass close to one means well-separated classes; the subtitle reports its mean and the relative entropy `1 - sum(-p log p) / (N log k)` (one for a perfect classification). `which = "correlation"` draws the heatmap of the correlation matrix of all the estimates, `cov2cor(vcov(x))` (see `plot.panglm()`).

Usage

```
## S3 method for class 'panfmm'
plot(
  x,
  which = c("coefficients", "posterior", "correlation"),
  parm = NULL,
  level = 0.95,
  cells = FALSE,
  main = NULL,
  xlab = NULL,
  ...
)
```

Arguments

`x`	a fitted `"panfmm"` object
`which`	`"coefficients"` (default), `"posterior"` or `"correlation"`
`parm`	names or positions of the terms to plot (without the class prefix); by default every term except `"(Intercept)"`. For `"correlation"`, names as in `coef(x)`.
`level`	confidence level of the intervals
`cells`	for `"correlation"`, print the correlation in each cell
`main`	optional plot title
`xlab`	optional horizontal-axis label
`...`	additional arguments passed to `graphics::plot.default()`

Value

`x`, invisibly

See Also

Other panfmm methods: `predict.panfmm()`, `summary.panfmm()`, `vcov.panfmm()`

Examples

```
data(copd)
fit <- panfmm(exacerbations ~ crp + treatment, data = copd,
              index = c("id", "visit"), k = 2, family = "poisson",
              nstart = 2, seed = 1)
plot(fit)
plot(fit, which = "posterior")
plot(fit, which = "correlation", cells = TRUE)
```

`plot.panglm` *Plot a fitted panglm model*

Description

Base-graphics coefficient and diagnostic plots of a `panglm()` fit.

Usage

```
## S3 method for class 'panglm'
plot(
  x,
 which = c("coefficients", "residuals", "fitted", "qq", "calibration", "rootogram",
    "density", "effects", "correlation"),
  level = 0.95,
  parm = NULL,
  type = c("pearson", "response"),
  bins = 10,
  max = NULL,
  bw = "nrd0",
  cells = FALSE,
  main = NULL,
  xlab = NULL,
  ...
)
```

Arguments

`x`	a fitted `"panglm"` object
`which`	one of `"coefficients"`, `"residuals"`, `"fitted"`, `"qq"`, `"calibration"`, `"rootogram"`, `"density"`, `"effects"` or `"correlation"`
`level`	confidence level for coefficient and calibration intervals
`parm`	names or positions of the terms of the coefficient and correlation plots; by default every term except `"(Intercept)"`
`type`	residuals of the `"residuals"` view, `"pearson"` (default) or `"response"`
`bins`	number of quantile bins of the calibration plot

max	largest count of the rootogram; by default the 99th percentile of the outcome
bw	bandwidth of the kernel densities of the "density" and "effects" views, passed to stats::density()
cells	for "correlation", print the correlation in each cell
main	optional plot title
xlab	optional horizontal-axis label
...	additional arguments passed to graphics::plot.default()

Details

- "coefficients": estimates with Wald confidence intervals (the dot-and-whisker layout shared by every panglm coefficient plot and by coefplot()); the left margin is sized to the longest term label.
- "residuals": residuals against fitted values with a stats::lowess() smooth. Pearson residuals by default (see residuals.panglm() for the variance functions); for the ordered and Tobit families, where they are not defined, response residuals are drawn with a message. For log-link families (Poisson, negative binomial, log-link gamma) the fitted values are on a logarithmic axis, which spreads the many small means of count data.
- "fitted": observed against fitted values with the 45-degree line.
- "qq": normal Q-Q plot of the Pearson residuals with a reference line through the quartiles; for the Gaussian and gamma families the residuals are divided by the square root of the dispersion (standardized Pearson residuals). For discrete outcomes the plot is lumpy by construction and mainly shows skewness and outliers.
- "calibration" (binomial families, logit or probit): observed proportion against mean predicted probability in bins quantile bins of the predicted probability, with exact (Clopper-Pearson) binomial confidence intervals and the 45-degree line.
- "rootogram" (Poisson and negative binomial families): hanging rootogram. The curve is sqrt(E_k), the expected frequency of count k summed over observations, E_k = sum_i P(y_i = k); the bars of height sqrt(O_k) (observed frequency) hang from it, so that bars ending above zero mark counts predicted too often and bars ending below zero counts predicted too rarely. P(y_i = k) is the Poisson or NB2 probability at the fitted mean for pooled and fixed-effects fits (the fixed-effects means include the estimated individual effects). For random-intercept fits it is the marginal probability of a single observation implied by the mixing distribution, which is exactly the distribution whose frequencies the rootogram compares: negative binomial with size d for the Poisson-gamma model, beta-negative binomial for the Hausman-Hall-Griliches negative binomial model. Random-slope fits have no closed-form marginal and are not supported.
- "density" (Gaussian, gamma, beta and Tobit families): the kernel density of the observed outcome (stats::density(), bandwidth bw, grey) against the model-implied marginal density (orange), the average over observations of the density of a single outcome at the fitted parameters, f(t) = mean_i f(t | x_i). Gaussian: dnorm(t, eta_i, sigma) with sigma^2 the residual variance; gamma: dgamma(t, shape = 1 / phi, rate = 1 / (phi mu_i)) with phi the Pearson dispersion of the fit (the stats::glm() convention, also used in the paper); beta: dbeta(t, mu_i phi, (1 - mu_i) phi) with phi the precision; Tobit: the normal density on (left, right) plus point masses P(y = left) = mean_i pnorm((left - eta_i) / sigma)

(and likewise at `right`), drawn as bars on the right-hand axis, observed share (grey) beside predicted probability (orange). For the Tobit family the observed kernel density uses the uncensored outcomes and is scaled by their share, so that both curves integrate with the masses to one. Random-effects fits use the marginal density, integrated over the random effect: in closed form for the Gaussian and Tobit families (variance `sigma_e^2 + sigma_u^2`, or `sigma_e^2 + z' Sigma z` with random slopes) and by Gauss-Hermite quadrature (at least 21 nodes) for the gamma and beta families (the gamma random-intercept fit uses its maximum likelihood shape). Fixed-effects fits condition on the estimated individual effects (the fitted means include them), so the implied density is the in-sample fit with `n + G` parameters; between fits are refused. A positive outcome whose 99th percentile exceeds 20 times its median is drawn on a logarithmic axis (`log = "x"` forces it, `log = ""` turns it off): both curves are then densities of `log(y)`, the implied one `exp(s) f(exp(s))`, and point masses at zero sit left of the axis. Count families are refused in favour of `which = "rootogram"`.

- `"effects"`: distribution of the estimated individual effects. Fixed-effects fits: histogram and kernel density (blue) of `fixef()`. Random-effects fits: histogram and kernel density of the posterior means `ranef()`, with the assumed distribution of the random effect (dashed orange): `N(0, sigma_u^2)` for the normal random intercepts, the log-gamma density of `log(nu)` for the Poisson-gamma model and the logistic-beta density of `log(delta)` for the Hausman-Hall-Griliches negative binomial model. Empirical Bayes predictions are shrunk towards their mean, so the histogram is narrower than the assumed density; the subtitle compares the standard deviation of the predictions with the model one. A histogram much narrower than the curve signals little information per individual, not misspecification; marked skewness or several modes do suggest a non-normal effect distribution. With random slopes there is one panel per effect, then the intercept plotted against each slope with the 95% contour of the estimated normal distribution.
- `"correlation"`: heatmap of the correlation matrix of the estimates, `cov2cor(vcov(x))`, with a diverging palette centred at zero (blue negative, red positive, on the fixed range `[-1, 1]`) and a colour key; `cells = TRUE` prints the values. Strong correlations flag nearly collinear terms whose separate estimates are imprecise. `parm` selects the terms (by default all except `"(Intercept)"`).

Value

`x`, invisibly

See Also

`fixef()` and `ranef()` for the individual effects.

Other panglm methods: `coefplot()`, `confint.panglm()`, `effectcurve()`, `fitted.panglm()`, `fixef()`, `glance.panglm()`, `logLik.panglm()`, `model.matrix.panglm()`, `partialeffects.panordered()`, `plot.panglm_ape()`, `plot.panglm_effectcurve()`, `predict.panglm()`, `predict.panordered()`, `residuals.panglm()`, `summary.panglm()`, `tidy.panglm()`, `vcov.panglm()`

Examples

```
data(copd)
fit <- panglm(fev1 ~ crp + age + treatment, data = copd,
              index = c("id", "visit"), model = "random",
              family = "gaussian")
```

```
plot(fit)
plot(fit, parm = c("crp", "treatment"))
plot(fit, which = "residuals")
plot(fit, which = "qq")

counts <- panglm(exacerbations ~ crp + treatment, data = copd,
                 index = c("id", "visit"), model = "pooling",
                 family = "negbin")
plot(counts, which = "rootogram")

any_exacerbation <- panglm(I(exacerbations > 0) ~ crp + treatment,
                           data = copd, index = c("id", "visit"),
                           model = "pooling", family = "binomial")
plot(any_exacerbation, which = "calibration", bins = 5)

plot(fit, which = "density")
censored <- panglm(pmax(fev1, 1.5) ~ crp + treatment, data = copd,
                   index = c("id", "visit"), model = "pooling",
                   family = tobitfamily(left = 1.5))
plot(censored, which = "density")

plot(fit, which = "effects")
hosp <- panglm(hospitalized ~ crp + treatment, data = copd,
               index = c("id", "visit"), model = "random",
               family = "binomial")
plot(hosp, which = "effects")

plot(fit, which = "correlation", cells = TRUE)
```

`plot.panglm_ape` *Plot average partial effects*

Description

Dot-and-whisker plot of the average partial effects returned by `partialeffects()`: one row per continuous or binary regressor and per non-reference factor level, the estimate as a dot, its Wald confidence interval as a horizontal segment and a dotted reference line at zero. The left margin is widened to fit the row labels and restored on exit. For two-part models the part of the mean (`"total"`, `"binary"` or `"positive"`) is shown in the axis label; plot one object per part to compare them.

Usage

```
## S3 method for class 'panglm_ape'
plot(x, labels = NULL, xlab = NULL, main = NULL, col = "black", pch = 19, ...)
```

Arguments

`x`	a `"panglm_ape"` object from `partialeffects()`

labels	row labels; by default the variable name for continuous regressors and "variable: contrast" for discrete ones
xlab	horizontal-axis label
main	plot title
col	colour of the dots and whiskers
pch	plotting symbol of the estimates
...	passed to graphics::plot.default()

Value

x, invisibly

See Also

Other panglm methods: coefplot(), confint.panglm(), effectcurve(), fitted.panglm(), fixef(), glance.panglm(), logLik.panglm(), model.matrix.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_effectcurve(), predict.panglm(), predict.panordered(), residuals.panglm(), summary.panglm(), tidy.panglm(), vcov.panglm()

Examples

```
data(copd)
fit <- panglm(hospitalized ~ treatment + age + crp + factor(smoker),
              data = copd, index = c("id", "visit"), family = "probit")
plot(partialeffects(fit))
```

---

plot.panglm_attgt *Plot an event study from group-time effects*

Description

Plots the dynamic aggregation of attgt() (ATT(e) by event time), the cohort or calendar aggregation, or the group-time effects ATT(g, t) themselves. The default style ("classic") draws filled points with capped pointwise confidence intervals, a solid line at zero and, for event time, a dashed line at treatment onset labelled "Intervention". When the fit has uniform confidence bands (cband = TRUE), they are drawn as light grey bars behind the pointwise intervals.

