Package 'cobin'

September 1, 2025

Title Cobin and Micobin Regression Models for Continuous Proportional Data

Version 1.0.1.3

Description Provides functions for cobin and micobin regression models, a new family of generalized linear models for continuous proportional data (Y in the closed unit interval [0, 1]). It also includes an exact, efficient sampler for the Kolmogorov-Gamma random variable. For details, see Lee et al. (2025+) <doi:10.48550/arXiv.2504.15269>.

License MIT + file LICENSE

URL https://github.com/changwoo-lee/cobin

BugReports https://github.com/changwoo-lee/cobin/issues

Encoding UTF-8

RoxygenNote 7.3.2

LinkingTo Rcpp

Imports coda, fields, lme4, Matrix, matrixStats, methods, Rcpp, spam, spNNGP, stats, utils

Suggests betareg

NeedsCompilation yes

Author Changwoo Lee [aut, cre],
Benjamin Dahl [aut],
Otso Ovaskainen [aut],
David Dunson [aut]

Maintainer Changwoo Lee <changwoo.lee@duke.edu>

Repository CRAN

Date/Publication 2025-09-01 17:10:27 UTC

Contents

`bft` *Cumulant (log partition) function of cobin*

Description

$B(x) = \log\{(\exp(x) - 1)/x)\}$

Usage

```
bft(x)
```

Arguments

`x`	input vector

Value

$B(x) = \log\{(\exp(x) - 1)/x)\}$

bftprime *First derivative of cobin cumulant (log partition) function*

Description

$B'(x) = 1/(1 - \exp(-x)) - 1/x$. When g is canonical link of cobin, this is same as g^{-1}

Usage

```
bftprime(x)

cobitlinkinv(x)
```

Arguments

x	input vector

Value

$B'(x) = 1/(1 - \exp(-x)) - 1/x$.

bftprimeinv *Inverse of first derivative of cobin cumulant (log partition) function*

Description

Calculates $(B')^{-1}(y)$ using numerical inversion (Newton-Raphson), where $B'(x) = 1/(1-\exp(-x))-1/x$. This is the cobit link function g, the canonical link function of cobin.

Usage

```
bftprimeinv(y, x0 = 0, tol = 1e-08, maxiter = 100)

cobitlink(y, x0 = 0, tol = 1e-08, maxiter = 100)
```

Arguments

y	input vector
x0	Defult 0, initial value
tol	tolerance, stopping criterion for Newton-Raphson
maxiter	max iteration of Newton-Raphson

Value

$(B')^{-1}(y)$

bftprimeprime *Second derivative of cobin cumulant (log partition) function*

Description

$B''(x) = 1/x^2 + 1/(2 - 2 * \cosh(x))$ used Taylor series expansion for x near 0 for stability

Usage

```
bftprimeprime(x)
```

Arguments

x	input vector

Value

$B''(x) = 1/x^2 + 1/(2 - 2 * \cosh(x))$

bftprimeprimeprime *Third derivative of cobin cumulant (log partition) function*

Description

$B'''(x) = 1/(4 * \tanh(x/2) * \sinh(x/2)^2) - 2/x^3$ used Taylor series expansion for x near 0 for stability

Usage

```
bftprimeprimeprime(x)
```

Arguments

x	input vector

Value

$B'''(x) = 1/(4 * \tanh(x/2) * \sinh(x/2)^2) - 2/x^3$

`cobinfamily` *cobin family class*

Description

Specifies the information required to fit a cobin generalized linear model with known lambda parameter, using glm().

Usage

```
cobinfamily(lambda = stop("'lambda' must be specified"), link = "cobit")
```

Arguments

`lambda`	The known value of lambda, must be integer
`link`	The link function to be used. Options are "cobit" (canonical link for cobin regression), "logit", "probit", "cauchit", "cloglog"

Value

An object of class "family", a list of functions and expressions needed by glm() to fit a cobin generalized linear model.

