Package 'capybara'

June 15, 2026

Type Package

Title Fast and Memory Efficient Fitting of Linear Models with High-Dimensional Fixed Effects

Version 2.0.0

Description Fast and user-friendly estimation of generalized linear models with multiple fixed effects and cluster the standard errors. The method to obtain the estimated fixed-effects coefficients is based on Stammann (2018) <doi:10.48550/arXiv.1707.01815>, Gaure (2013) <doi:10.1016/j.csda.2013.03.024>, Berge (2018) <https://ideas.repec.org/p/luc/wpaper/18-13.html>, and Correia et al. (2020) <doi:10.1177/1536867X20909691>. This implementation is described in Vargas Sepulveda (2025) <doi:10.1371/journal.pone.0331178>.

License Apache License (>= 2)

URL https://pacha.dev/capybara/, https://github.com/pachadotdev/capybara

BugReports https://github.com/pachadotdev/capybara/issues

Depends R (>= 4.0.0)

Imports Formula, generics, ggplot2, MASS, rlang, stats

Suggests broom, knitr, rmarkdown, tinytest, units

LinkingTo armadillo4r, cpp4r

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Encoding UTF-8

LazyData true

NeedsCompilation yes

Author Mauricio Vargas Sepulveda [aut, cre] (ORCID: <https://orcid.org/0000-0003-1017-7574>), Joao Santos Silva [ths], Yoto Yotov [ctb]

Maintainer Mauricio Vargas Sepulveda <m.vargas.sepulveda@gmail.com>

Repository CRAN

Date/Publication 2026-06-15 08:50:02 UTC

Contents

`capybara-package` *Generalized Linear Models (GLMs) with high-dimensional k-way fixed effects*

Description

Provides a routine to partial out factors with many levels during the optimization of the log-likelihood function of the corresponding GLM. The package is based on the algorithm described in Stammann (2018). It also offers an efficient algorithm to recover estimates of the fixed effects in a post-estimation routine and includes robust and multi-way clustered standard errors. This package is a ground up rewrite with multiple refactors, optimizations, and new features compared to the original package `alpaca`. In its current state, the package is stable and future changes will be limited to bug fixes and improvements, but not to altering the functions' arguments or outputs.

Author(s)

Maintainer: Mauricio Vargas Sepulveda <m.vargas.sepulveda@gmail.com> (ORCID)

Authors:

- Mauricio Vargas Sepulveda <m.vargas.sepulveda@gmail.com> (ORCID)

Other contributors:

- Joao Santos Silva [thesis advisor]
- Yoto Yotov [contributor]

See Also

Useful links:

- https://pacha.dev/capybara/
- https://github.com/pachadotdev/capybara
- Report bugs at https://github.com/pachadotdev/capybara/issues

---

`augment.feglm` *Broom Integration*

Description

The provided `broom` methods do the following:

1. `augment`: Takes the input data and adds additional columns with the fitted values and residuals.
2. `glance`: Extracts the deviance, null deviance, and the number of observations.‘
3. `tidy`: Extracts the estimated coefficients and their standard errors.

Usage

```
## S3 method for class 'feglm'
augment(x, newdata = NULL, ...)

## S3 method for class 'felm'
augment(x, newdata = NULL, ...)

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

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

## S3 method for class 'feglm'
tidy(x, conf_int = FALSE, conf_level = 0.95, ...)

## S3 method for class 'felm'
tidy(x, conf_int = FALSE, conf_level = 0.95, ...)
```

Arguments

x	A fitted model object.
newdata	Optional argument to use data different from the data used to fit the model.
...	Additional arguments passed to the method.
conf_int	Logical indicating whether to include the confidence interval.
conf_level	The confidence level for the confidence interval.

Value

A tibble with the respective information for the augment, glance, and tidy methods.

Examples

```
ross2004_subset <- ross2004[ross2004$year == 1999, ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

fit <- fepoisson(ltrade ~ ldist, ross2004_subset,
  control = fit_control(keep_data = TRUE)
)

broom::augment(fit)
broom::glance(fit)
broom::tidy(fit)
```

---

autoplot.feglm *Autoplot method for feglm objects*

Description

Extracts the estimated coefficients and their confidence intervals.

Extracts the estimated coefficients and their confidence intervals.

Usage

```
## S3 method for class 'feglm'
autoplot(object, ...)

## S3 method for class 'felm'
autoplot(object, ...)
```

Arguments

object	A fitted model object.
...	Additional arguments passed to the method. In this case, the additional argument is conf_level, which is the confidence level for the confidence interval.

Value

A ggplot object with the estimated coefficients and their confidence intervals.

A ggplot object with the estimated coefficients and their confidence intervals.

Examples

```
ross2004_subset <- ross2004[ross2004$year == 1999, ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

fit <- fepoisson(ltrade ~ ldist | ctry1, ross2004_subset)

autoplot(fit, conf_level = 0.99)

ross2004_subset <- ross2004[ross2004$year == 1999, ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

fit <- felm(ltrade ~ ldist | ctry1, ross2004_subset)

autoplot(fit, conf_level = 0.99)
```

---

correia2019 *Separation Example Datasets*

Description

Nonexistence of estimates of Poisson models across different statistical packages.

