Package 'capybara'

March 26, 2025

Type Package

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

Version 0.9.1

Imports data.table, Formula, generics, ggplot2, kendallknight, MASS, stats

Suggests broom, knitr, rmarkdown, testthat (>= 3.0.0), units

Depends R(>= 3.5.0)

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> and Gaure (2013) <doi:10.1016/j.csda.2013.03.024>.

License Apache License (>= 2)

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

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

LazyData true

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

LinkingTo cpp11, cpp11armadillo

VignetteBuilder knitr

Config/testthat/edition 3

Author Mauricio Vargas Sepulveda [aut, cre] (<`https://orcid.org/0000-0003-1017-7574`>)

Maintainer Mauricio Vargas Sepulveda <`m.sepulveda@mail.utoronto.ca`>

Repository CRAN

Date/Publication 2025-03-26 00:20:02 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. Further the package provides analytical bias corrections for binary choice models derived by Fernández-Val and Weidner (2016) and Hinz, Stammann, and Wanner (2020). 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.sepulveda@mail.utoronto.ca`> (ORCID)

See Also

Useful links:

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

apes — *Compute average partial effects after fitting binary choice models with a 1,2,3-way error component*

Description

apes is a post-estimation routine that can be used to estimate average partial effects with respect to all covariates in the model and the corresponding covariance matrix. The estimation of the covariance is based on a linear approximation (delta method) plus an optional finite population correction. Note that the command automatically determines which of the regressors are binary or non-binary.

Remark: The routine currently does not allow to compute average partial effects based on functional forms like interactions and polynomials.

Usage

```
apes(
  object = NULL,
  n_pop = NULL,
  panel_structure = c("classic", "network"),
  sampling_fe = c("independence", "unrestricted"),
  weak_exo = FALSE
)
```

Arguments

`object`	an object of class `"bias_corr"` or `"feglm"`; currently restricted to `binomial`.
`n_pop`	unsigned integer indicating a finite population correction for the estimation of the covariance matrix of the average partial effects proposed by Cruz-Gonzalez, Fernández-Val, and Weidner (2017). The correction factor is computed as follows: $(n^* - n)/(n^* - 1)$, where n^* and n are the sizes of the entire population and the full sample size. Default is `NULL`, which refers to a factor of zero and a covariance obtained by the delta method.
`panel_structure`	a string equal to `"classic"` or `"network"` which determines the structure of the panel used. `"classic"` denotes panel structures where for example the same cross-sectional units are observed several times (this includes pseudo panels). `"network"` denotes panel structures where for example bilateral trade flows are observed for several time periods. Default is `"classic"`.
`sampling_fe`	a string equal to `"independence"` or `"unrestricted"` which imposes sampling assumptions about the unobserved effects. `"independence"` imposes that all unobserved effects are independent sequences. `"unrestricted"` does not impose any sampling assumptions. Note that this option only affects the optional finite population correction. Default is `"independence"`.

`weak_exo`	logical indicating if some of the regressors are assumed to be weakly exogenous (e.g. predetermined). If object is of class `"bias_corr"`, the option will be automatically set to `TRUE` if the chosen bandwidth parameter is larger than zero. Note that this option only affects the estimation of the covariance matrix. Default is `FALSE`, which assumes that all regressors are strictly exogenous.

Value

The function `apes` returns a named list of class `"apes"`.

References

Cruz-Gonzalez, M., I. Fernández-Val, and M. Weidner (2017). "Bias corrections for probit and logit models with two-way fixed effects". The Stata Journal, 17(3), 517-545.

Czarnowske, D. and A. Stammann (2020). "Fixed Effects Binary Choice Models: Estimation and Inference with Long Panels". ArXiv e-prints.

Fernández-Val, I. and M. Weidner (2016). "Individual and time effects in nonlinear panel models with large N, T". Journal of Econometrics, 192(1), 291-312.

Fernández-Val, I. and M. Weidner (2018). "Fixed effects estimation of large-t panel data models". Annual Review of Economics, 10, 109-138.

Hinz, J., A. Stammann, and J. Wanner (2020). "State Dependence and Unobserved Heterogeneity in the Extensive Margin of Trade". ArXiv e-prints.

Neyman, J. and E. L. Scott (1948). "Consistent estimates based on partially consistent observations". Econometrica, 16(1), 1-32.

