Package 'webSDM'

September 18, 2023

Title Including Known Interactions in Species Distribution Models

Version 1.1-4

Description A collection of tools to fit and work with trophic Species Distribution Models. Trophic Species Distribution Models combine knowledge of trophic interactions with Bayesian structural equation models that model each species as a function of its prey (or predators) and environmental conditions. It exploits the topological ordering of the known trophic interaction network to predict species distribution in space and/or time, where the prey (or predator) distribution is unavailable. The method implemented by the package is described in Poggiato, Andréoletti, Pollock and Thuiller (2022) <doi:10.22541/au.166853394.45823739/v1>.

License GPL-3

Encoding UTF-8

Imports GGally, abind, bayesplot, brms, broom.mixed, dismo, dplyr, ggplot2, glmnet, gridExtra, igraph, jtools, rstanarm, rstantools, utils

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

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

Suggests knitr,
rmarkdown,
reshape2,
devtools,
loo,
randomForest,
network,
sna,
scales,
intergraph

VignetteBuilder knitr

R topics documented:

`buildFormula` *Builds SDM formulae*

Description

Builds the formula of both the abiotic and biotic terms to fit a single species SDM based on the input parameters. The function is called inside the SDMfit function

Usage

```
buildFormula(
  form.init,
  species,
  sp.formula = NULL,
  sp.partition = NULL,
  useBRMS
)
```

Arguments

`form.init`	The abiotic part of the formula
`species`	The preys (or predators) of the focal species
`sp.formula`	optional parameter for composite variables. See ?trophicSDM
`sp.partition`	optional parameter to specify groups of species for composite variables. See ?trophicSDM
`useBRMS`	whether brms is used (TRUE if penal = "coeff.signs" and method = "stan_glm).

Author(s)

Giovanni Poggiato and Jérémy Andréoletti

`coef.SDMfit` *Gets regression coefficients from a local model, i.e. a SDMfit object.*

Description

Gets regression coefficients (eventually standardised) of a local model, i.e. a SDMfit object. p-values or credible intervals are returned when available.

Usage

```
## S3 method for class 'SDMfit'
coef(object, standardise = FALSE, level = 0.95, ...)
```

Arguments

`object`	A SDMfit object, typically obtained with trophicSDM() and available in the field $model of a trophicSDMfit object
`standardise`	Whether to standardise regression coefficients. Default to FALSE. If TRUE, coefficients are standardised using the latent variable standardisation (see Grace et al. 2018) for more details.
`level`	The confidence level of credible intervals, only available for stan_glm method. Default to 0.95.
`...`	additional arguments

Value

A table containing the inferred coefficients (with credible intervals or p-values when available).

Author(s)

Giovanni Poggiato

References

Grace, J. B., Johnson, D. J., Lefcheck, J. S., and Byrnes, J. E. K.. 2018. Quantifying relative importance: computing standardized effects in models with binary outcomes. Ecosphere 9(6):e02283.

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
m = trophicSDM(Y,X,G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
# unstandardised regression coefficients
coef(m$model$Y5)
```

```
#standardised regression coefficients with 90% credible intervals
coef(m$model$Y5, standardised = TRUE, level = 0.9)
# Run the same model using glm as fitting method
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y,X,G, env.formula,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "glm")
# Now we have p-values instead of credible intervals
coef(m$model$Y5)

# Notice that unstandardised coefficients are always accessible
# in the fitted model:
m$model$Y5$coef
```

`coef.trophicSDMfit` *Gets regression coefficients from a fitted trophicSDM model.*

Description

Gets regression coefficients (eventually standardised) of a fitted trophicSDM. p-values or credible intervals are returned when available.

Usage

```
## S3 method for class 'trophicSDMfit'
coef(object, standardise = FALSE, level = 0.95, ...)
```

Arguments

`object`	A trophicSDMfit object obtained with trophicSDM()
`standardise`	Whether to standardise regression coefficients. Default to FALSE. If TRUE, coefficients are standardised using the latent variable standardisation (see Grace et al. 2018) for more details.
`level`	The confidence level of credible intervals, only available for stan_glm method. Default to 0.95.
`...`	additional arguments

Value

A list containing, for each species, the inferred coefficients (with credible intervals or p-values when available).

Author(s)

Giovanni Poggiato

References

Grace, J. B., Johnson, D. J., Lefcheck, J. S., and Byrnes, J. E. K.. 2018. Quantifying relative importance: computing standardized effects in models with binary outcomes. Ecosphere 9(6):e02283.

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y,X,G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
# unstandardised regression coefficients
coef(m)
#standardised regression coefficients with 90% credible intervals
coef(m, standardised = TRUE, level = 0.9)
# Run the same model using glm as fitting method
m = trophicSDM(Y, X, G, env.formula,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "glm")
# Now we have p-values instead of credible intervals
coef(m)

# Notice that unstandardised coefficients are always accessible
# in the fitted model:
m$coef
```