Usage

```
correia2019
```

Format

`correia2019`**:**

A list of data frames with three elements:

example1 Data frame used to show lack of convergence

example2 Data frame used to show lack of convergence with step-halving

fe1 Data frame with fixed effects used with the 'alpaca' package

Source

'correia2019' GitHub repository (https://github.com/sergiocorreia/correia2019/blob/master/guides/nonexistence_benchmark
packages)

feglm *GLM fitting with high-dimensional k-way fixed effects*

Description

feglm can be used to fit generalized linear models with many high-dimensional fixed effects. The term fixed effect means having one intercept for each level in each category.

Usage

```
feglm(
  formula = NULL,
  data = NULL,
  family = gaussian(),
  weights = NULL,
  vcov = NULL,
  beta_start = NULL,
  eta_start = NULL,
  offset = NULL,
  control = NULL
)
```

Arguments

formula	an object of class "formula": a symbolic description of the model to be fitted. formula must be of type response ~ slopes | fixed_effects | cluster.
data	an object of class "data.frame" containing the variables in the model. The expected input is a dataset with the variables specified in formula and a number of rows at least equal to the number of variables in the model.
family	the link function to be used in the model. Similar to glm.fit this has to be the result of a call to a family function. Default is gaussian(). See family for details of family functions.
weights	an optional string with the name of the prior weights variable in data.
vcov	an optional character string specifying the type of variance-covariance estimator. One of "iid" (default OLS, ignore cluster part of formula), "hetero" (heteroskedastic-robust HC0, computed in C++ - no cluster variable needed), "cluster" (one-way sandwich using the cluster variable in the formula), "m-estimator" (M-estimator one-way sandwich), or "dyadic" (Cameron-Miller dyadic sandwich; requires two entity variables in the third part of the formula). When NULL (default), the type is inferred from the formula: if a cluster variable is present the standard sandwich is used, otherwise the inverse Hessian (IID) is returned.
beta_start	an optional vector of starting values for the structural parameters in the linear predictor. Default is $\boldsymbol{\beta} = \mathbf{0}$.
eta_start	an optional vector of starting values for the linear predictor.

offset	an optional formula or numeric vector specifying an a priori known component to be included in the linear predictor. If a formula, it should be of the form ~ variable.
control	a named list of parameters for controlling the fitting process. See fit_control for details.

Details

If feglm does not converge this is often a sign of linear dependence between one or more regressors and a fixed effects category. In this case, you should carefully inspect your model specification.

Value

A named list of class "feglm". The list contains the following fifteen elements:

coefficients	a named vector of the estimated coefficients
eta	a vector of the linear predictor
weights	a vector of the weights used in the estimation
hessian	a matrix with the numerical second derivatives
deviance	the deviance of the model
null_deviance	the null deviance of the model
conv	a logical indicating whether the model converged
iter	the number of iterations needed to converge
nobs	a named vector with the number of observations used in the estimation indicating the dropped and perfectly predicted observations
fe_levels	a named vector with the number of levels in each fixed effects
nms_fe	a list with the names of the fixed effects variables
formula	the formula used in the model
data	the data used in the model after dropping non-contributing observations
family	the family used in the model
control	the control list used in the model
vcov_type	a character string indicating the variance-covariance type used: "iid", "hetero", "cluster", "m-estimator", or "dyadic"

References

Gaure, S. (2013). "OLS with Multiple High Dimensional Category Variables". Computational Statistics and Data Analysis, 66.

Marschner, I. (2011). "glm2: Fitting generalized linear models with convergence problems". The R Journal, 3(2).

Stammann, A., F. Heiss, and D. McFadden (2016). "Estimating Fixed Effects Logit Models with Large Panel Data". Working paper.

Stammann, A. (2018). "Fast and Feasible Estimation of Generalized Linear Models with High-Dimensional k-Way Fixed Effects". ArXiv e-prints.

Examples

```
# check the felm examples for the details about clustered standard errors
ross2004_subset <- ross2004[ross2004$year == 1999, ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

fit <- feglm(ltrade ~ ldist | ctry1, ross2004_subset, family = poisson())

summary(fit)
```

`felm` *LM fitting with high-dimensional k-way fixed effects*

Description

feglm can be used to fit linear models with many high-dimensional fixed effects. The estimation procedure is based on unconditional maximum likelihood and can be interpreted as a "weighted demeaning" approach.

