

# Package ‘penppml’

September 8, 2023

**Title** Penalized Poisson Pseudo Maximum Likelihood Regression

**Version** 0.2.3

**Description** A set of tools that enables efficient estimation of penalized Poisson Pseudo Maximum Likelihood regressions, using lasso or ridge penalties, for models that feature one or more sets of high-dimensional fixed effects. The methodology is based on Breinlich, Corradi, Rocha, Ruta, Santos Silva, and Zylkin (2021) <<http://hdl.handle.net/10986/35451>> and takes advantage of the method of alternating projections of Gaure (2013) <[doi:10.1016/j.csda.2013.03.024](https://doi.org/10.1016/j.csda.2013.03.024)> for dealing with HDFE, as well as the coordinate descent algorithm of Friedman, Hastie and Tibshirani (2010) <[doi:10.18637/jss.v033.i01](https://doi.org/10.18637/jss.v033.i01)> for fitting lasso regressions. The package is also able to carry out cross-validation and to implement the plugin lasso of Belloni, Chernozhukov, Hansen and Kozbur (2016) <[doi:10.1080/07350015.2015.1102733](https://doi.org/10.1080/07350015.2015.1102733)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**LazyDataCompression** gzip

**RoxygenNote** 7.2.3

**LinkingTo** Rcpp, RcppEigen

**Imports** Rcpp, glmnet, fixest, collapse, rlang, magrittr, matrixStats, dplyr, devtools

**Depends** R (>= 2.10)

**URL** <https://github.com/tomzylkin/penppml>

**BugReports** <https://github.com/tomzylkin/penppml/issues>

**Suggests** testthat (>= 3.0.0), MASS, knitr, rmarkdown, ggplot2, reshape2

**Config/testthat/edition** 3

**VignetteBuilder** knitr, rmarkdown

**NeedsCompilation** yes

**Author** Diego Ferreras Garrucho [aut],  
 Tom Zylkin [aut],  
 Joao Cruz [cre],  
 Nicolas Apfel [ctb]

**Maintainer** Joao Cruz <jm01780@surrey.ac.uk>

**Repository** CRAN

**Date/Publication** 2023-09-08 14:10:02 UTC

## R topics documented:

AtA . . . . .	3
bootstrap . . . . .	3
cluster_matrix . . . . .	5
collinearity_check . . . . .	6
compute_fes . . . . .	6
countries . . . . .	7
eigenMatMult . . . . .	8
fastolsCpp . . . . .	8
fastridge . . . . .	9
fastridgeCpp . . . . .	9
faststddev . . . . .	10
fastwmean . . . . .	10
genfes . . . . .	11
genmodel . . . . .	11
hdfepml . . . . .	12
hdfepml_int . . . . .	14
iceberg . . . . .	17
manyouter . . . . .	18
mlfitppml . . . . .	19
mlfitppml_int . . . . .	21
penhdfepml . . . . .	24
penhdfepml_cluster . . . . .	27
penhdfepml_cluster_int . . . . .	29
penhdfepml_int . . . . .	32
plugin_lasso_int . . . . .	35
select_fes . . . . .	36
standardize_wt . . . . .	36
trade . . . . .	37
xeex . . . . .	38
xvalidate . . . . .	38
<b>Index</b>	<b>42</b>

---

AtA	<i>Computing A'A</i>
-----	----------------------

---

**Description**

Computes A'A using C++.

**Usage**

AtA(A)

**Arguments**

A                    A matrix.

---

bootstrap	<i>Bootstrap Lasso Implementation (in development)</i>
-----------	--

---

**Description**

This function performs standard plugin lasso PPML estimation for bootreps samples drawn again with replacement and reports those regressors selected in at least a certain fraction of the bootstrap repetitions.

**Usage**

```
bootstrap(  
  data,  
  dep,  
  indep = NULL,  
  cluster_id = NULL,  
  fixed = NULL,  
  selectobs = NULL,  
  bootreps = 250,  
  boot_threshold = 0.01,  
  colcheck_x = FALSE,  
  colcheck_x_fes = FALSE,  
  post = FALSE,  
  gamma_val = NULL,  
  verbose = FALSE,  
  tol = 1e-06,  
  hdfetol = 0.01,  
  penweights = NULL,  
  maxiter = 1000,  
  phipost = TRUE  
)
```

**Arguments**

<code>data</code>	A data frame containing all relevant variables.
<code>dep</code>	A string with the names of the independent variables or their column numbers.
<code>indep</code>	A vector with the names or column numbers of the regressors. If left unspecified, all remaining variables (excluding fixed effects) are included in the regressor matrix.
<code>cluster_id</code>	A string denoting the cluster-id with which to perform cluster bootstrap.
<code>fixed</code>	A vector with the names or column numbers of factor variables identifying the fixed effects, or a list with the desired interactions between variables in <code>data</code> .
<code>selectobs</code>	Optional. A vector indicating which observations to use (either a logical vector or a numeric vector with row numbers, as usual when subsetting in R).
<code>bootreps</code>	Number of bootstrap repetitions.
<code>boot_threshold</code>	Minimal threshold. If a variable is selected in at least this fraction of times, it is reported at the end of the iterations.
<code>colcheck_x</code>	Logical. If TRUE, this checks collinearity between the independent variables and drops the collinear variables.
<code>colcheck_x_fes</code>	Logical. If TRUE, this checks whether the independent variables are perfectly explained by the fixed effects drops those that are perfectly explained.
<code>post</code>	Logical. If TRUE, estimates a post-penalty regression with the selected variables.
<code>gamma_val</code>	Numerical value that determines the regularization threshold as defined in Belloni, Chernozhukov, Hansen, and Kozbur (2016). NULL default sets parameter to $0.1/\log(n)$ .
<code>verbose</code>	Logical. If TRUE, it prints information to the screen while evaluating.
<code>tol</code>	Tolerance parameter for convergence of the IRLS algorithm.
<code>hdfetol</code>	Tolerance parameter for the within-transformation step, passed on to <code>collapse::fhdwithin</code> .
<code>penweights</code>	Optional: a vector of coefficient-specific penalties to use in plugin lasso when <code>method == "plugin"</code> .
<code>maxiter</code>	Maximum number of iterations (a number).
<code>phipost</code>	Logical. If TRUE, the plugin coefficient-specific penalty weights are iteratively calculated using estimates from a post-penalty regression. Otherwise, these are calculated using estimates from a penalty regression.

**Details**

This function enables users to implement the "bootstrap" step in the procedure described in Breinlich, Corradi, Rocha, Ruta, Santos Silva and Zylkin (2020). To do this, Plugin Lasso is run `B` times. The function can also perform a post-selection estimation.

**Value**

A matrix with coefficient estimates for all dependent variables.

## References

- Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.
- Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.
- Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.
- Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.
- Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.

## Examples

```
## Not run: bs1 <- bootstrap(data=trade3, dep="export",
  cluster_id="clus",
  fixed=list(c("exp", "time"),
  c("imp", "time"), c("exp", "imp")),
  indep=7:22, bootreps=10, colcheck_x = TRUE,
  colcheck_x_fes = TRUE,
  boot_threshold = 0.01,
  post=TRUE, gamma_val=0.01, verbose=FALSE)
## End(Not run)
```

---

cluster\_matrix

*Cluster-robust Standard Error Estimation*

---

## Description

cluster\_matrix is a helper for computation of cluster-robust standard errors.

## Usage

```
cluster_matrix(e, cluster, x)
```

## Arguments

e	Vector of residuals.
cluster	Vector of clusters.
x	Regressor matrix.

## Value

Gives the XeeX matrix.

---

collinearity_check	<i>Checking for Perfect Multicollinearity</i>
--------------------	---

---

### Description

collinearity\_check checks for perfect multicollinearity in a model with high-dimensional fixed effects. It calls `lfe::demeanlist` in order to partial out the fixed effects, and then uses `stats::lm.wfit` to discard linearly dependent variables.

