Package 'jtdm'

February 1, 2022

Type Package

Title Joint modelling of function traits

Version 0.1.0

Author Giovanni Poggiato

Maintainer Giovanni Poggiato <giov.poggiato@gmail.com>

Description Joint modelling of functional traits.

License What license is it under?

Encoding UTF-8

LazyData true

Imports arm, rjags, runjags, mvtnorm, mcmcplots, ggplot2, ggforce, coda, MASS, parallel

RoxygenNote 7.1.2

Depends R (>= 2.10)

R topics documented:

ellipse_plot 2
getB 3
get_sigma 4
joint_trait_prob 4
joint_trait_prob_gradient 6
jtdm 8
jtdmCV 8
jtdm_fit 9
jtdm_predict 10
partial_response 11
X 12
Y 13

Index **14**

ellipse_plot *Partial response curve of the pairwise most suitable community-level strategy and of the pairwise envelop of possible community-level strategy*

Description

Partial response curve of the pairwise most suitable community-level strategy and of the pairwise envelop of possible community-level strategy. In order to build the response curve, the function builts a dataframe where the focal variable varies along a gradient and the other (non-focal) variables are fixed to their mean (but see FixX parameter for fixing non-focal variables to user-defined values). The chosen traits are specified in indexTrait. Then uses the jtdm_predict function to compute the most suitable community-level strategy and the residual covariance matrix to build the envelop of possible CLS.

Usage

```
ellipse_plot(
  m,
  indexGradient,
  indexTrait,
  FullPost = F,
  grid.length = 20,
  FixX = NULL,
  confL = 0.95
)
```

Arguments

`m`	a model fitted with `jtdm_fit`
`indexGradient`	The name (as specified in the column names of X) of the focal variable.
`indexTrait`	A vector of the two names (as specified in the column names of Y) containing the two (or more!) traits we want to compute the community level strategy of.
`FullPost`	If FullPost = TRUE, the function returns samples from the predictive distribution of joint probabilities. If FullPost= FALSE, joint probabilities are computed only using the posterior mean of the parameters.
`grid.length`	The number of points along the gradient of the focal variable. Default to 20 (which ensures a fair visualisation).
`FixX`	Optional. A parameter to specify the value to which non-focal variables are fixed. This can be useful for example if we have some categorical variables (e.g. forest vs meadows) and we want to obtain the partial response curve for a given value of the variable. It has to be a list of the length and names of the columns of X. For example, if the columns of X are "MAT","MAP","Habitat" and we want to fix "Habitat" to 1, then FixX=list(MAT=NULL,MAP=NULL,Habitat=1.). Default to NULL.
`confL`	The confidence level of the confidence ellipse (i.e. of the envelop of possible community-level strategies). Default is 0.95.

Value

Plot of the partial response curve of the pairwise most suitable community-level strategy and of the pairwise envelop of possible community-level strategy

Examples

```
data(Y)
data(X)
# Short MCMC to obtain a fast example: results are unreliable !
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"),  adapt = 10,
        burnin = 100,
        sample = 100)

# plot the pairwise SLA-LNC partial response curve along the GDD gradient
ellipse_plot(m,indexTrait = c("SLA","LNC"),indexGradient="GDD")
#  plot the pairwise SLA-LNC partial response curve along the GDD gradient
#  in forest (i.e. when forest=1)
ellipse_plot(m,indexTrait = c("SLA","LNC"),indexGradient="GDD",
             FixX=list(GDD=NULL,FDD=NULL,forest=1))
```

---

getB   *Get the inferred regression coefficients*

Description

Get the samples from the posterior distribution of the regression coefficient matrix B, together with the posterior mean and quantiles. The regression coefficient matrix B is a matrix where the number of rows is defined by the number of traits that are modelled, and the number of columns is the number of columns of the matrix m$X (the number of explanatory variables after transformation via formula)

Usage

```
getB(m)
```

Arguments

`m`	a model fitted with `jtdm_fit`

Value

A list containing:

`Bsamples`	Sample from the posterior distribution of the regression coefficient matrix. It is an array where the first dimension is the number of traits, the second the number of columns in m$X (the number of variables after transformation via formula) and the third the number of MCMC samples.
`Bmean`	Posterior mean of the regression coefficient matrix.
`Bq975,Bq025`	97.5% and 0.25% posterior quantiles of the regression coefficient matrix.