`computeVariableImportance` *Computes variable importance of (groups of) variables of fitted a trophicSDM model.*

Description

Computes variable importance of (groups of) variables of fitted a trophicSDM model, for each species. Variable importance are computed as the standardised regression coefficients (summed across species of the same group). Standardisation is done using latent variable standardisation described in Grace et al. 2018.

Usage

```
computeVariableImportance(tSDM, groups = NULL)
```

Arguments

`tSDM`	A trophicSDMfit object obtained with trophicSDM()
`groups`	A list where each element is group. Each group is specified as a vector containing species or environmental covariates names of a given group. Each element of the list (i.e. each group) has to be named.

Value

A groups x species matrix containing variable importance for each groups of variables and each species.

Author(s)

Giovanni Poggiato

References

Grace, J. B., Johnson, D. J., Lefcheck, J. S., and Byrnes, J. E. K.. 2018. Quantifying relative importance: computing standardized effects in models with binary outcomes. Ecosphere 9(6):e02283.

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
#Compute the importance of each variable
computeVariableImportance(m)
#Compute the importance of three different set of variables
computeVariableImportance(m, groups =list("X" = c("X_1","X_2"),
                                          "Ybasal" = c("Y1","Y2","Y3"),
                                          "Ypredator"= c("Y4", "Y5", "Y6")))
```

`evaluateModelFit` *Evaluates prediction goodness of fit*

Description

Evaluate goodness of fit by comparing a true versus a predicted dataset of species distribution. Ypredicted is typically predicted using a prediction method of trophicSDM (in cross-validation if `trophicSDM_CV()` is used).

Usage

```
evaluateModelFit(tSDM, Ynew = NULL, Ypredicted = NULL)
```

Arguments

`tSDM`	A trophicSDMfit object obtained with `trophicSDM()`.
`Ynew`	A sites x species matrix containing the true species occurrences state. If set to NULL (default), it is set to the species distribution data Y on which the model is fitted.
`Ypredicted`	A sites x species matrix containing the predicted species occurrences state. If set to NULL (default), it is set to the fitted values, i.e. predictions on the dataset used to train the model.

Value

A table specifying the goodness of fit metrics for each species. For presence-absence data, the model computes TSS and AUC. For Gaussian data, the R2.

Author(s)

Giovanni Poggiato

References

Grace, J. B., Johnson, D. J., Lefcheck, J. S., and Byrnes, J. E. K.. 2018. Quantifying relative importance: computing standardized effects in models with binary outcomes. Ecosphere 9(6):e02283.

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 10,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
# Evaluate the quality of model predictions on the training
# Predict (fullPost = FALSE) as we used stan_glm to fit the model
# but here we are only intested in the posterior mean
Ypred = predict(m, fullPost = FALSE)
# format predictions to obtain a sites x species dataset whose
# columns are ordered as Ynew
Ypred = do.call(cbind,
                lapply(Ypred, function(x) x$predictions.mean))

Ypred = Ypred[,colnames(Y)]
evaluateModelFit(m, Ynew = Y, Ypredicted = Ypred)

# Note that this is equivalent to `evaluateModelFit(m)`
# If we fitted the model using "glm"
m = trophicSDM(Y, X, G, env.formula,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "glm")
Ypred = predict(m, fullPost = FALSE)
# format predictions to obtain a sites x species dataset whose
# columns are ordered as Ynew
Ypred = do.call(cbind, Ypred)
Ypred = Ypred[,colnames(Y)]

evaluateModelFit(m, Ynew = Y, Ypredicted = Ypred)
# Note that this is equivalent to:

evaluateModelFit(m)
```

G *Simulated environemntal covariates G*

Description

Simulated environemntal covariates G

Usage

```
data(G)
```

Format

A simulated graph of trophic interactions G

Author(s)

Giovanni Poggiato

Examples

```
data(G)
```

---

global *Global*

Description

Declare global variables

---

loo.trophicSDMfit *Computes an approximation of loo for the whole model*

Description

Only works if method = 'stan_glm'. The global loo is computed by summing the loo of all the local models (since the likelihood factorises, the log-likelihood can be summed)This is an implementation of the methods described in Vehtari, Gelman, and Gabry (2017) and Vehtari, Simpson, Gelman, Yao, and Gabry (2019).