Usage

```
felm(formula = NULL, data = NULL, weights = NULL, vcov = NULL, control = NULL)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model to be fitted. `formula` must be of type `response ~ slopes | fixed_effects | cluster`.
`data`	an object of class `"data.frame"` containing the variables in the model. The expected input is a dataset with the variables specified in `formula` and a number of rows at least equal to the number of variables in the model.
`weights`	an optional string with the name of the prior weights variable in `data`.
`vcov`	an optional character string specifying the type of variance-covariance estimator. One of `"iid"` (default OLS, ignore cluster part of formula), `"hetero"` (heteroskedastic-robust HC0, computed in C++ - no cluster variable needed), `"cluster"` (one-way sandwich using the cluster variable in the formula), `"m-estimator"` (M-estimator one-way sandwich), or `"dyadic"` (Cameron-Miller dyadic sandwich; requires two entity variables in the third part of the formula). When `NULL` (default), the type is inferred from the formula: if a cluster variable is present the standard sandwich is used, otherwise the inverse Hessian (IID) is returned.
`control`	a named list of parameters for controlling the fitting process. See fit_control for details.

Value

A named list of class `"felm"`. The list contains the following eleven elements:

`coefficients`	a named vector of the estimated coefficients
`fitted_values`	a vector of the estimated dependent variable
`weights`	a vector of the weights used in the estimation
`hessian`	a matrix with the numerical second derivatives
`null_deviance`	the null deviance of the model
`nobs`	a named vector with the number of observations used in the estimation indicating the dropped and perfectly predicted observations
`fe_levels`	a named vector with the number of levels in each fixed effect
`nms_fe`	a list with the names of the fixed effects variables
`formula`	the formula used in the model
`data`	the data used in the model after dropping non-contributing observations
`control`	the control list used in the model

References

Gaure, S. (2013). "OLS with Multiple High Dimensional Category Variables". Computational Statistics and Data Analysis, 66.

Marschner, I. (2011). "glm2: Fitting generalized linear models with convergence problems". The R Journal, 3(2).

Stammann, A., F. Heiss, and D. McFadden (2016). "Estimating Fixed Effects Logit Models with Large Panel Data". Working paper.

Stammann, A. (2018). "Fast and Feasible Estimation of Generalized Linear Models with High-Dimensional k-Way Fixed Effects". ArXiv e-prints.

Examples

```
ross2004_subset <- ross2004[ross2004$year == 1999, ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

# Model with fixed effects
fit <- felm(ltrade ~ ldist | ctry1, ross2004_subset)
summary(fit)

ross2004_subset <- ross2004[ross2004$year %in% c(1994, 1999), ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

# Model without fixed effects but with clustered standard errors
# Note: Use 0 to indicate no fixed effects when specifying clusters
fit <- felm(ltrade ~ ldist | 0 | year, ross2004_subset)
summary(fit)
```

`fenegbin` *Negative Binomial model fitting with high-dimensional k-way fixed effects*

Description

A routine that uses the same internals as feglm.

Usage

```
fenegbin(
  formula = NULL,
  data = NULL,
  weights = NULL,
  beta_start = NULL,
  eta_start = NULL,
  init_theta = NULL,
  link = c("log", "identity", "sqrt"),
  offset = NULL,
  control = NULL
)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model to be fitted. `formula` must be of type `response ~ slopes | fixed_effects | cluster`.
`data`	an object of class `"data.frame"` containing the variables in the model. The expected input is a dataset with the variables specified in `formula` and a number of rows at least equal to the number of variables in the model.
`weights`	an optional string with the name of the prior weights variable in `data`.
`beta_start`	an optional vector of starting values for the structural parameters in the linear predictor. Default is $\boldsymbol{\beta} = \mathbf{0}$.
`eta_start`	an optional vector of starting values for the linear predictor.
`init_theta`	an optional initial value for the theta parameter (see glm.nb).
`link`	the link function. Must be one of `"log"`, `"sqrt"`, or `"identity"`.
`offset`	an optional formula or numeric vector specifying an a priori known component to be included in the linear predictor. If a formula, it should be of the form `~ variable`.
`control`	a named list of parameters for controlling the fitting process. See fit_control for details.

Value

A named list of class "feglm". The list contains the following eighteen elements:

coefficients	a named vector of the estimated coefficients
eta	a vector of the linear predictor
weights	a vector of the weights used in the estimation
hessian	a matrix with the numerical second derivatives
deviance	the deviance of the model
null_deviance	the null deviance of the model
conv	a logical indicating whether the model converged
iter	the number of iterations needed to converge
theta	the estimated theta parameter
iter_outer	the number of outer iterations
conv_outer	a logical indicating whether the outer loop converged
nobs	a named vector with the number of observations used in the estimation indicating the dropped and perfectly predicted observations
fe_levels	a named vector with the number of levels in each fixed effects
nms_fe	a list with the names of the fixed effects variables
formula	the formula used in the model
data	the data used in the model after dropping non-contributing observations
family	the family used in the model
control	the control list used in the model

Examples

```
# check the felm examples for the details about clustered standard errors

ross2004_subset <- ross2004[ross2004$year == 1999, ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

fit <- fenegbin(ltrade ~ ldist | ctry1, ross2004_subset)

summary(fit)
```

`fepoisson` *Poisson model fitting high-dimensional with k-way fixed effects*

Description

A wrapper for feglm with `family = poisson()`.

Usage

```
fepoisson(
  formula = NULL,
  data = NULL,
  weights = NULL,
  vcov = NULL,
  beta_start = NULL,
  eta_start = NULL,
  offset = NULL,
  control = NULL
)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model to be fitted. `formula` must be of type `response ~ slopes | fixed_effects | cluster`.
`data`	an object of class `"data.frame"` containing the variables in the model. The expected input is a dataset with the variables specified in `formula` and a number of rows at least equal to the number of variables in the model.
`weights`	an optional string with the name of the prior weights variable in `data`.
`vcov`	an optional character string specifying the type of variance-covariance estimator. One of `"iid"` (default OLS, ignore cluster part of formula), `"hetero"` (heteroskedastic-robust HC0, computed in C++ - no cluster variable needed), `"cluster"` (one-way sandwich using the cluster variable in the formula), `"m-estimator"` (M-estimator one-way sandwich), or `"dyadic"` (Cameron-Miller dyadic sandwich; requires two entity variables in the third part of the formula). When `NULL` (default), the type is inferred from the formula: if a cluster variable is present the standard sandwich is used, otherwise the inverse Hessian (IID) is returned.
`beta_start`	an optional vector of starting values for the structural parameters in the linear predictor. Default is $\boldsymbol{\beta} = \mathbf{0}$.
`eta_start`	an optional vector of starting values for the linear predictor.
`offset`	an optional formula or numeric vector specifying an a priori known component to be included in the linear predictor. If a formula, it should be of the form `~ variable`.
`control`	a named list of parameters for controlling the fitting process. See fit_control for details.

Value

A named list of class "feglm".

Examples

