Package 'jSDM'

July 2, 2019

Type Package

Title Joint Species Distribution Models

Version 0.1.0

Date 2019-05-30

Imports Rcpp (>= 1.0.0), graphics, stats, coda, corrplot

LinkingTo Rcpp, RcppArmadillo, RcppGSL

NeedsCompilation yes

SystemRequirements GNU GSL

Suggests knitr, raster, sp, rmarkdown, bookdown, testthat, boral

Maintainer Ghislain Vieilledent <ghislain.vieilledent@cirad.fr>

Description Fits joint species distribution models ('jSDM')
in a hierarchical Bayesian framework (Warton et al. 2015
<doi:10.1016/j.tree.2015.09.007>). The Gibbs sampler is written
in C++. It uses 'Rcpp', 'Armadillo' and 'GSL' to maximize computation
efficiency.

License GPL-3 | file LICENSE

URL https://ecology.ghislainv.fr/jSDM,
https://github.com/ghislainv/jSDM

BugReports https://github.com/ghislainv/jSDM/issues

LazyLoad yes

Encoding UTF-8

RoxygenNote 6.1.1

VignetteBuilder knitr

Author Ghislain Vieilledent [aut, cre]
(<https://orcid.org/0000-0002-1685-4997>),
Jeanne Clément [aut] (<https://orcid.org/0000-0002-5228-5015>),
CIRAD [cph, fnd]

Repository CRAN

Date/Publication 2019-07-02 09:10:03 UTC

R topics documented:

`jSDM-package` *joint species distribution models*

Description

`jSDM` is an R package for fitting joint species distribution models in a hierarchical Bayesian framework.

Details

Package:	jSDM
Type:	Package
Version:	0.1.0
Date:	2019-01-11
License:	GPL-3
LazyLoad:	yes

Author(s)

Ghislain Vieilledent <ghislain.vieilledent@cirad.fr>
Matthieu Authier <matthieu.authier@univ-lr.fr>

`frogs` *frogs dataset*

Description

Presence or absence of 9 species of frogs on 104 sites, 3 covariates collected on those site and their coordinates.

Usage

```
frogs
```

Format

`frogs` is a data frame with 104 observations on the following 14 variables.

`Species_` 1 to 9 indicate by a 0 the absence of the species on one site and by a 1 its presence

`Covariates_` 1 and 3 continuous variables

`Covariates_` 2 discrete variables

`x` a numeric vector of first coordinates corresponding to each site

`y` a numeric vector of second coordinates corresponding to each site

Source

Wilkinson, D. P.; Golding, N.; Guillera-Arroita, G.; Tingley, R. and McCarthy, M. A. (2018) *A comparison of joint species distribution models for presence-absence data.* Methods in Ecology and Evolution.

Examples

```
data(frogs, package="jSDM")
head(frogs)
```

---

get_residual_cor *Calculate the residual correlation matrix from a LVM.*

Description

This function use coefficients $(\lambda_j^l) with j = 1, \ldots, n_s peciesandl = 1, \ldots, n_l atent$\$ corresponding to latent variables to calculate the variance-covariance matrix which controls correlation between species.

Usage

```
get_residual_cor(mod)
```

Arguments

`mod`	An object of class `"jSDM"`

Value

cov.mean	Average over the MCMC samples of the variance-covariance matrix.
cov.median	Median over the MCMC samples of the variance-covariance matrix.
cor.mean	Average over the MCMC samples of the residual correlation matrix.
cor.median	Median over the MCMC samples of the residual correlation matrix.

Author(s)

Jeanne Clement <jeanne.clement16@laposte.net>, adapted from function boral::get.residual.cor() written by Francis K. Hui <fhui28@gmail.com>.