Usage

```
## S3 method for class 'trophicSDMfit'
loo(x, ...)
```

Arguments

x	A trophicSDMfit object obtained with trophicSDM()
...	additional arguments

Value

The value of the loo for the whole model

Author(s)

Giovanni Poggiato

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
m = trophicSDM(Y,X,G, env.formula,
               family = binomial(link = "logit"), penal = NULL, iter = 50,
               mode = "prey", method = "stan_glm")
brms::loo(m)
```

`plot.SDMfit` *Plots the regression coefficients of a local model*

Description

Plots the regression coefficients of a local SDMfit model

Usage

```
## S3 method for class 'SDMfit'
plot(x, level = 0.95, ...)
```

Arguments

`x`	A SDMfit object, typically obtained with trophicSDM() and available in the field $model of a trophicSDMfit object
`level`	the confidence level of the confidence intervals
`...`	additional arguments

Value

A plot of the regression coefficients of the fitted local SDM

Author(s)

Giovanni Poggiato

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 50,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
# Plot species Y6

plot(m$model$Y6)
```

`plot.trophicSDMfit` *Plots the regression coefficients of a fitted trophicSDM model*

Description

Plots the regression coefficients of a fitted trophicSDM model. A subset of species to be plotted can be specified in the parameterspecies.

Usage

```
## S3 method for class 'trophicSDMfit'
plot(x, species = NULL, ...)
```

Arguments

x	A trophicSDMfit object obtained with trophicSDM()
species	A vector of species names to be plot. If NULL (default), all species are plotted.
...	additional arguments

Value

A plot of the regression coefficients of the fitted tropic SDM

Author(s)

Giovanni Poggiato

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 50,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
# Plot just the first three species

plot(m, species = c("Y1","Y2","Y3"))

# If species = NULL (default), all species are plotted.
```

plotG *Plots the metaweb G*

Description

Plots the metaweb G used to fit the trophicSDM model

Usage

```
plotG(tSDM)
```

Arguments

tSDM	A trophicSDMfit object obtained with trophicSDM()

Value

A ggnet object

Author(s)

Giovanni Poggiato

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")

plotG(m)
```

plotG_inferred *Plot the metaweb G according to the inferred coefficients*

Description

Plot the metaweb G with links colored accordingly to the inferred prey-predator regression coefficients of a fitted trophicSDM model. Plots the metaweb G, where each predator-prey link is colored according to whether the related regression coefficient if inferred as positive (in red), negative (in blue) or non-significant (dashed grey line) according to the confidence level specified in "level". Estimates of the significant standardised regression coefficients are pasted on the links. Only works if species are modeled as a function of their preys or predators without composite variables (i.e., the function only works if tSDM is fitted with sp.formula = NULL and sp.partition = NULL)

Usage

```
plotG_inferred(tSDM, level = 0.9)
```

Arguments

tSDM	A trophicSDMfit object obtained with trophicSDM()
level	The confidence level used to decide whether regression coefficients are non-significant or not. Default to 0.9.

Value

A ggnet object

Author(s)

Giovanni Poggiato

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "glm")

plotG_inferred(m)
```

---

predict.SDMfit *Predicts with a local model*

Description

Computes predicted values for a local model, i.e., a fitted SDMfit object This is sequentially called, for each species, by the function trophicSDM.predict

Usage

```
## S3 method for class 'SDMfit'
predict(object, newdata, pred_samples = NULL, prob.cov = TRUE, ...)
```

Arguments

object	A SDMfit object, typically obtained with trophicSDM() and available in the field $model of a trophicSDMfit object
newdata	A matrix containing both environmental covariates and the biotic variables that the local model uses to predict the species distribution.
pred_samples	Number of samples to draw from species posterior predictive distribution when method = "stan_glm". If NULL, set by the default to the number of iterations/10.

prob.cov	Only for presence-absence data. If set to FALSE, it gives back also predicted presence-absences (which is then used by trophicSDM.predict to predict the predators).
...	additional arguments

Value

A list containing for each species the predicted value at each sites. If method = "stan_glm", then each element of the list is a sites x pred_samples matrix containing the posterior predictive distribution of the species at each sites. If prob.cov = TRUE, it returns a list containing:

- predictions.probPredicted probabilities of presence
- predictions.probPredicted presence-absences

Author(s)

Giovanni Poggiato and Jérémy Andréoletti

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
# In order to predict non-basal species, we need to also provide
# the predicted occurrences of its preys. Here we compute the probability of
# presence of species Y4 at environemntal conditions c(0.5,0.5)
# when its prey Y3 is present.
predict(m$model$Y4, newdata = data.frame(X_1 = 0.5, X_2 = 0.5, Y3 = 1), pred_samples = 10)
```