### Usage

```
collinearity_check(
  y,
  x = NULL,
  fes = NULL,
  hdfetol,
  colcheck_x_fes = TRUE,
  colcheck_x = TRUE
)
```

### Arguments

y	Dependent variable (a numeric vector).
x	Regressor matrix.
fes	List of fixed effects.
hdfetol	Tolerance for the centering, passed on to <code>lfe::demeanlist</code> .
colcheck_x_fes	Logical. If TRUE, this checks whether the independent variables are perfectly explained by the fixed effects drops those that are perfectly explained.
colcheck_x	Logical. If TRUE, this checks collinearity between the independent variables and drops the collinear variables.

### Value

A numeric vector containing the variables that pass the collinearity check.

---

compute_fes	<i>Fixed Effects Computation</i>
-------------	----------------------------------

---

### Description

This function is a helper for `xvalidate` that computes FEs using PPML First Order Conditions (FOCs).

**Usage**

```
compute_fes(
  y,
  fes,
  x,
  b,
  insample_obs = rep(1, n),
  onlymus = FALSE,
  tol = 1e-08,
  verbose = FALSE
)
```

**Arguments**

y	Dependent variable (a vector).
fes	List of fixed effects.
x	Regressor matrix.
b	A vector of coefficient estimates.
insample_obs	Vector of observations used to estimate the b coefficients..
onlymus	Logical. If TRUE, returns only the conditional means.
tol	A tolerance parameter.
verbose	Logical. If TRUE, prints messages to the console while evaluating.

**Value**

If `onlymus = TRUE`, the vector of conditional means. Otherwise, a list with two elements:

- `mu`: conditional means.
- `fe_values`: fixed effects.

---

countries

*Country ISO Codes*

---

**Description**

An auxiliary data set with basic geographic information about country ISO 3166 codes included in the trade data set.

**Usage**

```
countries
```

**Format**

A data frame with 249 rows and 4 variables.

- iso: Country ISO 3166 code.
- name: Country name.
- region: Continent.
- subregion: sub-continental region.

**Source**

The source of the data set is Luke Duncalfe's ISO-3166-Countries-with-Regional-Codes repository on GitHub (<https://github.com/luke/ISO-3166-Countries-with-Regional-Codes#readme>).

---

eigenMatMult	<i>Faster Matrix Multiplication</i>
--------------	-------------------------------------

---

**Description**

Faster matrix multiplication using C++.

**Usage**

```
eigenMatMult(A, B)
```

```
eigenMapMatMult(A, B)
```

**Arguments**

A, B	Matrices.
------	-----------

---

fastolsCpp	<i>Faster Least Squares Estimation</i>
------------	--

---

**Description**

Finds Least Squares solutions using C++.

**Usage**

```
fastolsCpp(X, y)
```

**Arguments**

X	Regressor matrix.
y	Dependent variable (a vector).



**Value**

The vector of parameter (beta) estimates.

---

 fastridge

*Finding Ridge Regression Solutions*


---

**Description**

A wrapper around `fastridgeCpp`, for faster computation of the analytical solution for ridge regression.

**Usage**

```
fastridge(x, y, weights = rep(1/n, n), lambda, standardize = TRUE)
```

**Arguments**

<code>x</code>	Regressor matrix.
<code>y</code>	Dependent variable (a numeric vector).
<code>weights</code>	Vector of weights.
<code>lambda</code>	Penalty parameter.
<code>standardize</code>	Logical. If TRUE, <code>x</code> is standardized using the weights.

**Value**

A vector of coefficient (beta) estimates.

---

 fastridgeCpp

*Faster Ridge Regression*


---

**Description**

Finds Ridge solutions using C++.

**Usage**

```
fastridgeCpp(X, y, lambda)
```

**Arguments**

<code>X</code>	Regressor matrix.
<code>y</code>	Dependent variable (a vector).
<code>lambda</code>	Penalty parameter (a number).

**Value**

The vector of parameter (beta) estimates.

**faststddev***Faster Standard Deviation*

---

**Description**

Computes standard deviation using C++.

**Usage**

```
faststddev(X, w)
```

**Arguments**

X	Regressor matrix.
w	Weights.

**Value**

Vector of standard deviations of the parameter estimates.

---

**fastwmean***Faster Weighted Mean*

---

**Description**

Computes weighted mean using C++.

**Usage**

```
fastwmean(X, w)
```

**Arguments**

X	Regressor matrix.
w	Weights.

**Value**

Weighted mean.

---

genfes *Generating a List of Fixed Effects*

---

**Description**

genfes generates a list of fixed effects by creating interactions of paired factors.

**Usage**

```
genfes(data, inter)
```

**Arguments**

data	A data frame including the factors.
inter	A list: each element includes the variables to be interacted (both names and column)

**Value**

A list containing the desired interactions of vars, with the same length as inter.

---

genmodel *Generating Model Structure*

---

**Description**

genmodel transforms a data frame into the needed components for our main functions (a y vector, a x matrix and a fes list).

**Usage**

```
genmodel(  
  data,  
  dep = NULL,  
  indep = NULL,  
  fixed = NULL,  
  cluster = NULL,  
  selectobs = NULL  
)
```

**Arguments**

<code>data</code>	A data frame containing all relevant variables.
<code>dep</code>	A string with the name of the independent variable or a column number.
<code>indep</code>	A vector with the names or column numbers of the regressors. If left unspecified, all remaining variables (excluding fixed effects) are included in the regressor matrix.
<code>fixed</code>	A vector with the names or column numbers of factor variables identifying the fixed effects, or a list with the desired interactions between variables in <code>data</code> .
<code>cluster</code>	Optional. A string with the name of the clustering variable or a column number. It's also possible to input a vector with several variables, in which case the interaction of all of them is taken as the clustering variable.
<code>selectobs</code>	Optional. A vector indicating which observations to use.

**Value**

A list with four elements:

- `y`: y vector.
- `x`: x matrix.
- `fes`: list of fixed effects.
- `cluster`: cluster vector.

---

hdfeppl

*PPML Estimation with HDFE*

---

**Description**

`hdfeppl` fits an (unpenalized) Poisson Pseudo Maximum Likelihood (PPML) model with high-dimensional fixed effects (HDFE).

**Usage**

```
hdfeppl(  
  data,  
  dep = 1,  
  indep = NULL,  
  fixed = NULL,  
  cluster = NULL,  
  selectobs = NULL,  
  ...  
)
```

**Arguments**

<code>data</code>	A data frame containing all relevant variables.
<code>dep</code>	A string with the name of the independent variable or a column number.
<code>indep</code>	A vector with the names or column numbers of the regressors. If left unspecified, all remaining variables (excluding fixed effects) are included in the regressor matrix.
<code>fixed</code>	A vector with the names or column numbers of factor variables identifying the fixed effects, or a list with the desired interactions between variables in <code>data</code> .
<code>cluster</code>	Optional. A string with the name of the clustering variable or a column number. It's also possible to input a vector with several variables, in which case the interaction of all of them is taken as the clustering variable.
<code>selectobs</code>	Optional. A vector indicating which observations to use (either a logical vector or a numeric vector with row numbers, as usual when subsetting in R).
<code>...</code>	Further options. For a full list, see <a href="#">hdfepml_int</a> .

**Details**

This function is a thin wrapper around [hdfepml\\_int](#), providing a more convenient interface for data frames. Whereas the internal function requires some preliminary handling of data sets (`y` must be a vector, `x` must be a matrix and fixed effects `fes` must be provided in a list), the wrapper takes a full data frame in the `data` argument, and users can simply specify which variables correspond to `y`, `x` and the fixed effects, using either variable names or column numbers.

More formally, `hdfepml_int` performs iteratively re-weighted least squares (IRLS) on a transformed model, as described in Correia, Guimarães and Zylkin (2020) and similar to the `ppmlhdfc` package in Stata. In each iteration, the function calculates the transformed dependent variable, partials out the fixed effects (calling `collapse:fhwithin`) and then solves a weighted least squares problem (using fast C++ implementation).