Examples

```
# frogs data
data(frogs, package="jSDM")

# Arranging data
PA_frogs <- frogs[,4:12]

# Normalized continuous variables
Env_frogs <- cbind(scale(frogs[,1]),frogs[,2],scale(frogs[,3]))
colnames(Env_frogs) <- colnames(frogs[,1:3])

# Parameter inference
# Increase the number of iterations to reach MCMC convergence
mod_jSDM_block_frogs <- jSDM::jSDM_probit_block (
  # Response variable
  presence_site_sp = as.matrix(PA_frogs),
  # Explanatory variables
  site_suitability = ~.,
  site_data = as.data.frame(Env_frogs), n_latent=2,
  # Chains
  burnin=1000, mcmc=1000, thin=1,
  # Starting values
  alpha_start=0, beta_start=0,
  lambda_start=0, W_start=0,
  V_alpha_start=1,
  # Priors
  shape=0.5, rate=0.0005,
  mu_beta=0, V_beta=1.0E6,
  mu_lambda=0, V_lambda=10,
  # Various
  seed=1234, verbose=1)

# Calcul of residual correlation between species
result <- get_residual_cor(mod_jSDM_block_frogs)
result$cov.mean
result$cor.mean
```

jSDM_binomial *Binomial logistic regression model*

Description

The jSDM_binomial function performs a Binomial logistic regression in a Bayesian framework. The function calls a Gibbs sampler written in C++ code which uses an adaptive Metropolis algorithm to estimate the conditional posterior distribution of model's parameters.

Usage

```
jSDM_binomial(presences, trials, suitability, data,
burnin = 5000, mcmc = 10000, thin = 10,
beta_start, mubeta = 0, Vbeta = 1e+06, seed = 1234, ropt = 0.44, verbose = 1)
```

Arguments

presences	A vector indicating the number of successes (or presences) for each observation.
trials	A vector indicating the number of trials for each observation. t_n should be superior or equal to y_n, the number of successes for observation n. If $t_n = 0$, then $y_n = 0$.
suitability	A one-sided formula of the form '~x1+...+xp' with p terms specifying the explicative variables for the suitability process of the model.
data	A data frame containing the model's explicative variables.
burnin	The number of burnin iterations for the sampler.
mcmc	The number of Gibbs iterations for the sampler. Total number of Gibbs iterations is equal to burnin+mcmc. burnin+mcmc must be divisible by 10 and superior or equal to 100 so that the progress bar can be displayed.
thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
beta_start	Starting values for beta parameters of the suitability process. If beta_start takes a scalar value, then that value will serve for all of the betas.
mubeta	Means of the priors for the β parameters of the suitability process. mubeta must be either a scalar or a p-length vector. If mubeta takes a scalar value, then that value will serve as the prior mean for all of the betas. The default value is set to 0 for an uninformative prior.
Vbeta	Variances of the Normal priors for the β parameters of the suitability process. Vbeta must be either a scalar or a p-length vector. If Vbeta takes a scalar value, then that value will serve as the prior variance for all of the betas. The default variance is large and set to 1.0E6 for an uninformative flat prior.
seed	The seed for the random number generator. Default to 1234.
ropt	Target acceptance rate for the adaptive Metropolis algorithm. Default to 0.44.
verbose	A switch (0,1) which determines whether or not the progress of the sampler is printed to the screen. Default is 1: a progress bar is printed, indicating the step (in %) reached by the Gibbs sampler.

Details

We model an ecological process where the presence or absence of the species is explained by habitat suitability.

Ecological process:

$$y_i \sim \mathcal{B}inomial(\theta_i, t_i)$$

$$logit(\theta_i) = X_i\beta$$

Value

`mcmc`	An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The posterior sample of the deviance D, with $D = -2\log(\prod_i P(y_i\|\beta, t_i))$, is also provided.
`theta_latent`	Predictive posterior mean of the probability associated to the suitability process for each observation.
`spec`	Model's specifications

Author(s)

Ghislain Vieilledent <ghislain.vieilledent@cirad.fr>

References

Gelfand, A. E.; Schmidt, A. M.; Wu, S.; Silander, J. A.; Latimer, A. and Rebelo, A. G. (2005) Modelling species diversity through species level hierarchical modelling. *Applied Statistics*, 54, 1-20.

Latimer, A. M.; Wu, S. S.; Gelfand, A. E. and Silander, J. A. (2006) Building statistical models to analyze species distributions. *Ecological Applications*, 16, 33-50.