`cobinreg` *cobin generalized linear (mixed) models*

Description

Fit Bayesian cobin regression model under canonical link (cobit link) with Markov chain Monte Carlo (MCMC). It supports both fixed-effect only model

$$y_i \mid x_i \overset{ind}{\sim} cobin(x_i^T\beta, \lambda^{-1}),$$

for $i = 1, \ldots, n$, and random intercept model (v 1.0.x only supports random intercept),

$$y_{ij} \mid x_{ij}, u_i \overset{ind}{\sim} cobin(x_{ij}^T\beta + u_i, \lambda^{-1}), \quad u_i \overset{iid}{\sim} N(0, \sigma_u^2)$$

for $i = 1, \ldots, n$ (group), and $j = 1, \ldots, n_i$ (observation within group). See dcobin for details on cobin distribution.

Usage

```
cobinreg(
  formula,
  data,
  link = "cobit",
  contrasts = NULL,
  priors = list(beta_intercept_scale = 100, beta_scale = 100, beta_df = Inf),
  nburn = 1000,
  nsave = 1000,
  nthin = 1,
  MH = FALSE,
  lambda_fixed = NULL
)
```

Arguments

formula	an object of class "formula" or a two-sided linear formula object describing both the fixed-effects and random-effects part of the model; see "lmer"
data	data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model.
link	character, link function (default "cobit"). Only supports canonical link function "cobit" that is compatible with Kolmogorov-Gamma augmentation.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
priors	a list of prior hyperparameters. See Details
nburn	number of burn-in MCMC iterations.
nsave	number of posterior samples. Total MCMC iteration is nburn + nsave*nthin
nthin	thin-in rate. Total MCMC iteration is nburn + nsave*nthin
MH	logical, Metropolis-Hastings; experimental
lambda_fixed	logical, fixing lambda; experimental

Details

The prior setting can be controlled with "priors" argument. Prior for regression coefficients are independent normal or t prior centered at 0. "priors" is a named list of:

- beta_intercept_scale, Default 100, the scale of the intercept prior
- beta_scale, Default 100, the scale of nonintercept fixed-effect coefficients
- beta_df, Default Inf, degree of freedom of t prior. If `beta_df=Inf`, it corresponds to normal prior
- lambda_grid, Default 1:70, candidate for lambda (integer)
- lambda_logprior, Default $p(\lambda) \propto \lambda\Gamma(\lambda+1)/\Gamma(\lambda+5)$, log-prior of lambda. Default choice arises from beta negative binomial distribution; $(\lambda-1) \mid \psi \sim negbin(2,\psi), \psi \sim Beta(2,2)$.

if random intercept model, u ~ InvGamma(a_u,b_u) with

- a_u, Default 1, first parameter of Inverse Gamma prior of u
- b_u, Default 1, second parameter of Inverse Gamma prior of u

Value

Returns list of

post_save	a matrix of posterior samples (coda::mcmc) with nsave rows
loglik_save	a nsave x n matrix of pointwise log-likelihood values, can be used for WAIC calculation.
priors	list of hyperprior information
nsave	number of MCMC samples
t_mcmc	wall-clock time for running MCMC
t_premcmc	wall-clock time for preprocessing before MCMC
y	response vector
X	fixed effect design matrix

if random effect model, also returns

post_u_save	a matrix of posterior samples (coda::mcmc) of random effects
Z	random effect design matrix

Examples

```
requireNamespace("betareg", quietly = TRUE)
library(betareg) # for dataset example
data("GasolineYield", package = "betareg")

# basic model
out1 = cobinreg(yield ~ temp, data = GasolineYield,
               nsave = 2000, link = "cobit")
summary(out1$post_save)
plot(out1$post_save)

# random intercept model
out2 = cobinreg(yield ~ temp + (1 | batch), data = GasolineYield,
               nsave = 2000, link = "cobit")
summary(out2$post_save)
plot(out2$post_save)
```

---

dcobin *Density function of cobin (continuous binomial) distribution*

Description

Continuous binomial distribution with natural parameter θ and dispersion parameter $1/\lambda$, in short $Y \sim cobin(\theta, \lambda^{-1})$, has density

$$p(y; \theta, \lambda^{-1}) = h(y; \lambda) \exp(\lambda\theta y - \lambda B(\theta)), \quad 0 \le y \le 1$$

where $B(\theta) = \log\{(e^\theta - 1)/\theta\}$ and $h(y; \lambda) = \frac{\lambda}{(\lambda-1)!} \sum_{k=0}^{\lambda} (-1)^k \binom{\lambda}{k} \max(0, \lambda y - k)^{\lambda-1}$. When $\lambda = 1$, it becomes continuous Bernoulli distribution.