Usage

```
## S3 method for class 'panglm_attgt'
plot(
  x,
  type = c("dynamic", "group", "calendar", "gt", "heatmap"),
  min_e = -Inf,
  max_e = Inf,
  ncol = NULL,
```

```
  xlab = NULL,
  ylab = "Estimated ATT",
  main = NULL,
  ylim = NULL,
  legend = FALSE,
  style = c("classic", "points", "line"),
  ref = -1,
  onset = NULL,
  onset_label = NA,
  caps = NULL,
  cells = FALSE,
  digits = 2,
  ...
)

## S3 method for class 'summary.panglm_attgt'
plot(
  x,
  type = NULL,
  min_e = -Inf,
  max_e = Inf,
  xlab = NULL,
  ylab = "Estimated ATT",
  main = NULL,
  ylim = NULL,
  legend = FALSE,
  style = c("classic", "points", "line"),
  ref = -1,
  onset = NULL,
  onset_label = NA,
  caps = NULL,
  ...
)
```

Arguments

x	a "panglm_attgt" fit from attgt(), or its summary
type	what to plot (for a fit): "dynamic" (event study), "group", "calendar", "gt" (the group-time effects, one panel per cohort) or "heatmap" (the group-time effects as a cohort by period heat map)
min_e, max_e	event window (event time for "dynamic", calendar time for "gt" and "heatmap")
ncol	number of panel columns for type = "gt" (default: about the square root of the number of cohorts)
xlab, ylab, main	axis labels and title
ylim	y-axis limits; for the heat map, the limits of the colour scale (default: symmetric around zero)
legend	logical: add a compact legend of the symbols (the heat map always has its colour bar)

style	"classic" (default), "points" or "line" (see Styles); for the event-time, cohort and calendar plots
ref	event time of the reference period, marked when its estimate is exactly zero (see Styles); NULL marks none
onset	position of the dashed treatment-onset line on the event time axis: NULL gives 0 for style = "classic" and -0.5 (between the last pre-treatment and the first treated period) otherwise
onset_label	text written above the onset line; the default (NA) is "Intervention" for style = "classic" and none otherwise, NULL suppresses it
caps	logical: draw caps on the confidence-interval whiskers; the default (NULL) draws them for style = "classic" only
cells	logical: print the estimates in the cells of the heat map
digits	number of decimals of the printed estimates
...	passed to summary.panglm_attgt() (balance_e, level) and then to graphics::plot()

Details

With type = "gt", one small panel per treatment cohort shows ATT(g, t) against calendar time, with the cohort size in the panel title, a dashed line at adoption and a common y axis (as did::ggdid()). The uniform bands of a cband = TRUE fit are simultaneous over all ATT(g, t).

With type = "heatmap", the ATT(g, t) are shown as a cohort (rows, earliest adopters at the top) by calendar period (columns) heat map, on a blue-white-red scale centred at zero with a colour bar. Pre-treatment cells are hatched, a thick black edge marks the adoption of each cohort and grey cells have no estimate. With cells = TRUE the estimates are printed in the cells, in bold with a star when the pointwise confidence interval excludes zero.

Value

the plotted summary, invisibly; for type = "gt" and "heatmap", the plotted rows of the estimates of the fit (with a logical column pre for the heat map)

Styles

style = "classic" (the default) is the textbook dynamic difference-in-differences figure: filled points with capped whiskers, the reference period as an open circle at zero without a whisker, a solid line at zero and a dashed vertical line at event time 0 labelled "Intervention" above the plot. style = "points" draws hollow points before treatment and filled points after, an open diamond at the reference period, uncapped intervals, a dotted zero line and the onset line at -0.5, between the last pre-treatment and the first treated period. style = "line" joins the estimates by a line and shades the pointwise confidence intervals (and, lighter, the uniform bands), with the onset line at -0.5. The cohort panels of type = "gt" always use the "points" look.

The reference period is the event time ref when its estimate is exactly zero, that is when it is omitted or normalized. With base = "universal" in attgt() the effect at event time -1 is normalized to zero and drawn as the reference. With the default base = "varying" no period is omitted (each pre-treatment estimate is a short difference), so no reference period is drawn.

See Also

Other attgt methods: summary.panglm_attgt()

Examples

```
set.seed(2)
d <- expand.grid(id = 1:150, t = 1:8)
d$first <- rep(c(4, 6, NA), each = 50)[d$id]
d$y <- rnorm(150)[d$id] + 0.2 * d$t +
  ifelse(!is.na(d$first) & d$t >= d$first, 1, 0) + rnorm(nrow(d))
fit <- attgt(y ~ 1, d, index = c("id", "t"), treat = "first")
plot(fit)
plot(fit, min_e = -2, max_e = 2, legend = TRUE)
plot(fit, type = "gt")
plot(fit, type = "group")
plot(fit, type = "heatmap", cells = TRUE)
plot(fit, style = "points", legend = TRUE)
plot(fit, style = "line")
# with a universal base period, event time -1 is the reference (zero)
fu <- attgt(y ~ 1, d, index = c("id", "t"), treat = "first", base = "universal")
plot(fu, min_e = -3, max_e = 3)
```

---

plot.panglm_effectcurve

Plot an effect curve

Description

Draws the estimate as a line over a shaded pointwise confidence band, a rug of the percentiles of the observed regressor, and dotted reference lines: at zero and at the reference value for link-scale curves, at zero for slope curves. Curves by group share one panel, one colour each, with a legend. Curves of a discrete regressor are drawn as points with confidence whiskers.

Usage

```
## S3 method for class 'panglm_effectcurve'
plot(
  x,
  xlab = attr(x, "variable"),
  ylab = NULL,
  band_col = "grey85",
  rug = TRUE,
  col = NULL,
  legend = "auto",
  ...
)
```

Arguments

x	a "panglm_effectcurve" object from effectcurve()
xlab, ylab	axis labels; ylab defaults to a label for the scale
band_col	colour of the confidence band (single curve)
rug	if TRUE, mark the percentiles of the observed regressor on the horizontal axis
col	colour(s) of the curve(s); black for a single curve and the Okabe-Ito palette for curves by group by default
legend	position of the legend for curves by group: "auto" (the corner of the plot least covered by the curves and bands), a keyword accepted by graphics::legend(), or FALSE for none
...	passed to graphics::plot()

Value

x, invisibly

See Also

Other panglm methods: coefplot(), confint.panglm(), effectcurve(), fitted.panglm(), fixef(), glance.panglm(), logLik.panglm(), model.matrix.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_ape(), predict.panglm(), predict.panordered(), residuals.panglm(), summary.panglm(), tidy.panglm(), vcov.panglm()

Examples

```
data(copd)
fit <- panglm(fev1 ~ splines::ns(crp, 3), data = copd,
              index = c("id", "visit"), model = "within")
plot(effectcurve(fit, "crp"), band_col = "lightblue")
```

---

plot.panglm_etwfe *Plot ETWFE treatment effects*

Description

Plots the event-study aggregation of an etwfe() fit (or the cohort or calendar aggregation) with pointwise confidence intervals, in the styles of plot.panglm_attgt() (see its Styles section). The reference period (event time -1 with control = "never") is drawn as an open symbol at zero.

Usage

```
## S3 method for class 'panglm_etwfe'
plot(
  x,
  type = c("event", "group", "calendar", "heatmap"),
  min_e = -Inf,
  max_e = Inf,
  xlab = NULL,
  ylab = "Estimated ATT",
  main = NULL,
  ylim = NULL,
  legend = FALSE,
  style = c("classic", "points", "line"),
  ref = -1,
  onset = NULL,
  onset_label = NA,
  caps = NULL,
  cells = FALSE,
  digits = 2,
  ...
)

## S3 method for class 'summary.panglm_etwfe'
plot(
  x,
  type = NULL,
  min_e = -Inf,
  max_e = Inf,
  xlab = NULL,
  ylab = "Estimated ATT",
  main = NULL,
  ylim = NULL,
  legend = FALSE,
  style = c("classic", "points", "line"),
  ref = -1,
  onset = NULL,
  onset_label = NA,
  caps = NULL,
  ...
)
```

Arguments

x	a "panglm_etwfe" fit from etwfe(), or its summary
type	aggregation to plot (for a fit): "event", "group", "calendar" or "heatmap" (cohort by period heat map of ATT(g, t))
min_e, max_e	event window to plot (event time for the event-time plot, calendar time for the heat map)

xlab, ylab, main	axis labels and title
ylim	y-axis limits
legend	logical: add a compact legend of the symbols
style	"classic" (default), "points" or "line" (see Styles); for the event-time, cohort and calendar plots
ref	event time of the reference period, marked when its estimate is exactly zero: event time -1 with control = "never"; with control = "notyet" there are no pre-treatment effects
onset	position of the dashed treatment-onset line on the event time axis: NULL gives 0 for style = "classic" and -0.5 (between the last pre-treatment and the first treated period) otherwise
onset_label	text written above the onset line; the default (NA) is "Intervention" for style = "classic" and none otherwise, NULL suppresses it
caps	logical: draw caps on the confidence-interval whiskers; the default (NULL) draws them for style = "classic" only
cells	logical: print the estimates in the cells of the heat map
digits	number of decimals of the printed estimates
...	passed to graphics::plot()

Details

type = "heatmap" draws the cohort-by-period effects ATT(g, t) on the response scale as a heat map, as for plot.panglm_attgt(); with control = "never" the reference cell g - 1 of each cohort is zero.

Value

the plotted summary, invisibly; for the heat map, the table of ATT(g, t) with a logical column pre (pre-treatment cell)

See Also

Other etwfe methods: summary.panglm_etwfe()

Examples

```
set.seed(2)
d <- expand.grid(id = 1:150, t = 1:8)
d$first <- rep(c(4, 6, NA), each = 50)[d$id]
d$y <- rnorm(150)[d$id] + 0.2 * d$t +
  ifelse(!is.na(d$first) & d$t >= d$first, 1, 0) + rnorm(nrow(d))
fit <- etwfe(y ~ 1, d, index = c("id", "t"), treat = "first", control = "never")
plot(fit)
plot(fit, min_e = -3, max_e = 3, legend = TRUE)
plot(fit, style = "points", legend = TRUE)
plot(fit, style = "line")
plot(fit, type = "heatmap", cells = TRUE)
```

`plot.panglm_eventstudy` *Plot event-study coefficients*

Description

Plots the event-time coefficients of `eventstudy()` with pointwise confidence intervals, in the styles of `plot.panglm_attgt()` (see its Styles section). The omitted reference period (zero by construction) is drawn as an open symbol at zero, without an interval. With `bin = TRUE` the end points pool all earlier (later) event times.

Usage

```
## S3 method for class 'panglm_eventstudy'
plot(
  x,
  min_e = -Inf,
  max_e = Inf,
  xlab = "Event time",
  ylab = "Estimate",
  main = NULL,
  ylim = NULL,
  legend = FALSE,
  style = c("classic", "points", "line"),
  ref = x$ref,
  onset = NULL,
  onset_label = NA,
  caps = NULL,
  ...
)
```

Arguments

`x`	a `"panglm_eventstudy"` object from `eventstudy()`
`min_e, max_e`	event window to plot
`xlab, ylab, main`	axis labels and title
`ylim`	y-axis limits
`legend`	logical: add a compact legend of the symbols
`style`	`"classic"` (default), `"points"` or `"line"` (see Styles); for the event-time, cohort and calendar plots
`ref`	event time of the reference period, marked when its estimate is exactly zero; the default is the period omitted by `eventstudy()` (`x$ref`)
`onset`	position of the dashed treatment-onset line on the event time axis: `NULL` gives `0` for `style = "classic"` and `-0.5` (between the last pre-treatment and the first treated period) otherwise

onset_label	text written above the onset line; the default (NA) is "Intervention" for style = "classic" and none otherwise, NULL suppresses it
caps	logical: draw caps on the confidence-interval whiskers; the default (NULL) draws them for style = "classic" only
...	passed to graphics::plot()

Value

x, invisibly

See Also

Other eventstudy methods: eventstudy-methods

Examples

```
set.seed(1)
d <- expand.grid(id = 1:100, year = 2015:2020)
d$first <- ifelse(d$id <= 50, 2018, NA)
e <- d$year - d$first
d$visits <- 3 + rnorm(100)[d$id] + ifelse(!is.na(e) & e >= 0, 1, 0) +
  rnorm(nrow(d))
es <- eventstudy(visits ~ 1, data = d, index = c("id", "year"),
                 treat = "first", leads = 3, lags = 2)
plot(es, main = "Effect of the policy by event time")
plot(es, min_e = -2, legend = TRUE)
plot(es, style = "points")
plot(es, style = "line")
```

---

plot.panglm_hurdle *Plot a fitted panglm hurdle model*

Description

Displays coefficient estimates and Wald confidence intervals for the binary hurdle component, the positive-count component, or both, in the dot-and-whisker layout of plot.panglm(). which = "correlation" draws the heatmap of the correlation matrix of all the estimates (see plot.panglm()); the two parts are estimated separately, so the between-part block is zero.