See Also

`plot.mcmc`, `summary.mcmc`

Examples

```
#=============================================
# jSDM_binomial()
# Example with simulated data
#=============================================

#=================
#== Load libraries
library(jSDM)

#==================
#== Data simulation

#= Number of sites
```

```
nsite <- 200

#= Set seed for repeatability
seed <- 1234

#= Number of visits associated to each site
set.seed(seed)
visits <- rpois(nsite,3)
visits[visits==0] <- 1

#= Ecological process (suitability)
set.seed(seed)
x1 <- rnorm(nsite,0,1)
set.seed(2*seed)
x2 <- rnorm(nsite,0,1)
X <- cbind(rep(1,nsite),x1,x2)
beta.target <- c(-1,1,-1)
logit.theta <- X %*% beta.target
theta <- inv_logit(logit.theta)
set.seed(seed)
Y <- rbinom(nsite,visits,theta)

#= Data-sets
data.obs <- data.frame(Y,visits,x1,x2)

#==================================
#== Site-occupancy model

mod_jSDM_binomial <- jSDM_binomial(presences=data.obs$Y,
                                   trials=data.obs$visits,
                                   suitability=~x1+x2,
                                   data=data.obs,
                                   burnin=1000, mcmc=1000, thin=1,
                                   beta_start=0,
                                   mubeta=0, Vbeta=1.0E6,
                                   seed=1234, ropt=0.44, verbose=1)

#==========
#== Outputs

#= Parameter estimates
summary(mod_jSDM_binomial$mcmc)
pdf(file=file.path(tempdir(), "Posteriors_jSDM_binomial.pdf"))
plot(mod_jSDM_binomial$mcmc)
dev.off()

#== glm resolution to compare
mod_glm <- glm(cbind(Y,visits-Y)~x1+x2,family="binomial",data=data.obs)
summary(mod_glm)

#= Predictions
summary(mod_jSDM_binomial$theta_latent)
pdf(file=file.path(tempdir(), "Pred-Init.pdf"))
```

```
plot(theta, mod_jSDM_binomial$theta_latent)
abline(a=0 ,b=1, col="red")
dev.off()
```

`jSDM_probit_block` *Binomial probit regression model*

Description

The `jSDM_probit_block` function performs a Binomial probit regression in a Bayesian framework. The function calls a Gibbs sampler written in C++ code which uses conjugate priors to estimate the conditional posterior distribution of model's parameters.

Usage

```
jSDM_probit_block(presence_site_sp, site_suitability,
site_data, n_latent=2, burnin=5000, mcmc=15000, thin=10,
alpha_start=0, beta_start=0, lambda_start=0, W_start=0,
V_alpha_start=1, shape=0.5, rate=0.0005,
mu_beta=0, V_beta=1.0E6, mu_lambda=0, V_lambda=10,
seed=1234, verbose=1)
```

Arguments

`presence_site_sp`
: A matrix $n_{site} \times n_{species}$ indicating the presence by a 1 (or the absence by a 0) of each species on each site.

`n_latent`
: An integer indicating the number of latent variables.

`site_suitability`
: A one-sided formula of the form '~x1+...+xp' with p terms specifying the explicative variables for the suitability process of the model.

`site_data`
: A data frame containing the model's explicative variables by site.

`burnin`
: The number of burnin iterations for the sampler.

`mcmc`
: The number of Gibbs iterations for the sampler. Total number of Gibbs iterations is equal to `burnin+mcmc`. `burnin+mcmc` must be divisible by 10 and superior or equal to 100 so that the progress bar can be displayed.

`thin`
: The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.

`beta_start`
: Starting values for beta parameters of the suitability process for each species must be either a scalar or a $p \times n_{species}$ matrix. If `beta_start` takes a scalar value, then that value will serve for all of the betas.

`lambda_start`
: Starting values for lambda parameters corresponding to the latent variables for each species must be either a scalar or a $n_{latent} \times n_{species}$ upper triangular matrix with strictly positive values on the diagonal. If `lambda_start` takes a scalar value, then that value will serve for all of the lambdas except those concerned by the constraints explained above.

`alpha_start` Starting values for random site effect parameters must be either a scalar or a nsite-length vector. If `alpha_start` takes a scalar value, then that value will serve for all of the alphas.