Usage

```
dcobin(x, theta, lambda, log = FALSE)
```

Arguments

x	num (length n), between 0 and 1, evaluation point
theta	scalar or length n vector, num (length 1 or n), natural parameter
lambda	scalar or length n vector, integer, inverse of dispersion parameter
log	logical (Default FALSE), if TRUE, return log density

Details

For the evaluation of $h(y; \lambda)$, see `?cobin::dIH`.

Value

density of $cobin(\theta, \lambda^{-1})$

Examples

```
xgrid = seq(0, 1, length = 500)
plot(xgrid, dcobin(xgrid, 0, 1), type="l", ylim = c(0,3)) # uniform
lines(xgrid, dcobin(xgrid, 0, 3))
plot(xgrid, dcobin(xgrid, 2, 3), type="l")
lines(xgrid, dcobin(xgrid, -2, 3))
```

---

dIH *Density of Irwin-Hall distribution*

Description

Irwin-Hall distribution is defined as a sum of m uniform (0,1) distribution. Its density is given as

$$f(x; m) = \frac{1}{(m-1)!} \sum_{k=0}^{m} (-1)^k \binom{m}{k} \max(0, x-k)^{m-1}, 0 < x < m$$

The density of Bates distribution, defined as an average of m uniform (0,1) distribution, can be obtained from change-of-variable (y = x/m),

$$h(y; m) = \frac{m}{(m-1)!} \sum_{k=0}^{m} (-1)^k \binom{m}{k} \max(0, my-k)^{m-1}, 0 < y < 1$$

Usage

```
dIH(x, m, log = FALSE)
```

Arguments

x	vector of quantities, between 0 and m
m	integer, parameter
log	logical, return log density if TRUE

Details

Due to alternating series representation, m > 80 may yield numerical issues

Value

(log) density evaluated at x

Examples

```
m = 8
xgrid= seq(0, m, length = 500)
hist(colSums(matrix(runif(m*1000), nrow = m, ncol = 1000)), freq = FALSE)
lines(xgrid, dIH(xgrid, m, log = FALSE))
# Bates distribution
xgrid= seq(0, 1, length = 500)
hist(colMeans(matrix(runif(m*1000), nrow = m, ncol = 1000)), freq = FALSE)
lines(xgrid, m*dIH(xgrid*m, m, log = FALSE))
```

---

dmicobin *Density function of micobin (mixture of continuous binomial) distribution*

Description

Micobin distribution with natural parameter θ and dispersion psi, denoted as $micobin(\theta, \psi)$, is defined as a dispersion mixture of cobin:

$$Y \sim micobin(\theta, \psi) \iff Y|\lambda \sim cobin(\theta, \lambda^{-1}), (\lambda - 1) \sim negbin(2, \psi)$$

so that micobin density is a weighted sum of cobin density with negative binomial weights.

Usage

```
dmicobin(x, theta, psi, r = 2, log = FALSE, l_max = 70)
```

Arguments

x	num (length n), between 0 and 1, evaluation point
theta	scalar or length n vector, natural parameter
psi	scalar or length n vector, between 0 and 1, dispersion parameter
r	(Default 2) This should be always 2 to maintain interpretaton of psi. It is kept for future experiment purposes.
log	logical (Default FALSE), if TRUE, return log density
l_max	integer (Default 70), upper bound of lambda.