---

predict.trophicSDMfit *Computes predicted values from the fitted trophicSDMfit model*

Description

Computes predicted values from the fitted trophicSDMfit model at environmental conditions specified by Xnew. Once predictions have been obtained, their quality can eventually be evaluated with evaluateModelFit().

Usage

```
## S3 method for class 'trophicSDMfit'
predict(
  object,
  Xnew = NULL,
  prob.cov = FALSE,
  pred_samples = NULL,
  run.parallel = FALSE,
```

```
  verbose = FALSE,
  fullPost = TRUE,
  filter.table = NULL,
  ...
)
```

Arguments

object	A trophicSDMfit object obtained with trophicSDM()
Xnew	a matrix specifying the environmental covariates for the predictions to be made. If NULL (default), predictions are done on the training dataset (e.g. by setting Xnew = tSDM$data$X).
prob.cov	Parameter to predict with trophicSDM with presence-absence data. Whether to use predicted probability of presence (prob.cov = T) or the transformed presence-absences (default, prov.cov = F) to predict species distribution.
pred_samples	Number of samples to draw from species posterior predictive distribution when method = "stan_glm". If NULL, set by the default to the number of iterations/10.
run.parallel	Whether to use parallelise code when possible. Can speed up computation time.
verbose	Whether to print advances of the algorithm
fullPost	Optional parameter for stan_glm only. Whether to give back the full posterior predictive distribution (default, fullPost = TRUE) or just the posterior mean, and 2.5% and 97.5% quantiles,
filter.table	Optional, default to NULL, should be provided only if the users wants to filter some species predictions. A sites x species matrix of zeros and ones.
...	additional arguments

Value

A list containing for each species the predicted value at each sites. If method = "stan_glm", then each element of the list is a sites x pred_samples matrix containing the posterior predictive distribution of the species at each sites.

Author(s)

Giovanni Poggiato and Jérémy Andréoletti

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
# We can now evaluate species probabilities of presence for the environmental conditions c(0.5, 0.5)
predict(m, Xnew = data.frame(X_1 = 0.5, X_2 = 0.5))
# Obtain 50 draws from the posterior predictive distribution of species (pred_samples = 50)
# using predicted presence-absences of species to predict their predators (prob.cov = TRUE)
# Since we don't specify Xnew, the function sets Xnew = X by default
Ypred = predict(m, fullPost = TRUE, pred_samples = 50, prob.cov = FALSE)
```

```
# We can ask the function to only give back posterior mean and 95% credible intervals with
# fullPost = F

Ypred = predict(m, fullPost = TRUE, pred_samples = 50, prob.cov = FALSE)

# If we fit the model using in a frequentist  way (e.g. glm)
m = trophicSDM(Y, X, G, env.formula,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "glm")
# We are obliged to set pred_samples = 1
# (this is done by default if pred_samples is not provided)
# In the frequentist case, fullPost is useless.
 Ypred = predict(m, pred_samples = 1, prob.cov = FALSE)
```

`predictPotential` *Predicts species potential niche*

Description

Computes predicted values of the potential niches of species from the fitted trophicSDMfit model at environmental conditions specified by `Xnew`. Predictions are obtained by setting preys to present when mode = "prey" or setting predators to absent when mode = "predator".

Usage

