Package 'ngspatial'

October 13, 2022

Version 1.2-2

Date 2020-05-08

Title Fitting the Centered Autologistic and Sparse Spatial Generalized Linear Mixed Models for Areal Data

Type Package

Author John Hughes <drjphughesjr@gmail.com> and Xiaohui Cui

Maintainer John Hughes <drjphughesjr@gmail.com>

Depends Rcpp, batchmeans

Suggests parallel, pbapply

Description Provides tools for analyzing spatial data, especially non-Gaussian areal data. The current version supports the sparse restricted spatial regression model of Hughes and Haran (2013) <DOI:10.1111/j.1467-9868.2012.01041.x>, the centered autologistic model of Caragea and Kaiser (2009) <DOI:10.1198/jabes.2009.07032>, and the Bayesian spatial filtering model of Hughes (2017) <arXiv:1706.04651>.

License GPL (>= 2)

LinkingTo Rcpp, RcppArmadillo

RcppModules ngspatialmod

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-05-08 16:10:03 UTC

R topics documented:

`A` *Adjacency matrix for the infant mortality data.*

Description

A contains the adjacency matrix for the infant mortality data analyzed in (Hughes and Haran, 2013).

Usage

```
data(A)
```

Source

The data were obtained from the 2008 Area Resource File (ARF), a county-level database maintained by the Bureau of Health Professions, Health Resources and Services Administration, US Department of Health and Human Services.

Examples

```
data(A)
```

`adjacency.matrix` *Return an adjacency matrix for a square lattice.*

Description

Return an adjacency matrix for a square lattice.

Usage

```
adjacency.matrix(m, n = NULL)
```

Arguments

`m`	the number of rows in the lattice.
`n`	the number of columns in the lattice. Defaults to `NULL`. If missing, the lattice is assumed to be `m` by `m`.

Details

This function builds the adjacency matrix for the `m` by `n` square lattice.

Value

A matrix A of 0s and 1s, where A_{ij} is equal to 1 iff vertices i and j are adjacent.

`autologistic` *Fit a centered autologistic model using maximum pseudolikelihood estimation or MCMC for Bayesian inference.*

Description

Fit a centered autologistic model using maximum pseudolikelihood estimation or MCMC for Bayesian inference.

Usage

```
autologistic(formula, data, A, method = c("PL", "Bayes"), model = TRUE,
  x = FALSE, y = FALSE, verbose = FALSE, control = list())
```

Arguments

`formula`	an object of class `formula`: a symbolic description of the model to be fitted.
`data`	an optional data frame, list, or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `autologistic` is called.
`A`	the adjacency matrix for the underlying graph. The matrix need not be binary, but it must be numeric and symmetric.
`method`	the method to use for inference. "`PL`" (the default) enables maximum pseudolikelihood estimation, and "`Bayes`" enables Bayesian inference.
`model`	a logical value indicating whether the model frame should be included as a component of the returned value.
`x`	a logical value indicating whether the model matrix used in the fitting process should be returned as a component of the returned value.
`y`	a logical value indicating whether the response vector used in the fitting process should be returned as a component of the returned value.
`verbose`	a logical value indicating whether to print various messages to the screen, including progress updates when `method` is "`Bayes`". Defaults to `FALSE`.
`control`	a list of the following control parameters.
	`confint` (PL) the method for computing confidence intervals. The possible values are "`sandwich`" (the default), "`bootstrap`", and "`none`".

`sigma` (Bayes) the common standard deviation for β's prior distribution. Defaults to 1,000.

`eta.max` (Bayes) the right endpoint for η's prior distribution. Defaults to 2.

`bootit` (PL) the size of the bootstrap sample. This applies when `confint` is "`sandwich`" or "`bootstrap`", since samples from the fitted model are needed in both cases. Defaults to 1,000.

`trainit` (Bayes) the length of the training run. Defaults to 100,000.

`minit` (Bayes) the minimum number of posterior samples. Defaults to 100,000.

`maxit` (Bayes) the maximum number of posterior samples. Defaults to 1,000,000.

`tol` (Bayes) the tolerance. Defaults to 0.01.