See Also

`bias_corr`, `feglm`

Examples

```
# subset trade flows to avoid fitting time warnings during check
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]

trade_2006$trade <- ifelse(trade_2006$trade > 100, 1L, 0L)

# Fit 'feglm()'
mod <- feglm(trade ~ lang | year, trade_2006, family = binomial())

# Compute average partial effects
mod_ape <- apes(mod)
summary(mod_ape)

# Apply analytical bias correction
mod_bc <- bias_corr(mod)
summary(mod_bc)
```

```
# Compute bias-corrected average partial effects
mod_ape_bc <- apes(mod_bc)
summary(mod_ape_bc)
```

`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

```
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]

mod <- fepoisson(
  trade ~ log_dist + lang + cntg + clny | exp_year + imp_year,
  trade_2006
)

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

---

autoplot.feglm *Autoplot method for feglm objects*

Description

Extracts the estimated coefficients and their confidence intervals.

Extracts the estimated coefficients and their confidence

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

```
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]

mod <- fepoisson(
  trade ~ log_dist + lang + cntg + clny | exp_year + imp_year,
  trade_2006
)

autoplot(mod, conf_level = 0.99)

set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[trade_2006$trade > 0, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]
trade_2006$log_trade <- log(trade_2006$trade)

mod <- felm(
  log_trade ~ log_dist + lang + cntg + clny | exp_year + imp_year,
  trade_2006
)

autoplot(mod, conf_level = 0.90)
```

`bias_corr` *Asymptotic bias correction after fitting binary choice models with a 1,2,3-way error component*

Description

Post-estimation routine to substantially reduce the incidental parameter bias problem. Applies the analytical bias correction derived by Fernández-Val and Weidner (2016) and Hinz, Stammann, and Wanner (2020) to obtain bias-corrected estimates of the structural parameters and is currently restricted to `binomial` with 1,2,3-way fixed effects.

Usage

```
bias_corr(object = NULL, l = 0L, panel_structure = c("classic", "network"))
```

Arguments

`object`	an object of class `"feglm"`.
`l`	unsigned integer indicating a bandwidth for the estimation of spectral densities proposed by Hahn and Kuersteiner (2011). The default is zero, which should be used if all regressors are assumed to be strictly exogenous with respect to the idiosyncratic error term. In the presence of weakly exogenous regressors, e.g.

lagged outcome variables, we suggest to choose a bandwidth between one and four. Note that the order of factors to be partialed out is important for bandwidths larger than zero.

`panel_structure`
: a string equal to `"classic"` or `"network"` which determines the structure of the panel used. `"classic"` denotes panel structures where for example the same cross-sectional units are observed several times (this includes pseudo panels). `"network"` denotes panel structures where for example bilateral trade flows are observed for several time periods. Default is `"classic"`.

Value

A named list of classes `"bias_corr"` and `"feglm"`.

References

Czarnowske, D. and A. Stammann (2020). "Fixed Effects Binary Choice Models: Estimation and Inference with Long Panels". ArXiv e-prints.

Fernández-Val, I. and M. Weidner (2016). "Individual and time effects in nonlinear panel models with large N, T". Journal of Econometrics, 192(1), 291-312.

Fernández-Val, I. and M. Weidner (2018). "Fixed effects estimation of large-t panel data models". Annual Review of Economics, 10, 109-138.

Hahn, J. and G. Kuersteiner (2011). "Bias reduction for dynamic nonlinear panel models with fixed effects". Econometric Theory, 27(6), 1152-1191.

Hinz, J., A. Stammann, and J. Wanner (2020). "State Dependence and Unobserved Heterogeneity in the Extensive Margin of Trade". ArXiv e-prints.

Neyman, J. and E. L. Scott (1948). "Consistent estimates based on partially consistent observations". Econometrica, 16(1), 1-32.

See Also

`feglm`

Examples

```
# subset trade flows to avoid fitting time warnings during check
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]

trade_2006$trade <- ifelse(trade_2006$trade > 100, 1L, 0L)