Usage

```
## S3 method for class 'panglm_hurdle'
plot(
  x,
  which = c("both", "zero", "count", "correlation"),
  level = 0.95,
  parm = NULL,
  cells = FALSE,
  ...
)
```

Arguments

x	a fitted "panglm_hurdle" object
which	one of "both", "zero", "count" or "correlation"
level	confidence level for coefficient intervals
parm	names or positions of the terms to plot; by default every term except "(Intercept)". For "correlation", names as in coef(x) (with the zero: and count: prefixes).
cells	for "correlation", print the correlation in each cell
...	additional arguments passed to graphics::plot.default()

Value

x, invisibly

See Also

Other panhurdle methods: hurdle-methods

Examples

```
data(copd)
fit <- panhurdle(exacerbations ~ crp, data = copd,
                    index = c("id", "visit"))
plot(fit)
plot(fit, which = "correlation")
```

---

plot.pangmm *Plot the coefficients of a GMM, IV or selection model*

Description

Coefficient estimates with Wald confidence intervals, in the dot-and-whisker layout of plot.panglm(). For panselect() fits, which = "selection" shows the coefficients of the per-period probits of the selection equation (one marker per period, dodged within each term); they are drawn without intervals because the first-step probits are stored without standard errors. which = "both" shows the selection and outcome equations side by side.

Usage

```
## S3 method for class 'pangmm'
plot(x, parm = NULL, level = 0.95, main = NULL, xlab = NULL, ...)

## S3 method for class 'paniv'
plot(x, parm = NULL, level = 0.95, main = NULL, xlab = NULL, ...)

## S3 method for class 'panselect'
```

```
plot(
  x,
  which = c("both", "outcome", "selection"),
  parm = NULL,
  level = 0.95,
  xlab = NULL,
  ...
)
```

Arguments

x	a fitted "pangmm", "paniv" or "panselect" object
parm	names or positions of the terms to plot; by default every term except "(Intercept)"
level	confidence level of the intervals
main	optional plot title
xlab	optional horizontal-axis label
...	additional arguments passed to graphics::plot.default()
which	for panselect fits, "both" (default), "outcome" or "selection"

Value

x, invisibly

See Also

Other pangmm methods: predict.pangmm(), summary.pangmm(), vcov.pangmm()

Other paniv methods: predict.paniv(), summary.paniv(), vcov.paniv()

Other panselect methods: predict.panselect(), summary.panselect(), vcov.panselect()

Examples

```
data(copd)
gm <- pangmm(fev1 ~ lag(fev1, 1) + crp | lag(fev1, 2:99), data = copd,
             index = c("id", "visit"))
plot(gm, parm = c("lag(fev1, 1)", "crp"))

iv <- paniv(fev1 ~ crp + treatment | age + treatment, data = copd,
            index = c("id", "visit"), model = "pooling")
plot(iv)
set.seed(1)
d <- data.frame(id = rep(1:150, each = 4), wave = rep(1:4, 150))
d$treat <- rnorm(600)
d$reminder <- rnorm(600)
v <- rnorm(600)
d$respond <- as.numeric(0.3 + 0.5 * d$treat + d$reminder + v > 0)
d$qol <- ifelse(d$respond == 1, 1 + 0.5 * d$treat + 0.7 * v + rnorm(600), NA)
sel <- panselect(respond ~ treat + reminder, qol ~ treat, data = d,
                 index = c("id", "wave"), model = "pooling", boot = 0)
plot(sel)
```

`plot.panqr` *Plot a panel quantile regression fit*

Description

Base-graphics plot of the coefficients of a `panqr()` fit across the estimated quantiles, one panel per term: the estimates joined by a line, a shaded pointwise Wald confidence band (from the same variances as `confint()`), and a dashed reference line at zero. With a single quantile each panel shows the estimate and its interval. Without a covariance (`vcov = "none"`) only the estimates are drawn.

Usage

```
## S3 method for class 'panqr'
plot(
  x,
  parm = NULL,
  level = 0.95,
  ncol = NULL,
  col = "black",
  band = "grey85",
  main = NULL,
  xlab = expression(tau),
  ylab = "Estimate",
  ...
)
```

Arguments

`x`	a `"panqr"` fit
`parm`	names or positions of the terms to plot; all by default
`level`	confidence level of the bands
`ncol`	number of panel columns; by default up to three
`col`	colour of the estimates
`band`	colour of the confidence bands
`main`	optional titles, recycled over the terms; the term names by default
`xlab`	horizontal-axis label
`ylab`	vertical-axis label
`...`	additional arguments passed to `graphics::plot.default()`

Value

`x`, invisibly

See Also

Other panqr methods: `summary.panqr()`, `vcov.panqr()`

Examples

```
set.seed(2)
d <- data.frame(id = rep(1:100, each = 5), t = rep(1:5, 100))
d$x <- rnorm(500) + rep(rnorm(100), each = 5)
d$y <- d$x + rep(rnorm(100), each = 5) + (1 + 0.2 * abs(d$x)) * rnorm(500)
fit <- panqr(y ~ x, data = d, index = c("id", "t"),
             tau = c(0.1, 0.25, 0.5, 0.75, 0.9), model = "mmqr")
plot(fit)
```

---

plot.twopart *Plot a two-part model*

Description

Coefficient estimates with Wald confidence intervals of the binary part (probability of a positive outcome), of the positive part, or of both side by side, in the dot-and-whisker layout of `plot.panglm()`. The intervals use `vcov()`, so that they are the joint-likelihood intervals of correlated two-part fits.

Usage

```
## S3 method for class 'twopart'
plot(
  x,
  which = c("both", "binary", "positive", "density", "correlation"),
  parm = NULL,
  level = 0.95,
  bw = "nrd0",
  cells = FALSE,
  ...
)
```

Arguments

`x`	a fitted `"twopart"` object
`which`	`"both"` (default), `"binary"`, `"positive"`, `"density"` or `"correlation"`
`parm`	names or positions of the terms to plot; by default every term except `"(Intercept)"`. With `which = "both"`, names absent from one part are ignored in that part. For `"correlation"`, names as in `coef(x)` (with the `binary:` and `positive:` prefixes).
`level`	confidence level of the intervals
`bw`	bandwidth of the observed kernel density of the `"density"` view, passed to `stats::density()`
`cells`	for `"correlation"`, print the correlation in each cell
`...`	additional arguments passed to `graphics::plot.default()`

Details

`which = "density"` compares the observed outcome with the distribution implied by the model, a point mass at zero plus a density on the positive values. The bars at zero are the observed share of zeros (grey) and the predicted probability `mean_i (1 - p_i)` (orange, on the right-hand axis); the curves are the kernel density of the positive outcomes (`stats::density()`, bandwidth `bw`) scaled by their share (grey) and the implied density `mean_i p_i g_i(t)` (orange), where `p_i` is the probability of a positive outcome and `g_i` the gamma density (shape `1 / phi`, `phi` the Pearson dispersion of the positive part) or the lognormal density (`sigma^2` the residual variance of `log y`). Random-effects fits integrate over the random intercepts: independent intercepts by Gauss-Hermite quadrature (closed form for the lognormal part), correlated intercepts over their joint normal distribution. Skewed costs are drawn on a logarithmic axis (densities of `log(y)`, the bars at zero left of the axis) unless `log = ""` is passed.

`which = "correlation"` draws the heatmap of the correlation matrix of the estimates of both parts, `cov2cor(vcov(x))` (see `plot.panglm()`).

Value

`x`, invisibly

See Also

Other twopart methods: `confint.twopart()`, `predict.twopart()`

Examples

```
data(copd)
set.seed(1)
copd$cost <- ifelse(copd$exacerbations > 0,
                    exp(6 + 0.1 * copd$crp + rnorm(nrow(copd), 0, 0.5)), 0)
tp <- twopart(cost ~ treatment + age + crp, data = copd,
              index = c("id", "visit"), positive = "gamma")
plot(tp)
plot(tp, which = "positive", parm = "crp")
plot(tp, which = "density")
plot(tp, which = "correlation")
```

---

predict.panfmm *Predictions from a latent class panel model*

Description

Predictions from a latent class panel model

Usage

```
## S3 method for class 'panfmm'
predict(
  object,
  newdata = NULL,
  type = c("response", "class", "prior", "posterior", "components"),
  weights = c("prior", "posterior"),
  ...
)
```

Arguments

object	a "panfmm" fit
newdata	optional data.frame; by default the estimation sample (in its original row order). Offsets are included in the class means: the estimation-sample offsets without newdata, the offset() terms of the formula evaluated in newdata otherwise (an offset argument of panfmm() cannot be evaluated in newdata and is not applied, with a warning)
type	"response" for the marginal mean sum_c w_ic mu_c(x_it); "class" for the modal class of each observation's individual (an integer, the class with the largest posterior probability in the estimation sample, or the largest prior probability, from the concomitant variables when there are some, with newdata); "posterior" / "prior" for the matrix of class probabilities of each observation's individual; "components" for the matrix of class-specific means mu_c(x_it) (one column per class). In panglm 1.5.0 and earlier, type = "class" returned the class-specific means now given by "components"; a message recalls this once per session.
weights	for type = "response": "prior" (the class probabilities, from the concomitant variables when there are some) or "posterior" (the in-sample posterior probabilities; only without newdata)
...	unused

Value

a numeric vector ("response"), an integer vector of class labels 1, ..., k ("class"), or a matrix with one column per class ("posterior", "prior", "components")

See Also

Other panfmm methods: plot.panfmm(), summary.panfmm(), vcov.panfmm()

Examples

```
set.seed(3)
d <- data.frame(id = rep(1:150, each = 3), t = rep(1:3, 150), x = rnorm(450))
d$y <- rpois(450, exp(ifelse(rep(rbinom(150, 1, 0.4), each = 3) == 1, 1.5, 0) + 0.3 * d$x))
fit <- panfmm(y ~ x, data = d, index = c("id", "t"), nstart = 2, seed = 1)
head(predict(fit))
```

```
table(predict(fit, type = "class"))              # modal class per row
head(predict(fit, type = "posterior"))           # class probabilities
predict(fit, newdata = data.frame(x = c(-1, 0, 1)), type = "components")
```

`predict.panglm` *Predict from a fitted panglm model*

Description

Pooled predictions use the ordinary GLM mean. Random-effects response predictions are population averages over the fitted latent-effect distribution. Fixed-effects predictions include the estimated individual and, where applicable, time effects. New fixed-effect levels produce an error unless `allow.new.levels = TRUE`, in which case their contribution is set to zero.

Usage

```
## S3 method for class 'panglm'
predict(
  object,
  newdata = NULL,
  type = c("link", "response"),
  allow.new.levels = FALSE,
  ...
)
```

Arguments

`object`	a fitted `"panglm"` object
`newdata`	optional data frame; when omitted, return in-sample values
`type`	either `"link"` or `"response"`
`allow.new.levels`	if `TRUE`, assign zero to unseen fixed-effect levels; otherwise unseen levels are an error
`...`	unused

Details

Exact conditional binomial fits provide in-sample conditional inclusion probabilities. They cannot provide predictions for arbitrary `newdata` because the individual intercept is conditioned out and the probability depends on the complete response set and its success total.

Value

a numeric vector of predictions

See Also

Other panglm methods: coefplot(), confint.panglm(), effectcurve(), fitted.panglm(), fixef(), glance.panglm(), logLik.panglm(), model.matrix.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_ape(), plot.panglm_effectcurve(), predict.panordered(), residuals.panglm(), summary.panglm(), tidy.panglm(), vcov.panglm()

Examples

```
data(copd)
fit <- panglm(fev1 ~ crp, data = copd, index = c("id", "visit"),
              model = "random", family = "gaussian")
predict(fit, newdata = copd[1:4, ], type = "response")
```

---

predict.pangmm *Predictions from a dynamic panel GMM fit*

Description

Linear predictions from a pangmm() fit on newdata, a long data frame with the panel index columns and the variables of the first part of the formula. Lags are taken within individuals over the sorted periods of newdata, as in the fit.

Usage

```
## S3 method for class 'pangmm'
predict(object, newdata, type = c("level", "difference"), ...)
```

Arguments

object	a "pangmm" fit
newdata	a data frame in long format; the fit does not store its data, so newdata is required (pass the fitting data for in-sample predictions)
type	"level" or "difference"
...	unused

Details

- type = "level": x_it'b + d_t (+ c), the regressors (with their lags, including the lagged dependent variable) times their coefficients, plus the time effect and, for system GMM, the constant. The individual effect is not identified by GMM and is excluded, so for difference GMM (no constant) this is the level up to an individual intercept. On the fitting data these are the fitted values of the level equations of system GMM.
- type = "difference": the prediction of the first difference y_it - y_i,t-1, the difference of the level predictions, in which the individual effect cancels. On the fitting data these are the fitted values of the differenced equations.

