Package: jtdm (via r-universe)

October 19, 2026

Type Package

Title Joint Modelling of Functional Traits

Version 0.1-3

Description Fitting and analyzing a Joint Trait Distribution Model. The Joint Trait Distribution Model is implemented in the Bayesian framework using conjugate priors and posteriors, thus guaranteeing fast inference. In particular the package computes joint probabilities and multivariate confidence intervals, and enables the investigation of how they depend on the environment through partial response curves. The method implemented by the package is described in Poggiato et al. (2023) <doi:10.1111/geb.13706>.

License GPL-3

Encoding UTF-8

LazyData true

Imports ggforce, mniw, mvtnorm, parallel, stats, utils, ggplot2, gridExtra, reshape2

RoxygenNote 7.2.3

Depends R (>= 2.10)

URL https://github.com/giopogg/jtdm, https://giopogg.github.io/jtdm/

VignetteBuilder knitr

BugReports https://github.com/giopogg/jtdm/issues

Suggests knitr, rmarkdown, devtools

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libicu-dev

Repository https://giopogg.r-universe.dev

Date/Publication 2024-09-09 09:47:21 UTC

RemoteUrl https://github.com/giopogg/jtdm

RemoteRef HEAD

RemoteSha 1c04e3da1db5d0604a415a8e54c929c266facd91

Contents

ellipse_plot . . . 2
get_sigma . . . 3
getB . . . 4
global . . . 5
joint_trait_prob . . . 5
joint_trait_prob_gradient . . . 7
jtdm_fit . . . 9
jtdm_predict . . . 10
jtdmCV . . . 12
partial_response . . . 13
plot.jtdm_fit . . . 14
summary.jtdm_fit . . . 15
X . . . 16
Y . . . 16

Index **17**

`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 builds 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 CWM combinations.

Usage

```
ellipse_plot(
  m,
  indexGradient,
  indexTrait,
  FullPost = FALSE,
  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 visualization).
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"),  sample = 1000)

# 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))
```

---

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

Value

A list containing:

Ssamples	Sample from the posterior distribution of the residual covariance matrix. It is an array where the first two dimensions are the rows and columns of the matrix, and the third dimensions are the samples from the posterior distribution
Smean	Posterior mean of the residual covariance matrix.
Sq975, Sq025	97.5% and 0.25% posterior quantiles of the residual covariance 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"), sample = 1000)
# get the inferred residual covariance
Sigma =get_sigma(m)
```

---

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 modeled, 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)
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"), sample = 1000)
# get the inferred regression coefficients
B=getB(m)
```

---

global *Global*

Description

Declare global variables

---

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 = FALSE,
  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, thus allowing the computation of credible intervals. If FullPost= FALSE, joint probabilities are computed only using the posterior mean of the parameters. FullPost cannot be equal to "mean" here.
samples	Optional, default to NULL, only works when FullPost=FALSE. Defines the number of posterior samples to compute the posterior distribution of joint probabilities. Needs to be between 1 the total number of samples drawn from the posterior distribution.
parallel	Optional, only works when FullPost = TRUE. When parallel = TRUE, the function uses mclapply to parallelise the calculation of the posterior distribution joint probabilities.

Details

This function is time consuming when FullPost = TRUE. Consider setting parallel = TRUE and/or to set samples to a value smaller than the total number of posterior samples .

Value

A list containing:

PROBsamples	Samples from the posterior distribution of the joint probability.NULL if FullPost=FALSE.
PROBmean	Posterior mean of the joint probability.
PROBq975, PROBq025	97.5% and 0.25% posterior quantiles of the joint probability. NULL if FullPost=FALSE.

Examples

```
data(Y)
data(X)
#We sample only few samples from the posterior in order to reduce
# the computational time of the examples.
```

```
#Increase the number of samples to obtain robust results
m = jtdm_fit(Y = Y, X = X, formula = as.formula("~GDD+FDD+forest"), sample = 10)
# 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)),
                         FullPost = TRUE)
```

`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 builds 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 = FALSE,
  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, thus allowing the computation of credible intervals. If FullPost= FALSE, joint probabilities are computed only using the posterior mean of the parameters. FullPost cannot be equal to "mean" here.

`samples` Optional, default to NULL, only works when FullPost=FALSE. Defines the number of samples to compute the posterior distribution of joint probabilities. Needs to be between 1 the total number of samples drawn from the posterior distribution.

`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 = TRUE`. Consider setting `parallel = TRUE` and/or to set `samples` to a value smaller than the total number of posterior samples.