`V_alpha_start` Starting value for variance of random site effect must be a stricly positive scalar.

`W_start` Starting values for latent variables must be either a scalar or a $nsite \times n_latent$ matrix. If `W_start` takes a scalar value, then that value will serve for all of the Ws.

`shape` Shape parameter of the Inverse-Gamma prior for the random site effect variance `V_alpha`. Must be a stricly positive scalar. Default to 0.5 for weak informative prior.

`rate` Rate parameter of the Inverse-Gamma prior for the random site effect variance `V_alpha`. Must be a stricly positive scalar. Default to 0.0005 for weak informative prior.

`mu_beta` Means of the Normal priors for the β parameters of the suitability process. `mubeta` must be either a scalar or a p-length vector. If `mubeta` takes a scalar value, then that value will serve as the prior mean for all of the betas. The default value is set to 0 for an uninformative prior.

`V_beta` Variances of the Normal priors for the β parameters of the suitability process. `Vbeta` must be either a scalar or a $p \times p$ symmetric positive semi-definite square matrix. If `Vbeta` takes a scalar value, then that value will serve as the prior variance for all of the betas, so the variance covariance matrix used in this case is diagonal with the specified value on the diagonal. The default variance is large and set to 1.0E6 for an uninformative flat prior.

`mu_lambda` Means of the Normal priors for the λ parameters corresponding to the latent variables. `mulambda` must be either a scalar or a n_latent-length vector. If `mulambda` takes a scalar value, then that value will serve as the prior mean for all of the lambdas. The default value is set to 0 for an uninformative prior.

`V_lambda` Variances of the Normal priors for the λ parameters corresponding to the latent variables. `Vlambda` must be either a scalar or a $n_{latent} \times n_{latent}$ symmetric positive semi-definite square matrix. If `Vlambda` takes a scalar value, then that value will serve as the prior variance for all of the lambdas, so the variance covariance matrix used in this case is diagonal with the specified value on the diagonal. The default variance is large and set to 10 for an uninformative flat prior.

`seed` The seed for the random number generator. Default to 1234.

`verbose` A switch (0,1) which determines whether or not the progress of the sampler is printed to the screen. Default is 1: a progress bar is printed, indicating the step (in %) reached by the Gibbs sampler.

Details

We model an ecological process where the presence or absence of the species is explained by habitat suitability.

Ecological process:

$$y_{i,j} \sim \mathcal{B}ernoulli(\theta_{i,j})$$
$$probit(\theta_{i,j}) = \beta_{0j} + X_i\beta_j + W_i\lambda_j + \alpha_i$$

Value

`mcmc.alpha`	An mcmc object that contains the posterior samples for alphas. This object can be summarized by functions provided by the coda package.
`mcmc.Valpha`	An mcmc object that contains the posterior samples for variance of random site effect.
`mcmc.latent`	A list by latent variable of mcmc objects that contains the posterior samples for latent variables Ws.
`mcmc.sp`	A list by species of mcmc objects that contains the posterior samples for betas and lambdas.
`mcmc.Deviance`	The posterior sample of the deviance D, with $D = -2\log(\prod_{i,j} P(y_{i,j}\|\beta_j, \lambda_j, \alpha_i, W_i))$, is also provided.
`Z_latent`	Predictive posterior mean of the latent variable Z.
`probit_theta_pred`	Predictive posterior mean of the probability to each species to be present on each site, transformed by probit link function.
`model_spec`	Model's specifications

Author(s)

Jeanne Clement <jeanne.clement16@laposte.net> Ghislain Vieilledent <ghislain.vieilledent@cirad.fr>

References

Chib, S. and Greenberg, E. (1998) Analysis of multivariate probit models. *Biometrika*, 85, 347-361.

Warton, D. I.; Blanchet, F. G.; O'Hara, R. B.; O'Hara, R. B.; Ovaskainen, O.; Taskinen, S.; Walker, S. C. and Hui, F. K. C. (2015) So Many Variables: Joint Modeling in Community Ecology. *Trends in Ecology & Evolution*, 30, 766-779.