Periods that precede the first time effect of the fit have a time effect of zero (the reference); a later period that is not in the fit gives a missing prediction, with a warning. A prediction is missing whenever a required lag is unavailable.

Value

a numeric vector with one prediction per row of `newdata`

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

See Also

Other pangmm methods: `plot.pangmm()`, `summary.pangmm()`, `vcov.pangmm()`

Examples

```
set.seed(3)
N <- 100; TT <- 6
d <- data.frame(id = rep(seq_len(N), each = TT), t = rep(seq_len(TT), N))
d$x <- rnorm(N * TT)
d$y <- ave(rep(rnorm(N), each = TT) + 0.3 * d$x + rnorm(N * TT), d$id,
           FUN = function(e) stats::filter(e, 0.4, method = "recursive"))
fit <- pangmm(y ~ lag(y, 1) + x | lag(y, 2:99), data = d, index = c("id", "t"),
              transformation = "ld")
head(predict(fit, newdata = d))
head(predict(fit, newdata = d, type = "difference"))
```

`predict.paniv` *Predictions from a panel instrumental-variables fit*

Description

Linear predictions from a `paniv()` fit. For `model = "pooling"`, `"random"` and `"hausmantaylor"` the prediction is the linear index `x'b` (with the intercept), the conditional mean given the regressors that does not involve the individual effect; this is what `predict()` returns for `AER::ivreg()` and `plm::plm()` fits of these models. For `model = "within"` the estimated individual effects `a_i = mean_t(y_it - x_it'b)` are added (and, for `effect = "twoways"`, the time effects, from the two-way decomposition of `y - x'b`), as in `predict.panglm()`.

Usage

```
## S3 method for class 'paniv'
predict(object, newdata = NULL, allow.new.levels = FALSE, ...)
```

Arguments

`object`	a `"paniv"` fit
`newdata`	optional data frame

allow.new.levels
: for within fits, if TRUE an individual (or time period) of newdata that is not in the fit gets an effect of zero, so that its prediction is the linear index x'b; if FALSE (default) it is an error

...
: unused

Details

Without newdata the in-sample fitted values are returned in the row order of the fitting data (with NA for the rows dropped by na.action). With newdata, the regressors of the first part of the formula are evaluated on newdata; factor regressors must carry the levels used in the fit. For within fits newdata must contain the panel index columns.

Value

a numeric vector of predictions, one per row of newdata (NA where a regressor is missing)

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

See Also

Other paniv methods: plot.pangmm(), summary.paniv(), vcov.paniv()

Examples

```
set.seed(1)
d <- data.frame(id = rep(1:50, each = 4), t = rep(1:4, 50))
d$z <- rnorm(200); d$u <- rnorm(200)
d$x <- d$z + d$u + rnorm(200)
d$y <- 1 + d$x + d$u + rep(rnorm(50), each = 4) + rnorm(200)
fit <- paniv(y ~ x | z, data = d, index = c("id", "t"), model = "within")
head(predict(fit))
predict(fit, newdata = data.frame(id = 1:2, t = 5, x = 0))
```

---

predict.panordered *Predict from an ordered panel model*

Description

Predict from an ordered panel model

Usage

```
## S3 method for class 'panordered'
predict(object, newdata = NULL, type = c("prob", "class", "link"), ...)
```

Arguments

object	a "panordered" fit (from panglm() with an ordered family)
newdata	optional data frame; defaults to the fitted data
type	"prob" for the matrix of category probabilities (population-averaged over the random intercept for model = "random"), "class" for the most likely category, or "link" for x'beta
...	unused

Value

a matrix with one column per category ("prob"), a factor ("class") or a numeric vector ("link")

See Also

Other panglm methods: coefplot(), confint.panglm(), effectcurve(), fitted.panglm(), fixef(), glance.panglm(), logLik.panglm(), model.matrix.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_ape(), plot.panglm_effectcurve(), predict.panglm(), residuals.panglm(), summary.panglm(), tidy.panglm(), vcov.panglm()

Examples

```
data(copd)
copd$lf <- cut(copd$fev1, 3, labels = c("low", "mid", "high"))
fit <- panglm(lf ~ crp + age, data = copd, index = c("id", "visit"),
              model = "pooling", family = "ologit")
head(predict(fit, type = "prob"))
predict(fit, newdata = copd[1:3, ], type = "class")
```

---

predict.panselect *Predictions from a panel sample-selection fit*

Description

Predictions from a panselect() fit on newdata, a long data frame with the panel index columns and the variables of both equations (the outcome and the selection indicator themselves are not needed).

Usage

```
## S3 method for class 'panselect'
predict(
  object,
  newdata,
  type = c("unconditional", "conditional", "selection"),
  ...
)
```

Arguments

object	a "panselect" fit
newdata	a data frame in long format (see Details); the fit does not store its data, so newdata is required (pass the fitting data for in-sample predictions)
type	"unconditional", "conditional" or "selection"
...	unused

Details

- type = "unconditional": the linear index of the outcome equation, x'b, the mean of the outcome in the whole population (selected or not) given the regressors. With model = "pooling" it includes the intercept and, with mundlak = TRUE, the Mundlak means; with model = "within" it excludes the individual effect (and the intercept, which the within transformation removes).
- type = "conditional": the mean in the selected population, x'b + sum_t g_t lambda_it, adding the inverse Mills ratio terms with their estimated coefficients (one per period with lambda = "period", a single one with lambda = "common").
- type = "selection": the selection probability pnorm(z'd_t) of the first-step probit of each row's period.

The inverse Mills ratios and probabilities use the first-step probit of the row's period; the Mundlak means are the means of the time-varying selection regressors over the rows of each individual in newdata, so newdata should hold every period of an individual (as in the fit). A period without a probit in the fit (every observation selected) has an inverse Mills ratio of zero and a missing selection probability; a period that is not in the fit gives a missing prediction for the conditional and selection types.

Value

a numeric vector with one prediction per row of newdata

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

See Also

Other panselect methods: plot.pangmm(), summary.panselect(), vcov.panselect()

Examples

```
set.seed(1)
d <- data.frame(id = rep(1:150, each = 3), t = rep(1:3, 150))
d$x <- rnorm(450); d$z <- rnorm(450); v <- rnorm(450)
d$s <- as.numeric(0.5 + d$x + d$z + v > 0)
d$y <- ifelse(d$s == 1, 1 + d$x + 0.5 * v + rnorm(450), NA)
fit <- panselect(s ~ x + z, y ~ x, data = d, index = c("id", "t"),
                 model = "pooling", boot = 0)
head(predict(fit, newdata = d, type = "conditional"))
head(predict(fit, newdata = d, type = "selection"))
```

`predict.twopart` *Predict from a two-part model*

Description

Predict from a two-part model

Usage

```
## S3 method for class 'twopart'
predict(object, newdata = NULL, type = c("response", "prob", "positive"), ...)
```

Arguments

`object`	a `"twopart"` fit
`newdata`	optional data frame; defaults to the (complete-case) data used for fitting. For `cre = TRUE` fits, the individual means are recomputed within `newdata` unless its `mean_*` columns are supplied.
`type`	`"response"` for `E[y] = P(y > 0) E[y | y > 0]`, `"prob"` for `P(y > 0)`, or `"positive"` for `E[y | y > 0]` (retransformed for the lognormal part)
`...`	unused

Value

a numeric vector with one prediction per row of `newdata`

See Also

Other twopart methods: `confint.twopart()`, `plot.twopart()`

Examples

```
data(copd)
set.seed(1)
copd$cost <- ifelse(copd$exacerbations > 0,
                    exp(6 + 0.1 * copd$crp + rnorm(nrow(copd), 0, 0.5)), 0)
tp <- twopart(cost ~ treatment + crp, data = copd, index = c("id", "visit"))
head(predict(tp))                    # E[y]
head(predict(tp, type = "prob"))     # P(y > 0)
head(predict(tp, type = "positive")) # E[y | y > 0]
```

`residuals.panglm` *Extract residuals from a panglm model*

Description

Response residuals `y - mu` or Pearson residuals `(y - mu) / sqrt(V(mu))`, with `mu` the stored fitted values and `V` the variance function of the family, unscaled by the dispersion as in `stats::residuals.glm()`: `1` (Gaussian), `mu` (Poisson), `mu + mu^2 / theta` (NB2 negative binomial), `mu (1 - mu)` (binomial), `mu^2` (gamma) and `mu (1 - mu) / (1 + phi)` (beta). For random-effects Poisson-gamma and beta-negative-binomial random-intercept fits, whose fitted values are marginal means, `V` is the marginal variance implied by the mixing distribution; where it is not finite (beta parameter `a <= 2`) the Poisson variance is used. For random-slope fits `V` is the conditional variance function evaluated at the marginal mean. Pearson residuals are not defined for the ordered and Tobit families.

Usage

```
## S3 method for class 'panglm'
residuals(object, type = c("response", "pearson"), ...)
```

Arguments

`object`	a fitted `"panglm"` object
`type`	`"response"` (default) or `"pearson"`
`...`	unused

Value

a numeric vector, one value per row of the original data (in that row order), `NA` at rows dropped for missingness.

See Also

Other panglm methods: `coefplot()`, `confint.panglm()`, `effectcurve()`, `fitted.panglm()`, `fixef()`, `glance.panglm()`, `logLik.panglm()`, `model.matrix.panglm()`, `partialeffects.panordered()`, `plot.panglm()`, `plot.panglm_ape()`, `plot.panglm_effectcurve()`, `predict.panglm()`, `predict.panordered()`, `summary.panglm()`, `tidy.panglm()`, `vcov.panglm()`

Examples

```
data(copd)
fit <- panglm(exacerbations ~ crp, data = copd,
              index = c("id", "visit"), model = "pooling",
              family = "poisson")
head(residuals(fit))
head(residuals(fit, type = "pearson"))
```

summary.panfmm *Summarize a panfmm fit*

Description

Summarize a panfmm fit

Usage

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

Arguments

object	a "panfmm" fit
...	unused

Value

an object of class "summary.panfmm": a list with one coefficient table per class (classes: estimate, standard error, z value and p-value, ancillary parameter on the log scale), the table of the class-probability (multinomial logit) parameters (prior), the mean class probabilities, the class sizes by modal posterior class, and the fit information (call, loglik, AIC, BIC, nobs, n_groups, vcov_type).

See Also

Other panfmm methods: plot.panfmm(), predict.panfmm(), vcov.panfmm()

Examples

```
set.seed(3)
d <- data.frame(id = rep(1:150, each = 3), t = rep(1:3, 150), x = rnorm(450))
d$y <- rpois(450, exp(ifelse(rep(rbinom(150, 1, 0.4), each = 3) == 1, 1.5, 0) + 0.3 * d$x))
summary(panfmm(y ~ x, data = d, index = c("id", "t"), nstart = 2, seed = 1))
```

summary.panglm *Summarize a panglm fit*

Description

Produces a classic coefficient table (Estimate, Std. Error, z value, Pr(>|z|)) in the style of summary.glm().

Usage

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

Arguments

object	a "panglm" object
...	unused

Value

an object of class "summary.panglm", a list with components call (the matched call), model, effect, and family (as stored on the fit), coefficients (a numeric matrix with one row per parameter and columns Estimate, Std. Error, z value, Pr(>|z|)), nobs, n_groups, df.residual, loglik, dispersion (NULL when not applicable, as for negative binomial fits), iterations, and vcov_type (which covariance estimator produced the reported standard errors). It has an associated print.summary.panglm() method and is normally printed rather than used programmatically. Also included are sigma (random-intercept standard deviation) and shape (gamma shape) when estimated, varcomp (covariance matrix of the random intercept and slopes) for random-slope fits, for family = "negbin" the NB2 negbin_theta (variance mu + mu^2 / theta) of pooled, fixed-effects and random-slope fits, with its standard error negbin_theta_se (pooled and fixed-effects fits, from the observed information at the fitted means, as in MASS::glm.nb()), or negbin_ab, the beta mixing parameters a and b of the random-intercept model, and, for cre = TRUE fits, cre_test: the Wald (Mundlak) test that the individual-mean coefficients are jointly zero, computed with the covariance stored on the fit.