`parallel` (PL) a logical value indicating whether to parallelize the bootstrap. This defaults to `TRUE` if the **parallel** package is available.

`type` (PL) the cluster type, one of "`FORK`", "`MPI`", "`NWS`", "`PSOCK`", or "`SOCK`" (default).

`nodes` (PL) the number of slave nodes to create.

Details

This function fits the centered autologistic model of Caragea and Kaiser (2009) using maximum pseudolikelihood estimation or MCMC for Bayesian inference. The joint distribution for the centered autologistic model is

$$\pi(Z \mid \theta) = c(\theta)^{-1} \exp\left(Z'X\beta - \eta Z'A\mu + \frac{\eta}{2} Z'AZ\right),$$

where $\theta = (\beta', \eta)'$ is the parameter vector, $c(\theta)$ is an intractable normalizing function, Z is the response vector, X is the design matrix, β is a $(p-1)$-vector of regression coefficients, A is the adjacency matrix for the underlying graph, μ is the vector of independence expectations, and η is the spatial dependence parameter.

Maximum pseudolikelihood estimation sidesteps the intractability of $c(\theta)$ by maximizing the product of the conditional likelihoods. Confidence intervals are then obtained using a parametric bootstrap or a normal approximation, i.e., sandwich estimation. The bootstrap datasets are generated by perfect sampling (`rautologistic`). The bootstrap samples can be generated in parallel using the **parallel** package.

Bayesian inference is obtained using the auxiliary variable algorithm of Moller et al. (2006). The auxiliary variables are generated by perfect sampling.

The prior distributions are (1) zero-mean normal with independent coordinates for β, and (2) uniform for η. The common standard deviation for the normal prior can be supplied by the user as control parameter `sigma`. The default is 1,000. The uniform prior has support [0, 2] by default, but the right endpoint can be supplied (as control parameter `eta.max`) by the user.

The posterior covariance matrix of θ is estimated using samples obtained during a training run. The default number of iterations for the training run is 100,000, but this can be controlled by the user (via parameter `trainit`). The estimated covariance matrix is then used as the proposal variance for a Metropolis-Hastings random walk. The proposal distribution is normal. The posterior samples

obtained during the second run are used for inference. The length of the run can be controlled by the user via parameters `minit`, `maxit`, and `tol`. The first determines the minimum number of iterations. If `minit` has been reached, the sampler will terminate when `maxit` is reached or all Monte Carlo standard errors are smaller than `tol`, whichever happens first. The default values for `minit`, `maxit`, and `tol` are 100,000; 1,000,000; and 0.01, respectively.

Value

`autologistic` returns an object of class "`autologistic`", which is a list containing the following components.

`coefficients`	the point estimate of θ.
`fitted.values`	the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
`linear.predictors`	the linear fit on link scale.
`residuals`	the response residuals.
`iter`	the size of the bootstrap/posterior sample.
`sample`	(where relevant) an `iter` by p matrix containing the bootstrap/posterior samples.
`mcse`	(where relevant) a p-vector of Monte Carlo standard errors.
`S`	(where relevant) the estimated sandwich matrix.
`accept`	(Bayes) the acceptance rate for the MCMC sampler.
`y`	if requested (the default), the `y` vector used.
`X`	if requested, the model matrix.
`model`	if requested (the default), the model frame.
`call`	the matched call.
`formula`	the formula supplied.
`method`	the method used for inference.
`convergence`	the integer code returned by `optim` subsequent to optimizing the pseudolikelihood.
`message`	a character string to go along with `convergence`.
`terms`	the `terms` object used.
`data`	the `data` argument.
`xlevels`	(where relevant) a record of the levels of the factors used in fitting.
`control`	a list containing the names and values of the control parameters.
`value`	the value of the negative log pseudolikelihood at its minimum.

References

Caragea, P. and Kaiser, M. (2009) Autologistic models with interpretable parameters. *Journal of Agricultural, Biological, and Environmental Statistics*, **14**(3), 281–300.

Hughes, J., Haran, M. and Caragea, P. C. (2011) Autologistic models for binary data on a lattice. *Environmetrics*, **22**(7), 857–871.

Moller, J., Pettitt, A., Berthelsen, K., and Reeves, R. (2006) An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika*, **93**(2), 451–458.