Examples

```
data(Y)
data(X)
# Short MCMC to obtain a fast example: results are unreliable !
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"),  adapt = 10,
        burnin = 100,
        sample = 100)
# get the inferred regression coefficients
B=getB(m)
```

---

get_sigma *Get the inferred residual covariance matrix*

Description

Get the samples from the posterior distribution of the residual covariance matrix, together with the posterior mean and quantiles.

Usage

```
get_sigma(m)
```

Arguments

m	a model fitted with `jtdm_fit`

Examples

```
data(Y)
data(X)
# Short MCMC to obtain a fast example: results are unreliable !
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"),  adapt = 10,
        burnin = 100,
        sample = 100)
# get the inferred residual covariance
Sigma =get_sigma(m)
```

---

joint_trait_prob *Computes joint probabilities.*

Description

Computes the joint probability of CWM traits in regions in the community-trait space specified by bounds and in sites specified in Xnew.

Usage

```
joint_trait_prob(
  m,
  indexTrait,
  bounds,
  Xnew = NULL,
  FullPost = T,
  mcmc.samples = NULL,
  parallel = FALSE
)
```

Arguments

`m`	a model fitted with `jtdm_fit`
`indexTrait`	A vector of the names (as specified in the column names of Y) of the two (or more!) traits we want to compute the joint probabilities of.
`bounds`	The parameter to specify a region in the community-trait space where the function computes the joint probabilities of traits. It is a list of the length of "indexTrait", each element of the list is a vector of length two. The vector represents the inferior and superior bounds of the region for the specified trait. For example, if we consider two traits, bounds=list(c(10,Inf),c(10,Inf)) corresponds to the region in the community-trait space where both traits both take values greater than 10.
`Xnew`	Optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
`FullPost`	If FullPost = TRUE, the function returns samples from the predictive distribution of joint probabilities. If FullPost= FALSE, joint probabilities are computed only using the posterior mean of the parameters. FullPost cannot be equal to "mean" here.
`mcmc.samples`	Optional, default to NULL, only works when FullPost=FALSE. Defines the number of MCMC samples to compute the posterior distribution of joint probabilities. Needs to be between 1 and m$model$sample x length(m$model$mcmc)
`parallel`	Optional, only works when FullPost = TRUE. When TRUE, the function uses mclapply to parallelise the calculation of the posterior distribution joint probabilities.

Details

This function is time consuming when `FullPost=T`. Consider setting `parallel=T` and/or to set `mcmc.samples` to a value smaller than the length of the MCMC chains.

Examples

```
data(Y)
data(X)
# Short MCMC to obtain a fast example: results are unreliable !
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"),  adapt = 10,
        burnin = 100,
        sample = 100)
# Compute probability of SLA and LNC to be joint-high at sites in the studies
joint = joint_trait_prob(m,indexTrait=c("SLA","LNC"),
                          bounds=list(c(mean(Y[,"SLA"]),Inf),c(mean(Y[,"SLA"]),Inf)))
```

joint_trait_prob_gradient

Computes partial response curves of joint probabilities

Description

Computes the partial responses curves of joint probability of CWM traits as a function of a focal variable. The regions in which joint probabilities are computed are specified by bounds. In order to build the response curve, the function builts a dataframe where the focal variable varies along a gradient and the other (non-focal) variables are fixed to their mean (but see FixX parameter for fixing non-focal variables to user-defined values). Then, uses joint_trait_prob to compute the joint probability in these dataset.