See Also

Other panglm methods: coefplot(), confint.panglm(), effectcurve(), fitted.panglm(), fixef(), glance.panglm(), logLik.panglm(), model.matrix.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_ape(), plot.panglm_effectcurve(), predict.panglm(), predict.panordered(), residuals.panglm(), tidy.panglm(), vcov.panglm()

Examples

```
data(copd)
fit <- panglm(fev1 ~ treatment + age + smoker + crp, data = copd,
              index = c("id", "visit"), model = "pooling", family = "gaussian")
summary(fit)
```

---

summary.panglm_attgt *Aggregate group-time average treatment effects*

Description

Combines the ATT(g, t) of an attgt() fit into summary effects, as in Callaway and Sant'Anna (2021, Section 3) and did::aggte():

- "simple": the average of all post-treatment ATT(g, t) weighted by cohort size;
- "dynamic": the event study, ATT(e) for each event time e = t - g, averaging across cohorts weighted by cohort size; the overall effect is the mean of the post-treatment ATT(e), e >= 0;

- "group": the average effect of each cohort over its post-treatment periods; the overall effect weights cohorts by size;
- "calendar": the average effect in each calendar period across the cohorts treated by then (weighted by size); the overall effect is the mean over periods.

Usage

```
## S3 method for class 'panglm_attgt'
summary(
  object,
  type = c("simple", "dynamic", "group", "calendar"),
  min_e = -Inf,
  max_e = Inf,
  balance_e = NULL,
  level = object$level,
  ...
)
```

Arguments

object	a "panglm_attgt" fit from attgt()
type	"simple", "dynamic", "group" or "calendar"
min_e, max_e	for "dynamic": event-time window
balance_e	for "dynamic": if not NULL, only cohorts observed for at least balance_e periods after treatment are used, so that the composition of cohorts is the same at every event time up to balance_e
level	confidence level; defaults to that of the fit
...	unused

Details

Standard errors come from the influence functions and account for the estimation of the cohort-size weights. When the fit was bootstrapped (attgt() with bstrap = TRUE), the standard errors of the aggregated effects come from the same multiplier draws and, with cband = TRUE, the effects by level get uniform confidence bands (band_lower, band_upper) from the simultaneous critical value crit, as in did::aggte(). Clustering follows the fit.

Value

an object of class "summary.panglm_attgt": a list with overall (a one-row data frame with estimate, se, lower, upper), estimates (the effects by event time, cohort or period: column level plus the same columns; NULL for "simple"), type, pretrend, inf (the influence functions of the overall effect), crit (the simultaneous critical value of the uniform bands, or NULL), boot and the fit settings. The influence functions of the effects by level are the attribute "inf" of estimates.

See Also

Other attgt methods: plot.panglm_attgt()

Examples

```
set.seed(2)
d <- expand.grid(id = 1:150, t = 1:8)
d$first <- rep(c(4, 6, NA), each = 50)[d$id]
d$y <- rnorm(150)[d$id] + 0.2 * d$t +
  ifelse(!is.na(d$first) & d$t >= d$first, 1, 0) + rnorm(nrow(d))
fit <- attgt(y ~ 1, d, index = c("id", "t"), treat = "first")
summary(fit, type = "dynamic")
summary(fit, type = "calendar")
```

`summary.panglm_etwfe` *Average treatment effects on the treated from an ETWFE fit*

Description

Averages, over the treated observations, the difference between the predicted mean with and without the observation's cohort by period effect (on the response scale). The averages are taken over all treated observations (`"simple"`), by cohort (`"group"`), by calendar period (`"calendar"`) or by time since treatment (`"event"`), with delta-method standard errors. With sampling weights in `etwfe()`, the averages are weighted. They correspond to `etwfe::emfx()`. With `control = "never"`, the event study also reports the pre-treatment placebo effects, with event time `-1` as the reference (zero).

Usage

```
## S3 method for class 'panglm_etwfe'
summary(
  object,
  type = c("simple", "group", "calendar", "event"),
  level = object$level,
  byxvar = !is.null(object$xvar),
  ...
)
```

Arguments

`object`	a `"panglm_etwfe"` fit from `etwfe()`
`type`	`"simple"`, `"group"`, `"calendar"` or `"event"`
`level`	confidence level; defaults to that of the fit
`byxvar`	logical; for a fit with `xvar`, report the ATTs separately for each value of the covariate (by aggregation level and value). The default is `TRUE` for such fits; `overall` is always the ATT over all treated observations.
`...`	unused

Value

an object of class "summary.panglm_etwfe": a list with overall (the simple ATT: estimate, se, statistic, p.value, lower, upper), estimates (the same columns by level of the aggregation, and by value of the xvar covariate in a column named after it when byxvar = TRUE; NULL for "simple" without byxvar), cells and the fit information.

See Also

Other etwfe methods: plot.panglm_etwfe()

Examples

```
set.seed(2)
d <- expand.grid(id = 1:150, t = 1:8)
d$first <- rep(c(4, 6, NA), each = 50)[d$id]
d$y <- rbinom(nrow(d), 1, plogis(-1 + 0.1 * d$t +
  0.8 * (!is.na(d$first) & d$t >= d$first)))
fit <- etwfe(y ~ 1, d, index = c("id", "t"), treat = "first", family = "binomial")
summary(fit, type = "event")
```

---

summary.pangmm *Summarize a pangmm fit*

Description

Summarize a pangmm fit

Usage

```
## S3 method for class 'pangmm'
summary(object, time.dummies = FALSE, ...)
```

Arguments

object	a "pangmm" fit
time.dummies	logical: show the time effects (and the system-GMM constant) in the coefficient table
...	unused

Value

an object of class "summary.pangmm": a list with the coefficient table (coefficients: estimate, standard error, z value and p-value), the Arellano-Bond tests (artest), sargan, hansen, wald and the fit information (call, label, vcov_type, nobs, n_groups, n_periods, n_instruments).

See Also

Other pangmm methods: plot.pangmm(), predict.pangmm(), vcov.pangmm()

Examples

```
set.seed(3)
N <- 200; TT <- 6
d <- data.frame(id = rep(seq_len(N), each = TT), t = rep(seq_len(TT), N))
mu <- rep(rnorm(N), each = TT)
d$y <- ave(mu + rnorm(N * TT), d$id, FUN = function(e) stats::filter(e, 0.4,
  method = "recursive"))
summary(pangmm(y ~ lag(y, 1) | lag(y, 2:99), data = d, index = c("id", "t")))
```

`summary.paniv` *Summarize a paniv fit*

Description

Summarize a paniv fit

Usage

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

Arguments

`object`	a "paniv" fit
`...`	unused

Value

an object of class "summary.paniv": a list with the coefficient table (coefficients: estimate, standard error, z value and p-value), first_stage, overid, ercomp, endogenous, instruments, varlist (Hausman-Taylor) and the fit information (call, model, nobs, n_groups, df.residual, vcov_type).

See Also

Other paniv methods: plot.pangmm(), predict.paniv(), vcov.paniv()

Examples

```
set.seed(2)
d <- data.frame(id = rep(1:50, each = 4), t = rep(1:4, 50))
d$z <- rnorm(200); d$u <- rnorm(200)
d$x <- d$z + d$u + rnorm(200); d$y <- 1 + d$x + d$u + rnorm(200)
summary(paniv(y ~ x | z, data = d, index = c("id", "t"), model = "within"))
```

summary.panqr *Summarize a panqr fit*

Description

Summarize a panqr fit

Usage

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

Arguments

object	a "panqr" fit
...	unused

Value

an object of class "summary.panqr": a list with one coefficient table per quantile (coefficients: estimate, standard error, z value and p-value), the location and scale tables and the quantiles q of the standardized errors for MM-QR, and the fit information (call, model, tau, nobs, n_groups, vcov_type, R, cluster, n_clusters, dropped).

See Also

Other panqr methods: plot.panqr(), vcov.panqr()

Examples

```
set.seed(2)
d <- data.frame(id = rep(1:100, each = 5), t = rep(1:5, 100))
d$x <- rnorm(500) + rep(rnorm(100), each = 5)
d$y <- d$x + rep(rnorm(100), each = 5) + (1 + 0.2 * abs(d$x)) * rnorm(500)
summary(panqr(y ~ x, data = d, index = c("id", "t"), tau = 0.9, model = "mmqr"))
```

summary.panselect *Summarize a panselect fit*

Description

Summarize a panselect fit

Usage

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

Arguments

object	a "panselect" fit
...	unused

Value

an object of class "summary.panselect": a list with the coefficient table (coefficients: estimate, standard error, z value and p-value, using the default covariance of the fit), the selection test (selection_test), the probit coefficients (probit) and the fit information.

See Also

Other panselect methods: plot.pangmm(), predict.panselect(), vcov.panselect()

Examples

```
set.seed(3)
d <- data.frame(id = rep(1:100, each = 3), t = rep(1:3, 100))
d$x <- rnorm(300); d$z <- rnorm(300); v <- rnorm(300)
d$s <- as.numeric(0.5 + d$x + d$z + v > 0)
d$y <- ifelse(d$s == 1, 1 + d$x + 0.5 * v + rnorm(300), NA)
summary(panselect(s ~ x + z, y ~ x, data = d, index = c("id", "t"),
                  boot = 29, seed = 1))
```

---

tidy.panglm *Tidy a panglm fit into a data frame of coefficients*

Description

Implements the generics::tidy() generic so panglm results work with broom/modelsummary table-building workflows.

Usage

```
## S3 method for class 'panglm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

x	a "panglm" fit
conf.int	if TRUE, add conf.low/conf.high columns
conf.level	confidence level for the interval
...	unused

Value

a data.frame with one row per coefficient

See Also

tidy-methods for the other model classes.

Other panglm methods: coefplot(), confint.panglm(), effectcurve(), fitted.panglm(), fixef(), glance.panglm(), logLik.panglm(), model.matrix.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_ape(), plot.panglm_effectcurve(), predict.panglm(), predict.panordered(), residuals.panglm(), summary.panglm(), vcov.panglm()

Examples

```
data(copd)
fit <- panglm(fev1 ~ treatment + age + smoker + crp, data = copd,
              index = c("id", "visit"), model = "pooling", family = "gaussian")
if (requireNamespace("generics", quietly = TRUE)) {
  generics::tidy(fit, conf.int = TRUE)
}
```

---

tidy.panglm_hurdle *Tidy and glance methods for the panglm model classes*

Description

Implement the generics::tidy() and generics::glance() generics for the fits of twopart(), pangmm(), paniv(), panqr(), panselect(), panfmm(), attgt(), etwfe() and eventstudy(), the hurdle model, and for the test objects of the Hausman, joint Wald and dispersion tests, so that they work with broom and modelsummary.

Usage

```
## S3 method for class 'panglm_hurdle'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'panglm_hurdle'
glance(x, ...)

## S3 method for class 'twopart'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'twopart'
glance(x, ...)

## S3 method for class 'pangmm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'pangmm'
glance(x, ...)

## S3 method for class 'paniv'
```

```
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'paniv'
glance(x, ...)

## S3 method for class 'panqr'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'panqr'
glance(x, ...)

## S3 method for class 'panselect'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'panselect'
glance(x, ...)

## S3 method for class 'panfmm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'panfmm'
glance(x, ...)

## S3 method for class 'panglm_attgt'
tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  type = c("attgt", "simple", "dynamic", "group", "calendar"),
  ...
)

## S3 method for class 'panglm_attgt'
glance(x, ...)

## S3 method for class 'panglm_etwfe'
tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  type = c("simple", "group", "calendar", "event", "coefficients"),
  ...
)

## S3 method for class 'panglm_etwfe'
glance(x, ...)

## S3 method for class 'panglm_eventstudy'
```

```
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'panglm_eventstudy'
glance(x, ...)