Value

density of $micobin(\theta, \psi)$

Examples

```
hist(rcobin(1000, 2, 3), freq = FALSE)
xgrid = seq(0, 1, length = 500)
lines(xgrid, dcobin(xgrid, 2, 3))
```

---

glm.cobin *Find the MLE of cobin GLM*

Description

Find the maximum likelihood estimate of a cobin generalized linear model with unknown dispersion. This is a modification of `stats::glm` to include estimation of the additional parameter, lambda, for a cobin generalized linear model, in a similar manner to the `MASS::glm.nb`. Note that MLE of regression coefficient does not depends on lambda.

Usage

```
glm.cobin(
  formula,
  data,
  weights,
  subset,
  na.action,
  start = NULL,
  etastart,
  mustart,
  control = glm.control(...),
  method = "glm.fit",
  model = TRUE,
  x = FALSE,
  y = TRUE,
```

```
  contrasts = NULL,
  ...,
  lambda_list = 1:70,
  link = "cobit",
  verbose = TRUE
)
```

Arguments

formula, data, weights, subset, na.action, start, etastart, mustart, control, method, model, x, y, contrasts, ...	arguments for the stats::glm without family and offset.
lambda_list	(Default 1:70) an integer vector of candidate lambda values. Note that MLE of coefficient does not depends on lambda
link	character, link function. Default cobit. Must be one of "cobit", "logit", "probit", "cloglog", "cauchit".
verbose	logical, if TRUE, print the MLE of lambda.

Details

Since dispersion parameter lambda is discrete, it does not provide standard error of lambda. With cobit link, we strongly encourage Bayesian approaches, using `cobin::cobinreg()` function.

Value

The object is like the output of glm but contains additional components, the ML estimate of lambda and the log-likelihood values for each lambda in the lambda_list.

Examples

```
requireNamespace("betareg", quietly = TRUE)
library(betareg)# for dataset example
data(GasolineYield, package = "betareg")
# cobin regression, frequentist
out_freq = glm.cobin(yield ~ temp, data = GasolineYield, link = "cobit")
summary(out_freq)
# cobin regression, Bayesian
out = cobinreg(yield ~ temp, data = GasolineYield,
               nsave = 10000, link = "cobit")
summary(out$post_save)
plot(out$post_save)
```

micobinreg *micobin generalized linear (mixed) models*

Description

Fit Bayesian micobin regression model under canonical link (cobit link) with Markov chain Monte Carlo (MCMC). It supports both fixed-effect only model

$$y_i \mid x_i \overset{ind}{\sim} micobin(x_i^T\beta, \psi),$$

for $i = 1, \ldots, n$, and random intercept model (v 1.0.x only supports random intercept),

$$y_{ij} \mid x_{ij}, u_i \overset{ind}{\sim} micobin(x_{ij}^T\beta + u_i, \psi), \quad u_i \overset{iid}{\sim} N(0, \sigma_u^2)$$

for $i = 1, \ldots, n$ (group), and $j = 1, \ldots, n_i$ (observation within group). See dmicobin for details on micobin distribution.

Usage

```
micobinreg(
  formula,
  data,
  link = "cobit",
  contrasts = NULL,
  priors = list(beta_intercept_scale = 100, beta_scale = 100, beta_df = Inf),
  nburn = 1000,
  nsave = 1000,
  nthin = 1,
  psi_fixed = NULL
)
```

Arguments

formula	an object of class "formula" or a two-sided linear formula object describing both the fixed-effects and random-effects part of the model; see "lmer"
data	data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model.
link	character, link function (default "cobit"). Only supports canonical link function "cobit" that is compatible with Kolmogorov-Gamma augmentation.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
priors	a list of prior hyperparameters. See Details
nburn	number of burn-in MCMC iterations.
nsave	number of posterior samples. Total MCMC iteration is nburn + nsave*nthin
nthin	thin-in rate. Total MCMC iteration is nburn + nsave*nthin
psi_fixed	logical, fixing psi; experimental

Details

The prior setting can be controlled with "priors" argument. Prior for regression coefficients are independent normal or t prior centered at 0. "priors" is a named list of:

- beta_intercept_scale, Default 100, the scale of the intercept prior
- beta_scale, Default 100, the scale of nonintercept fixed-effect coefficients
- beta_df, Default Inf, degree of freedom of t prior. If `beta_df=Inf`, it corresponds to normal prior
- lambda_max, Default 70, upper bound for lambda (integer)
- psi_ab, Default c(2,2), beta shape parameters for ψ (length 2 vector).

if random intercept model, u ~ InvGamma(a_u,b_u) with

- a_u, Default 1, first parameter of Inverse Gamma prior of u
- b_u, Default 1, second parameter of Inverse Gamma prior of u

Value

Returns list of

`post_save`	a matrix of posterior samples (coda::mcmc) with nsave rows
`loglik_save`	a nsave x n matrix of pointwise log-likelihood values, can be used for WAIC calculation.
`priors`	list of hyperprior information
`nsave`	number of MCMC samples
`t_mcmc`	wall-clock time for running MCMC
`t_premcmc`	wall-clock time for preprocessing before MCMC
`y`	response vector
`X`	fixed effect design matrix

if random effect model, also returns

`post_u_save`	a matrix of posterior samples (coda::mcmc) of random effects
`Z`	random effect design matrix

Examples

```
requireNamespace("betareg", quietly = TRUE)
library(betareg)# for dataset example
data("GasolineYield", package = "betareg")

# basic model
out1 = micobinreg(yield ~ temp, data = GasolineYield,
               nsave = 2000, link = "cobit")
summary(out1$post_save)
plot(out1$post_save)
```

```
# random intercept model
out2 = micobinreg(yield ~ temp + (1 | batch), data = GasolineYield,
               nsave = 2000, link = "cobit")
summary(out2$post_save)
plot(out2$post_save)
```

`pcobin` *Cumulative distribution function of cobin (continuous binomial) distribution*

Description

Continuous binomial distribution with natural parameter θ and dispersion parameter $1/\lambda$, in short $Y \sim cobin(\theta, \lambda^{-1})$, has density

$$p(y; \theta, \lambda^{-1}) = h(y; \lambda) \exp(\lambda\theta y - \lambda B(\theta)), \quad 0 \leq y \leq 1$$

where $B(\theta) = \log\{(e^\theta - 1)/\theta\}$ and $h(y; \lambda) = \frac{\lambda}{(\lambda-1)!} \sum_{k=0}^{\lambda} (-1)^k \binom{\lambda}{k} \max(0, \lambda y - k)^{\lambda-1}$. When $\lambda = 1$, it becomes continuous Bernoulli distribution.

Usage

```
pcobin(q, theta, lambda)
```

Arguments

`q`	num (length n), between 0 and 1, evaluation point
`theta`	scalar, natural parameter
`lambda`	integer, inverse of dispersion parameter

Value

c.d.f. of $cobin(\theta, \lambda^{-1})$

Examples

```
xgrid = seq(0, 1, length = 500)
out = pcobin(xgrid, 1, 2)
plot(ecdf(rcobin(10000, 1, 2)))
lines(xgrid, out, col = 2)
```

`pmicobin` *Cumulative distribution function of micobin (mixture of continuous binomial) distribution*

Description

Micobin distribution with natural parameter θ and dispersion psi, denoted as $micobin(\theta, \psi)$, is defined as a dispersion mixture of cobin:

$$Y \sim micobin(\theta, \psi) \iff Y|\lambda \sim cobin(\theta, \lambda^{-1}), (\lambda - 1) \sim negbin(2, \psi)$$

so that micobin cdf is a weighted sum of cobin cdf with negative binomial weights.

Usage

```
pmicobin(q, theta, psi, r = 2, l_max = 70)
```

Arguments

`q`	num (length n), between 0 and 1, evaluation point
`theta`	scalar, natural parameter
`psi`	scalar, dispersion parameter
`r`	(Default 2) This should be always 2 to maintain interpretaton of psi. It is kept for future experiment purposes.
`l_max`	integer (Default 70), upper bound of lambda.