# Fit 'feglm()'
mod <- feglm(trade ~ lang | year, trade_2006, family = binomial())

# Apply analytical bias correction
mod_bc <- bias_corr(mod)
summary(mod_bc)
```

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 estimation procedure is based on unconditional maximum likelihood and can be interpreted as a "weighted demeaning" approach.

Remark: The term fixed effect is used in econometrician's sense of having intercepts for each level in each category.

Usage

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

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model to be fitted. `formula` must be of type `y ~ x	k`, where the second part of the formula refers to factors to be concentrated out. It is also possible to pass clustering variables to `feglm` as `y ~ x	k	c`.
`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`.
`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.
`control`	a named list of parameters for controlling the fitting process. See `feglm_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
`lvls_k`	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

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

```
# subset trade flows to avoid fitting time warnings during check
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]

mod <- feglm(
  trade ~ log_dist + lang + cntg + clny | exp_year + imp_year,
  trade_2006,
  family = poisson(link = "log")
)
```

```
summary(mod)

mod <- feglm(
  trade ~ log_dist + lang + cntg + clny | exp_year + imp_year | pair,
  trade_panel,
  family = poisson(link = "log")
)

summary(mod, type = "clustered")
```

feglm_control *Set* feglm *Control Parameters*

Description

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

Usage

```
feglm_control(
  dev_tol = 1e-06,
  center_tol = 1e-06,
  iter_max = 25L,
  iter_center_max = 10000L,
  iter_inner_max = 50L,
  iter_interrupt = 1000L,
  limit = 10L,
  trace = FALSE,
  drop_pc = TRUE,
  keep_mx = 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-08`.
`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-08`.
`iter_max`	unsigned integer indicating the maximum number of iterations in the maximization routine. The default is `25L`.
`iter_center_max`	unsigned integer indicating the maximum number of iterations in the centering algorithm. The default is `10000L`.

iter_inner_max	unsigned integer indicating the maximum number of iterations in the inner loop of the centering algorithm. The default is 50L.
iter_interrupt	unsigned integer indicating the maximum number of iterations before the algorithm is interrupted. The default is 1000L.
limit	unsigned integer indicating the maximum number of iterations of theta.ml. The default is 10L.
trace	logical indicating if output should be produced in each iteration. Default is FALSE.
drop_pc	logical indicating to drop observations that are perfectly classified/separated and hence do not contribute to the log-likelihood. This option is useful to reduce the computational costs of the maximization problem and improves the numerical stability of the algorithm. Note that dropping perfectly separated observations does not affect the estimates. The default is TRUE.
keep_mx	logical indicating if the centered regressor matrix should be stored. The centered regressor matrix is required for some covariance estimators, bias corrections, and average partial effects. This option saves some computation time at the cost of memory. The default is TRUE.

Value

A named list of control parameters.

See Also

feglm

Examples

```
feglm_control(0.05, 0.05, 10L, 10L, TRUE, TRUE, TRUE)
```

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

Description

A wrapper for feglm with family = gaussian().

Usage

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

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model to be fitted. `formula` must be of type `y ~ x | k`, where the second part of the formula refers to factors to be concentrated out. It is also possible to pass clustering variables to `feglm` as `y ~ x | k | c`.
`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`.
`control`	a named list of parameters for controlling the fitting process. See `feglm_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
`lvls_k`	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

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

# subset trade flows to avoid fitting time warnings during check
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]

mod <- felm(
  log(trade) ~ log_dist + lang + cntg + clny | exp_year + imp_year,
  trade_2006
)

summary(mod)
```

`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"),
  control = NULL
)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model to be fitted. `formula` must be of type `y ~ x	k`, where the second part of the formula refers to factors to be concentrated out. It is also possible to pass clustering variables to `feglm` as `y ~ x	k	c`.
`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"`.
`control`	a named list of parameters for controlling the fitting process. See `feglm_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
`lvls_k`	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 feglm examples for the details about clustered standard errors

# subset trade flows to avoid fitting time warnings during check
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 700), ]

mod <- fenegbin(
  trade ~ log_dist + lang + cntg + clny | exp_year + imp_year,
  trade_2006
)

summary(mod)
```

---

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,
  beta_start = NULL,
  eta_start = NULL,
  control = NULL
)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the model to be fitted. `formula` must be of type `y ~ x	k`, where the second part of the formula refers to factors to be concentrated out. It is also possible to pass clustering variables to `feglm` as `y ~ x	k	c`.
`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.
`control`	a named list of parameters for controlling the fitting process. See `feglm_control` for details.