Usage

```
joint_trait_prob_gradient(
  m,
  indexTrait,
  indexGradient,
  bounds,
  grid.length = 200,
  XFocal = NULL,
  FixX = NULL,
  FullPost = T,
  mcmc.samples = NULL,
  parallel = FALSE
)
```

Arguments

`m`	A model fitted with `jtdm_fit`
`indexTrait`	A vector of the names (as specified in the column names of Y) of the two (or more!) traits we want to compute the joint probabilities of.
`indexGradient`	The name (as specified in the column names of X) of the focal variable.
`bounds`	The parameter to specify a region in the community-trait space where the function computes the joint probabilities of traits. It is a list of the length of "indexTrait", each element of the list is a vector of length two. The vector represents the inferior and superior bounds of the region for the specified trait. For example, if we consider two traits, bounds=list(c(10,Inf),c(10,Inf)) corresponds to the region in the community-trait space where both traits both take values greater than 10.
`grid.length`	The number of points along the gradient of the focal variable. Default to 200.
`XFocal`	Optional. A gradient of the focal variable provided by the user. If provided, the function will used this gradient instead of building a regular one. Default to NULL.
`FixX`	Optional. A parameter to specify the value to which non-focal variables are fixed. This can be useful for example if we have some categorical variables (e.g. forest vs meadows) and we want to obtain the partial response curve for a given value of the variable. It has to be a list of the length and names of the columns of

	X. For example, if the columns of X are "MAT","MAP","Habitat" and we want to fix "Habitat" to 1, then FixX=list(MAT=NULL,MAP=NULL,Habitat=1.). Default to NULL.
`FullPost`	If FullPost = TRUE, the function returns samples from the predictive distribution of joint probabilities. If FullPost= FALSE, joint probabilities are computed only using the posterior mean of the parameters. FullPost cannot be equal to "mean" here.
`mcmc.samples`	Optional, default to NULL, only works when FullPost=FALSE. Defines the number of MCMC samples to compute the posterior distribution of joint probabilities. Needs to be between 1 and m$model$sample x length(m$model$mcmc)
`parallel`	Optional, only works when FullPost = TRUE. When TRUE, the function uses mclapply to parallelise the calculation of the posterior distribution joint probabilities.

Details

This function is time consuming when `FullPost=T`. Consider setting `parallel=T` and/or to set `mcmc.samples` to a value smaller than the length of the MCMC chains.

Value

A list containing:

`GradProbssamples`	Sample from the posterior distribution of the joint probability along the gradient. It is a vector whose lenght is the number of MCMC samples. NULL if FullPost=FALSE.
`GradProbsmean`	Posterior mean of the joint probability along the gradient.
`GradProbsq975,GradProbsq025`	97.5% and 0.25% posterior quantiles of the joint probability along the gradient. NULL if FullPost=FALSE.
`gradient`	The gradient of the focal variable built by the function.

Examples

```
data(Y)
data(X)
# Short MCMC to obtain a fast example: results are unreliable !
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"),  adapt = 10,
        burnin = 100,
        sample = 100)
# Compute probability of SLA and LNC to be joint-high at sites in the studies

# Compute the joint probability of SLA and LNC
  #to be joint-high along the GDD gradient
joint = joint_trait_prob_gradient(m,indexTrait=c("SLA","LNC"),
                                  indexGradient="GDD",
                             bounds=list(c(mean(Y[,"SLA"]),Inf),c(mean(Y[,"SLA"]),Inf)))
# Compute the joint probability of SLA and LNC to be joint-high along the
# GDD gradient when forest = 1 (i.e. in forests)
joint = joint_trait_prob_gradient(m,indexTrait=c("SLA","LNC"),
                                  indexGradient="GDD",
                             bounds=list(c(mean(Y[,"SLA"]),Inf),c(mean(Y[,"SLA"]),Inf)),
FixX=list(GDD=NULL,FDD=NULL,forest=1))
```

`jtdm` *jtdm.*

Description

Package to fit a Join Trait Distribution Model and to analyse its result to understand and predict the community-level strategy. See Poggiato et al. In prep.