Value

c.d.f. of $micobin(\theta, \psi)$

Examples

```
xgrid = seq(0, 1, length = 500)
out = pmicobin(xgrid, 1, 1/2)
plot(ecdf(rmicobin(10000, 1, 1/2)))
lines(xgrid, out, col = 2)
```

`rcobin` *Random variate generation for cobin (continuous binomial) distribution*

Description

Continuous binomial distribution with natural parameter θ and dispersion parameter $1/\lambda$, in short $Y \sim cobin(\theta, \lambda^{-1})$, has density

$$p(y; \theta, \lambda^{-1}) = h(y; \lambda) \exp(\lambda\theta y - \lambda B(\theta)), \quad 0 \leq y \leq 1$$

where $B(\theta) = \log\{(e^{\theta} - 1)/\theta\}$ and $h(y; \lambda) = \frac{\lambda}{(\lambda-1)!} \sum_{k=0}^{\lambda} (-1)^k \binom{\lambda}{k} \max(0, \lambda y - k)^{\lambda-1}$. When $\lambda = 1$, it becomes continuous Bernoulli distribution.

Usage

```
rcobin(n, theta, lambda)
```

Arguments

`n`	integer, number of samples
`theta`	scalar or length n vector, natural parameter.
`lambda`	scalar or length n vector, inverse of dispersion parameter. Must be integer, length should be same as theta

Details

The random variate generation is based on the fact that $cobin(\theta, \lambda^{-1})$ is equal in distribution to the sum of λ $cobin(\theta, 1)$ random variables, scaled by λ^{-1}. Random variate generation for continuous Bernoulli is done by inverse cdf transform method.

Value

random samples from $cobin(\theta, \lambda^{-1})$.

Examples

```
hist(rcobin(1000, 2, 3), freq = FALSE)
xgrid = seq(0, 1, length = 500)
lines(xgrid, dcobin(xgrid, 2, 3))
```

rkgcpp *Sample Kolmogorov-Gamma random variables*

Description

A random variable X follows Kolmogorov-Gamma(b,c) distribution, in short KG(b,c), if

$$X \overset{d}{=} \frac{1}{2\pi^2} \sum_{k=1}^{\infty} \frac{\epsilon_k}{k^2 + c^2/(4\pi^2)}, \quad \epsilon_k \overset{iid}{\sim} Gamma(b, 1)$$

where $\overset{d}{=}$ denotes equality in distribution. The random variate generation is based on alternating series method, a fast and exact method (without infinite sum truncation) implemented in cpp. This function only supports integer b, which is sufficient for cobin and micobin regression models.

Usage

```
rkgcpp(n, b, c)
```

Arguments

n	The number of samples.
b	First parameter, positive integer (1,2,...). Length must be 1 or n.
c	Second parameter, real, associated with tilting. Length must be 1 or n.

Value

It returns n independent Kolmogorov-Gamma(b[i],c[i]) samples. If input b or c is scalar, it is assumed to be length n vector with same entries.

Examples

```
rkgcpp(100, 1, 2)
rkgcpp(100, 1, rnorm(100))
rkgcpp(100, rep(c(1,2),50), rnorm(100))
```

rmicobin *Random variate generation for micobin (mixture of continuous binomial) distribution*

Description

Micobin distribution with natural parameter θ and dispersion psi, denoted as $micobin(\theta, \psi)$, is defined as a dispersion mixture of cobin:

$$Y \sim micobin(\theta, \psi) \iff Y|\lambda \sim cobin(\theta, \lambda^{-1}), (\lambda - 1) \sim negbin(2, \psi)$$

Usage

```
rmicobin(n, theta, psi, r = 2)
```

Arguments

n	integer, number of samples
theta	scalar or length n vector, natural parameter
psi	scalar or length n vector, between 0 and 1, dispersion parameter
r	(Default 2) This should be always 2 to maintain interpretaton of psi. It is kept for future experiment purposes.

Value

random samples from $micobin(\theta, \psi)$.