```
predictPotential(
  tSDM,
  Xnew = NULL,
  pred_samples = NULL,
  verbose = FALSE,
  fullPost = TRUE
)
```

Arguments

`tSDM`	A trophicSDMfit object obtained with trophicSDM()
`Xnew`	a matrix specifying the environmental covariates for the predictions to be made. If NULL (default), predictions are done on the training dataset (e.g. by setting Xnew = tSDM$data$X).
`pred_samples`	Number of samples to draw from species posterior predictive distribution when method = "stan_glm". If NULL, set by the default to the number of iterations/10.
`verbose`	Whether to print advances of the algorithm.
`fullPost`	Optional parameter for stan_glm only. Whether to give back the full posterior predictive distribution (default, fullPost = TRUE) or just the posterior mean, and 2.5% and 97.5% quantiles.

Value

A list containing for each species the predicted value at each sites. If method = "stan_glm", then each element of the list is a sites x pred_samples matrix containing the posterior predictive distribution of the species at each sites.

Author(s)

Giovanni Poggiato and Jérémy Andréoletti

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
# Obtain 100 draws from the posterior predictive distribution of species potential niche
# (pred_samples = 50)
# Since we don't specify Xnew, the function sets Xnew = X by default
Ypred = predictPotential(m, fullPost = TRUE, pred_samples = 50)
# We can ask the function to only give back posterior mean and 95% credible intervals with
# fullPost = FALSE

Ypred = predictPotential(m, fullPost = FALSE, pred_samples = 50)

#' We can now evaluate species probabilities of presence for the enviromental
# conditions c(0.5, 0.5)
predictPotential(m, Xnew = data.frame(X_1 = 0.5, X_2 = 0.5), pred_samples = 50)

# If we fit the model using in a frequentist  way (e.g. glm)
m = trophicSDM(Y, X, G, env.formula,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "glm")
# We are obliged to set pred_samples = 1
# (this is done by default if pred_samples is not provided)
# In the frequentist case, fullPost is useless.
Ypred = predictPotential(m, pred_samples = 1)
```

`print.SDMfit` *Prints a SDMfit object*

Description

Prints a SDMfit object

Usage

```
## S3 method for class 'SDMfit'
print(x, ...)
```

Arguments

`x`	A SDMfit object, typically obtained with trophicSDM() and available in the field $model of a trophicSDMfit object
`...`	additional arguments

Value

Prints a summary of the local SDM

Author(s)

Giovanni Poggiato

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
m$model$Y1
```

---

print.trophicSDMfit *Prints a fitted trophicSDM model*

Description

Prints a fitted trophicSDM model

Usage

```
## S3 method for class 'trophicSDMfit'
print(x, ...)
```

Arguments

x	A trophicSDMfit object obtained with trophicSDM()
...	additional arguments

Value

Prints a summary of the fitted trophic SDM

Author(s)

Giovanni Poggiato

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
trophicSDM(Y, X, G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
```

SDMfit *Fitting a single-species SDM*

Description

SDMfit is used to fit a single species SDM, what we call a 'local model' of trophicSDM. It returns an object of class 'SDMfit'. Requires basically the same inputs of trophicSDM, with the requirement to specify with the parameter 'focal' the species that is modeled by the SDMfit.

Usage

```
SDMfit(
  focal,
  Y,
  X,
  G,
  formula.foc,
  sp.formula = NULL,
  sp.partition = NULL,
  mode = "prey",
  method = "stan_glm",
  family,
  penal = NULL,
  iter = 1000,
  chains = 2,
  verbose = TRUE
)
```

Arguments

focal	the name of the species to be modeled
Y	The sites x species matrix containing observed species distribution (e.g. presence-absence).
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.
G	The species interaction network (aka metaweb). Needs to be an igraph object. Links must go from predator to preys. It needs to be a directed acyclic graph.
formula.foc	The formula for the abiotic part of the species distribution model.
sp.formula	(optional) It allows to specify a particular definition of the biotic part of the model, e.g., using composite variables (e.g., richness), or an interaction of the biotic and abiotic component. More details in 'Details'.
sp.partition	(optional) a list to specify groups of species that are used to compute composite variables, e.g., a species can be modeled as a function of the richness of each group of preys. It has to be a list, each element is a vector containing the names of species in the group.
mode	"prey" if bottom-up control (default), "predators" otherwise. Notice that G needs to be such that links point from predators to prey.

method	which SDM method to use. For now the available choices are: "glm" (frequentist) or "stan_glm" (full Bayesian MCMC, default). Notice that using "glm" does not allow error propagation when predicting.
family	the family parameter of the glm function (see glm). family=gaussian(link ="identity") for gaussian data or family=binomial(link = "logit") or binomial(link = "probit") for presence-absence data.
penal	(optional, default to NULL) Penalisation method to shrink regression coefficients.If NULL (default), the model does not penalise the regression coefficient. For now, available penalisation method are "horshoe" for stan_glm, "elasticnet" for glm and "coeff.signs" (prey coefficients are set to positive and predator coefficients to negative) for glm and stan_glm.
iter	(for method="stan_glm" only) Number of iterations for each MCMC chain if stan_glm is used
chains	(for method="stan_glm" only) Number of MCMC chains (default to 2)
verbose	Whether to print algorithm progresses

Details

"sp.formula" and "sp.partition" can be combined to define any kind of composite variables for the biotic part of the formula. "sp.formula" can be :

- A string defining a formula as function of "richness". E.g., sp.formula="richness+I(richness)^2" (species are modeled as a function of a quadratic polynomial of their prey richness), "I(richness>0)" (species are modeled as a function of a dummy variable that is equal to 1 when at least one species is present). Importantly, when group of preys (or predators) are specified by "sp.partition", species are modeled as a function of the composite variable specified by "sp.formula" for each of their prey groups.