## S3 method for class 'panglm_hausman'
tidy(x, ...)

## S3 method for class 'panglm_hausman'
glance(x, ...)

## S3 method for class 'panglm_jointtest'
tidy(x, ...)

## S3 method for class 'panglm_jointtest'
glance(x, ...)

## S3 method for class 'panglm_dispersiontest'
tidy(x, ...)

## S3 method for class 'panglm_dispersiontest'
glance(x, ...)
```

Arguments

x	a fitted model or test object
conf.int	if TRUE, add conf.low/conf.high columns
conf.level	confidence level for the interval
...	further arguments passed to the summary() aggregation (attgt, etwfe), otherwise unused
type	for attgt() and etwfe() fits, which effects to tidy (see Details)

Details

tidy() returns one row per coefficient with columns term, estimate, std.error, statistic (the Wald z), p.value and, with conf.int = TRUE, the Wald interval conf.low, conf.high. The estimates and standard errors are those of coef() and vcov(), so the interval equals confint(). Extra columns identify the rows where a fit has several blocks of coefficients: component (twopart: "binary" or "positive"; panfmm: "class" or "prior"), class (panfmm), tau (panqr: one row per term and quantile), group, time, event (attgt, etwfe, eventstudy).

For attgt(), type = "attgt" (the default) gives one row per group-time effect ATT(g, t), named as in coef(); the other types are the aggregations of summary.panglm_attgt(). For etwfe(), type = "simple" (the default) gives the overall ATT, "group", "calendar" and "event" the aggregations of summary.panglm_etwfe(), and "coefficients" the coefficients of the underlying regression. For eventstudy() the reference period is a row with estimate and standard error zero and a missing statistic. A statistic is missing whenever its standard error is zero or missing.

For the test objects, `tidy()` and `glance()` both return a one-row data frame in the style of `broom`'s `htest` methods: `statistic`, `p.value`, `parameter` (the degrees of freedom; for an F test also `den.df`), `distribution` and `method`.

`glance()` returns one row with the estimator, the variance type, the numbers of observations and individuals, the log-likelihood, AIC and BIC where the model is fitted by maximum likelihood (twopart, panfmm), and the specification tests reported by `summary()` (for example the Sargan, Hansen and Arellano-Bond tests of `pangmm()`, the pre-trend test of `attgt()` and `eventstudy()`).

Value

a `data.frame`: one row per coefficient (or effect) for `tidy()` of a fit, one row for `glance()` and for the test objects (see Details).

Validation

Numerical accuracy is checked in the package tests and in the validation suite; see the validation article on the package website.

See Also

`tidy.panglm()` and `glance.panglm()` for `panglm()` fits.

Examples

```
data(copd)
if (requireNamespace("generics", quietly = TRUE)) {
  set.seed(1)
  d <- data.frame(id = rep(1:100, each = 5), t = rep(1:5, 100), x = rnorm(500))
  d$y <- d$x + rep(rnorm(100), each = 5) + rnorm(500)
  fit <- panqr(y ~ x, data = d, index = c("id", "t"), tau = c(0.25, 0.75))
  generics::tidy(fit, conf.int = TRUE)
  generics::glance(fit)
}
```

---

treatmentplot *Plot the treatment timing of a staggered design*

Description

Draws one row per individual and one column per period, sorted by treatment cohort (earliest adopters at the top, never-treated units at the bottom), with treated cells shaded. The right axis gives each cohort and its size. With many individuals the unit labels are dropped and the tiles are drawn as a raster, so the plot stays readable for thousands of units. Periods in which a unit is not observed are left blank.

Usage

```
treatmentplot(
  data,
  index,
  treat,
  labels = NULL,
  col = c("#08519C", "#C6DBEF", "grey88"),
  legend = TRUE,
  xlab = "Period",
  ylab = NULL,
  main = NULL,
  ...
)
```

Arguments

`data`	a data frame in long format
`index`	`c(individual, time)`: column names, the time numeric
`treat`	name of the column with the first treated period, as in `attgt()` and `etwfe()` (`0`, `NA` or `Inf` for never-treated units)
`labels`	logical: show the individual labels on the left axis. The default shows them for at most 60 individuals.
`col`	colours of the treated, not-yet-treated and never-treated cells
`legend`	logical: add a legend below the plot
`xlab`, `ylab`, `main`	axis labels and title
`...`	passed to `graphics::image()`

Value

a data frame with one row per cohort (`cohort`, `n`; `NA` for the never treated), invisibly

See Also

Other policy evaluation: `attgt()`, `did()`, `etwfe()`, `eventplot()`, `eventstudy()`, `panelplot()`, `trendplot()`

Examples

```
d <- expand.grid(id = 1:30, year = 2010:2019)
d$first <- rep(c(2013, 2015, 2017, 0, 0, 0), each = 5)[d$id]
treatmentplot(d, index = c("id", "year"), treat = "first")
```

`trendplot` *Plot raw outcome trends of a difference-in-differences design*

Description

Draws the mean outcome over time, the classical first look at a difference-in-differences design before any model is fitted.

Usage

```
trendplot(
  data,
  index,
  treat,
  outcome,
  type = c("cohort", "treated", "event"),
  weights = NULL,
  band = FALSE,
  level = 0.95,
  min_e = -Inf,
  max_e = Inf,
  col = NULL,
  legend = TRUE,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`data`	a data frame in long format
`index`	`c(individual, time)`: column names, the time numeric
`treat`	name of the column with the first treated period, as in `attgt()` and `treatmentplot()` (`0`, `NA` or `Inf` for never-treated units)
`outcome`	name of the outcome column (numeric or logical)
`type`	`"cohort"`, `"treated"` or `"event"` (see Details)
`weights`	optional sampling weights: the name of a column of `data` or a non-negative numeric vector with one value per row
`band`	logical: add pointwise confidence bands (shaded) around each line
`level`	confidence level of the bands
`min_e, max_e`	event window for `type = "event"`
`col`	line colours, one per line (cohorts first, the never treated last). The default gives ordered colours to the cohorts and black to the never treated.

legend	logical: add a legend
xlab, ylab, main	axis labels and title
ylim	y-axis limits
...	passed to graphics::plot()

Details

- type = "cohort": one line per adoption cohort and one for the never-treated units, against calendar time. Points are hollow before the cohort adopts and filled from adoption on.
- type = "treated": the ever-treated and never-treated means against calendar time. A dashed vertical line marks a single adoption date (half a period before it); with staggered adoption each adoption date gets a dotted line (the ever-treated mean then mixes treated and not yet treated observations).
- type = "event": the treated units aligned on event time (period minus first treated period), with a dashed line at treatment onset (-0.5). Never-treated units have no event time, so their mean cannot be aligned directly; the never-treated line is instead their mean in the same calendar periods as the treated observations: at event time e, each treated observation contributes the never-treated mean of its own period (a calendar-matched comparison, as in stacked event studies). The treated mean at each event time pools cohorts observed in different calendar periods, so its composition changes with e.

Means are weighted by weights when given. The standard error of each mean is cluster-robust by individual,

$$\widehat{se}(\bar{y}_c) = \sqrt{\frac{G_c}{G_c - 1} \sum_g \Big(\sum_{i \in g} w_i (y_i - \bar{y}_c) \Big)^2 \Big/ W_c^2},$$

with G_c the number of individuals contributing to the mean and W_c the sum of their weights: the usual standard error of a mean (with one observation per individual it is s / sqrt(n)), and the one of fixest::feols(y ~ 1, cluster = ~id) fitted on the observations of the mean. The calendar-matched never-treated line of type = "event" is a fixed linear combination of period means; its standard error uses the same formula with the influence of each never-treated observation.

Value

a data frame of the plotted means, invisibly: group (the line label), time (calendar period, or event time for type = "event"), mean, se and n (number of observations)

See Also

treatmentplot() for the treatment timing, panelplot() for the unit by period values, attgt() and etwfe() for the estimators.

Other policy evaluation: attgt(), did(), etwfe(), eventplot(), eventstudy(), panelplot(), treatmentplot()

Examples

```
set.seed(2)
d <- expand.grid(id = 1:90, t = 1:8)
d$first <- rep(c(4, 6, 0), each = 30)[d$id]
```

```
d$y <- rnorm(90)[d$id] + 0.2 * d$t +
  ifelse(d$first > 0 & d$t >= d$first, 1, 0) + rnorm(nrow(d))
trendplot(d, index = c("id", "t"), treat = "first", outcome = "y")
trendplot(d, c("id", "t"), "first", "y", type = "treated", band = TRUE)
m <- trendplot(d, c("id", "t"), "first", "y", type = "event")
head(m)
```

`twopart` *Two-part model for panel cost data*

Description

Fits the two-part model of Duan et al. (1983), the standard model for health care expenditures with a mass of zeros: a binary model for `P(y > 0)` on all observations, and a model for `E[y | y > 0]` on the positive observations. The unconditional mean is `E[y] = P(y > 0) * E[y | y > 0]`.

Usage

```
twopart(
  formula,
  data,
  index,
  binary_formula = NULL,
  model = c("pooling", "random"),
  link = c("probit", "logit"),
  positive = c("gamma", "lognormal"),
  retransform = c("smearing", "normal"),
  correlated = FALSE,
  cre = FALSE,
  weights = NULL,
  vcov = c("classical", "HC1", "cluster"),
  R = 21,
  maxit = 100,
  tol = 1e-10
)
```

Arguments

`formula`	model formula for the positive part, e.g. `cost ~ age + treatment`; also used for the binary part unless `binary_formula` is given
`data`	a data.frame
`index`	length-1 or length-2 character vector of panel index columns
`binary_formula`	optional right-hand-side formula for the binary part, e.g. `~ age + treatment + smoker`
`model`	`"pooling"` or `"random"`, used for both parts
`link`	link of the binary part, `"probit"` or `"logit"`

positive	"gamma" (gamma GLM, log link) or "lognormal" (linear model for log(y))
retransform	for positive = "lognormal": "smearing" or "normal"
correlated	logical; with model = "random" and positive = "lognormal", let the random intercepts of the two parts be correlated and fit them jointly
cre	logical; add individual means of the time-varying regressors to both parts
weights	optional observation weights (model = "pooling" only): the name of a column of data or a numeric vector with one value per row of data
vcov	covariance type passed to panglm() for both parts
R, maxit, tol	passed to panglm()

Details

The positive part is either a gamma GLM with a log link (positive = "gamma", no retransformation needed) or a linear model for log(y) (positive = "lognormal"). For the log model, E[y | y > 0] = exp(x'beta) * phi needs a retransformation factor phi: Duan's smearing estimate, the (weighted) mean of the exponentiated log-scale residuals (retransform = "smearing"), or the normal-theory factor exp(sigma^2 / 2) (retransform = "normal"; with model = "random", sigma^2 is the sum of the individual and idiosyncratic variance components). Both assume homoskedastic log-scale errors.

With model = "random" each part has its own random intercept and the parts are estimated separately; the combined mean multiplies the two population-averaged means, which assumes the two random intercepts are independent. With correlated = TRUE (random effects, lognormal positive part) the two random intercepts are instead bivariate normal with correlation rho and both parts are estimated jointly by maximum likelihood (Van Ourti 2004): individuals with a high propensity to use care may also have high spending when they do. The positive-part random intercept is integrated out in closed form and the binary-part one by adaptive Gauss-Hermite quadrature. Predictions then integrate over the joint distribution (normal-theory retransformation). A likelihood-ratio test of rho = 0 compares logLik() with the separate fit. cre = TRUE adds the individual means of the time-varying regressors (computed over all observations of the individual, including zeros) to both parts, the Mundlak correlated random effects device.

Value

an object of class "twopart" with components binary and positive (the two panglm() fits), retransformation (the factor phi, 1 for the gamma part) and the settings used. Methods: print, summary, coef, vcov, confint, predict, fitted, logLik, nobs, partialeffects(), tidy and glance.

Validation

Both parts are panglm() fits and share its validation. The package tests also check that the pooled two-part prediction is the product of the two stats::glm() fits, the smearing retransformation of the lognormal part, and the correlated random-intercept likelihood against two-dimensional numerical integration and against GLMMadaptive::mixed_model() with the hurdle.lognormal() family (logit link, correlated random intercepts), whose log-likelihood it reproduces at the same parameters.

References

Duan, N., Manning, W. G., Morris, C. N. and Newhouse, J. P. (1983). A comparison of alternative models for the demand for medical care. *Journal of Business and Economic Statistics*, 1, 115-126. doi:10.1080/07350015.1983.10509330

Duan, N. (1983). Smearing estimate: a nonparametric retransformation method. *Journal of the American Statistical Association*, 78, 605-610. doi:10.1080/01621459.1983.10478017

Van Ourti, T. (2004). Measuring horizontal inequity in Belgian health care using a Gaussian random effects two part count data model. *Health Economics*, 13, 705-724. doi:10.1002/hec.920

See Also

`predict.twopart()`, `confint.twopart()` and `partialeffects()` for the fitted model, tidy-methods for `tidy()` and `glance()`.