Examples

```
hist(rmicobin(1000, 2, 1/3), freq = FALSE)
xgrid = seq(0, 1, length = 500)
lines(xgrid, dmicobin(xgrid, 2, 1/3))
```

---

spcobinreg *spatial cobin regression model*

Description

Fit Bayesian spatial cobin regression model under canonical link (cobit link) with Markov chain Monte Carlo (MCMC).

$$y(s_i) \mid x(s_i), u(s_i) \overset{ind}{\sim} cobin(x(s_i)^T\beta + u(s_i), \lambda^{-1}), \quad u(\cdot) \sim GP$$

for $i = 1, \ldots, n$. See dcobin for details on cobin distribution. It currently only supports mean zero GP with exponential covariance

$$cov(u(s_i), u(s_j)) = \sigma_u^2 \exp(-\phi_u d(s_i, s_j))$$

where ϕ_u corresponds to inverse range parameter.

Usage

```
spcobinreg(
  formula,
  data,
  link = "cobit",
  coords,
  NNGP = FALSE,
```

```
  contrasts = NULL,
  priors = list(beta_intercept_scale = 10, beta_scale = 2.5, beta_df = Inf),
  nngp.control = list(n.neighbors = 15, ord = order(coords[, 1])),
  nburn = 1000,
  nsave = 1000,
  nthin = 1
)
```

Arguments

formula	an object of class "formula" or a two-sided linear formula object describing both the fixed-effects and random-effects part of the model; see "lmer"
data	data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model.
link	character, link function (default "cobit"). Only supports canonical link function "cobit" that is compatible with Kolmogorov-Gamma augmentation.
coords	a n x 2 matrix of Euclidean coordinates
NNGP	logical, if TRUE, use NNGP prior for the spatial random effects; see spNNGP
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
priors	a list of prior hyperparameters. See Details
nngp.control	a list of control parameters for NNGP prior (only when NNGP = TRUE). This should be a named list of n.neighbors and ord, with default of 15 and first coordiate-based ordering. See spNNGP for details.
nburn	number of burn-in MCMC iterations.
nsave	number of posterior samples. Total MCMC iteration is nburn + nsave*nthin
nthin	thin-in rate. Total MCMC iteration is nburn + nsave*nthin

Details

The prior setting can be controlled with "priors" argument. Prior for regression coefficients are independent normal or t prior centered at 0. "priors" is a named list of:

- beta_intercept_scale, Default 100, the scale of the intercept prior
- beta_scale, Default 100, the scale of nonintercept fixed-effect coefficients
- beta_df, Default Inf, degree of freedom of t prior. If `beta_df=Inf`, it corresponds to normal prior
- lambda_grid, Default 1:70, candidate for lambda (integer)
- lambda_logprior, Default $p(\lambda) \propto \lambda\Gamma(\lambda+1)/\Gamma(\lambda+5)$, log-prior of lambda. Default choice arises from beta negative binomial distribution; $(\lambda-1) \mid \psi \sim negbin(2,\psi), \psi \sim Beta(2,2)$.
- logprior_sigma.sq, Default half-Cauchy on the sd(u) $= \sigma_u$, log prior of var(u)$= \sigma_u^2$
- phi_lb, lower bound of uniform prior of ϕ_u (inverse range parameter of spatial random effect). Can be same as phi_ub
- phi_ub, lower bound of uniform prior of ϕ_u (inverse range parameter of spatial random effect). Can be same as phi_lb

Value

Returns list of

post_save	a matrix of posterior samples (coda::mcmc) with nsave rows
post_u_save	a matrix of posterior samples (coda::mcmc) of random effects, with nsave rows
loglik_save	a nsave x n matrix of pointwise log-likelihood values, can be used for WAIC calculation.
priors	list of hyperprior information
nsave	number of MCMC samples
t_mcmc	wall-clock time for running MCMC
t_premcmc	wall-clock time for preprocessing before MCMC
y	response vector
X	fixed effect design matrix
coords	a n x 2 matrix of Euclidean coordinates

if NNGP = TRUE, also returns

nngp.control	a list of control parameters for NNGP prior
spNNGPfit	an "NNGP" class with empty samples, placeholder for prediction