See Also

`plot.mcmc`, `summary.mcmc`

Examples

```
#=============================================
# jSDM_probit_block()
# Example with simulated data
#=============================================

#=================
#== Load libraries
library(jSDM)

#==================
#== Data simulation

#= Number of sites
```

```
nsite <- 50

#= Set seed for repeatability
seed <- 1234
set.seed(seed)

#= Number of species
nsp<- 5

#= Number of latent variables
n_latent <- 2

#= Ecological process (suitability)
x1 <- rnorm(nsite,0,1)
x2 <- rnorm(nsite,0,1)
X <- data.frame(Int=rep(1,nsite),x1=x1,x2=x2)
W <- cbind(rnorm(nsite,0,1),rnorm(nsite,0,1))
data <- cbind (X,W)
beta.target <- t(matrix(runif(nsp*ncol(X),-2,2), byrow=TRUE, nrow=nsp))
l.zero <- 0
l.diag <- runif(2,0,2)
l.other <- runif(nsp*n_latent-3,-2,2)
lambda.target <- t(matrix(c(l.diag[1],l.zero,
l.other[1],l.diag[2],l.other[-1]), byrow=TRUE, nrow=nsp))
param.target <- rbind(beta.target,lambda.target)
Valpha.target <- 0.5
V <- 1
alpha.target <- rnorm(nsite,0,sqrt(Valpha.target))
probit_theta <- as.matrix(X) %*% beta.target + W %*% lambda.target + alpha.target
e <- matrix(rnorm(nsp*nsite,0,sqrt(V)),nsite,nsp)
Z_true <- probit_theta + e
Y <- matrix (NA, nsite,nsp)
for (i in 1:nsite){
  for (j in 1:nsp){
    if ( Z_true[i,j] > 0) {Y[i,j] <- 1}
    else {Y[i,j] <- 0}
  }
}

#==================================
#== Site-occupancy model

# Increase number of iterations (burnin and mcmc) to get convergence
mod_jSDM_probit_block <- jSDM::jSDM_probit_block ( presence_site_sp = Y ,
                                                   site_suitability = ~ x1 + x2,
                                                   site_data = X[,-1], n_latent=2,
                                                   burnin=3000, mcmc=3000, thin=3,
                                                   alpha_start=0, beta_start=0,
                                                   lambda_start=0, W_start=0,
                                                   V_alpha_start=1,
                                                   shape=0.5, rate=0.0005,
                                                   mu_beta=0, V_beta=1.0E6,
                                                   mu_lambda=0, V_lambda=10,
```

```
                                                seed=1234, verbose=1)

# ==================================================
# Result analysis
# ==================================================

#==========
#== Outputs

#= Parameter estimates

## alpha
summary(mod_jSDM_probit_block$mcmc.alpha)
pdf(file=file.path(tempdir(), "Posteriors_alpha_jSDM_probit_block.pdf"))
plot(alpha.target,
summary(mod_jSDM_probit_block$mcmc.alpha)[[1]][,"Mean"],
xlab ="alphas target", ylab ="alphas estimated")
abline(a=0,b=1,col='red')
dev.off()

## Valpha
summary(mod_jSDM_probit_block$mcmc.Valpha)
pdf(file=file.path(tempdir(), "Posteriors_Valpha_jSDM_probit_block.pdf"))
par(mfrow=c(1,2))
coda::traceplot(mod_jSDM_probit_block$mcmc.Valpha)
coda::densplot(mod_jSDM_probit_block$mcmc.Valpha)
abline(v=Valpha.target,col='red')
dev.off()


## beta_j
summary(mod_jSDM_probit_block$mcmc.sp$sp_1[,1:ncol(X)])
pdf(file=file.path(tempdir(), "Posteriors_beta_jSDM_probit_block.pdf"))
par(mfrow=c(ncol(X),2))
for (j in 1:nsp) {
  for (p in 1:ncol(X)) {
    coda::traceplot(coda::as.mcmc(mod_jSDM_probit_block$mcmc.sp[[paste0("sp_",j)]][,p]))
    coda::densplot(coda::as.mcmc(mod_jSDM_probit_block$mcmc.sp[[paste0("sp_",j)]][,p]),
   main = paste(colnames(mod_jSDM_probit_block$mcmc.sp[[paste0("sp_",j)]])[p],", species : ",j))
    abline(v=beta.target[p,j],col='red')
  }
}
dev.off()

## lambda_j
summary(mod_jSDM_probit_block$mcmc.sp$sp_1[,(ncol(X)+1):(ncol(X)+n_latent)])
summary(mod_jSDM_probit_block$mcmc.sp$sp_2[,(ncol(X)+1):(ncol(X)+n_latent)])
pdf(file=file.path(tempdir(), "Posteriors_lambda_jSDM_probit_block.pdf"))
par(mfrow=c(n_latent*2,2))
for (j in 1:nsp) {
  for (l in 1:n_latent) {
   coda::traceplot(coda::as.mcmc(mod_jSDM_probit_block$mcmc.sp[[paste0("sp_",j)]][,ncol(X)+l]))
   coda::densplot(coda::as.mcmc(mod_jSDM_probit_block$mcmc.sp[[paste0("sp_",j)]][,ncol(X)+l]),
```

```
    main=paste(colnames(mod_jSDM_probit_block$mcmc.sp[[paste0("sp_",j)]])[ncol(X)+l],",
    species : ",j))
    abline(v=lambda.target[l,j],col='red')
  }
}
dev.off()

## W latent variables
pdf(file=file.path(tempdir(), "Posteriors_lv_jSDM_probit_block.pdf"))
par(mfrow=c(1,2))
for (l in 1:n_latent) {
plot(W[,l],
summary(mod_jSDM_probit_block$mcmc.latent[[paste0("lv_",l)]])[[1]][,"Mean"],
main = paste0("Latent variable W_", l),
xlab =paste0("W_", l, " target"), ylab =paste0("W_", l, " estimated"))
abline(a=0,b=1,col='red')
}
dev.off()

## Deviance
summary(mod_jSDM_probit_block$mcmc.Deviance)
plot(mod_jSDM_probit_block$mcmc.Deviance)

#= Predictions

pdf(file=file.path(tempdir(), "Pred-Init.pdf"))
## probit_theta
summary(mod_jSDM_probit_block$probit_theta_pred)
par(mfrow=c(1,1))
plot(probit_theta,mod_jSDM_probit_block$probit_theta_pred)
abline(a=0,b=1,col='red')

## Z
summary(mod_jSDM_probit_block$Z_latent)
plot(Z_true,mod_jSDM_probit_block$Z_latent)
abline(a=0,b=1,col='red')
dev.off()
```

