Package 'cxr'

October 12, 2022

Type Package

Title A Toolbox for Modelling Species Coexistence in R

Version 1.0.0

URL https://github.com/RadicalCommEcol/cxr

Description Recent developments in modern coexistence theory have advanced our understanding on how species are able to persist and co-occur with other species at varying abundances. However, applying this mathematical framework to empirical data is still challenging, precluding a larger adoption of the theoretical tools developed by empiricists. This package provides a complete toolbox for modelling interaction effects between species, and calculate fitness and niche differences.
The functions are flexible, may accept covariates, and different fitting algorithms can be used.
A full description of the underlying methods is available in García-Callejas, D., Godoy, O., and Bartomeus, I. (2020) <doi:10.1111/2041-210X.13443>.

License MIT + file LICENSE

BugReports https://github.com/RadicalCommEcol/cxr/issues

Encoding UTF-8

Depends R (>= 3.5)

Imports optimx, stats, mvtnorm

RoxygenNote 7.1.1

Suggests ggplot2, tidyr, dplyr, magrittr, knitr, stringr, rmarkdown, testthat (>= 0.8.0), BB, ucminf, dfoptim, minqa, hydroPSO, GenSA, DEoptimR, nloptr

VignetteBuilder knitr

NeedsCompilation no

Author David Garcia-Callejas [aut, cre] (<https://orcid.org/0000-0001-6982-476X>), Ignasi Bartomeus [aut] (<https://orcid.org/0000-0001-7893-4389>), Oscar Godoy [aut] (<https://orcid.org/0000-0003-4988-6626>), Maxime Lancelot [ctb]

Maintainer David Garcia-Callejas <david.garcia.callejas@gmail.com>

Repository CRAN

Date/Publication 2021-04-16 09:20:02 UTC

R topics documented:

---

abundance *Abundance measurements*

Description

A dataset containing abundances for each plant species, where each species was sampled at its developmental peak.

- plot: one of 9 plots of the study area
- subplot: one of 36 1x1 m subplots of each plot
- species: plant species
- individuals: number of individuals observed

Usage

```
data(abundance)
```

Format

A data frame with 5184 rows and 4 variables

Note

For details, see Lanuza et al. 2018 Ecology Letters.

`abundance_projection` *Title Project abundances from population dynamics models*

Description

The function projects a number of steps of a time-discrete model, with model parameters taken from a 'cxr_pm_multifit' object or as function arguments.

Usage

```
abundance_projection(
  cxr_fit = NULL,
  model_family = NULL,
  alpha_form = NULL,
  lambda_cov_form = NULL,
  alpha_cov_form = NULL,
  lambda = NULL,
  alpha_matrix = NULL,
  lambda_cov = NULL,
  alpha_cov = NULL,
  covariates = NULL,
  timesteps = 2,
  initial_abundances = 0
)
```

Arguments

`cxr_fit`	object of type 'cxr_pm_multifit'. If this is not specified, all parameters below are needed.
`model_family`	acronym for model family. Included by default in 'cxr' are 'BH' (Beverton-Holt), 'RK' (Ricker), 'LW' (Law-Watkinson), 'LV' (Lotka-Volterra).
`alpha_form`	character, either "none","global", or "pairwise".
`lambda_cov_form`	character, either "none" or "global".
`alpha_cov_form`	character, either "none","global", or "pairwise".
`lambda`	named vector with lambda values for all taxa to be projected.
`alpha_matrix`	square matrix with taxa names in rows and columns.

lambda_cov	optional named matrix with covariates in columns and taxa in rows, representing the effect of each covariate on the lambda parameter of each taxa.
alpha_cov	optional list. Each element of the named list represents the effects of a covariate over alpha values. Thus, each list element contains a square matrix of the same dimensions as 'alpha_matrix', as returned from the function 'cxr_pm_fit'. Note that for alpha_cov_form = "global", all columns in this matrix are the same, as there is a single value per species.
covariates	matrix or dataframe with covariates in columns and timesteps in rows.
timesteps	number of timesteps to project.
initial_abundances	named vector of initial abundances for all taxa.

Value

named matrix with projected abundance values for each taxa at each timestep.