Author(s)

Giovanni Poggiato <`giov.poggiato@gmail.com`>

`jtdmCV` *K-fold cross validation predictions and goodness of fit metrics*

Description

Run K-fold cross validation predictions of the model m on a specified dataset.

Usage

```
jtdmCV(
  m,
  K = 5,
  adapt = 200,
  burnin = 500,
  sample = 500,
  n.chains = 2,
  partition = NULL
)
```

Arguments

`m`	a model fitted with `jtdm_fit`
`K`	The number of folds of the K-fold cross validation
`adapt, burnin, sample, n.chains`	Parameters of the MCMC sampler. See `?run.jags` for details
`partition`	A partition of the dataset specified by the user. It is a vector (whose length are the number of sites), where each element specifies the fold index of the site.
`Ynew`	Optional. The observed response variables at sites specified in Xnew. It is used to compute goodness of fit metrics when validation= T.

Value

A list containing:

`Pred`	Sample from the posterior predictive distribution in cross validation. It is an array where the first dimension is the number of sites in Xnew, the second is the number of traits modelled and the third the number of MCMC samples. NULL if FullPost=FALSE.
`PredMean`	Posterior mean of posterior predictive distribution in cross validation.
`Predq975,Predq025`	97.5% and 0.25% posterior quantiles of the posterior predictive distribution in cross validation. NULL if FullPost=FALSE.
`R2`	R squared of predictions in cross validation.
`RMSE`	Root square mean error between squared of predictions in cross validation.

Examples

```
data(Y)
data(X)
# Short MCMC to obtain a fast example: results are unreliable !
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"),  adapt = 10,
        burnin = 100,
        sample = 100)
# Run 5-fold cross validation on m
pred = jtdmCV(m, K=5)
```

`jtdm_fit` *Fitting joint trait distribution models*

Description

jtdm_fit is used to fit a Joint trait distribution model. Requires the response variable Y (the sites x traits matrix) and the explanatory variables X.

Usage

```
jtdm_fit(
  Y,
  X,
  formula,
  adapt = 200,
  burnin = 5000,
  sample = 5000,
  n.chains = 2,
  monitor = c("B", "Sigma", "pd")
)
```

Arguments

`Y`	The sites x traits matrix containing community (weighted) means of each trait at each site.
`X`	The design matrix, i.e. sites x predictor matrix containing the value of each explanatory variable (e.g. the environmental conditions) at each site.
`formula`	An object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'
`adapt, burnin, sample, n.chains, monitor`	Parameters of the MCMC sampler. See `?run.jags` for details

Details

A formula has an implied intercept term. To remove this use either y ~ x - 1 or y ~ 0 + x. See formula for more details of allowed formulae.

Examples

```
data(Y)
data(X)
# Short MCMC to obtain a fast example: results are unreliable !
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"),  adapt = 10,
        burnin = 100,
        sample = 100)
```

`jtdm_predict` *Predict method for joint trait distribution model*

Description

Obtains predictions from a fitted joint trait distribution model and optionally computes their R squared and root mean square error (RMSE)

Usage

```
jtdm_predict(m = m, Xnew = NULL, Ynew = NULL, validation = F, FullPost = T)
```

Arguments

`m`	a model fitted with `jtdm_fit`
`Xnew`	optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used
`Ynew`	Optional. The observed response variables at sites specified in Xnew. It is used to compute goodness of fit metrics when validation= T.
`validation`	boolean parameter to decide whether we want to compute goodness of fit measures. If true, then Ynew is needed.
`FullPost`	The type of predictions to be obtain. If FullPost = TRUE, the function returns samples from the predictive distribution. If FullPost="mean", the function computes the posterior distribution of the regression term $BXnew$). If FullPost=F, the function only returns the posterior mean of the regression term ($BmeanXnew$).