```
# check the feglm examples for the details about clustered standard errors

ross2004_subset <- ross2004[ross2004$year == 1999, ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

fit <- fepoisson(ltrade ~ ldist, ross2004_subset)

summary(fit)
```

---

fepoisson_asymmetric *Asymmetric Poisson Pseudo-Maximum Likelihood (APPML) Estimation*

Description

Fits an asymmetric Poisson pseudo-maximum likelihood model with high-dimensional fixed effects using expectile regression. This approach extends standard PPML by allowing different weights for positive and negative residuals, enabling estimation of conditional expectiles rather than the conditional mean.

Usage

```
fepoisson_asymmetric(
  formula = NULL,
  data = NULL,
  weights = NULL,
  beta_start = NULL,
  eta_start = NULL,
  offset = NULL,
  control = NULL
)
```

Arguments

formula	an object of class "formula": a symbolic description of the model to be fitted. formula must be of type response ~ slopes | fixed_effects | cluster.
data	an object of class "data.frame" containing the variables in the model. The expected input is a dataset with the variables specified in formula and a number of rows at least equal to the number of variables in the model.
weights	an optional string with the name of the prior weights variable in data.

`beta_start`	an optional vector of starting values for the structural parameters in the linear predictor. Default is $\boldsymbol{\beta} = \mathbf{0}$.
`eta_start`	an optional vector of starting values for the linear predictor.
`offset`	an optional formula or numeric vector specifying an a priori known component to be included in the linear predictor. If a formula, it should be of the form `~ variable`.
`control`	a named list of parameters for controlling the fitting process. See fit_control for details.

Details

The APPML estimator minimizes an asymmetric loss function based on expectiles. For a given expectile τ, observations with negative residuals receive weight τ while observations with positive residuals receive weight $1 - \tau$. The algorithm iteratively:

1. Computes residuals from the current fit
2. Updates weights as $w_i = |\tau - \mathbf{1}(r_i < 0)|$
3. Re-fits the weighted Poisson model
4. Checks convergence using $(b - b_{old})' V^{-1} (b - b_{old}) < \epsilon$

The expectile parameter is specified via `control = fit_control(expectile = ...)`. When `expectile = 0.5`, the estimator is equivalent to standard PPML. Values below 0.5 estimate lower conditional expectiles (more sensitive to small values), while values above 0.5 estimate upper conditional expectiles (more sensitive to large values).

Value

A named list of class `"feglm"` containing:

`coefficients`	named vector of estimated coefficients
`vcov`	variance-covariance matrix of coefficients
`eta`	linear predictor
`fitted_values`	fitted values from the final iteration
`residuals`	residuals from the final fit
`weights`	observation weights used in final fit
`appml_weights`	asymmetric weights used in APPML algorithm
`deviance`	the deviance of the model
`null_deviance`	the null deviance of the model
`conv`	logical indicating whether inner GLM converged
`conv_outer`	logical indicating whether APPML outer loop converged
`iter`	number of inner iterations
`iter_outer`	number of outer APPML iterations
`expectile`	the expectile value used

`objective_function`
: final value of the convergence criterion

`negative_residuals_share`
: proportion of negative residuals in final fit

`nobs`
: a named vector with the number of observations

`fe_levels`
: a named vector with the number of levels in each fixed effect

`nms_fe`
: a list with the names of the fixed effects variables

`formula`
: the formula used in the model

`family`
: the family used in the model (Poisson)

`control`
: the control list used in the model

References

Newey, W. K., & Powell, J. L. (1987). Asymmetric least squares estimation and testing. *Econometrica*, 55(4), 819-847.

See Also

fepoisson, feglm, fit_control

Examples

```
ross2004_subset <- ross2004[ross2004$year == 1999, ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

# Lower expectile (10th) - more weight on negative residuals
fit10 <- fepoisson_asymmetric(
  ltrade ~ ldist | ctry1, ross2004_subset,
  control = fit_control(expectile = 0.1)
)

summary(fit10)
```

`fe_table` *Generate fixed effects tables*

Description

Generate fixed effects tables

Usage

```
fe_table(
  ...,
  n = 5,
  coef_digits = 3,
  latex = FALSE,
  model_names = NULL,
  caption = NULL,
  label = NULL,
  position = "htbp"
)
```

Arguments

...	One or more model objects of `felm` or `feglm` class fitted with `return_fe = TRUE` in `control`.
n	Number of levels to show per fixed effect dimension. Use `Inf` to show all levels. The default is 5.
coef_digits	Number of digits for fixed effect coefficients. The default is 3.
latex	Whether to output as LaTeX code. The default is `FALSE`.
model_names	Optional vector of custom model names.
caption	Optional caption for the table (LaTeX only).
label	Optional label for cross-referencing (LaTeX only).
position	LaTeX float position specifier (LaTeX only). The default is `"htbp"`.

Value

A formatted fixed effects table of class `summary_table`.

Examples