---

avg_fitness_diff *Average fitness differences*

Description

computes the average fitness differences among two or more species according to the formulation of the MCT (Chesson 2012, Godoy and Levine 2014), and according to the structural approach (Saavedra et al. 2017). For the MCT version, the average fitness ratio is decomposed in a 'demographic ratio' and a 'competitive response ratio', the product of which is the average fitness ratio (Godoy and Levine 2014). This formulation is only valid for competitive interaction coefficients (i.e. positive alpha values in the interaction matrix). The structural analog can be computed for any interaction matrix, on the other hand. Note that the 'demographic ratio' is model-specific (Hart et al. 2018).

Usage

```
avg_fitness_diff(
  cxr_multifit = NULL,
  cxr_sp1 = NULL,
  cxr_sp2 = NULL,
  pair_lambdas = NULL,
  pair_matrix = NULL,
  model_family = NULL
)
```

Arguments

cxr_multifit	cxr_pm_multifit object, with parameters for a series of species.
cxr_sp1	cxr_pm_fit object giving the parameters from the first species.
cxr_sp2	cxr_pm_fit object giving the parameters from the second species.

pair_lambdas	numeric vector of length 2 giving lambda values for the two species.
pair_matrix	2x2 matrix with intra and interspecific interaction coefficients between the two species.
model_family	model family for which to calculate fitness differences.

Details

This function, as in niche_overlap and competitive_ability, accepts three different parameterizations:

- A cxr_pm_multifit object, from which average fitness differences will be computed across all species pairs.
- two cxr_pm_fit objects, one for each species.
- explicit lambda and alpha values, as well as the model family from which these parameters were obtained.

If using the third parameterization, the function will try to find a model-specific function for obtaining the demographic ratio, by looking at the 'model_family' parameter. If this specific function is not found, it will resort to the standard Lotka-Volterra formulation (lambda in the numerator term). Overall, we strongly suggest that you use the standard formulation ONLY if you are completely confident that your custom model is consistent with it. Otherwise, you should include your own formulation of the demographic ratio (see vignette 4).

Value

data frame with variable number of rows, and columns specifying the different components of the MCT average fitness ratio, as well as its structural analog. The average fitness ratio informs quantitatively about the better competitor. If the ratio is < 1, sp2 is the better competitor; if = 1, both species are equivalent competitors, if > 1, sp1 is the better competitor.

Examples

```
avg_fitness_diff(pair_lambdas = runif(2,1,10),
                 pair_matrix = matrix(runif(4,0,1),nrow = 2),
                 model_family = "BH")
```

BH_er_lambdacov_global_effectcov_global_responsecov_global

Effect response Beverton-Holt model with covariate effects on lambda, effect, and response

---

Description

The function for calculating fecundity given effect and response values is taken from Godoy et al. (2014). Note that, as e and r are not pair-specific, all species parameters are fit in the same function.

Usage

```
BH_er_lambdacov_global_effectcov_global_responsecov_global(
  par,
  fitness,
  target,
  density,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector with initial parameters in the order: lambda,lambda_cov,effect,effect_cov,response,response_co
fitness	1d vector with fitness observations
target	matrix with species in rows, observations in columns. Value is 1 if a species is focal for a given observation, 0 otherwise.
density	matrix with species in rows, observations in columns. Value is density of each sp as neighbour for each observation.
covariates	numeric dataframe or matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","lambda_cov","effect","eff "response","response_cov".

Value

log-likelihood value

BH_er_lambdacov_none_effectcov_none_responsecov_none

Effect response model without covariate effects

Description

The function for calculating fecundity given effect and response values is taken from Godoy et al. (2014). Note that, as e and r are not pair-specific, all species parameters are fit in the same function.

Usage

```
BH_er_lambdacov_none_effectcov_none_responsecov_none(
  par,
  fitness,
  target,
  density,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector with initial parameters in the order: lambda,effect,response,sigma.
fitness	1d vector with fitness observations.
target	matrix with species in rows, observations in columns. Value is 1 if a species is focal for a given observation, 0 otherwise.
density	matrix with species in rows, observations in columns. Value is density of each sp as neighbour for each observation.
covariates	included for compatibility, not used in this model.
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","effect","response".