Details

To obtain a full assesment of the posterior distribution, the function should be ran with FullPost=TRUE, altough this can be time consuming. FullPost="mean" is used to compute partial response curves, while FullPost=FALSE is used to compute goodness of fit metrics.

Examples

```
data(Y)
data(X)
# Short MCMC to obtain a fast example: results are unreliable !
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"),  adapt = 10,
        burnin = 100,
        sample = 100)
# marginal predictions of traits in the sites of X
pred = jtdm_predict(m)
```

---

partial_response   *Computes and plots the trait-environment relationship of a given CWM trait and a given environmental variable*

Description

Computes and plots the trait-environment relationship of a given CWM trait and a focal environmental variable. In order to build the response curve, the function builts a dataframe where the focal environmental variable varies along a gradient and the other (non-focal) variables are fixed to their mean (but see FixX parameter for fixing non-focal variables to user-defined values).

Usage

```
partial_response(
  m,
  indexGradient,
  indexTrait,
  XFocal = NULL,
  grid.length = 200,
  FixX = NULL,
  FullPost = "mean"
)
```

Arguments

`m`	a model fitted with `jtdm_fit`
`indexGradient`	The name (as specified in the column names of X) of the focal variable.
`indexTrait`	The name (as specified in the column names of Y) of the focal trait.
`XFocal`	Optional. A gradient of the focal variable provided by the user. If provided, the function will used this gradient instead of building a regular one. Default to NULL.
`grid.length`	The number of points along the gradient of the focal variable. Default to 200.

`FixX`	Optional. A parameter to specify the value to which non-focal variables are fixed. This can be useful for example if we have some categorical variables (e.g. forest vs meadows) and we want to obtain the partial response curve for a given value of the variable. It has to be a list of the length and names of the columns of X. For example, if the columns of X are "MAT","MAP","Habitat" and we want to fix "Habitat" to 1, then FixX=list(MAT=NULL,MAP=NULL,Habitat=1.). Default to NULL.
`FullPost`	The type of predictions to be obtain. If FullPost = TRUE, the function returns samples from the predictive distribution. If FullPost="mean", the function computes the posterior distribution of the regression term B%*%X). Default to "mean", here FullPost cannot be FALSE.

Value

A list containing:

`p`	A plot of the trait-environment relationship.
`predictions`	A data frame containing the predicted trait-environmental relationships including the gradient of the focal environmental variable, mean trait predictions and quantiles (can be useful to code customized plot).

Examples

```
data(Y)
data(X)
# Short MCMC to obtain a fast example: results are unreliable !
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"),  adapt = 10,
        burnin = 100,
        sample = 100)
# SLA-GDD relationship
plot = partial_response(m,indexGradient="GDD",indexTrait="SLA")
plot$p
# SLA-GDD relationship in forest (i.e. when forest=1)
plot = partial_response(m,indexGradient="GDD",indexTrait="SLA",
                        FixX=list(GDD=NULL,FDD=NULL,forest=1))
plot$p
```

`X` *Site x environmental covariates dateset*

Description

Includes the Growing Degree Days (GDD) during the growing season and Freezing Degree Days (FDD) during the growing season averaged over the period 1989-2019

Usage

```
data(X)
```

```
data(X)
```

Format

A matrix

Author(s)

Orchamp consortium

Examples

```
data(X)
```

`Y` *Site x CWM traits dataset*

Description

A site x CWM traits dataset computed using pinpoint abundances of plants and species mean

Usage

```
data(Y)
```

Format

A matrix

Author(s)

Orchamp Consortium

Examples

```
data(Y)
```

Index

∗ **datasets**
  X, 12
  Y, 13

ellipse_plot, 2

get_sigma, 4
getB, 3

joint_trait_prob, 4
joint_trait_prob_gradient, 6
jtdm, 8
jtdm_fit, 9
jtdm_predict, 10
jtdmCV, 8

partial_response, 11

X, 12

Y, 13