```
ross2004_subset <- ross2004[ross2004$year == 1999, ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

m1 <- felm(ltrade ~ ldist | ctry1, ross2004_subset)
m2 <- fepoisson(ltrade ~ ldist | ctry1, ross2004_subset)

fe_table(m1, m2, model_names = c("Linear", "Poisson"))
```

`fit_control` *Set* `feglm` *Control Parameters*

Description

Set and change parameters used for fitting feglm, felm, and fenegbin. Termination conditions are similar to glm.

Usage

```
fit_control(
  dev_tol = 1e-06,
  center_tol = 1e-06,
  collin_tol = 1e-08,
  step_halving_factor = 0.5,
  alpha_tol = 1e-05,
  iter_max = 50L,
  iter_center_max = 10000L,
  iter_inner_max = 50L,
  iter_alpha_max = 10000L,
  step_halving_memory = 0.9,
  max_step_halving = 2L,
  start_inner_tol = 1e-05,
  grand_acc_period = 4L,
  centering = "berge",
  sep_tol = 1e-08,
  sep_zero_tol = 1e-08,
  sep_mu_tol = 1e-06,
  sep_max_iter = 200L,
  sep_simplex_max_iter = 2000L,
  sep_use_relu = TRUE,
  sep_use_simplex = TRUE,
  sep_use_mu = TRUE,
  return_fe = TRUE,
  keep_tx = FALSE,
  keep_data = FALSE,
  return_hessian = FALSE,
  check_separation = TRUE,
  init_theta = 0,
  vcov_type = NULL,
  expectile = NULL,
  expectile_tol = 1e-12,
  expectile_iter_max = 50L,
  expectile_glm_iter_max = NULL,
  expectile_trace = FALSE
)
```

Arguments

dev_tol	tolerance level for the first stopping condition of the maximization routine. The stopping condition is based on the relative change of the deviance in iteration r and can be expressed as follows: $\|dev_r - dev_{r-1}\|/(0.1 + \|dev_r\|) < tol$. The default is `1.0e-06`.
center_tol	tolerance level for the stopping condition of the centering algorithm. The stopping condition is based on the relative change of the centered variable similar to the `'lfe'` package. The default is `1.0e-05`.
collin_tol	tolerance level for detecting collinearity. The default is `1.0e-08`.
step_halving_factor	numeric indicating the factor by which the step size is halved to iterate towards convergence. This is used to control the step size during optimization. The default is `0.5`.
alpha_tol	tolerance for fixed effects (alpha) convergence. The default is `1.0e-05`.
iter_max	integer indicating the maximum number of iterations in the maximization routine. The default is `50L`.
iter_center_max	integer indicating the maximum number of iterations in the centering algorithm. The default is `10000L`.
iter_inner_max	integer indicating the maximum number of iterations in the inner loop of the centering algorithm. The default is `50L`.
iter_alpha_max	maximum iterations for fixed effects computation. The default is `10000L`.
step_halving_memory	numeric memory factor for step-halving algorithm. Controls how much of the previous iteration is retained. The default is `0.9`.
max_step_halving	maximum number of post-convergence step-halving attempts. The default is `2`.
start_inner_tol	starting tolerance for inner solver iterations. The default is `1.0e-05`.
grand_acc_period	integer indicating the period (in iterations) for grand acceleration in the centering algorithm. Grand acceleration applies a second-level Irons-Tuck extrapolation on the overall convergence trajectory. Lower values (e.g., 4-10) may speed up convergence for difficult problems. Set to a very large value (e.g., 10000) to effectively disable. The default is `4L`.
centering	character string indicating the centering algorithm to use for demeaning fixed effects. `"stammann"` (default) uses alternating projections with Gauss-Seidel sweeps plus Irons-Tuck and grand acceleration on coefficient vectors. Each iteration updates each fixed-effect dimension in sequence. `"berge"` uses a fixed-point reformulation as described in Berge (2018): all FE updates are composed into a single map $F = f_T \circ f_I$, reducing the problem to finding $\beta^* = F(\beta^*)$. The Irons and Tuck (1969) acceleration is then applied to the composed iteration. Both methods use warm-starting and grand acceleration.
sep_tol	tolerance for separation detection. The default is `1.0e-08`.

sep_zero_tol tolerance for treating values as zero in separation detection. The default is 1.0e-08.

sep_mu_tol tolerance for mu-based separation detection during IRLS iterations. Observations with y == 0 and eta <= log(sep_mu_tol) are flagged as separated. Based on ppmlhdfe's mu separation method. The default is 1.0e-06.

sep_max_iter maximum iterations for ReLU separation detection algorithm. The default is 200L.

sep_simplex_max_iter
maximum iterations for simplex separation detection algorithm. The default is 2000L.

sep_use_relu logical indicating whether to use the ReLU algorithm for separation detection. The default is TRUE.

sep_use_simplex
logical indicating whether to use the simplex algorithm for separation detection. The default is TRUE.

sep_use_mu logical indicating whether to use mu-based separation detection during IRLS iterations. This catches observations where predicted values become extremely small (suggesting perfect prediction of zeros). Following ppmlhdfe methodology. The default is TRUE.

return_fe logical indicating if the fixed effects should be returned. This can be useful when fitting general equilibrium models where skipping the fixed effects for intermediate steps speeds up computation. Set it to FALSE to minimize memory usage. The default is TRUE.

keep_tx logical indicating if the centered regressor matrix should be stored. The default is FALSE to minimize memory usage.

keep_data logical indicating if the filtered data should be stored in the result object. Required for predict() methods. Set to TRUE when planning to use prediction functions. The default is FALSE to minimize memory usage for production/benchmark use.

return_hessian logical indicating if the Hessian matrix should be returned. The Hessian is a P*P matrix used to compute the variance-covariance matrix. The default is FALSE to minimize memory usage (vcov is still computed and returned).

check_separation
logical indicating whether to perform separation detection for Poisson models. When TRUE (default), observations with perfect prediction are automatically detected and excluded from estimation. Set to FALSE to skip this check and speed up computation when separation is known not to be an issue. The default is TRUE.

init_theta Initial value for the negative binomial dispersion parameter (theta). The default is 0.0.

vcov_type Optional character string specifying the type of variance-covariance estimator to be used. When NULL (default), the covariance matrix is the inverse Hessian (IID) when no cluster variable is present, or a clustered sandwich when one is. Other values: "hetero" - heteroskedastic-robust HC0 sandwich (no cluster variable needed); "m-estimator" - one-way M-estimator sandwich (cluster variable required); "m-estimator-dyadic" - dyadic-robust Cameron-Miller

sandwich (two entity columns required in the third part of the formula like `z ~ x + y | fe | cl1 + cl2`).

`expectile` numeric value between 0 and 1 (exclusive) specifying the expectile for asymmetric Poisson pseudo-maximum likelihood (APPML) estimation. When `NULL` (default), standard symmetric estimation is performed. Values below 0.5 give more weight to negative residuals (lower quantiles), while values above 0.5 give more weight to positive residuals (upper quantiles). For example, `expectile = 0.1` estimates the 10th expectile, `expectile = 0.5` is equivalent to standard Poisson PML, and `expectile = 0.9` estimates the 90th expectile.

`expectile_tol` tolerance level for the stopping condition of the expectile iteration algorithm.

The convergence criterion uses a hybrid approach: the algorithm converges when the squared norm of coefficient changes $\|b - b_{old}\|^2$ is below a threshold computed as the maximum of an absolute floor (`1e-14`) and a relative tolerance scaled by the L2 norm of coefficients. Specifically: threshold $= \max(10^{-14}, (\text{tol} \cdot \|b\|_2)^2)$. This hybrid criterion ensures numerical robustness across different compiler optimizations (e.g., FMA on macOS) and handles both cases with small and large coefficients appropriately. The default is `1.0e-12`.

`expectile_iter_max`

integer indicating the maximum number of iterations for the expectile reweighting algorithm. The default is `50L`.

`expectile_glm_iter_max`

integer indicating the maximum number of inner GLM (IRLS) iterations per APPML outer iteration. When `NULL` (default), the value of `iter_max` is used, meaning the inner GLM runs to full convergence before APPML weights are updated. Setting this to `1L` enables a single-step mode where APPML asymmetric weights are updated after every Newton step, which typically reduces the total number of iterations needed.

`expectile_trace`

logical indicating whether to print iteration information during expectile estimation. The default is `FALSE`.

Value

A named list of control parameters.

Memory-Efficient vs Full Models

By default, `fit_control()` returns "thin" model objects with minimal memory footprint:

- `keep_data = FALSE` - data not stored
- `return_fe = FALSE` - fixed effects not stored
- `return_hessian = FALSE` - Hessian not stored
- `keep_tx = FALSE` - centered matrix not stored

See Also

feglm, felm, and fenegbin

Examples