**Value**

A list with the following elements:

- `coefficients`: a  $1 \times \text{ncol}(x)$  matrix with coefficient (beta) estimates.
- `residuals`: a  $1 \times \text{length}(y)$  matrix with the residuals of the model.
- `mu`: a  $1 \times \text{length}(y)$  matrix with the final values of the conditional mean  $\mu$ .
- `deviance`:
- `bic`: Bayesian Information Criterion.
- `x_resid`: matrix of demeaned regressors.
- `z_resid`: vector of demeaned (transformed) dependent variable.
- `se`: standard errors of the coefficients.

## References

- Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.
- Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.
- Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.
- Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.
- Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.

## Examples

```
## Not run:
# To reduce run time, we keep only countries in the Americas:
americas <- countries$iso[countries$region == "Americas"]
test <- hdfepml(data = trade[, -(5:6)],
               dep = "export",
               fixed = list(c("exp", "time"),
                           c("imp", "time"),
                           c("exp", "imp")),
               selectobs = (trade$imp %in% americas) & (trade$exp %in% americas))

## End(Not run)
```

---

hdfepml\_int

*PPML Estimation with HDFE*

---

## Description

`hdfepml_int` is the internal algorithm called by `hdfepml` to fit an (unpenalized) Poisson Pseudo Maximum Likelihood (PPML) regression with high-dimensional fixed effects (HDFE). It takes a vector with the dependent variable, a regressor matrix and a set of fixed effects (in list form: each element in the list should be a separate HDFE).

## Usage

```
hdfepml_int(
  y,
  x = NULL,
  fes = NULL,
  tol = 1e-08,
```

```

    hdfetol = 1e-04,
    mu = NULL,
    saveX = TRUE,
    colcheck = TRUE,
    colcheck_x = colcheck,
    colcheck_x_fes = colcheck,
    init_z = NULL,
    verbose = FALSE,
    maxiter = 1000,
    cluster = NULL,
    vcv = TRUE
)

```

### Arguments

y	Dependent variable (a vector)
x	Regressor matrix.
fes	List of fixed effects.
tol	Tolerance parameter for convergence of the IRLS algorithm.
hdfetol	Tolerance parameter for the within-transformation step, passed on to <code>collapse::fhdwithin</code> .
mu	A vector of initial values for mu that can be passed to the command.
saveX	Logical. If TRUE, it returns the values of x and z after partialling out the fixed effects.
colcheck	Logical. If TRUE, performs both checks in <code>colcheck_x</code> and <code>colcheck_x_fes</code> . If the user specifies <code>colcheck_x</code> and <code>colcheck_x_fes</code> individually, this option is overwritten.
colcheck_x	Logical. If TRUE, this checks collinearity between the independent variables and drops the collinear variables.
colcheck_x_fes	Logical. If TRUE, this checks whether the independent variables are perfectly explained by the fixed effects drops those that are perfectly explained.
init_z	Optional: initial values of the transformed dependent variable, to be used in the first iteration of the algorithm.
verbose	Logical. If TRUE, it prints information to the screen while evaluating.
maxiter	Maximum number of iterations (a number).
cluster	Optional: a vector classifying observations into clusters (to use when calculating SEs).
vcv	Logical. If TRUE (the default), it returns standard errors.

### Details

More formally, `hdfppml_int` performs iteratively re-weighted least squares (IRLS) on a transformed model, as described in Correia, Guimarães and Zylkin (2020) and similar to the `ppmlhdfc` package in Stata. In each iteration, the function calculates the transformed dependent variable, partials out the fixed effects (calling `collapse::fhdwithin`, which uses the algorithm in Gaure (2013)) and then solves a weighted least squares problem (using fast C++ implementation).

**Value**

A list with the following elements:

- coefficients: a  $1 \times \text{ncol}(x)$  matrix with coefficient (beta) estimates.
- residuals: a  $1 \times \text{length}(y)$  matrix with the residuals of the model.
- mu: a  $1 \times \text{length}(y)$  matrix with the final values of the conditional mean  $\mu$ .
- deviance:
- bic: Bayesian Information Criterion.
- x\_resid: matrix of demeaned regressors.
- z\_resid: vector of demeaned (transformed) dependent variable.
- se: standard errors of the coefficients.

**References**

Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.

Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.

Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.

Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.

Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.

**Examples**

```
## Not run:
# To reduce run time, we keep only countries in the Americas:
americas <- countries$iso[countries$region == "Americas"]
trade <- trade[(trade$imp %in% americas) & (trade$exp %in% americas), ]
# Now generate the needed x, y and fes objects:
y <- trade$export
x <- data.matrix(trade[, -1:-6])
fes <- list(exp_time = interaction(trade$exp, trade$time),
           imp_time = interaction(trade$imp, trade$time),
           pair     = interaction(trade$exp, trade$imp))
# Finally, the call to hdfepml_int:
reg <- hdfepml_int(y = y, x = x, fes = fes)

## End(Not run)
```



---

iceberg *Iceberg Lasso Implementation (in development)*

---

## Description

A function performs standard plugin lasso PPML estimation (without fixed effects) for several dependent variables in a single step. This is still IN DEVELOPMENT: at the current stage, only coefficient estimates are provided and there is no support for clustered errors.

## Usage

```
iceberg(data, dep, indep = NULL, selectobs = NULL, ...)
```

## Arguments

data	A data frame containing all relevant variables.
dep	A string with the names of the independent variables or their column numbers.
indep	A vector with the names or column numbers of the regressors. If left unspecified, all remaining variables (excluding fixed effects) are included in the regressor matrix.
selectobs	Optional. A vector indicating which observations to use (either a logical vector or a numeric vector with row numbers, as usual when subsetting in R).
...	Further arguments, including: <ul style="list-style-type: none"> <li>• tol: Tolerance parameter for convergence of the IRLS algorithm.</li> <li>• glmnettol: Tolerance parameter to be passed on to <code>glmnet::glmnet</code>.</li> <li>• penweights: Optional: a vector of coefficient-specific penalties to use in plugin lasso.</li> <li>• colcheck: Logical. If TRUE, checks for perfect multicollinearity in <code>x</code>.</li> <li>• K: Maximum number of iterations.</li> <li>• verbose: Logical. If TRUE, prints information to the screen while evaluating.</li> <li>• lambda: Penalty parameter (a number).</li> <li>• icepost: Logical. If TRUE, it carries out a post-lasso estimation with just the selected variables and reports the coefficients from this regression.</li> </ul>

## Details

This functions enables users to implement the "iceberg" step in the two-step procedure described in Breinlich, Corradi, Rocha, Ruta, Santos Silva and Zylkin (2020). To do this after using the plugin method in `mlfitppml`, just select all the variables with non-zero coefficients in `dep` and the remaining regressors in `indep`. The function will then perform separate lasso estimation on each of the selected dependent variables and report the coefficients.

## Value

A matrix with coefficient estimates for all dependent variables.

## References

- Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.
- Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.
- Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.
- Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.
- Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.

## Examples

```
iceberg_results <- iceberg(data = trade[, -(1:6)],
                          dep = c("ad_prov_14", "cp_prov_23", "tbt_prov_07",
                                  "tbt_prov_33", "tf_prov_41", "tf_prov_45"),
                          selectobs = (trade$time == "2016"))
```

---

manyouter

*Many Outer Products*

---

## Description

Compute a large number of outer products (useful for clustered SEs) using C++.

## Usage

```
manyouter(A, B, c)
```

## Arguments

A, B	Numeric vectors.
c	Integer.

**Description**

mlfitppml is a general-purpose wrapper function for penalized PPML estimation. This is a flexible tool that allows users to select:

- Penalty type: either lasso or ridge.
- Penalty parameter: users can provide a single global value for lambda (a single regression is estimated), a vector of lambda values (the function estimates the regression using each of them, sequentially) or even coefficient-specific penalty weights.
- Method: plugin lasso estimates can be obtained directly from this function too.
- Cross-validation: if this option is enabled, the function uses IDs provided by the user to perform k-fold cross-validation and reports the resulting RMSE for all lambda values.