- A more flexible option is to specify sp.formula as a list (whose names are species' names) that contains for each species the definition biotic part of the model. Notice that, in this case, the function does not check that the model is a DAG. This allow to define any kind of composite variable, or to model interactions between environmental covariates and preys (or predators).

Value

A list containing 'm', a "SDMfit" object and 'form.all', a string describing the formula of the SDMfit object. The "SDM" fit object contains:

model	The output of the function used to fit the SDM. E.g., an object of class "glm" is method = "glm", an object of class "stanreg" if method = "stan_glm".
Y	A numeric vector of standard errors on parameters
form.all	The formula used to fit the SDM (both abiotic and biotic terms)
method, family, penal, iter, chains	The input parameters used to fit the SDM.
sp.name	The name of the species modeled
data	The model.frame data.frame used to fit the model
coef	The inferred coefficients (with credible intervals or p-values when available)
AIC	The AIC of the local model
log.lik	The log.likelihood of the local model

Author(s)

Giovanni Poggiato and Jérémy Andréoletti

Examples

```
data(Y,X,G)
# Run a local model (i.e. a SDM) for species Y6
mySDM = SDMfit("Y6", Y, X, G, "~X_1 + X_2", mode = "prey",
       method = "stan_glm", family = binomial(link = "logit"))
mySDM$m
```

---

summary.SDMfit *Summary of a fitted SDMfit model*

Description

Summary of a fitted SDMfit model

Usage

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

Arguments

object	A SDMfit object, typically obtained with trophicSDM() and available in the field $model of a trophicSDMfit object
...	additional arguments

Value

Prints a summary of the local SDM

Author(s)

Giovanni Poggiato

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
summary(m$model$Y1)
```

summary.trophicSDMfit *Summary of a fitted trophicSDM model*

Description

Summary of a fitted trophicSDM model

Usage

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

Arguments

object	A trophicSDMfit object obtained with trophicSDM()
...	additional arguments

Value

Prints a summary of the fitted trophic SDM

Author(s)

Giovanni Poggiato

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)
m = trophicSDM(Y, X, G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
summary(m)
```

trophicSDM *Fitting a trophic Species distribution model*

Description

trophicSDM is used to fit a trophic species distribution model. Requires the species distribution data Y (the sites x species matrix), explanatory variables X and a directed acyclic graph G containing species interactions (i.e., the metaweb, with links going from predators to prey). The function fits the distribution of each species as a function of their preys (with mode = "prey", by default) or predators (if set mode = "predator").

Usage

```
trophicSDM(
  Y,
  X,
  G,
  env.formula = NULL,
  sp.formula = NULL,
  sp.partition = NULL,
  penal = NULL,
  mode = "prey",
  method = "stan_glm",
  family,
  iter = 500,
  chains = 2,
  run.parallel = FALSE,
  verbose = FALSE
)
```

Arguments

Y	The sites x species matrix containing observed species distribution (e.g. presence-absence).
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.
G	The species interaction network (aka metaweb). Needs to be an igraph object. Links must go from predator to preys. It needs to be a directed acyclic graph.
env.formula	The definition of the abiotic part of the model. It can be : • a string specifying the formula (e.g. "~ X_1 + X_2"). In this case, the same environmental variables are used for every species. • A list that contains for each species the formula that describes the abiotic part of the model. In this case, different species can be modeled as a function of different environmental covariates. The names of the list must coincide with the names of the species.
sp.formula	(optional) It allows to specify a particular definition of the biotic part of the model, e.g., using composite variables (e.g., richness), or an interaction of the biotic and abitic component. More details in 'Details'.
sp.partition	(optional) a list to specify groups of species that are used to compute composite variables, e.g., a species can be modeled as a function of the richness of each group of preys. It has to be a list, each element is a vector containing the names of species in the group. More details in 'Details'.
penal	Penalisation method to shrink regression coefficients. If NULL (default), the model does not penalise the regression coefficient. For now, available penalization method are "horshoe" for method stan_glm, "elasticnet" for method glm. It is also possible to constrain the sign of biotic coefficients (prey coefficients are set to positive and predator coefficients to negative) by setting "coeff.signs" for methods glm and stan_glm.
mode	"prey" if bottom-up control (default), "predators" otherwise. Notice that G needs to be such that links point from predators to prey.