```
ross2004_subset <- ross2004[ross2004$year == 1999, ]
ross2004_subset <- ross2004_subset[ross2004_subset$ltrade >
  quantile(ross2004_subset$ltrade, 0.75), ]

felm(ltrade ~ ldist | ctry1, ross2004_subset,
  control = fit_control(dev_tol = 1e-10, center_tol = 1e-10)
)
```

---

ross2004 *Subset of WTO data from Ross (2004)*

Description

Data from Ross (2004) used to show the different variance-covariance estimators from Cameron and Miller (2014).

Usage

```
ross2004
```

Format

`ross2004`**:**

A data frame with 234,597 rows and 22 columns:

ltrade Log of bilateral trade between i and j at time t

bothin Binary variable which is unity if both i and j are GATT/WTO members at t

onein Binary variable which is unity if either i or j is a GATT/WTO member at t

gsp Binary variable which is unity if i was a GSP beneficiary of j or vice versa at t

ldist Log of distance between i and j

lrgdp Log of real GDP

lrgdppc Log of real GDP per capita

regional Binary variable which is unity if i and j are in the same region

custrict Binary variable which is unity if i and j are in the same customs union

comlang Binary variable which is unity if i and j share a common language

border Binary variable which is unity if i and j share a border

landl Number of landlocked countries in the country-pair (0, 1, or 2)

island Number of island nations in the pair (0, 1, or 2)

lareap Log of the area of the country (in square kilometers)

comcol Binary variable which is unity if i and j were ever colonies after 1945 with the same colonizer

curcol Binary variable which is unity if i and j are colonies at time t

colony Binary variable which is unity if i ever colonized j or vice versa

comctry Binary variable which is unity if i and j remained part of the same nation during the sample (e.g., France and Guadeloupe)

ctry1 Country 1 ISO-3 code

ctry2 Country 2 ISO-3 code

pair Country pair undirected dyads

year Year of observation

Source

Do We Really Know That the WTO Increases Trade? (DOI: 10.1257/000282804322970724)

`sandwich_vcov` *Recompute Sandwich Variance-Covariance Matrix*

Description

Recompute the variance-covariance matrix for a fitted model using a different clustering structure or covariance type. This allows changing the vcov estimator without re-fitting the model, provided the model was fit with `keep_tx = TRUE` and `return_hessian = TRUE` in the control parameters.

Usage