**Usage**

```
mlfitppml(
  data,
  dep = 1,
  indep = NULL,
  fixed = NULL,
  cluster = NULL,
  selectobs = NULL,
  ...
)
```

**Arguments**

data	A data frame containing all relevant variables.
dep	A string with the name of the independent variable or a column number.
indep	A vector with the names or column numbers of the regressors. If left unspecified, all remaining variables (excluding fixed effects) are included in the regressor matrix.
fixed	A vector with the names or column numbers of factor variables identifying the fixed effects, or a list with the desired interactions between variables in data.
cluster	Optional. A string with the name of the clustering variable or a column number. It's also possible to input a vector with several variables, in which case the interaction of all of them is taken as the clustering variable.
selectobs	Optional. A vector indicating which observations to use (either a logical vector or a numeric vector with row numbers, as usual when subsetting in R).
...	Further arguments, including: <ul style="list-style-type: none"> <li>• penalty: A string indicating the penalty type. Currently supported: "lasso" and "ridge".</li> </ul>

- `method`: The user can set this equal to "plugin" to perform the plugin algorithm with coefficient-specific penalty weights (see details). Otherwise, a single global penalty is used.
- `post`: Logical. If TRUE, estimates a post-penalty regression with the selected variables.
- `xval`: Logical. If TRUE, cross-validation is performed using the IDs provided in the `IDs` argument as folds. Note that, by default, observations are assigned individual IDs, which makes the cross-validation algorithm very time-consuming.

For a full list of options, see [mlfitppml\\_int](#).

## Details

This function is a thin wrapper around `mlfitppml_int`, providing a more convenient interface for data frames. Whereas the internal function requires some preliminary handling of data sets (`y` must be a vector, `x` must be a matrix and `fes` must be provided in a list), the wrapper takes a full data frame in the `data` argument, and users can simply specify which variables correspond to `y`, `x` and the fixed effects, using either variable names or column numbers.

For technical details on the algorithms used, see [hdfepml](#) (post-lasso regression), [penhdfepml](#) (standard penalized regression), [penhdfepml\\_cluster](#) (plugin lasso), and [xvalidate](#) (cross-validation).

## Value

A list with the following elements:

- `beta`: if `post = FALSE`, a  $\text{length}(\text{lambdas}) \times \text{ncol}(x)$  matrix with coefficient (beta) estimates from the penalized regressions. If `post = TRUE`, this is the matrix of coefficients from the post-penalty regressions.
- `beta_pre`: if `post = TRUE`, a  $\text{length}(\text{lambdas}) \times \text{ncol}(x)$  matrix with coefficient (beta) estimates from the penalized regressions.
- `bic`: Bayesian Information Criterion.
- `lambdas`: vector of penalty parameters.
- `ses`: standard errors of the coefficients of the post-penalty regression. Note that these are only provided when `post = TRUE`.
- `rmse`: if `xval = TRUE`, a matrix with the root mean squared error (RMSE - column 2) for each value of `lambda` (column 1), obtained by cross-validation.
- `phi`: coefficient-specific penalty weights (only if `method == "plugin"`).

## References

Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.

Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.

Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.

Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.

Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.

## Examples

```
## Not run:
# To reduce run time, we keep only countries in the Americas:
americas <- countries$iso[countries$region == "Americas"]
# Now we can use our main functions on the reduced trade data set:
test <- mlfitppml(data = trade[, -(5:6)],
  dep = "export",
  fixed = list(c("exp", "time"),
              c("imp", "time"),
              c("exp", "imp")),
  selectobs = (trade$imp %in% americas) & (trade$exp %in% americas),
  lambdas = c(0.01, 0.001),
  tol = 1e-6, hdfetol = 1e-2)

## End(Not run)
```

---

mlfitppml\_int

*General Penalized PPML Estimation*


---

## Description

mlfitppml\_int is the internal wrapper called by mlfitppml for penalized PPML estimation. This in turn calls penhdfepml\_int, penhdfepml\_cluster\_int and hdfepml\_int as needed. It takes a vector with the dependent variable, a regressor matrix and a set of fixed effects (in list form: each element in the list should be a separate HDFE). This is a flexible tool that allows users to select:

- Penalty type: either lasso or ridge.
- Penalty parameter: users can provide a single global value for lambda (a single regression is estimated), a vector of lambda values (the function estimates the regression using each of them, sequentially) or even coefficient-specific penalty weights.
- Method: plugin lasso estimates can be obtained directly from this function too.
- Cross-validation: if this option is enabled, the function uses IDs provided by the user to perform k-fold cross-validation and reports the resulting RMSE for all lambda values.

**Usage**

```
mlfitppml_int(
  y,
  x,
  fes,
  lambdas,
  penalty = "lasso",
  tol = 1e-08,
  hdfetol = 1e-04,
  colcheck = TRUE,
  colcheck_x = colcheck,
  colcheck_x_fes = colcheck,
  post = TRUE,
  cluster = NULL,
  method = "bic",
  IDs = 1:n,
  verbose = FALSE,
  xval = FALSE,
  standardize = TRUE,
  vcv = TRUE,
  phipost = TRUE,
  penweights = NULL,
  K = 15,
  gamma_val = NULL,
  mu = NULL
)
```

**Arguments**

y	Dependent variable (a vector)
x	Regressor matrix.
fes	List of fixed effects.
lambdas	Vector of penalty parameters.
penalty	A string indicating the penalty type. Currently supported: "lasso" and "ridge".
tol	Tolerance parameter for convergence of the IRLS algorithm.
hdfetol	Tolerance parameter for the within-transformation step, passed on to <code>collapse::fhdwithin</code> .
colcheck	Logical. If TRUE, performs both checks in <code>colcheck_x</code> and <code>colcheck_x_fes</code> . If the user specifies <code>colcheck_x</code> and <code>colcheck_x_fes</code> individually, this option is overwritten.
colcheck_x	Logical. If TRUE, this checks collinearity between the independent variables and drops the collinear variables.
colcheck_x_fes	Logical. If TRUE, this checks whether the independent variables are perfectly explained by the fixed effects drops those that are perfectly explained.
post	Logical. If TRUE, estimates a post-penalty regression with the selected variables.

cluster	Optional: a vector classifying observations into clusters (to use when calculating SEs).
method	The user can set this equal to "plugin" to perform the plugin algorithm with coefficient-specific penalty weights (see details). Otherwise, a single global penalty is used.
IDs	A vector of fold IDs for k-fold cross validation. If left unspecified, each observation is assigned to a different fold (warning: this is likely to be very resource-intensive).
verbose	Logical. If TRUE, it prints information to the screen while evaluating.
xval	Logical. If TRUE, it carries out cross-validation.
standardize	Logical. If TRUE, x variables are standardized before estimation.
vcv	Logical. If TRUE (the default), the post-estimation model includes standard errors.
phipost	Logical. If TRUE, the plugin coefficient-specific penalty weights are iteratively calculated using estimates from a post-penalty regression when method == "plugin". Otherwise, these are calculated using estimates from a penalty regression.
penweights	Optional: a vector of coefficient-specific penalties to use in plugin lasso when method == "plugin".
K	Maximum number of iterations for the plugin algorithm to converge.
gamma_val	Numerical value that determines the regularization threshold as defined in Belloni, Chernozhukov, Hansen, and Kozbur (2016). NULL default sets parameter to $0.1/\log(n)$ .
mu	A vector of initial values for mu that can be passed to the command.

## Details

For technical details on the algorithms used, see [hdfeppml\\_int](#) (post-lasso regression), [penhdfeppml\\_int](#) (standard penalized regression), [penhdfeppml\\_cluster\\_int](#) (plugin lasso), and [xvalidate](#) (cross-validation).