Value

A named list of class `"feglm"`.

Examples

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

# subset trade flows to avoid fitting time warnings during check
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]

mod <- fepoisson(
  trade ~ log_dist + lang + cntg + clny | exp_year + imp_year,
  trade_2006
)

summary(mod)
```

`fixed_effects` *Recover the estimates of the fixed effects after fitting (G)LMs*

Description

The system might not have a unique solution since we do not take collinearity into account. If the solution is not unique, an estimable function has to be applied to our solution to get meaningful estimates of the fixed effects.

Usage

```
fixed_effects(object = NULL, control = NULL)
```

Arguments

`object`	an object of class "`feglm`".
`control`	a list of control parameters. If `NULL`, the default control parameters are used.

Value

A named list containing named vectors of estimated fixed effects.

References

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

Gaure, S. (n. d.). "Multicollinearity, identification, and estimable functions". Unpublished.

See Also

`felm`, `feglm`

Examples

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

# subset trade flows to avoid fitting time warnings during check
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]

mod <- fepoisson(
  trade ~ log_dist + lang + cntg + clny | exp_year + imp_year,
  trade_2006
)

fixed_effects(mod)
```

`trade_panel` *Trade Panel 1986-2006*

Description

Aggregated exports at origin-destination-year level for 1986-2006.

Usage

```
trade_panel
```

Format

`trade_panel`**:**

A data frame with 14,285 rows and 7 columns:

trade Nominal trade flows in current US dollars

dist Population-weighted bilateral distance between country ’i’ and ’j’, in kilometers

cntg Indicator. Equal to 1 if country ’i’ and ’j’ share a common border

lang Indicator. Equal to 1 if country ’i’ and ’j’ speak the same official language

clny Indicator. Equal to 1 if country ’i’ and ’j’ share a colonial relationship

year Year of observation

exp_year Exporter ISO country code and year

imp_year Importer ISO country code and year

Source

Advanced Guide to Trade Policy Analysis (ISBN: 978-92-870-4367-2)

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 from the hessian, the scores, or a combination of both after convergence.

Usage

```
## S3 method for class 'feglm'
vcov(
  object,
  type = c("hessian", "outer.product", "sandwich", "clustered"),
  ...
)
```

Arguments

`object`	an object of class `"feglm"`.
`type`	the type of covariance estimate required. `"hessian"` refers to the inverse of the negative expected hessian after convergence and is the default option. `"outer.product"` is the outer-product-of-the-gradient estimator. `"sandwich"` is the sandwich estimator (sometimes also referred as robust estimator), and `"clustered"` computes a clustered covariance matrix given some cluster variables.
`...`	additional arguments.

Value

A named matrix of covariance estimates.

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

```
# same as the example in feglm but extracting the covariance matrix

# subset trade flows to avoid fitting time warnings during check
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]

mod <- fepoisson(
  trade ~ log_dist + lang + cntg + clny | exp_year + imp_year | pair,
  trade_2006
)

round(vcov(mod, type = "clustered"), 5)
```

`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 from the hessian, the scores, or a combination of both after convergence.

Usage

```
## S3 method for class 'felm'
vcov(
  object,
  type = c("hessian", "outer.product", "sandwich", "clustered"),
  ...
)
```

Arguments

`object`	an object of class `"felm"`.
`type`	the type of covariance estimate required. `"hessian"` refers to the inverse of the negative expected hessian after convergence and is the default option. `"outer.product"` is the outer-product-of-the-gradient estimator. `"sandwich"` is the sandwich estimator (sometimes also referred as robust estimator), and `"clustered"` computes a clustered covariance matrix given some cluster variables.
`...`	additional arguments.

Value

A named matrix of covariance estimates.

See Also

felm

Examples

```
# same as the example in felm but extracting the covariance matrix

# subset trade flows to avoid fitting time warnings during check
set.seed(123)
trade_2006 <- trade_panel[trade_panel$year == 2006, ]
trade_2006 <- trade_2006[sample(nrow(trade_2006), 500), ]

mod <- felm(
  trade ~ log_dist + lang + cntg + clny | exp_year + imp_year | pair,
  trade_2006
)

round(vcov(mod, type = "clustered"), 5)
```

Index