```
sandwich_vcov(
  object,
  cluster1 = NULL,
  cluster2 = NULL,
  type = c("hetero", "clustered", "m-estimator", "dyadic", "m-estimator-dyadic"),
  ...
)
```

Arguments

`object`	a fitted model object of class `"feglm"` or `"felm"`.
`cluster1`	a vector or factor for the first clustering variable. Required for `"clustered"` and `"dyadic"` types.
`cluster2`	a vector or factor for the second clustering variable. Required for `"dyadic"` type.
`type`	character string specifying the covariance type. One of: • `"hetero"`: heteroskedasticity-robust (no clustering, also known as `"HC0"`) • `"clustered"` or `"m-estimator"`: one-way cluster-robust • `"dyadic"` or `"m-estimator-dyadic"`: dyadic cluster-robust for network/trade data
`...`	additional arguments (currently ignored).

Details

The model must be fit with `fit_control(keep_tx = TRUE, return_hessian = TRUE)` to store the centered design matrix, Hessian, and working residuals needed for vcov recomputation. Note that `keep_data = TRUE` is NOT required - residuals are stored automatically when `keep_tx = TRUE`.

For dyadic clustering (used in gravity/trade models), `cluster1` and `cluster2` represent the two entity dimensions (e.g., exporter and importer). The function handles the full dyadic correlation structure including cross-entity correlations.

Value

A named matrix of covariance estimates.

References

Cameron, C., J. Gelbach, and D. Miller (2011). "Robust Inference With Multiway Clustering". Journal of Business & Economic Statistics 29(2).

Cameron, C. and D. Miller (2014). "Robust Inference for Dyadic Data". Unpublished manuscript.

See Also

feglm, felm, fit_control

Examples

```
# Refitting models

ross2004_s1 <- ross2004[ross2004$year == 1994, ]
ross2004_s1 <- ross2004_s1[ross2004_s1$ltrade >
  quantile(ross2004_s1$ltrade, 0.75), ]

ross2004_s2 <- ross2004[ross2004$year == 1999, ]
ross2004_s2 <- ross2004_s2[ross2004_s2$ltrade >
  quantile(ross2004_s2$ltrade, 0.75), ]

ross2004_subset <- rbind(ross2004_s1, ross2004_s2)

fepoisson(
  ltrade ~ ldist | ctry1, ross2004_subset,
  control = fit_control(vcov_type = "hetero")
)

fepoisson(
  ltrade ~ ldist | ctry1, ross2004_subset,
  control = fit_control(vcov_type = "m-estimator")
)

# Reusing models

# Store required components
fit <- fepoisson(
```

```
  ltrade ~ ldist | ctry1, ross2004_subset,
  control = fit_control(keep_tx = TRUE, return_hessian = TRUE)
)

# Heteroskedastic-robust HC0 sandwich (no cluster variable needed)
sandwich_vcov(fit, type = "hetero")

#' One-way cluster
sandwich_vcov(fit, cluster1 = ross2004_subset$year, type = "clustered")
```

`summary_table` *Generate formatted regression tables*

Description

Generate formatted regression tables

Usage

```
summary_table(
  ...,
  coef_digits = 3,
  se_digits = 3,
  stars = TRUE,
  latex = FALSE,
  model_names = NULL,
  caption = NULL,
  label = NULL,
  position = "htbp"
)
```

Arguments

`...`	One or more model objects of `felm` or `feglm` class.
`coef_digits`	Number of digits for coefficients. The default is 3.
`se_digits`	Number of digits for standard errors. The default is 3.
`stars`	Whether to include significance stars. The default is `TRUE`.
`latex`	Whether to output as LaTeX code. The default is `FALSE`.
`model_names`	Optional vector of custom model names
`caption`	Optional caption for the table (LaTeX only)
`label`	Optional label for cross-referencing (LaTeX only)
`position`	LaTeX float position specifier (LaTeX only). The default is `"htbp"`.

Value

A formatted table

Examples

```
ross2004_subset <- ross2004[ross2004$year == 1999 &
  ross2004$ltrade > quantile(ross2004$ltrade, 0.75), ]

m1 <- felm(ltrade ~ ldist | ctry1, ross2004_subset)
m2 <- fepoisson(ltrade ~ ldist | ctry1, ross2004_subset)

summary_table(m1, m2, model_names = c("Linear", "Poisson"))
```

`update.feglm` *Update a fitted* `feglm` *model*

Description

S3 method for `update()` that understands the `|`-separated formula syntax used by `feglm()`. Uses `Formula::update.Formula()` for proper handling of multi-part formulas. Identical semantics to `update.felm()`.

Usage