method	which SDM method to use. For now the available choices are: "glm" (frequentist) or "stan_glm" (full bayesian MCMC, default). Notice that using "glm" does not allow error propagation when predicting.
family	the family parameter of the glm function (see glm). gaussian(link = "identity") for gaussian data. binomial(link = "logit") or binomial(link = "probit") for presence-absence data.
iter	(for "stan_glm" only) Number of iterations for each MCMC chain if stan_glm is used
chains	(for "stan_glm" only) Number of MCMC chains (default to 2)
run.parallel	Whether species models are fitted in parallel (can speed computational up time). Default to FALSE.
verbose	Whether to print algorithm progresses

Details

"sp.formula" and "sp.partition" can be combined to define any kind of composite variables for the biotic part of the formula. "sp.formula" can be :

- A string defining a formula as function of "richness", e.g., "richness+I(richness)^2" (species are modeled as a function of a quadratic polynomial of their prey richness), "I(richness>0)" (species are modeled as a function of a dummy variable that is equal to 1 when at least one species is present). Importantly, when group of preys (or predators) are specified by "sp.partition", species are modeled as a function of the composite variable specified by "sp.formula" for each of their prey (or predator) groups.
- A more flexible option is to specify sp.formula as a list (whose names are species' names) that contains for each species the definition of the biotic part of the model. Notice that, in this case, the function does not check that the model is a DAG. This allow to define any kind of composite variable, or to model interactions between environmental covariates and preys (or predators).

Value

A "trophicSDMfit" object, containing:

model	A list containing the local models (i.e. a SDM for each species). Each local model is an object of class "SDMfit". See ?SDMfit for more informations.
Y	A numeric vector of standard errors on parameters
form.all	A list describing each species formula (both biotic and abiotic terms)
data	A list containing all the data used to fit the model
model.call	A list containing the modeling choices of the fitted model (e.g. method, penalisation...)
coef	A list containing, for each species, the inferred coefficients (with credible intervals or p-values when available)
MCMC.diag	MCMC convergence metrics, only available for MCMC methods
AIC	Model's AIC
log.lik	Model's log.likelihood

Author(s)

Giovanni Poggiato and Jérémy Andréoletti

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using stan_glm as fitting method and no penalisation
# Increase the number of iterations to obtain reliable results.
m = trophicSDM(Y,X,G, env.formula, iter = 100,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")
print(m)

# Access local models (e.g. species "Y5")
m$model$Y5
coef(m$model$Y5)
# The fitted model can be plotted with `plot(m)`

# Fit a sparse model in the Bayesian framework with the horshoe prior

m = trophicSDM(Y,X,G, env.formula,
               family = binomial(link = "logit"), penal = "horshoe",
               mode = "prey", method = "stan_glm")

# Fit frequentist glm
m = trophicSDM(Y,X,G, env.formula,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "glm")

# With elasticnet penalty
m = trophicSDM(Y,X,G, env.formula,
               family = binomial(link = "logit"), penal = "elasticnet",
               mode = "prey", method = "glm")

#### Composite variables
# See vignette 'Composite variables' for a complete introduction to the use of composite variables
# Model species as a function of a quadratic polynomial of prey richness
m = trophicSDM(Y,X,G, env.formula,
               family = binomial(link = "logit"), penal = NULL,
               sp.formula = "richness + I(richness^2)",
               mode = "prey", method = "glm")
m$form.all
# Notice that for predators that feed on a single prey (with presence-absence data),
# their richness and the square of their richness is exactly the same variable
# In this case, `trophicSDM()` removes the redundant variable but prints a warning message

# Model species as a function of a dummy variable saying whether they have at leaste one prey
m = trophicSDM(Y,X,G, env.formula,
               family = binomial(link = "logit"), penal = NULL,
               sp.formula = "I(richness>0)",
               mode = "prey", method = "glm")
m$form.all

# Define group of preys and model species as a function of the richness (with a quadratic term)
# of these groups of preys separately