`logit` *Generalized logit and inverse logit function*

Description

Compute generalized logit and generalized inverse logit functions.

Usage

```
logit(x, min = 0, max = 1)
inv_logit(x, min = 0, max = 1)
```

Arguments

x	value(s) to be transformed
min	Lower end of logit interval
max	Upper end of logit interval

Details

The generalized logit function takes values on [min, max] and transforms them to span [-Inf,Inf] it is defined as:

$$y = log(\frac{p}{(1-p)})$$

where

$$p = \frac{(x - min)}{(max - min)}$$

The generalized inverse logit function provides the inverse transformation:

$$x = p'(max - min) + min$$

where

$$p' = \frac{exp(y)}{(1 + exp(y))}$$

Value

Transformed value(s).

Author(s)

Gregory R. Warnes <greg@warnes.net>

Examples

```
x <- seq(0,10, by=0.25)
xt <- jSDM::logit(x, min=0, max=10)
cbind(x,xt)
y <- jSDM::inv_logit(xt, min=0, max=10)
cbind(x,xt,y)
```

plot_residual_cor *Plot the residual correlation matrix from a LVM.*

Description

Plot the posterior mean estimator of residual correlation matrix reordered by first principal component.

Usage

```
plot_residual_cor(mod)
```

Arguments

mod	An object of class "jSDM".