```
## S3 method for class 'feglm'
update(object, formula. = . ~ ., vcov = NULL, family = NULL, ...)
```

Arguments

`object`	A fitted `feglm` object.
`formula.`	Update formula; only the segments you want to change need to differ from `. .` Examples: • `. ~ . | country + year` - change FE, keep regressors. • `. ~ . | . | ctry1 + ctry2` - keep FE, change cluster. • `. ~ . - bothin | year` - drop a regressor, keep FE.
`vcov`	Optional new `vcov` value (e.g. `"cluster"`). If omitted the original value is reused.
`family`	Optional new family (e.g. `binomial()`). If omitted the original family is reused.
`...`	Additional arguments forwarded to `felm()`.

Value

A refitted `feglm` object.

`update.felm` *Update a fitted* `felm` *model*

Description

S3 method for `update()` that understands the `|`-separated formula syntax used by `felm()`. R's built-in `stats::update.formula()` breaks on these formulas because the `|` parts look like factor arithmetic. This method uses the `Formula::update.Formula()` method which correctly handles multi-part formulas.

The `.` placeholder works as usual:

- `. ~ .` - keep the current response and RHS regressors.
- The second `|` segment replaces (or keeps, if `.`) the fixed-effects.
- The third `|` segment replaces (or keeps, if `.`) the cluster variables.

Usage

```
## S3 method for class 'felm'
update(object, formula. = . ~ ., vcov = NULL, ...)
```

Arguments

`object`	A fitted `felm` object.
`formula.`	Update formula; only the segments you want to change need to differ from `.`. Examples: • `. ~ . | country + year` - change FE, keep regressors. • `. ~ . | . | ctry1 + ctry2` - keep FE, change cluster. • `. ~ . - bothin | year` - drop a regressor, keep FE.
`vcov`	Optional new `vcov` value (e.g. `"cluster"`). If omitted the original value is reused.
`...`	Additional arguments forwarded to `felm()`.

Value

A refitted `felm` object.

update.felm_formula *Update a* felm_formula *object*

Description

S3 method for update() on felm_formula objects. Uses Formula::update.Formula() to properly handle multi-part formulas with |.

The . placeholder behaves as in stats::update.formula():

- . ~ . - keep current response and RHS unchanged.
- Second | segment - replaces (or keeps with .) the fixed effects.
- Third | segment - replaces (or keeps with .) the cluster variables.

Usage

```
## S3 method for class 'felm_formula'
update(object, formula., ...)
```

Arguments

object	A felm_formula object.
formula.	Update specification, e.g. . ~ . | . | ctry1 + ctry2.
...	Ignored.

Value

A new felm_formula object.

vcov.feglm *Covariance matrix for GLMs*

Description

Covariance matrix for the estimator of the structural parameters from objects returned by feglm. The covariance is computed during model fitting - either the inverse Hessian (default) or the sandwich estimator if cluster variables are specified in the formula.

Usage

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

Arguments

`object`	an object of class `"feglm"`.
`...`	additional arguments (currently ignored).

Value

A named matrix of covariance estimates.

References

Cameron, C., J. Gelbach, and D. Miller (2011). "Robust Inference With Multiway Clustering". Journal of Business & Economic Statistics 29(2).

See Also

feglm

Examples

```
# Model with clustering - returns sandwich covariance
ross2004_s1 <- ross2004[ross2004$year == 1994, ]
ross2004_s1 <- ross2004_s1[ross2004_s1$ltrade >
  quantile(ross2004_s1$ltrade, 0.75), ]

ross2004_s2 <- ross2004[ross2004$year == 1999, ]
ross2004_s2 <- ross2004_s2[ross2004_s2$ltrade >
  quantile(ross2004_s2$ltrade, 0.75), ]

ross2004_subset <- rbind(ross2004_s1, ross2004_s2)

fit <- fepoisson(ltrade ~ ldist + border | ctry1 | year, ross2004_subset)

vcov(fit)
```

`vcov.felm` *Covariance matrix for LMs*

Description

Covariance matrix for the estimator of the structural parameters from objects returned by felm. The covariance is computed during model fitting - either the inverse Hessian (default) or the sandwich estimator if cluster variables are specified in the formula.

Usage

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

Arguments

`object`	an object of class `"felm"`.
`...`	additional arguments (currently ignored).

Value

A named matrix of covariance estimates.

References

Cameron, C., J. Gelbach, and D. Miller (2011). "Robust Inference With Multiway Clustering". Journal of Business & Economic Statistics 29(2).

See Also

felm

Examples

```
# Model with clustering - returns sandwich covariance
ross2004_s1 <- ross2004[ross2004$year == 1994, ]
ross2004_s1 <- ross2004_s1[ross2004_s1$ltrade >
  quantile(ross2004_s1$ltrade, 0.75), ]

ross2004_s2 <- ross2004[ross2004$year == 1999, ]
ross2004_s2 <- ross2004_s2[ross2004_s2$ltrade >
  quantile(ross2004_s2$ltrade, 0.75), ]

ross2004_subset <- rbind(ross2004_s1, ross2004_s2)

fit <- felm(ltrade ~ ldist | ctry1 | year, ross2004_subset)

vcov(fit)
```

Index