## Value

A list with the following elements:

- **beta**: if post = FALSE, a  $\text{length}(\text{lambdas}) \times \text{ncol}(x)$  matrix with coefficient (beta) estimates from the penalized regressions. If post = TRUE, this is the matrix of coefficients from the post-penalty regressions.
- **beta\_pre**: if post = TRUE, a  $\text{length}(\text{lambdas}) \times \text{ncol}(x)$  matrix with coefficient (beta) estimates from the penalized regressions.
- **bic**: Bayesian Information Criterion.
- **lambdas**: vector of penalty parameters.
- **ses**: standard errors of the coefficients of the post-penalty regression. Note that these are only provided when post = TRUE.
- **rmse**: if xval = TRUE, a matrix with the root mean squared error (RMSE - column 2) for each value of lambda (column 1), obtained by cross-validation.
- **phi**: coefficient-specific penalty weights (only if method == "plugin").

## References

- Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.
- Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.
- Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.
- Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.
- Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.

## Examples

```
## Not run:
# First, we need to transform the data (this is what mlfitppml handles internally). Start by
# filtering the data set to keep only countries in the Americas:
americas <- countries$iso[countries$region == "Americas"]
trade <- trade[(trade$imp %in% americas) & (trade$exp %in% americas), ]
# Now generate the needed x, y and fes objects:
y <- trade$export
x <- data.matrix(trade[, -1:-6])
fes <- list(exp_time = interaction(trade$exp, trade$time),
           imp_time = interaction(trade$imp, trade$time),
           pair      = interaction(trade$exp, trade$imp))
# Finally, we try mlfitppml_int with a lasso penalty (the default) and two lambda values:
reg <- mlfitppml_int(y = y, x = x, fes = fes, lambdas = c(0.1, 0.01))

# We can also try plugin lasso:
\donttest{reg <- mlfitppml_int(y = y, x = x, fes = fes, cluster = fes$pair, method = "plugin")}

# For an example with cross-validation, please see the vignette.

## End(Not run)
```

## Description

penhdfepml fits a penalized PPML regression for a given type of penalty and a given value of the penalty parameter. The penalty can be either lasso or ridge, and the plugin method can be enabled via the method argument.



**Usage**

```
penhdfepml(
  data,
  dep = 1,
  indep = NULL,
  fixed = NULL,
  cluster = NULL,
  selectobs = NULL,
  ...
)
```

**Arguments**

<code>data</code>	A data frame containing all relevant variables.
<code>dep</code>	A string with the name of the independent variable or a column number.
<code>indep</code>	A vector with the names or column numbers of the regressors. If left unspecified, all remaining variables (excluding fixed effects) are included in the regressor matrix.
<code>fixed</code>	A vector with the names or column numbers of factor variables identifying the fixed effects, or a list with the desired interactions between variables in data.
<code>cluster</code>	Optional. A string with the name of the clustering variable or a column number. It's also possible to input a vector with several variables, in which case the interaction of all of them is taken as the clustering variable.
<code>selectobs</code>	Optional. A vector indicating which observations to use (either a logical vector or a numeric vector with row numbers, as usual when subsetting in R).
<code>...</code>	Further options, including: <ul style="list-style-type: none"> <li>• <code>penalty</code>: A string indicating the penalty type. Currently supported: "lasso" and "ridge".</li> <li>• <code>method</code>: The user can set this equal to "plugin" to perform the plugin algorithm with coefficient-specific penalty weights (see details). Otherwise, a single global penalty is used.</li> </ul>

For a full list of options, see [penhdfepml\\_int](#).

**Details**

This function is a thin wrapper around [penhdfepml\\_int](#), providing a more convenient interface for data frames. Whereas the internal function requires some preliminary handling of data sets (`y` must be a vector, `x` must be a matrix and `fes` must be provided in a list), the wrapper takes a full data frame in the `data` argument, and users can simply specify which variables correspond to `y`, `x` and the fixed effects, using either variable names or column numbers.

More formally, `penhdfepml_int` performs iteratively re-weighted least squares (IRLS) on a transformed model, as described in Breinlich, Corradi, Rocha, Ruta, Santos Silva and Zylkin (2021). In each iteration, the function calculates the transformed dependent variable, partials out the fixed effects (calling `lfe::fhdwithin`) and then and then calls `glmnet::glmnet` if the selected penalty is lasso (the default). If the user has selected ridge, the analytical solution is instead computed directly using fast C++ implementation.

For information on how the plugin lasso method works, see [penhdfepml\\_cluster](#).

### Value

If `method == "lasso"` (the default), an object of class `elnet` with the elements described in [glmnet](#), as well as:

- `mu`: a  $1 \times \text{length}(y)$  matrix with the final values of the conditional mean  $\mu$ .
- `deviance`.
- `bic`: Bayesian Information Criterion.
- `phi`: coefficient-specific penalty weights (only if `method == "plugin"`).
- `x_resid`: matrix of demeaned regressors.
- `z_resid`: vector of demeaned (transformed) dependent variable.

If `method == "ridge"`, a list with the following elements:

- `beta`: a  $1 \times \text{ncol}(x)$  matrix with coefficient (beta) estimates.
- `mu`: a  $1 \times \text{length}(y)$  matrix with the final values of the conditional mean  $\mu$ .
- `deviance`.
- `bic`: Bayesian Information Criterion.
- `x_resid`: matrix of demeaned regressors.
- `z_resid`: vector of demeaned (transformed) dependent variable.

### References

- Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.
- Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.
- Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.
- Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.
- Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.

### Examples

```
## Not run:
# To reduce run time, we keep only countries in the Americas:
americas <- countries$iso[countries$region == "Americas"]
test <- penhdfepml(data = trade[, -(5:6)],
                  dep = "export",
                  fixed = list(c("exp", "time"),
```

```

                                c("imp", "time"),
                                c("exp", "imp")),
lambda = 0.05,
selectobs = (trade$imp %in% americas) & (trade$exp %in% americas))

## End(Not run)

```

---

penhdfepml\_cluster     *Plugin Lasso Estimation*

---

## Description

Performs plugin lasso - PPML estimation with HDFE. This is an internal function, called by `mlfitppml` and `penhdfepml` when users select the `method = "plugin"` option, but it's made available as a stand-alone option for advanced users who may prefer to avoid some overhead imposed by the wrappers.

## Usage

```

penhdfepml_cluster(
  data,
  dep = 1,
  indep = NULL,
  fixed = NULL,
  cluster = NULL,
  selectobs = NULL,
  ...
)

```

## Arguments

<code>data</code>	A data frame containing all relevant variables.
<code>dep</code>	A string with the name of the independent variable or a column number.
<code>indep</code>	A vector with the names or column numbers of the regressors. If left unspecified, all remaining variables (excluding fixed effects) are included in the regressor matrix.
<code>fixed</code>	A vector with the names or column numbers of factor variables identifying the fixed effects, or a list with the desired interactions between variables in <code>data</code> .
<code>cluster</code>	A string with the name of the clustering variable or a column number. It's also possible to input a vector with several variables, in which case the interaction of all of them is taken as the clustering variable. Note that this is <b>NOT OPTIONAL</b> in this case: our plugin algorithm requires clusters to be specified.
<code>selectobs</code>	Optional. A vector indicating which observations to use (either a logical vector or a numeric vector with row numbers, as usual when subsetting in R).
<code>...</code>	Further options. For a full list of options, see <a href="#">penhdfepml_cluster_int</a> .

## Details

This function is a thin wrapper around `penppml_cluster_int`, providing a more convenient interface for data frames. Whereas the internal function requires some preliminary handling of data sets (`y` must be a vector, `x` must be a matrix and `fes` must be provided in a list), the wrapper takes a full data frame in the `data` argument, and users can simply specify which variables correspond to `y`, `x` and the fixed effects, using either variable names or column numbers.

The plugin method uses coefficient-specific penalty weights that account for heteroskedasticity. The penalty parameters are calculated automatically by the function using statistical theory - for a brief discussion of this, see Breinlich, Corradi, Rocha, Ruta, Santos Silva and Zylkin (2021), and for a more in-depth analysis, check Belloni, Chernozhukov, Hansen, and Kozbur (2016), which introduced the specific implementation used in this package. Heuristically, the penalty parameters are set at a level high enough so that the absolute value of the score for each regressor must be statistically large relative to its standard error in order for the regressors to be selected.