See Also

rautologistic, residuals.autologistic, summary.autologistic, vcov.autologistic

Examples

```
# Use the 20 x 20 square lattice as the underlying graph.

n = 20
A = adjacency.matrix(n)

# Assign coordinates to each vertex such that the coordinates are restricted to the unit square
# centered at the origin.

x = rep(0:(n - 1) / (n - 1), times = n) - 0.5
y = rep(0:(n - 1) / (n - 1), each = n) - 0.5
X = cbind(x, y)                          # Use the vertex locations as spatial covariates.
beta = c(2, 2)                                  # These are the regression coefficients.
col1 = "white"
col2 = "black"
colfunc = colorRampPalette(c(col1, col2))

# Simulate a dataset with the above mentioned regression component and eta equal to 0.6. This
# value of eta corresponds to dependence that is moderate in strength.

theta = c(beta, eta = 0.6)
set.seed(123456)
Z = rautologistic(X, A, theta)

# Find the MPLE, and do not compute confidence intervals.

fit = autologistic(Z ~ X - 1, A = A, control = list(confint = "none"))
summary(fit)

# The following examples are not executed by default since the computation is time consuming.

# Compute confidence intervals based on the normal approximation. Estimate the "filling" in the
# sandwich matrix using a parallel parametric bootstrap, where the computation is distributed
# across six cores on the host workstation.

# set.seed(123456)
# fit = autologistic(Z ~ X - 1, A = A, verbose = TRUE,
#                    control = list(confint = "sandwich", nodes = 6))
# summary(fit)

# Compute confidence intervals based on a parallel parametric bootstrap. Use a bootstrap sample
# of size 500, and distribute the computation across six cores on the host workstation.

# set.seed(123456)
# fit = autologistic(Z ~ X - 1, A = A, verbose = TRUE,
#                    control = list(confint = "bootstrap", bootit = 500, nodes = 6))
# summary(fit)
```

```
# Do MCMC for Bayesian inference. The length of the training run will be 10,000, and
# the length of the subsequent inferential run will be at least 10,000.

# set.seed(123456)
# fit = autologistic(Z ~ X - 1, A = A, verbose = TRUE, method = "Bayes",
#                    control = list(trainit = 10000, minit = 10000, sigma = 1000))
# summary(fit)
```

`infant` *Infant mortality data.*

Description

infant contains the infant mortality data analyzed in (Hughes and Haran, 2013).

Usage

```
data(infant)
```

Source

The data were obtained from the 2008 Area Resource File (ARF), a county-level database maintained by the Bureau of Health Professions, Health Resources and Services Administration, US Department of Health and Human Services.

Examples

```
data(infant)
```

`negbinomial` *Family function for negative binomial GLMs.*

Description

Provides the information required to apply a sparse SGLMM to conditionally negative binomial outcomes.

Usage

```
negbinomial(theta = stop("'theta' must be specified."), link = "log")
```

Arguments

`theta`	the dispersion parameter (must be positive).
`link`	the link function, as a character string, name, or one-element character vector, specifying one of `log`, `sqrt`, or `identity`, or an object of class "`link-glm`"

Value

An object of class "family", a list of functions and expressions needed to fit a negative binomial GLM.

`rautologistic` *Return a perfect sample from a centered autologistic model.*

Description

Return a perfect sample from a centered autologistic model.

Usage

```
rautologistic(X, A, theta)
```

Arguments

`X`	the design matrix.
`A`	the adjacency matrix for the underlying graph. The matrix need not be binary, but it must be numeric and symmetric.
`theta`	the vector of parameter values: $\theta = (\beta', \eta)'$.

Details

This function implements a perfect sampler for the centered autologistic model. The sampler employs coupling from the past.

Value

A vector that is distributed exactly according to the centered autologistic model with the given design matrix and parameter values.

References

Moller, J. (1999) Perfect simulation of conditionally specified models. *Journal of the Royal Statistical Society, Series B, Methodological*, **61**, 251–264.

Propp, J. G. and Wilson, D. B. (1996) Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, **9**(1-2), 223–252.