Examples

```
# Please see https://github.com/changwoo-lee/cobin-reproduce
```

---

spmicobinreg *spatial micobin regression model*

Description

Fit Bayesian spatial micobin regression model under canonical link (cobit link) with Markov chain Monte Carlo (MCMC).

$$y(s_i) \mid x(s_i), u(s_i) \overset{ind}{\sim} micobin(x(s_i)^T\beta + u(s_i), \psi), \quad u(\cdot) \sim GP$$

for $i = 1, \ldots, n$. See dmicobin for details on micobin distribution. It currently only supports mean zero GP with exponential covariance

$$cov(u(s_i), u(s_j)) = \sigma_u^2 \exp(-\phi_u d(s_i, s_j))$$

where ϕ_u corresponds to inverse range parameter.

Usage

```
spmicobinreg(
  formula,
  data,
  link = "cobit",
  coords,
  NNGP = FALSE,
  contrasts = NULL,
  priors = list(beta_intercept_scale = 10, beta_scale = 2.5, beta_df = Inf),
  nngp.control = list(n.neighbors = 15, ord = order(coords[, 1])),
  nburn = 1000,
  nsave = 1000,
  nthin = 1
)
```

Arguments

formula	an object of class "formula" or a two-sided linear formula object describing both the fixed-effects and random-effects part of the model; see "lmer"
data	data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model.
link	character, link function (default "cobit"). Only supports canonical link function "cobit" that is compatible with Kolmogorov-Gamma augmentation.
coords	a n x 2 matrix of Euclidean coordinates
NNGP	logical, if TRUE, use NNGP prior for the spatial random effects; see spNNGP
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
priors	a list of prior hyperparameters. See Details
nngp.control	a list of control parameters for NNGP prior (only when NNGP = TRUE). This should be a named list of n.neighbors and ord, with default of 15 and first coordiate-based ordering.. See spNNGP for details.
nburn	number of burn-in MCMC iterations.
nsave	number of posterior samples. Total MCMC iteration is nburn + nsave*nthin
nthin	thin-in rate. Total MCMC iteration is nburn + nsave*nthin

Details

The prior setting can be controlled with "priors" argument. Prior for regression coefficients are independent normal or t prior centered at 0. "priors" is a named list of:

- beta_intercept_scale, Default 100, the scale of the intercept prior
- beta_scale, Default 100, the scale of nonintercept fixed-effect coefficients
- beta_df, Default Inf, degree of freedom of t prior. If `beta_df=Inf`, it corresponds to normal prior
- lambda_max, Default 70, upper bound for lambda (integer)
- psi_ab, Default c(2,2), beta shape parameters for ψ (length 2 vector).

- logprior_sigma.sq, Default half-Cauchy on the sd(u) $= \sigma_u$, log prior of var(u)$= \sigma_u^2$
- phi_lb, lower bound of uniform prior of ϕ_u (inverse range parameter of spatial random effect). Can be same as phi_ub
- phi_ub, lower bound of uniform prior of ϕ_u (inverse range parameter of spatial random effect). Can be same as phi_lb

Value

Returns list of

`post_save`	a matrix of posterior samples (coda::mcmc) with nsave rows
`post_u_save`	a matrix of posterior samples (coda::mcmc) of random effects, with nsave rows
`loglik_save`	a nsave x n matrix of pointwise log-likelihood values, can be used for WAIC calculation.
`priors`	list of hyperprior information
`nsave`	number of MCMC samples
`t_mcmc`	wall-clock time for running MCMC
`t_premcmc`	wall-clock time for preprocessing before MCMC
`y`	response vector
`X`	fixed effect design matrix
`coords`	a n x 2 matrix of Euclidean coordinates

if NNGP = TRUE, also returns

`nngp.control`	a list of control parameters for NNGP prior
`spNNGPfit`	an "NNGP" class with empty samples, placeholder for prediction

Examples

```
# Please see https://github.com/changwoo-lee/cobin-reproduce
```

`Vft` *Variance function of cobin*

Description

$B''(B'^{-1}(\mu))$

Usage

```
Vft(mu)
```

Arguments

`mu`	input vector

Value

$B''(B'^{-1}(\mu))$

Index