## Value

An object of class `elnet` with the elements described in [glmnet](#), as well as the following:

- `mu`: a  $1 \times \text{length}(y)$  matrix with the final values of the conditional mean  $\mu$ .
- `deviance`.
- `bic`: Bayesian Information Criterion.
- `phi`: coefficient-specific penalty weights.
- `x_resid`: matrix of demeaned regressors.
- `z_resid`: vector of demeaned (transformed) dependent variable.

## References

- Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.
- Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.
- Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.
- Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.
- Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.

## Examples

```
## Not run:
# To reduce run time, we keep only countries in the Americas:
americas <- countries$iso[countries$region == "Americas"]
test <- penhdfepml_cluster(data = trade[, -(5:6)],
```

```

dep = "export",
fixed = list(c("exp", "time"),
            c("imp", "time"),
            c("exp", "imp")),
cluster = c("exp", "imp"),
selectobs = (trade$imp %in% americas) & (trade$exp %in% americas),
tol = 1e-5, hdfetol = 1e-1)

## End(Not run)

```

---

penhdfepml\_cluster\_int

*Plugin Lasso Estimation*


---

### Description

Performs plugin lasso - PPML estimation with HDFE. This is an internal function, called by `mlfitppml_int` and `penhdfepml_int` when users select the `method = "plugin"` option, but it's made available as a stand-alone option for advanced users who may prefer to avoid some overhead imposed by the wrappers.

### Usage

```

penhdfepml_cluster_int(
  y,
  x,
  fes,
  cluster,
  tol = 1e-08,
  hdfetol = 1e-04,
  glmnettol = 1e-12,
  penalty = "lasso",
  penweights = NULL,
  saveX = TRUE,
  mu = NULL,
  colcheck = TRUE,
  colcheck_x = colcheck,
  colcheck_x_fes = colcheck,
  K = 15,
  init_z = NULL,
  post = FALSE,
  verbose = FALSE,
  lambda = NULL,
  phipost = TRUE,
  gamma_val = NULL
)

```

**Arguments**

<code>y</code>	Dependent variable (a vector)
<code>x</code>	Regressor matrix.
<code>fes</code>	List of fixed effects.
<code>cluster</code>	Optional: a vector classifying observations into clusters (to use when calculating SEs).
<code>tol</code>	Tolerance parameter for convergence of the IRLS algorithm.
<code>hdfetol</code>	Tolerance parameter for the within-transformation step, passed on to <code>collapse::fhwithin</code> .
<code>glmnettol</code>	Tolerance parameter to be passed on to <code>glmnet</code> .
<code>penalty</code>	Only "lasso" is supported at the present stage.
<code>penweights</code>	Optional: a vector of coefficient-specific penalties to use in plugin lasso when <code>method == "plugin"</code> .
<code>saveX</code>	Logical. If TRUE, it returns the values of <code>x</code> and <code>z</code> after partialling out the fixed effects.
<code>mu</code>	A vector of initial values for <code>mu</code> that can be passed to the command.
<code>colcheck</code>	Logical. If TRUE, performs both checks in <code>colcheck_x</code> and <code>colcheck_x_fes</code> . If the user specifies <code>colcheck_x</code> and <code>colcheck_x_fes</code> individually, this option is overwritten.
<code>colcheck_x</code>	Logical. If TRUE, this checks collinearity between the independent variables and drops the collinear variables.
<code>colcheck_x_fes</code>	Logical. If TRUE, this checks whether the independent variables are perfectly explained by the fixed effects drops those that are perfectly explained.
<code>K</code>	Maximum number of iterations.
<code>init_z</code>	Optional: initial values of the transformed dependent variable, to be used in the first iteration of the algorithm.
<code>post</code>	Logical. If TRUE, estimates a post-penalty regression with the selected variables.
<code>verbose</code>	Logical. If TRUE, it prints information to the screen while evaluating.
<code>lambda</code>	Penalty parameter (a number).
<code>phipost</code>	Logical. If TRUE, the plugin coefficient-specific penalty weights are iteratively calculated using estimates from a post-penalty regression. Otherwise, these are calculated using estimates from a penalty regression.
<code>gamma_val</code>	Numerical value that determines the regularization threshold as defined in Belloni, Chernozhukov, Hansen, and Kozbur (2016). NULL default sets parameter to $0.1/\log(n)$ .

**Details**

The plugin method uses coefficient-specific penalty weights that account for heteroskedasticity. The penalty parameters are calculated automatically by the function using statistical theory - for a brief discussion of this, see Breinlich, Corradi, Rocha, Ruta, Santos Silva and Zylkin (2021), and for a more in-depth analysis, check Belloni, Chernozhukov, Hansen, and Kozbur (2016), which introduced the specific implementation used in this package. Heuristically, the penalty parameters are set at a level high enough so that the absolute value of the score for each regressor must be statistically large relative to its standard error in order for the regressors to be selected.

**Value**

An object of class `elnet` with the elements described in [glmnet](#), as well as the following:

- `mu`: a  $1 \times \text{length}(y)$  matrix with the final values of the conditional mean  $\mu$ .
- `deviance`.
- `bic`: Bayesian Information Criterion.
- `phi`: coefficient-specific penalty weights.
- `x_resid`: matrix of demeaned regressors.
- `z_resid`: vector of demeaned (transformed) dependent variable.

**References**

Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.

Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.

Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.

Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.

Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.

**Examples**

```
## Not run:
# To reduce run time, we keep only countries in Latin America and the Caribbean:
LatAmericaCar <- countries$iso[countries$sub.region == "Latin America and the Caribbean"]
trade <- trade[(trade$imp %in% LatAmericaCar) & (trade$exp %in% LatAmericaCar), ]
# Now generate the needed x, y and fes objects:
y <- trade$export
x <- data.matrix(trade[, -1:-6])
fes <- list(exp_time = interaction(trade$exp, trade$time),
           imp_time = interaction(trade$imp, trade$time),
           pair      = interaction(trade$exp, trade$imp))
# Finally, we try penhdfepml_cluster_int:
reg <- penhdfepml_cluster_int(y = y, x = x, fes = fes, cluster = fes$pair)

## End(Not run)
```

---

 penhdfepml\_int

*One-Shot Penalized PPML Estimation with HDFE*


---

### Description

penhdfepml\_int is the internal algorithm called by penhdfepml to fit a penalized PPML regression for a given type of penalty and a given value of the penalty parameter. It takes a vector with the dependent variable, a regressor matrix and a set of fixed effects (in list form: each element in the list should be a separate HDFE). The penalty can be either lasso or ridge, and the plugin method can be enabled via the method argument.

### Usage

```
penhdfepml_int(
  y,
  x,
  fes,
  lambda,
  tol = 1e-08,
  hdfetol = 1e-04,
  glmnettol = 1e-12,
  penalty = "lasso",
  penweights = NULL,
  saveX = TRUE,
  mu = NULL,
  colcheck = TRUE,
  colcheck_x = colcheck,
  colcheck_x_fes = colcheck,
  init_z = NULL,
  post = FALSE,
  verbose = FALSE,
  phipost = TRUE,
  standardize = TRUE,
  method = "placeholder",
  cluster = NULL,
  debug = FALSE,
  gamma_val = NULL
)
```

### Arguments

y	Dependent variable (a vector)
x	Regressor matrix.
fes	List of fixed effects.
lambda	Penalty parameter (a number).
tol	Tolerance parameter for convergence of the IRLS algorithm.