Examples

```
# Use the 20 x 20 square lattice as the underlying graph.

n = 20
A = adjacency.matrix(n)

# Assign coordinates to each vertex such that the coordinates are restricted to the unit square
# centered at the origin.

x = rep(0:(n - 1) / (n - 1), times = n) - 0.5
y = rep(0:(n - 1) / (n - 1), each = n) - 0.5
X = cbind(x, y)                          # Use the vertex locations as spatial covariates.
beta = c(2, 2)                                  # These are the regression coefficients.

# Simulate a dataset with the above mentioned regression component and eta equal to 0.6. This
# value of eta corresponds to dependence that is moderate in strength.

theta = c(beta, eta = 0.6)
set.seed(123456)
Z = rautologistic(X, A, theta)
```

---

residuals.autologistic

Extract model residuals.

Description

Extract model residuals.

Usage

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

Arguments

object	an object of class autologistic, typically the result of a call to autologistic.
type	the type of residuals that should be returned. The alternatives are "deviance" (default), "pearson", and "response".
...	additional arguments.

Value

A vector of residuals.

See Also

autologistic, residuals.glm

`residuals.sparse.sglmm` *Extract model residuals.*

Description

Extract model residuals.

Usage

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

Arguments

`object`	an object of class `sparse.sglmm`, typically the result of a call to `sparse.sglmm`.
`type`	the type of residuals that should be returned. The alternatives are "`deviance`" (default), "`pearson`", and "`response`".
`...`	additional arguments.

Value

A vector of residuals.

See Also

sparse.sglmm, residuals.glm

`sparse.sglmm` *Fit a sparse SGLMM.*

Description

Fit a sparse SGLMM.

Usage

```
sparse.sglmm(formula, family = gaussian, data, offset, A, method = c("BSF",
  "RSR"), attractive = 50, repulsive = 0, tol = 0.01, minit = 10000,
  maxit = 1e+06, tune = list(), hyper = list(), model = TRUE,
  x = FALSE, y = FALSE, verbose = FALSE, parallel = FALSE)
```

Arguments

formula	an object of class formula: a symbolic description of the model to be fitted.
family	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function, or the result of a call to a family function. (See family for details of family functions.) Supported families are binomial, gaussian (default), negbinomial, and poisson.
data	an optional data frame, list, or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which sparse.sglmm is called.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
A	the adjacency matrix for the underlying graph.
method	the basis to use. The options are Bayesian spatial filtering ("BSF") and restricted spatial regression ("RSR").
attractive	the number of attractive Moran eigenvectors to use. The default is 50. See 'Details' for more information.
repulsive	the number of repulsive Moran eigenvectors to use. The default is 0. See 'Details' for more information.
tol	a tolerance. If all Monte Carlo standard errors are smaller than tol, no more samples are drawn from the posterior. The default is 0.01.
minit	the minimum sample size. This should be large enough to permit accurate estimation of Monte Carlo standard errors. The default is 10,000.
maxit	the maximum sample size. Sampling from the posterior terminates when all Monte Carlo standard errors are smaller than tol or when maxit samples have been drawn, whichever happens first. The default is 1,000,000.
tune	(where relevant) a list containing sigma.s, sigma.h, and sigma.theta. These are the standard deviations for the γ, δ, and θ proposals, respectively.
hyper	a list containing sigma.b, the prior standard deviation for β, and (where relevant) a.h and b.h, the parameters of the gamma prior for τ_h.
model	a logical value indicating whether the model frame should be included as a component of the returned value.
x	a logical value indicating whether the model matrix used in the fitting process should be returned as a component of the returned value.
y	a logical value indicating whether the response vector used in the fitting process should be returned as a component of the returned value.
verbose	a logical value indicating whether to print MCMC progress to the screen. Defaults to FALSE.

`parallel` (for parallelized computation of the Moran operator) a list containing `type` and `nodes`, the cluster type and number of slave nodes, respectively. The former must be one of "`FORK`", "`MPI`", "`NWS`", "`PSOCK`", or "`SOCK`" (default). The latter must be a whole number greater than 1. This argument defaults to `FALSE`, in which case the matrix multiplications are not parallelized.