Author(s)

Jeanne Clement <jeanne.clement16@laposte.net>

Examples

```
# frogs data
data(frogs, package="jSDM")

# Arranging data
PA_frogs <- frogs[,4:12]

# Normalized continuous variables
Env_frogs <- cbind(scale(frogs[,1]),frogs[,2],scale(frogs[,3]))
colnames(Env_frogs) <- colnames(frogs[,1:3])

# Parameter inference
# Increase the number of iterations to reach MCMC convergence
mod_jSDM_block_frogs <- jSDM::jSDM_probit_block (
  # Response variable
  presence_site_sp = as.matrix(PA_frogs),
  # Explanatory variables
  site_suitability = ~.,
  site_data = as.data.frame(Env_frogs), n_latent=2,
  # Chains
  burnin=1000, mcmc=1000, thin=1,
  # Starting values
  alpha_start=0, beta_start=0,
  lambda_start=0, W_start=0,
  V_alpha_start=1,
  # Priors
  shape=0.5, rate=0.0005,
  mu_beta=0, V_beta=1.0E6,
```

```
  mu_lambda=0, V_lambda=10,
  # Various
  seed=1234, verbose=1)

# Representation of residual correlation between species
jSDM::plot_residual_cor(mod_jSDM_block_frogs)
```

`predict.jSDM` *Predict method for models fitted with jSDM*

Description

Predicted values for models fitted with jSDM

Usage

```
## S3 method for class 'jSDM'
predict(object, newdata=NULL, Id_species, Id_sites, type="mean",
  probs=c(0.025,0.975), ...)
```

Arguments

`object`	An object of class `"jSDM"`.
`newdata`	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
`Id_species`	An vector of character or integer indicating for which species the probabilities of presence on chosen sites will be predicted.
`Id_sites`	An vector of integer indicating for which sites the probabilities of presence of specified species will be predicted.
`type`	Type of prediction. Can be `"mean"` for predictive posterior mean, `"quantile"` for producing sample quantiles from the predictive posterior corresponding to the given probabilities (see `probs` argument) or `"posterior"` for the full predictive posterior for each prediction. Using `"quantile"` or `"posterior"` might lead to memory problem depending on the number of predictions and the number of samples for the jSDM model's parameters.
`probs`	Numeric vector of probabilities with values in [0,1] and used when `type="quantile"`.
`...`	Further arguments passed to or from other methods.

Value

Return a vector for the predictive posterior mean when `type="mean"`, a data-frame with the mean and quantiles when `type="quantile"` or an `mcmc` object (see `coda` package) with posterior distribution for each prediction when `type="posterior"`.

Author(s)

Jeanne Clement <jeanne.clement16@laposte.net> Ghislain Vieilledent <ghislain.vieilledent@cirad.fr>

See Also

jSDM

Examples

```
# frogs data
data(frogs, package="jSDM")

# Arranging data
PA_frogs <- frogs[,4:12]

# Normalized continuous variables
Env_frogs <- cbind(scale(frogs[,1]),frogs[,2],scale(frogs[,3]))
colnames(Env_frogs) <- colnames(frogs[,1:3])

# Parameter inference
# Increase the number of iterations to reach MCMC convergence
mod_jSDM_block_frogs <- jSDM::jSDM_probit_block(
  # Response variable
  presence_site_sp = as.matrix(PA_frogs),
  # Explanatory variables
  site_suitability = ~.,
  site_data = as.data.frame(Env_frogs), n_latent=2,
  # Chains
  burnin=1000, mcmc=1000, thin=1,
  # Starting values
  alpha_start=0, beta_start=0,
  lambda_start=0, W_start=0,
  V_alpha_start=1,
  # Priors
  shape=0.5, rate=0.0005,
  mu_beta=0, V_beta=1.0E6,
  mu_lambda=0, V_lambda=10,
  # Various
  seed=1234, verbose=1)

# Select site and species for predictions
## 30 sites
Id_sites <- sample.int(nrow(PA_frogs), 30)
## 5 species
Id_species <- sample(colnames(PA_frogs), 5)

# Predictions
theta_pred <- jSDM::predict.jSDM(mod_jSDM_block_frogs,
  Id_species=Id_species, Id_sites=Id_sites, type="mean")
hist(theta_pred, main="Predicted theta with simulated covariates")
```

Index