hdfetol	Tolerance parameter for the within-transformation step, passed on to <code>collapse::fhdwithin</code> .
glmnettol	Tolerance parameter to be passed on to <code>glmnet</code> .
penalty	A string indicating the penalty type. Currently supported: "lasso" and "ridge".
penweights	Optional: a vector of coefficient-specific penalties to use in plugin lasso when <code>method == "plugin"</code> .
saveX	Logical. If TRUE, it returns the values of x and z after partialling out the fixed effects.
mu	A vector of initial values for mu that can be passed to the command.
colcheck	Logical. If TRUE, performs both checks in <code>colcheck_x</code> and <code>colcheck_x_fes</code> . If the user specifies <code>colcheck_x</code> and <code>colcheck_x_fes</code> individually, this option is overwritten.
colcheck_x	Logical. If TRUE, this checks collinearity between the independent variables and drops the collinear variables.
colcheck_x_fes	Logical. If TRUE, this checks whether the independent variables are perfectly explained by the fixed effects drops those that are perfectly explained.
init_z	Optional: initial values of the transformed dependent variable, to be used in the first iteration of the algorithm.
post	Logical. If TRUE, estimates a post-penalty regression with the selected variables.
verbose	Logical. If TRUE, it prints information to the screen while evaluating.
hipost	Logical. If TRUE, the plugin coefficient-specific penalty weights are iteratively calculated using estimates from a post-penalty regression when <code>method == "plugin"</code> . Otherwise, these are calculated using estimates from a penalty regression.
standardize	Logical. If TRUE, x variables are standardized before estimation.
method	The user can set this equal to "plugin" to perform the plugin algorithm with coefficient-specific penalty weights (see details). Otherwise, a single global penalty is used.
cluster	Optional: a vector classifying observations into clusters (to use when calculating SEs).
debug	Logical. If TRUE, this helps with debugging penalty weights by printing output of the first iteration to the console and stopping the estimation algorithm.
gamma_val	Numerical value that determines the regularization threshold as defined in Belloni, Chernozhukov, Hansen, and Kozbur (2016). NULL default sets parameter to $0.1/\log(n)$ .

## Details

More formally, `penhdfepml_int` performs iteratively re-weighted least squares (IRLS) on a transformed model, as described in Breinlich, Corradi, Rocha, Ruta, Santos Silva and Zylkin (2020). In each iteration, the function calculates the transformed dependent variable, partials out the fixed effects (calling `collapse::fhdwithin`) and then and then calls `glmnet` if the selected penalty is lasso (the default). If the user selects ridge, the analytical solution is instead computed directly using fast C++ implementation.

For information on the plugin lasso method, see [penhdfepml\\_cluster\\_int](#).

**Value**

If `method == "lasso"` (the default), an object of class `elnet` with the elements described in [glmnet](#), as well as:

- `mu`: a  $1 \times \text{length}(y)$  matrix with the final values of the conditional mean  $\mu$ .
- `deviance`.
- `bic`: Bayesian Information Criterion.
- `phi`: coefficient-specific penalty weights (only if `method == "plugin"`).
- `x_resid`: matrix of demeaned regressors.
- `z_resid`: vector of demeaned (transformed) dependent variable.

If `method == "ridge"`, a list with the following elements:

- `beta`: a  $1 \times \text{ncol}(x)$  matrix with coefficient (beta) estimates.
- `mu`: a  $1 \times \text{length}(y)$  matrix with the final values of the conditional mean  $\mu$ .
- `deviance`.
- `bic`: Bayesian Information Criterion.
- `x_resid`: matrix of demeaned regressors.
- `z_resid`: vector of demeaned (transformed) dependent variable.

**References**

- Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.
- Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.
- Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.
- Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.
- Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.

**Examples**

```
## Not run:
# To reduce run time, we keep only countries in the Americas:
americas <- countries$iso[countries$region == "Americas"]
trade <- trade[(trade$imp %in% americas) & (trade$exp %in% americas), ]
# Now generate the needed x, y and fes objects:
y <- trade$export
x <- data.matrix(trade[, -1:-6])
fes <- list(exp_time = interaction(trade$exp, trade$time),
           imp_time = interaction(trade$imp, trade$time),
```

```

      pair      = interaction(trade$exp, trade$imp))
# Finally, we try penhdfeppl_int with a lasso penalty (the default):
reg <- penhdfeppl_int(y = y, x = x, fes = fes, lambda = 0.1)

# We can also try ridge:
\donttest{reg <- penhdfeppl_int(y = y, x = x, fes = fes, lambda = 0.1, penalty = "ridge")}

## End(Not run)

```

---

plugin\_lasso\_int      *Iceberg Lasso Implementation (in development)*

---

## Description

This is the internal function upon which the iceberg wrapper is built. It performs standard plugin lasso PPML estimation without fixed effects, relying on `glmnet::glmnet`. As the other internals in the package, it needs a y vector and an x matrix.

## Usage

```

plugin_lasso_int(
  y,
  x,
  tol = 1e-08,
  glmnettol = 1e-12,
  penweights = NULL,
  colcheck = FALSE,
  K = 50,
  verbose = FALSE,
  lambda = NULL,
  icepost = FALSE
)

```

## Arguments

y	Dependent variable (a vector).
x	Regressor matrix.
tol	Tolerance parameter for convergence of the IRLS algorithm.
glmnettol	Tolerance parameter to be passed on to <code>glmnet::glmnet</code> .
penweights	Optional: a vector of coefficient-specific penalties to use in plugin lasso.
colcheck	Logical. If TRUE, checks for perfect multicollinearity in x.
K	Maximum number of iterations.
verbose	Logical. If TRUE, prints information to the screen while evaluating.
lambda	Penalty parameter (a number).
icepost	Logical. If TRUE, it carries out a post-lasso estimation with just the selected variables and reports the coefficients from this regression.

**Value**

A list with 14 elements, including beta, which is the only one we use in the wrapper. For a full list, see [glmnet](#).

---

select_fes	<i>Filtering fixed effect lists</i>
------------	-------------------------------------

---

**Description**

A helper function for `xvalidate` that filters a list of fixed effects and returns the modified list. Used to split the fixed effects for cross-validation.

**Usage**

```
select_fes(fe_list, select_obs, list = TRUE)
```

**Arguments**

<code>fe_list</code>	A list of fixed effects.
<code>select_obs</code>	A vector of selected observations / rows.
<code>list</code>	Logical. If TRUE, it returns a list. Otherwise, a data frame.

**Value**

A modified list of fixed effects.

---

standardize_wt	<i>Weighted Standardization</i>
----------------	---------------------------------

---

**Description**

Performs weighted standardization of x variables. Used in `fastridge`.

**Usage**

```
standardize_wt(x, weights = rep(1/n, n), intercept = TRUE, return.sd = FALSE)
```

**Arguments**

<code>x</code>	Regressor matrix.
<code>weights</code>	Weights.
<code>intercept</code>	Logical. If TRUE, adds an intercept.
<code>return.sd</code>	Logical. If TRUE, it returns standard errors for the means.

**Value**

If `return.sd == FALSE`, it gives the matrix of standardized regressors. If `return.sd == TRUE`, then it returns the vector of standard errors of the means of the variables.

---

trade	<i>International trade agreements data set</i>
-------	--

---

**Description**

A panel data set containing bilateral trade flows between 210 exporters and 262 importers between 1964 and 2016. The data set also contains information about trade agreements in force between country pairs, as well as 16 dummies for specific provisions in those agreements (a small selection from a broader data set).

**Usage**

```
trade
```

**Format**

A data frame with 194,092 rows and 22 variables:

**exp** Exporter country (ISO 3166 code)

**imp** Importer country (ISO 3166 code).

**time** Year.

**export** Merchandise trade exports in USD.

**id** Agreement ID code.

**agreement** Agreement name.

**ad\_prov\_14** Anti-dumping actions allowed and with specific provisions for material injury.

**cp\_prov\_23** Does the agreement contain provisions that promote transparency?

**tbt\_prov\_07** Technical Regulations - Is the use of international standards promoted?

**tbt\_prov\_33** Does the agreement go beyond the TBT (Technical Barriers to Trade) Agreement?

**tf\_prov\_41** Harmonization and common legal framework

**tf\_prov\_45** Issuance of proof of origin

**ser\_prov\_47** Does the agreement contain a standstill provision?

**inv\_prov\_22** Does the agreement grant Fair and Equitable Treatment (FET)?

**et\_prov\_38** Prohibits export-related performance requirements, subject to exemptions.

**ipr\_prov\_44** Stipulates that GIs can be registered and protected through a TM system

**env\_prov\_18** Does the agreement require states to control ozone-depleting substances?

**ipr\_prov\_15** Incorporates/reaffirms all multilateral agreements to which both parties are a party (general obligation)

**moc\_prov\_21** Does the transfer provision explicitly exclude “good faith and non-discriminatory application of its laws” related to bankruptcy, insolvency or creditor rights protection?