Details

This function fits the sparse restricted spatial regression model of Hughes and Haran (2013), or the Bayesian spatial filtering model of Hughes (2017). The first stage of the model is

$$g(\mu_i) = x_i'\beta + m_i'\gamma \qquad (i = 1, \ldots, n)$$

or, in vectorized form,

$$g(\mu) = X\beta + M\gamma,$$

where X is the design matrix, β is a p-vector of regression coefficients, the columns of M are q eigenvectors of the Moran operator, and γ are spatial random effects. Arguments `attractive` and `repulsive` can be used to control the number of eigenvectors used. The default values are 50 and 0, respectively, which corresponds to pure spatial smoothing. Inclusion of some repulsive eigenvectors can be advantageous in certain applications.

The second stage, i.e., the prior for γ, is

$$p(\gamma \mid \tau_s) \propto \tau_s^{q/2} \exp\left(-\frac{\tau_s}{2}\gamma' M'QM\gamma\right),$$

where τ_s is a smoothing parameter and Q is the graph Laplacian.

The prior for β is spherical p-variate normal with mean zero and common standard deviation `sigma.b`, which defaults to 1,000. The prior for τ_s is gamma with parameters 0.5 and 2,000. The same prior is used for θ (when family is `negbinomial`).

When the response is normally distributed, the identity link is assumed, in which case the first stage is

$$\mu = X\beta + M\gamma + M\delta,$$

where δ are heterogeneity random effects. When the response is Poisson distributed, heterogeneity random effects are optional. In any case, the prior on δ is spherical q-variate normal with mean zero and common variance $1/\tau_h$. The prior for τ_h is gamma with parameters a_h and b_h, the values of which are controlled by the user through argument `hyper`.

If the response is Bernoulli, negative binomial, or Poisson, β and γ are updated using Metropolis-Hastings random walks with normal proposals. The proposal covariance matrix for β is the estimated asymptotic covariance matrix from a `glm` fit to the data (see `vcov`). The proposal for γ is spherical normal with common standard deviation `sigma.s`.

The updates for τ_s and τ_h are Gibbs updates irrespective of the response distribution.

If the response is Poisson distributed and heterogeneity random effects are included, those random effects are updated using a Metropolis-Hastings random walk with a spherical normal proposal.

The common standard deviation is `sigma.h`.

If the response is normally distributed, all updates are Gibbs updates.

If the response is negative binomial, the dispersion parameter θ is updated using a Metropolis-Hastings random walk with a normal proposal. Said proposal has standard deviation `sigma.theta`, which can be provided by the user as an element of argument `tune`.

Value

`sparse.sglmm` returns an object of class "`sparse.sglmm`", which is a list containing the following components.

`coefficients`	the estimated regression coefficients.
`fitted.values`	the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
`linear.predictors`	the linear fit on link scale.
`residuals`	the response residuals.
`iter`	the size of the posterior sample.
`beta.sample`	an `iter` by p matrix containing the posterior samples for β.
`gamma.sample`	an `iter` by q matrix containing the posterior samples for γ.
`delta.sample`	(where relevant) an `iter` by q matrix containing the posterior samples for δ.
`theta.sample`	(where relevant) a vector containing the posterior samples for θ.
`tau.s.sample`	a vector containing the posterior samples for τ_s.
`tau.h.sample`	(where relevant) a vector containing the posterior samples for τ_h.
`gamma.est`	the estimate of γ.
`delta.est`	(where relevant) the estimate of δ.
`tau.s.est`	the estimate of τ_s.
`tau.h.est`	(where relevant) the estimate of τ_h.
`theta.est`	(where relevant) the estimate of θ.
`beta.mcse`	the Monte Carlo standard errors for β.
`gamma.mcse`	the Monte Carlo standard errors for γ.
`delta.mcse`	(where relevant) the Monte Carlo standard errors for δ.
`tau.s.mcse`	the Monte Carlo standard error for τ_s.
`tau.h.mcse`	(where relevant) the Monte Carlo standard error for τ_h.
`theta.mcse`	(where relevant) the Monte Carlo standard error for θ.
`D.bar`	the goodness of fit component of the DIC.
`pD`	the penalty component of the DIC.
`dic`	the deviance information criterion.
`beta.accept`	the acceptance rate for β.