# Species Y1 and Y2 belong to the same group, species Y3 and Y4 are both alone in their group and
# species Y5 and Y6 form another group
```

```
sp.partition = list(c("Y1","Y2"),c("Y3"),c("Y4"), c("Y5","Y6"))

m = trophicSDM(Y,X,G, env.formula,
               family = binomial(link = "logit"), penal = NULL,
               sp.partition = sp.partition,
               sp.formula = "richness + I(richness^2)",
               mode = "prey", method = "glm")
m$form.all
```

`trophicSDM_CV` *Compute K-fold cross-validation predicted values from a fitted trophicSDM model*

Description

Once the CV predicted values are obtained, their quality can be evaluated with `evaluateModelFit()`.

Usage

```
trophicSDM_CV(
  tSDM,
  K,
  partition = NULL,
  prob.cov = FALSE,
  pred_samples = NULL,
  iter = NULL,
  chains = NULL,
  run.parallel = FALSE,
  verbose = FALSE
)
```

Arguments

`tSDM`	A trophicSDMfit object obtained with trophicSDM()
`K`	The number of folds for the K-fold cross validation
`partition`	Optional parameter. A partition vector to specify a partition in K fold for cross validation
`prob.cov`	Parameter to predict with trophicSDM with presence-absence data. Whether to use predicted probability of presence (prob.cov = T) or the transformed presence-absences (default, prov.cov = F) to predict species distribution.
`pred_samples`	Number of samples to draw from species posterior predictive distribution when method = "stan_glm". If NULL, set by the default to the number of iterations/10.
`iter`	For method = "stan_glm": number of iterations of each MCMC chains to fit the trophicSDM model. Default to the number of iterations used to fit the provided trophicSDMfit object
`chains`	For method = "stan_glm": number of MCMC chains to fit the trophicSDM model. Default to the number of iterations used to fit the provided trophicSDMfit object
`run.parallel`	Whether to use parallelise code when possible. Default to TRUE. Can speed up computation time
`verbose`	Whether to print advances of the algorithm

Value

A list containing:

meanPred	a sites x species matrix of predicted occurrences of species for each site (e.g. probability of presence). With stan_glm the posterior predictive mean is return
Pred975,Pred025	Only for method = "stan_glm", the 97.5% and 2.5% quantiles of the predictive posterior distribution
partition	the partition vector used to compute the K fold cross-validation

Author(s)

Giovanni Poggiato

Examples

```
data(Y, X, G)
# define abiotic part of the model
env.formula = "~ X_1 + X_2"
# Run the model with bottom-up control using glm as fitting method and no penalisation
# (set iter = 1000 to obtain reliable results)

m = trophicSDM(Y, X, G, env.formula, iter = 50,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "stan_glm")

# Run a 3-fold (K=3) cross validation. Predictions is done using presence-absences of preys
# (prob.cov = FALSE, see ?predict.trophicSDM) with 50 draws from the posterior distribution
# (pred_samples = 50)
CV = trophicSDM_CV(m, K = 3, prob.cov = FALSE, pred_samples = 10, run.parallel = FALSE)
# Use predicted values to evaluate model goodness of fit in cross validation
Ypred = CV$meanPred[,colnames(Y)]

evaluateModelFit(m, Ynew = Y, Ypredicted = Ypred)

# Now with K = 2 and by specifying the partition of site
m = trophicSDM(Y, X, G, env.formula, iter = 50,
               family = binomial(link = "logit"), penal = NULL,
               mode = "prey", method = "glm")
partition = c(rep(1,500),rep(2,500))
CV = trophicSDM_CV(m, K = 2, partition = partition, prob.cov = FALSE,
                   pred_samples = 10, run.parallel = FALSE)
Ypred = CV$meanPred[,colnames(Y)]
evaluateModelFit(m, Ynew = Y, Ypredicted = Ypred)
```

---

webSDM *webSDM.*

Description

Package to fit a trophic Species Distribution Model, analyse it and predict. See Poggiato et al. In prep.

X — *Simulated environmental covariates X*

Description

Simulated environmental covariates X

Usage

```
data(X)
```

Format

A site x covariates matrix X

Author(s)

Giovanni Poggiato

Examples

```
data(X)
```

Y — *Simulated species distribution Y*

Description

Simulated species distribution Y

Usage

```
data(Y)
```

Format

A site x species matrix Y, a site x covariates matrix X and a trophic interaction network G (object igraph)

Author(s)

Giovanni Poggiato

Examples

```
data(Y)
```

Index