Value

A list containing:

`GradProbssamples` Sample from the posterior distribution of the joint probability along the gradient. It is a vector whose length is the number of posterior 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)
# We sample only few samples from the posterior in order to reduce
# the computational time of the examples.
```

```
# Increase the number of samples to obtain robust results
m = jtdm_fit(Y = Y, X = X, formula = as.formula("~GDD+FDD+forest"),  sample = 10)
# 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)),
                                  FullPost = TRUE)

# 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),
                                  FullPost = TRUE)
```

`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.This function samples from the posterior distribution of the parameters, which has been analytically determined. Therefore, there is no need for classical MCMC convergence checks.

Usage

```
jtdm_fit(Y, X, formula, sample = 1000)
```

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'.
`sample`	Number of samples from the posterior distribution. Since we sample from the exact posterior distribution, the number of samples is relative lower than MCMC samplers. As a rule of thumb, 1000 samples should provide correct inference.

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.

Value

A list containing:

`model`	An object of class "jtdm_fit", containing the samples from the posterior distribution of the regression coefficients (B) and residual covariance matrix (Sigma), together with the likelihood of the model.
`Y`	A numeric vector of standard errors on parameters
`X_raw`	The design matrix specified as input
`X`	The design matrix transformed as specified in formula
`formula`	The formula specified as input

Examples

```
data(Y)
data(X)
m = jtdm_fit(Y = Y, X = X, formula = as.formula("~GDD+FDD+forest"), sample = 1000)
```

`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 = FALSE,
  FullPost = "mean"
)
```

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, the credible intervals are thus the predictive credible interval. If FullPost="mean", the function computes the posterior distribution of the regression term $BXnew$), i.e., classical credible intervals. If FullPost=FALSE, the function only returns the posterior mean of the regression term ($BmeanXnew$), i.e., no credible intervals.

Details

To obtain a full assessment of the posterior distribution, the function should be ran with FullPost=TRUE, although 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.

Value

A list containing:

`Pred` Sample from the posterior distribution of the posterior predictive distribution. 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

`Predq975`, `Predq025` 97.5% and 0.25% posterior quantiles of the posterior predictive distribution. NULL if FullPost=FALSE.

`R2` R squared of predictions (squared Pearson correlation between Ynew and the predictions). NULL if validation=FALSE.

`RMSE` Root square mean error between squared of predictions. NULL if validation=FALSE.

Examples

```
data(Y)
data(X)
m = jtdm_fit(Y = Y, X = X, formula=as.formula("~GDD+FDD+forest"), sample = 1000)
# marginal predictions of traits in the sites of X
pred = jtdm_predict(m)
```

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, sample = 1000, partition = NULL)
```

Arguments

m	a model fitted with jtdm_fit
K	The number of folds of the K-fold cross validation
sample	Number of samples from the posterior distribution. Since we sample from the exact posterior distribution, the number of samples is relative lower than MCMC samplers. As a rule of thumb, 1000 samples should provide correct inference.
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.

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 modeled 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)
m = jtdm_fit(Y=Y, X=X, formula=as.formula("~GDD+FDD+forest"), sample = 1000)
# Run 3-fold cross validation on m
pred = jtdmCV(m, K = 5, sample = 1000)
```

`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 builds 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"), sample = 1000)
# 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
```

---

plot.jtdm_fit *Plots the parameters of a fitted jtdm*

Description

Plots the regression coefficients and covariance matrix of a fitted jtdm

Usage

```
## S3 method for class 'jtdm_fit'
plot(x, ...)
```

Arguments

x	a model fitted with `jtdm_fit`
...	additional arguments

Value

A plot of the regression coefficients and covariance matrix of the fitted model

Author(s)

Giovanni Poggiato

Examples

```
data(Y)
data(X)
m = jtdm_fit(Y=Y, X=X,
             formula=as.formula("~GDD+FDD+forest"), sample = 1000)
plot(m)
```

`summary.jtdm_fit` *Prints the summary of a fitted jtdm*

Description

Prints the summary of a fitted jtdm

Usage

```
## S3 method for class 'jtdm_fit'
summary(object, ...)
```

Arguments

`object`	a model fitted with `jtdm_fit`
`...`	additional arguments

Value

A printed summary of the fitted jtdm

Author(s)

Giovanni Poggiato

Examples

```
data(Y)
data(X)
m = jtdm_fit(Y=Y, X=X,
             formula=as.formula("~GDD+FDD+forest"), sample = 1000)
summary(m)
```

X *Site x environmental covariates dataset*

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, 16
Y, 16

ellipse_plot, 2

get_sigma, 3
getB, 4
global, 5

joint_trait_prob, 5
joint_trait_prob_gradient, 7
jtdm_fit, 9
jtdm_predict, 10
jtdmCV, 12

partial_response, 13
plot.jtdm_fit, 14

summary.jtdm_fit, 15

X, 16

Y, 16