gamma.accept	the acceptance rate for γ.
delta.accept	(where relevant) the acceptance rate for δ.
theta.accept	(where relevant) the acceptance rate for θ.
y	if requested (the default), the y vector used.
X	if requested, the model matrix.
M	if requested, the matrix of Moran eigenvectors.
eigen.values	if requested, the spectrum of the Moran operator.
hyper	a list containing the names and values of the hyperparameters.
tune	a list containing the names and values of the tuning parameters.
model	if requested (the default), the model frame.
call	the matched call.
formula	the formula supplied.
terms	the terms object used.
data	the data argument.
offset	the offset vector used.
xlevels	(where relevant) a record of the levels of the factors used in fitting.

References

Hughes, J. and Haran, M. (2013) Dimension reduction and alleviation of confounding for spatial generalized linear mixed models. *Journal of the Royal Statistical Society, Series B*, **75**(1), 139–159.

See Also

residuals.sparse.sglmm, summary.sparse.sglmm, vcov.sparse.sglmm

Examples

```
## Not run:

The following code duplicates the analysis described in (Hughes and Haran, 2013). The data are
infant mortality data for 3,071 US counties. We do a spatial Poisson regression with offset.

data(infant)
infant$low_weight = infant$low_weight / infant$births
attach(infant)
Z = deaths
X = cbind(1, low_weight, black, hispanic, gini, affluence, stability)
data(A)
set.seed(123456)
fit = sparse.sglmm(Z ~ X - 1 + offset(log(births)), family = poisson, A = A, method = "RSR",
                   tune = list(sigma.s = 0.02), verbose = TRUE)
summary(fit)

## End(Not run)
```

summary.autologistic *Print a summary of a centered autologistic model fit.*

Description

Print a summary of a centered autologistic model fit.

Usage

```
## S3 method for class 'autologistic'
summary(object, alpha = 0.05, digits = 4, ...)
```

Arguments

object	an object of class autologistic, typically the result of a call to autologistic.
alpha	the significance level for the quantile/HPD intervals. The default is 0.05.
digits	the number of significant digits to display. The default is 4.
...	additional arguments.

Details

This function displays (1) the call to autologistic, (2) the values of the control parameters, (3) a table of estimates, and (4) the size of the bootstrap/posterior sample. Each row of the table of estimates shows a parameter estimate, the confidence or HPD interval for the parameter, and, where applicable, the Monte Carlo standard error.

See Also

autologistic

summary.sparse.sglmm *Print a summary of a sparse SGLMM fit.*

Description

Print a summary of a sparse SGLMM fit.

Usage

```
## S3 method for class 'sparse.sglmm'
summary(object, alpha = 0.05, digits = 4, ...)
```

Arguments

object	an object of class sparse.sglmm, typically the result of a call to sparse.sglmm.
alpha	the significance level used to compute the HPD intervals. The default is 0.05.
digits	the number of significant digits to display. The default is 4.
...	additional arguments.

Details

This function displays (1) the call to sparse.sglmm, (2) the values of the hyperparameters and tuning parameters, (3) a table of estimates, (4) the DIC value for the fit, and (5) the number of posterior samples. Each row of the table of estimates shows an estimated regression coefficient, the HPD interval for the coefficient, and the Monte Carlo standard error.

See Also

sparse.sglmm

---

vcov.autologistic *Return the estimated covariance matrix for an* autologistic *model object.*

Description

Return the estimated covariance matrix for an autologistic model object.

Usage

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

Arguments

object	a fitted autologistic model object.
...	additional arguments.

Value

An estimate of the covariance matrix of the parameters (in a Bayesian setting), or an estimate of the covariance matrix of the maximum pseudolikelihood estimator of the parameters.

`vcov.sparse.sglmm` *Return the covariance matrix of the regression parameters of a* `sparse.sglmm` *model object.*

Description

Return the covariance matrix of the regression parameters of a `sparse.sglmm` model object.

Usage

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

Arguments

`object`	a fitted `sparse.sglmm` model object.
`...`	additional arguments.

Value

An estimate of the posterior covariance matrix of the regression coefficients.

Index