Other panel models: `panfmm()`, `panfmmselect()`, `panglm()`, `pangmm()`, `panhurdle()`, `paniv()`, `panqr()`, `panselect()`

Examples

```
data(copd)
set.seed(1)
copd$cost <- ifelse(copd$exacerbations > 0,
                    exp(6 + 0.1 * copd$crp + rnorm(nrow(copd), 0, 0.5)), 0)
tp <- twopart(cost ~ treatment + age + crp, data = copd,
              index = c("id", "visit"), positive = "gamma")
summary(tp)
head(predict(tp))
partialeffects(tp)
```

---

vcov.panfmm *Covariance of a panfmm fit*

Description

Covariance of a panfmm fit

Usage

```
## S3 method for class 'panfmm'
vcov(object, type = NULL, cluster = NULL, ...)
```

Arguments

`object`	a `"panfmm"` fit
`type`	`NULL` (the covariance stored on the fit), `"classical"` (inverse of the negative Hessian of the observed-data log-likelihood) or `"cluster"` (sandwich with individual scores summed within clusters, factor `G / (G - 1)`; with weights the individual scores are multiplied by the individual weights)

cluster	optional clustering vector with one value per observation used in the fit (in the row order of the data), constant within individuals; defaults to the individual
...	unused

Value

the covariance matrix of all parameters

See Also

Other panfmm methods: plot.panfmm(), predict.panfmm(), summary.panfmm()

Examples

```
set.seed(3)
d <- data.frame(id = rep(1:150, each = 3), t = rep(1:3, 150), x = rnorm(450))
d$y <- rpois(450, exp(ifelse(rep(rbinom(150, 1, 0.4), each = 3) == 1, 1.5, 0) + 0.3 * d$x))
fit <- panfmm(y ~ x, data = d, index = c("id", "t"), nstart = 2, seed = 1)
sqrt(diag(vcov(fit, type = "cluster")))
```

---

vcov.panglm *Recompute the coefficient covariance matrix for a panglm fit*

Description

With type = NULL (the default) the covariance the object was fit with is returned (set by the vcov argument of panglm()). type = "classical" returns the model-based covariance, also for a fit made with a robust vcov. type = "HC1" (heteroskedasticity-robust / White) and type = "cluster" (cluster-robust sandwich, clustered by the panel individual unless cluster is given) are available for pooled and fixed-effects models. For random-effects models only "cluster" is available: the units of the integrated likelihood are individuals, so the sandwich is built from individual scores (clusters must contain whole individuals) with the factor G / (G - 1). It guards against a misspecified random-effects distribution and is the right choice with sampling weights. For the unweighted Gaussian (Swamy-Arora) fit it is the cluster sandwich of the quasi-demeaned regression, as in Stata's xtreg, re vce(cluster id). For exact conditional binomial models, the independent likelihood units are panel strata rather than observations. Their HC1 covariance therefore uses stratum-level scores. A custom cluster may combine complete strata, but it may not split observations from one conditioned stratum. For the two-way conditional logit (effect = "twoways") the time dummies are parameters of the conditional likelihood: the sandwich is built from the full (slopes + time dummies) stratum scores and bread, its small-sample factors count all of them, and the slope block is returned.

Usage

```
## S3 method for class 'panglm'
vcov(object, type = NULL, cluster = NULL, ...)
```

Arguments

object	a "panglm" fit
type	NULL (the covariance stored on the fit) or one of "classical", "HC1", "cluster"
cluster	optional vector giving the clustering variable (one value per original observation, i.e. nrow(data) long); defaults to the panel individual identifier used to fit the model
...	unused

Value

a numeric covariance matrix for the estimated coefficients, with row and column names matching names(coef(object)). For type = "classical" this is the model-based covariance; for "HC1" and "cluster" it is a sandwich (robust or cluster-robust) covariance estimate of the same dimension.

See Also

Other panglm methods: coefplot(), confint.panglm(), effectcurve(), fitted.panglm(), fixef(), glance.panglm(), logLik.panglm(), model.matrix.panglm(), partialeffects.panordered(), plot.panglm(), plot.panglm_ape(), plot.panglm_effectcurve(), predict.panglm(), predict.panordered(), residuals.panglm(), summary.panglm(), tidy.panglm()

Examples

```
data(copd)
fit <- panglm(fev1 ~ crp, data = copd, index = c("id", "visit"),
              model = "within", family = "gaussian")
vcov(fit)
vcov(fit, type = "cluster")
```

---

vcov.pangmm *Covariance of a pangmm fit*

Description

Covariance of a pangmm fit

Usage

```
## S3 method for class 'pangmm'
vcov(object, type = NULL, ...)
```

Arguments

object	a "pangmm" fit
type	NULL (the covariance stored on the fit), "robust" (the one-step sandwich, or the Windmeijer-corrected two-step covariance) or "classical" (the homoskedastic one-step covariance, or the uncorrected two-step covariance); "cluster" is a synonym of "robust"
...	unused

Value

the covariance matrix of the coefficients

See Also

Other pangmm methods: plot.pangmm(), predict.pangmm(), summary.pangmm()

Examples

```
set.seed(3)
N <- 200; TT <- 6
d <- data.frame(id = rep(seq_len(N), each = TT), t = rep(seq_len(TT), N))
mu <- rep(rnorm(N), each = TT)
d$y <- ave(mu + rnorm(N * TT), d$id, FUN = function(e) stats::filter(e, 0.4,
  method = "recursive"))
fit <- pangmm(y ~ lag(y, 1) | lag(y, 2:99), data = d, index = c("id", "t"),
              model = "twostep")
sqrt(diag(vcov(fit, type = "classical")))
sqrt(diag(vcov(fit, type = "robust")))
```

---

vcov.paniv *Covariance of a paniv fit*

Description

Covariance of a paniv fit

Usage

```
## S3 method for class 'paniv'
vcov(object, type = NULL, cluster = NULL, ...)
```

Arguments

object	a "paniv" fit
type	NULL (the covariance stored on the fit), "classical" or "cluster"
cluster	optional clustering vector with one value per observation used in the fit (in the row order of the data); defaults to the panel individual
...	unused

Value

the covariance matrix of the coefficients

See Also

Other paniv methods: `plot.pangmm()`, `predict.paniv()`, `summary.paniv()`

Examples

```
set.seed(2)
d <- data.frame(id = rep(1:50, each = 4), t = rep(1:4, 50))
d$z <- rnorm(200); d$u <- rnorm(200)
d$x <- d$z + d$u + rnorm(200); d$y <- 1 + d$x + d$u + rnorm(200)
fit <- paniv(y ~ x | z, data = d, index = c("id", "t"), model = "pooling")
vcov(fit, type = "cluster")
```

---

vcov.panqr *Covariance of a panqr fit*

Description

Covariance of a panqr fit

Usage

```
## S3 method for class 'panqr'
vcov(object, tau = NULL, ...)
```

Arguments

`object`	a "panqr" fit
`tau`	NULL or one of the fitted quantiles
`...`	unused

Value

the covariance matrix of the coefficients at quantile `tau`; when several quantiles were fitted and `tau` is `NULL`, a list of matrices, one per quantile

See Also

Other panqr methods: `plot.panqr()`, `summary.panqr()`

Examples

```
set.seed(2)
d <- data.frame(id = rep(1:100, each = 5), t = rep(1:5, 100))
d$x <- rnorm(500) + rep(rnorm(100), each = 5)
d$y <- d$x + rep(rnorm(100), each = 5) + (1 + 0.2 * abs(d$x)) * rnorm(500)
fit <- panqr(y ~ x, data = d, index = c("id", "t"), tau = c(0.25, 0.75),
             model = "mmqr")
vcov(fit, tau = 0.75)
```

`vcov.panselect` *Covariance of a panselect fit*

Description

Covariance of a panselect fit

Usage

```
## S3 method for class 'panselect'
vcov(object, type = NULL, ...)
```

Arguments

`object`	a `"panselect"` fit
`type`	`NULL` (the default covariance of the fit), `"boot"` (panel bootstrap), `"cluster"` (cluster-robust by individual, ignoring the estimation of the first step) or `"classical"` (homoskedastic, also ignoring the first step)
`...`	unused

Value

the covariance matrix of the outcome-equation coefficients

See Also

Other panselect methods: `plot.pangmm()`, `predict.panselect()`, `summary.panselect()`

Examples

```
set.seed(3)
d <- data.frame(id = rep(1:100, each = 3), t = rep(1:3, 100))
d$x <- rnorm(300); d$z <- rnorm(300); v <- rnorm(300)
d$s <- as.numeric(0.5 + d$x + d$z + v > 0)
d$y <- ifelse(d$s == 1, 1 + d$x + 0.5 * v + rnorm(300), NA)
fit <- panselect(s ~ x + z, y ~ x, data = d, index = c("id", "t"), boot = 0)
vcov(fit, type = "cluster")
```

Index

∗ **attgt methods**
  plot.panglm_attgt, 94
  summary.panglm_attgt, 118
∗ **datasets**
  cigar, 8
  copd, 14
  docvisits, 18
  healthspend, 37
  medicaid, 45
  meps, 49
∗ **etwfe methods**
  plot.panglm_etwfe, 98
  summary.panglm_etwfe, 120
∗ **eventstudy methods**
  eventstudy-methods, 32
  plot.panglm_eventstudy, 101
∗ **panel models**
  panfmm, 57
  panfmmselect, 61
  panglm, 62
  pangmm, 69
  panhurdle, 75
  paniv, 77
  panqr, 80
  panselect, 84
  twopart, 132
∗ **panfmm methods**
  plot.panfmm, 89
  predict.panfmm, 107
  summary.panfmm, 117
  vcov.panfmm, 134
∗ **panglm methods**
  coefplot, 10
  confint.panglm, 12
  effectcurve, 22
  fitted.panglm, 33
  fixef, 34
  glance.panglm, 36
  logLik.panglm, 44
  model.matrix.panglm, 53
  partialeffects.panordered, 87
  plot.panglm, 90
  plot.panglm_ape, 93
  plot.panglm_effectcurve, 97
  predict.panglm, 109
  predict.panordered, 112
  residuals.panglm, 116
  summary.panglm, 117
  tidy.panglm, 124
  vcov.panglm, 135
∗ **pangmm methods**
  plot.pangmm, 103
  predict.pangmm, 110
  summary.pangmm, 121
  vcov.pangmm, 136
∗ **panhurdle methods**
  hurdle-methods, 41
  plot.panglm_hurdle, 102
∗ **paniv methods**
  plot.pangmm, 103
  predict.paniv, 111
  summary.paniv, 122
  vcov.paniv, 137
∗ **panqr methods**
  plot.panqr, 105
  summary.panqr, 123
  vcov.panqr, 138
∗ **panselect methods**
  plot.pangmm, 103
  predict.panselect, 113
  summary.panselect, 123
  vcov.panselect, 139
∗ **policy evaluation**
  attgt, 4
  did, 16
  etwfe, 24
  eventplot, 28
  eventstudy, 30

panelplot, 56
treatmentplot, 128
trendplot, 130
∗ **tests**
jointtest, 42
lineartest, 43
pandispersiontest, 55
panhausman, 73
∗ **twopart methods**
confint.twopart, 13
plot.twopart, 106
predict.twopart, 115

attgt, 4, *18*, *27*, *29*, *31*, *57*, *129*, *131*
attgt(), *14*, *16*, *25*, *27, 28*, *31*, *46, 47*, *56*, *94–96*, *118, 119*, *125*, *127–131*

betafamily *(*panglm-families*)*, 67
betafamily(), *63*
binomialfamily *(*panglm-families*)*, 67
binomialfamily(), *63*

cigar, 8, *14*
coef.panglm_hurdle *(*hurdle-methods*)*, 41
coefplot, 10, *12*, *23*, *33*, *36, 37*, *44*, *54*, *88*, *92*, *94*, *98*, *110*, *113*, *116*, *118*, *125*, *136*
coefplot(), *89*, *91*
confint.panglm, *11*, 12, *23*, *33*, *36, 37*, *44*, *54*, *88*, *92*, *94*, *98*, *110*, *113*, *116*, *118*, *125*, *136*
confint.panglm_eventstudy *(*eventstudy-methods*)*, 32
confint.panglm_hurdle *(*hurdle-methods*)*, 41
confint.twopart, 13, *107*, *115*
confint.twopart(), *134*
copd, 14

did, *7*, 16, *27*, *29*, *31*, *57*, *129*, *131*
did(), *4*, *7*, *24*, *27*, *30, 31*, *46, 47*
docvisits, *14*, 18

effectcurve, *11, 12*, 22, *33*, *36, 37*, *44*, *54*, *88*, *92*, *94*, *98*, *110*, *113*, *116*, *118*, *125*, *136*
effectcurve(), *65*, *88*, *98*
etwfe, *7*, *18*, 24, *29*, *31*, *57*, *129*, *131*
etwfe(), *7*, *14*, *16, 17*, *28*, *31*, *46*, *98, 99*, *120*, *125*, *127*, *129*, *131*