**ste\_prov\_30** Does the agreement regulate subsidization to state enterprises?

**lm\_prov\_10** Does the agreement include reference to internationally recognized labor standards?

**cp\_prov\_26** Does the agreement regulate consumer protection?

### Source

Data on international trade flows was obtained from Comtrade. Provision data comes from: Mattoo, A., N. Rocha, M. Ruta (2020). Handbook of deep trade agreements. Washington, DC: World Bank.

---

xeex

*XeeX Matrix Computation*

---

### Description

Given matrix  $ee'$  and matrix  $X$ , compute  $X(k)'ee'X(k)$  for each regressor  $X$ .

### Usage

`xeex(X, e, S)`

### Arguments

$X$	Regressor matrix.
$e$	Residuals.
$S$	Cluster sizes.

### Value

The matrix product  $X(k)'ee'X(k)$ .

---

xvalidate

*Implementing Cross Validation*

---

### Description

This is the internal function called by `mlfitppml_int` to perform cross-validation, if the option is enabled. It is available also on a stand-alone basis in case it is needed, but generally users will be better served by using the wrapper `mlfitppml`.

**Usage**

```
xvalidate(
  y,
  x,
  fes,
  IDs,
  testID = NULL,
  tol = 1e-08,
  hdfetol = 1e-04,
  colcheck_x = TRUE,
  colcheck_x_fes = TRUE,
  init_mu = NULL,
  init_x = NULL,
  init_z = NULL,
  verbose = FALSE,
  cluster = NULL,
  penalty = "lasso",
  method = "placeholder",
  standardize = TRUE,
  penweights = rep(1, ncol(x_reg)),
  lambda = 0
)
```

**Arguments**

y	Dependent variable (a vector)
x	Regressor matrix.
fes	List of fixed effects.
IDs	A vector of fold IDs for k-fold cross validation. If left unspecified, each observation is assigned to a different fold (warning: this is likely to be very resource-intensive).
testID	Optional. A number indicating which ID to hold out during cross-validation. If left unspecified, the function cycles through all IDs and reports the average RMSE.
tol	Tolerance parameter for convergence of the IRLS algorithm.
hdfetol	Tolerance parameter for the within-transformation step, passed on to <code>collapse::fhdwthin</code> .
colcheck_x	Logical. If TRUE, this checks collinearity between the independent variables and drops the collinear variables.
colcheck_x_fes	Logical. If TRUE, this checks whether the independent variables are perfectly explained by the fixed effects drops those that are perfectly explained.
init_mu	Optional: initial values of the conditional mean $\mu$ , to be used as weights in the first iteration of the algorithm.
init_x	Optional: initial values of the independent variables.
init_z	Optional: initial values of the transformed dependent variable, to be used in the first iteration of the algorithm.

verbose	Logical. If TRUE, it prints information to the screen while evaluating.
cluster	Optional: a vector classifying observations into clusters (to use when calculating SEs).
penalty	A string indicating the penalty type. Currently supported: "lasso" and "ridge".
method	The user can set this equal to "plugin" to perform the plugin algorithm with coefficient-specific penalty weights (see details). Otherwise, a single global penalty is used.
standardize	Logical. If TRUE, x variables are standardized before estimation.
penweights	Optional: a vector of coefficient-specific penalties to use in plugin lasso when method == "plugin".
lambda	Penalty parameter, to be passed on to <code>penhdfepml_int</code> or <code>penhdfepml_cluster_int</code> .

### Details

`xvalidate` carries out cross-validation with the user-provided IDs by holding out each one of them, sequentially, as in the k-fold procedure (unless `testID` is specified, in which case it just uses this ID for validation). After filtering out the holdout sample, the function simply calls `penhdfepml_int` and `penhdfepml_cluster_int` to estimate the coefficients, it predicts the conditional means for the held-out observations and finally it calculates the root mean squared error (RMSE).

### Value

A list with two elements:

- `rmse`: root mean squared error (RMSE).
- `mu`: conditional means.

### References

- Breinlich, H., Corradi, V., Rocha, N., Ruta, M., Santos Silva, J.M.C. and T. Zylkin (2021). "Machine Learning in International Trade Research: Evaluating the Impact of Trade Agreements", Policy Research Working Paper; No. 9629. World Bank, Washington, DC.
- Correia, S., P. Guimaraes and T. Zylkin (2020). "Fast Poisson estimation with high dimensional fixed effects", *STATA Journal*, 20, 90-115.
- Gaure, S (2013). "OLS with multiple high dimensional category variables", *Computational Statistics & Data Analysis*, 66, 8-18.
- Friedman, J., T. Hastie, and R. Tibshirani (2010). "Regularization paths for generalized linear models via coordinate descent", *Journal of Statistical Software*, 33, 1-22.
- Belloni, A., V. Chernozhukov, C. Hansen and D. Kozbur (2016). "Inference in high dimensional panel models with an application to gun control", *Journal of Business & Economic Statistics*, 34, 590-605.



**Examples**

```
# First, we need to transform the data. Start by filtering the data set to keep only countries in
# the Americas:
americas <- countries$iso[countries$region == "Americas"]
trade <- trade[(trade$imp %in% americas) & (trade$exp %in% americas), ]
# Now generate the needed x, y and fes objects:
y <- trade$export
x <- data.matrix(trade[, -1:-6])
fes <- list(exp_time = interaction(trade$exp, trade$time),
           imp_time = interaction(trade$imp, trade$time),
           pair      = interaction(trade$exp, trade$imp))
# We also need to create the IDs. We split the data set by agreement, not observation:
id <- unique(trade[, 5])
nfolds <- 10
unique_ids <- data.frame(id = id, fold = sample(1:nfolds, size = length(id), replace = TRUE))
cross_ids <- merge(trade[, 5, drop = FALSE], unique_ids, by = "id", all.x = TRUE)
# Finally, we try xvalidate with a lasso penalty (the default) and two lambda values:
## Not run: reg <- xvalidate(y = y, x = x, fes = fes, lambda = 0.001,
                          IDs = cross_ids$fold, verbose = TRUE)
## End(Not run)
```

# Index

- \* **datasets**
  - countries, [7](#)
  - trade, [37](#)
- AtA, [3](#)
- bootstrap, [3](#)
- cluster\_matrix, [5](#)
- collinearity\_check, [6](#)
- compute\_fes, [6](#)
- countries, [7](#)
- eigenMapMatMult (eigenMatMult), [8](#)
- eigenMatMult, [8](#)
- fastolsCpp, [8](#)
- fastridge, [9](#)
- fastridgeCpp, [9](#)
- faststddev, [10](#)
- fastwmean, [10](#)
- genfes, [11](#)
- genmodel, [11](#)
- glmnet, [26](#), [28](#), [31](#), [34](#), [36](#)
- hdfepml, [12](#), [20](#)
- hdfepml\_int, [13](#), [14](#), [23](#)
- iceberg, [17](#)
- manyouter, [18](#)
- mlfitppml, [19](#)
- mlfitppml\_int, [20](#), [21](#)
- penhdfepml, [20](#), [24](#)
- penhdfepml\_cluster, [20](#), [26](#), [27](#)
- penhdfepml\_cluster\_int, [23](#), [27](#), [29](#), [33](#),  
[40](#)
- penhdfepml\_int, [23](#), [25](#), [32](#), [40](#)
- plugin\_lasso\_int, [35](#)
- select\_fes, [36](#)
- standardize\_wt, [36](#)
- trade, [37](#)
- xeex, [38](#)
- xvalidate, [20](#), [23](#), [38](#)