eventplot, *7*, *18*, *27*, 28, *31*, *57*, *129*, *131*
eventstudy, *7*, *18*, *27*, *29*, 30, *57*, *129*, *131*
eventstudy(), *7*, *14*, *16*, *27–29*, *32*, *47*, *101*, *125*, *127, 128*
eventstudy-methods, *31*, 32

fitted.panglm, *11, 12*, *23*, 33, *36, 37*, *44*, *54*, *88*, *92*, *94*, *98*, *110*, *113*, *116*, *118*, *125*, *136*
fixef, *11, 12*, *23*, *33*, 34, *37*, *44*, *54*, *88*, *92*, *94*, *98*, *110*, *113*, *116*, *118*, *125*, *136*
fixef(), *92*

gammafamily *(*panglm-families*)*, 67
gammafamily(), *63*
gaussianfamily *(*panglm-families*)*, 67
gaussianfamily(), *63*
glance.panfmm *(*tidy.panglm_hurdle*)*, 125
glance.panglm, *11, 12*, *23*, *33*, *36*, 36, *44*, *54*, *88*, *92*, *94*, *98*, *110*, *113*, *116*, *118*, *125*, *136*
glance.panglm(), *128*
glance.panglm_attgt *(*tidy.panglm_hurdle*)*, 125
glance.panglm_dispersiontest *(*tidy.panglm_hurdle*)*, 125
glance.panglm_etwfe *(*tidy.panglm_hurdle*)*, 125
glance.panglm_eventstudy *(*tidy.panglm_hurdle*)*, 125
glance.panglm_hausman *(*tidy.panglm_hurdle*)*, 125
glance.panglm_hurdle *(*tidy.panglm_hurdle*)*, 125
glance.panglm_jointtest *(*tidy.panglm_hurdle*)*, 125
glance.pangmm *(*tidy.panglm_hurdle*)*, 125
glance.paniv *(*tidy.panglm_hurdle*)*, 125
glance.panqr *(*tidy.panglm_hurdle*)*, 125
glance.panselect *(*tidy.panglm_hurdle*)*, 125
glance.twopart *(*tidy.panglm_hurdle*)*, 125
graphics::image(), *56*, *129*
graphics::legend(), *98*
graphics::plot(), *96*, *98*, *100*, *102*, *131*
graphics::plot.default(), *89*, *91*, *94*, *103–106*

healthspend, *14*, 37

hurdle-methods, 41, *76*

jointtest, 42, *44*, *55*, *75*
jointtest(), *31*, *43*

lineartest, *43*, 43, *55*, *75*
logLik.panglm, *11, 12*, *23*, *33*, *36, 37*, 44, *54*, *88*, *92*, *94*, *98*, *110*, *113*, *116*, *118*, *125*, *136*
logLik.panglm_eventstudy *(*eventstudy-methods*)*, 32

medicaid, *14*, 45
meps, *14*, 49
model.matrix.panglm, *11, 12*, *23*, *33*, *36, 37*, *44*, 53, *88*, *92*, *94*, *98*, *110*, *113*, *116*, *118*, *125*, *136*

negbinfamily *(*panglm-families*)*, 67
negbinfamily(), *55*, *63*
nobs.panglm_eventstudy *(*eventstudy-methods*)*, 32
nobs.panglm_hurdle *(*hurdle-methods*)*, 41

orderedfamily *(*panglm-families*)*, 67
orderedfamily(), *63*

pandispersiontest, *43, 44*, 55, *75*
pandispersiontest(), *76*
panelplot, *7*, *18*, *27*, *29*, *31*, 56, *129*, *131*
panelplot(), *131*
panfmm, 57, *61*, *66*, *72*, *76*, *79*, *83*, *86*, *134*
panfmm(), *11*, *15*, *61*, *65*, *108*, *125*
panfmmselect, *60*, 61, *66*, *72*, *76*, *79*, *83*, *86*, *134*
panglm, *60, 61*, 62, *72*, *76*, *79*, *83*, *86*, *134*
panglm(), *11*, *17, 18*, *30, 31*, *33, 34*, *46, 47*, *51*, *58*, *62*, *67*, *69*, *71*, *75–77*, *81*, *90*, *113*, *128*, *133*, *135*
panglm-deprecated, *69*
panglm-families, *66*, 67
pangmm, *60, 61*, *66*, 69, *76*, *79*, *83*, *86*, *134*
pangmm(), *11*, *65*, *110*, *125*, *128*
panhausman, *43, 44*, *55*, 73
panhausman(), *15*
panhurdle, *60, 61*, *66*, *72*, 75, *79*, *83*, *86*, *134*
panhurdle(), *15*, *34, 35*, *51*
paniv, *60, 61*, *66*, *72*, *76*, 77, *83*, *86*, *134*
paniv(), *11*, *65*, *111*, *125*
panqr, *60, 61*, *66*, *72*, *76*, *79*, 80, *86*, *134*
panqr(), *16*, *51*, *65*, *105*, *125*
panselect, *60, 61*, *66*, *72*, *76*, *79*, *83*, 84, *134*
panselect(), *51*, *103*, *113*, *125*
partialeffects *(*partialeffects.panordered*)*, 87
partialeffects(), *23*, *51*, *64–66*, *68*, *93*, *133, 134*
partialeffects.panordered, *11, 12*, *23*, *33*, *36, 37*, *44*, *54*, 87, *92*, *94*, *98*, *110*, *113*, *116*, *118*, *125*, *136*
plm::pgmm(), *70–72*
plm::pht(), *78*
plm::plm(), *78*
plot.panfmm, 89, *108*, *117*, *135*
plot.panglm, *11, 12*, *23*, *33*, *36, 37*, *44*, *54*, *88*, 90, *94*, *98*, *110*, *113*, *116*, *118*, *125*, *136*
plot.panglm(), *35, 36*, *89*, *102, 103*, *106, 107*
plot.panglm_ape, *11, 12*, *23*, *33*, *36, 37*, *44*, *54*, *88*, *92*, 93, *98*, *110*, *113*, *116*, *118*, *125*, *136*
plot.panglm_ape(), *88*
plot.panglm_attgt, 94, *119*
plot.panglm_attgt(), *29*, *98*, *100, 101*
plot.panglm_effectcurve, *11, 12*, *23*, *33*, *36, 37*, *44*, *54*, *88*, *92*, *94*, 97, *110*, *113*, *116*, *118*, *125*, *136*
plot.panglm_etwfe, 98, *121*
plot.panglm_eventstudy, *33*, 101
plot.panglm_eventstudy(), *31*
plot.panglm_hurdle, *42*, 102
plot.panglm_hurdle(), *76*
plot.pangmm, 103, *111, 112*, *114*, *121, 122*, *124*, *137–139*
plot.paniv *(*plot.pangmm*)*, 103
plot.panqr, 105, *123*, *138*
plot.panqr(), *83*
plot.panselect *(*plot.pangmm*)*, 103
plot.summary.panglm_attgt *(*plot.panglm_attgt*)*, 94
plot.summary.panglm_etwfe *(*plot.panglm_etwfe*)*, 98
plot.twopart, *13*, 106, *115*
poissonfamily *(*panglm-families*)*, 67
poissonfamily(), *63*
predict.panfmm, *89*, 107, *117*, *135*
predict.panfmm(), *60*
predict.panglm, *11, 12*, *23*, *33*, *36, 37*, *44*,

54, *88*, *92*, *94*, *98*, 109, *113*, *116*, *118*, *125*, *136*
predict.panglm(), *66*, *111*
predict.pangmm, *104*, 110, *121*, *137*
predict.pangmm(), *72*
predict.paniv, *104*, 111, *122*, *138*
predict.paniv(), *79*
predict.panordered, *11, 12*, *23*, *33*, *36, 37*, *44*, *54*, *88*, *92*, *94*, *98*, *110*, 112, *116*, *118*, *125*, *136*
predict.panordered(), *68*
predict.panselect, *104*, 113, *124*, *139*
predict.panselect(), *86*
predict.twopart, *13*, *107*, 115
predict.twopart(), *134*
print.summary.panglm_eventstudy *(*eventstudy-methods*)*, 32
print.summary.panglm_hurdle *(*hurdle-methods*)*, 41

quantreg::rq.fit(), *81, 82*

ranef *(*fixef*)*, 34
ranef(), *92*
residuals.panglm, *11, 12*, *23*, *33*, *36, 37*, *44*, *54*, *88*, *92*, *94*, *98*, *110*, *113*, 116, *118*, *125*, *136*
residuals.panglm(), *91*

stats::density(), *91*, *106, 107*
stats::family, *67*
stats::glm(), *11*, *59*, *85*
stats::lowess(), *91*
stats::model.frame(), *58*, *63*, *77*, *81*, *84*
stats::printCoefmat(), *32*
summary.glm(), *117*
summary.panfmm, *89*, *108*, 117, *135*
summary.panfmm(), *60*
summary.panglm, *11, 12*, *23*, *33*, *36, 37*, *44*, *54*, *88*, *92*, *94*, *98*, *110*, *113*, *116*, 117, *125*, *136*
summary.panglm(), *18*, *64*, *66*
summary.panglm_attgt, *97*, 118
summary.panglm_attgt(), *4–7*, *46*, *96*, *127*
summary.panglm_etwfe, *100*, 120
summary.panglm_etwfe(), *25, 26*, *127*
summary.panglm_eventstudy *(*eventstudy-methods*)*, 32
summary.panglm_hurdle *(*hurdle-methods*)*, 41
summary.pangmm, *104*, *111*, 121, *137*
summary.pangmm(), *72*
summary.paniv, *104*, *112*, 122, *138*
summary.paniv(), *79*
summary.panqr, *106*, 123, *138*
summary.panqr(), *83*
summary.panselect, *104*, *114*, 123, *139*
summary.panselect(), *86*

tidy-methods, *31*, *37*, *60*, *66*, *72*, *79*, *83*, *86*, *125*, *134*
tidy-methods *(*tidy.panglm_hurdle*)*, 125
tidy.panfmm *(*tidy.panglm_hurdle*)*, 125
tidy.panglm, *11, 12*, *23*, *33*, *36, 37*, *44*, *54*, *88*, *92*, *94*, *98*, *110*, *113*, *116*, *118*, 124, *136*
tidy.panglm(), *128*
tidy.panglm_attgt *(*tidy.panglm_hurdle*)*, 125
tidy.panglm_dispersiontest *(*tidy.panglm_hurdle*)*, 125
tidy.panglm_etwfe *(*tidy.panglm_hurdle*)*, 125
tidy.panglm_eventstudy *(*tidy.panglm_hurdle*)*, 125
tidy.panglm_hausman *(*tidy.panglm_hurdle*)*, 125
tidy.panglm_hurdle, 125
tidy.panglm_jointtest *(*tidy.panglm_hurdle*)*, 125
tidy.pangmm *(*tidy.panglm_hurdle*)*, 125
tidy.paniv *(*tidy.panglm_hurdle*)*, 125
tidy.panqr *(*tidy.panglm_hurdle*)*, 125
tidy.panselect *(*tidy.panglm_hurdle*)*, 125
tidy.twopart *(*tidy.panglm_hurdle*)*, 125
tobitfamily *(*panglm-families*)*, 67
tobitfamily(), *63*
treatmentplot, *7*, *18*, *27*, *29*, *31*, *57*, 128, *131*
treatmentplot(), *57*, *130, 131*
trendplot, *7*, *18*, *27*, *29*, *31*, *57*, *129*, 130
trendplot(), *57*
twopart, *60, 61*, *66*, *72*, *76*, *79*, *83*, *86*, 132
twopart(), *11*, *13*, *16*, *51*, *76*, *125*

vcov.panfmm, *89*, *108*, *117*, 134
vcov.panfmm(), *58*, *60*

vcov.panglm, *11, 12*, *23*, *33*, *36, 37*, *44*, *54*, *88*, *92*, *94*, *98*, *110*, *113*, *116*, *118*, *125*, 135
vcov.panglm(), *12*, *22*, *42, 43*, *64*, *66*
vcov.panglm_eventstudy *(*eventstudy-methods*)*, 32
vcov.panglm_hurdle *(*hurdle-methods*)*, 41
vcov.pangmm, *104*, *111*, *121*, 136
vcov.pangmm(), *70*, *72*
vcov.paniv, *104*, *112*, *122*, 137
vcov.paniv(), *77*, *79*
vcov.panqr, *106*, *123*, 138
vcov.panqr(), *83*
vcov.panselect, *104*, *114*, *124*, 139
vcov.panselect(), *86*