Value

log-likelihood value

BH_pm_alpha_global_lambdacov_none_alphacov_none

Beverton-Holt model with a global alpha and no covariate effects

---

Description

Beverton-Holt model with a global alpha and no covariate effects

Usage

```
BH_pm_alpha_global_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector of initial parameters: lambda, alpha, and sigma.
fitness	1d vector of fitness observations, in log scale.
neigh_intra_matrix	included for compatibility, not used in this model.
neigh_inter_matrix	matrix of arbitrary columns, number of neighbours for each observation. As in this model there is a single alpha argument, do not distinguish neighbour identity
covariates	included for compatibility, not used in this model.
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","alpha_inter".

Value

log-likelihood value

`BH_pm_alpha_none_lambdacov_none_alphacov_none` *Beverton-Holt model with no alphas and no covariate effects*

Description

Beverton-Holt model with no alphas and no covariate effects

Usage

```
BH_pm_alpha_none_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: lambda and sigma
`fitness`	1d vector of fitness observations, in log scale
`neigh_intra_matrix`	included for compatibility, not used in this model.
`neigh_inter_matrix`	included for compatibility, not used in this model.
`covariates`	included for compatibility, not used in this model
`fixed_parameters`	included for compatibility, not used in this model

Value

log-likelihood value

`BH_pm_alpha_pairwise_lambdacov_global_alphacov_global` *Beverton-Holt model with pairwise alphas and global covariate effects on lambda and alpha*

Description

Beverton-Holt model with pairwise alphas and global covariate effects on lambda and alpha

Usage

```
BH_pm_alpha_pairwise_lambdacov_global_alphacov_global(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: lambda, lambda_cov, alpha, alpha_cov, and sigma
`fitness`	1d vector of fitness observations, in log scale
`neigh_intra_matrix`	optional matrix of one column, number of intraspecific neighbours for each observation
`neigh_inter_matrix`	matrix of arbitrary columns, number of interspecific neighbours for each observation
`covariates`	optional matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation
`fixed_parameters`	optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

`BH_pm_alpha_pairwise_lambdacov_global_alphacov_pairwise` *Beverton-Holt model with pairwise alphas, covariate effects on lambda, and pairwise covariate effects on alpha*

Description

Beverton-Holt model with pairwise alphas, covariate effects on lambda, and pairwise covariate effects on alpha

Usage

```
BH_pm_alpha_pairwise_lambdacov_global_alphacov_pairwise(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: lambda, lambda_cov, alpha, alpha_cov, and sigma
`fitness`	1d vector of fitness observations, in log scale
`neigh_intra_matrix`	optional matrix of one column, number of intraspecific neighbours for each observation
`neigh_inter_matrix`	matrix of arbitrary columns, number of interspecific neighbours for each observation
`covariates`	optional matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation
`fixed_parameters`	optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

`BH_pm_alpha_pairwise_lambdacov_none_alphacov_none` *Beverton-Holt model with pairwise alphas and no covariate effects*

Description

Beverton-Holt model with pairwise alphas and no covariate effects

Usage

```
BH_pm_alpha_pairwise_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: 'lambda', 'alpha_intra' (optional), 'alpha_inter', and 'sigma'
`fitness`	1d vector of fitness observations, in log scale
`neigh_intra_matrix`	optional matrix of one column, number of intraspecific neighbours for each observation
`neigh_inter_matrix`	matrix of arbitrary columns, number of interspecific neighbours for each observation
`covariates`	included for compatibility, not used in this model
`fixed_parameters`	optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

BH_project_alpha_global_lambdacov_none_alphacov_none

Beverton-Holt model for projecting abundances, with a global alpha and no covariate effects

Description

Beverton-Holt model for projecting abundances, with a global alpha and no covariate effects

Usage

```
BH_project_alpha_global_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	single numeric value.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

BH_project_alpha_none_lambdacov_none_alphacov_none

Beverton-Holt model for projecting abundances, with no alpha and no covariate effects

Description

Beverton-Holt model for projecting abundances, with no alpha and no covariate effects

Usage

```
BH_project_alpha_none_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	included for compatibility, not used in this model.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

BH_project_alpha_pairwise_lambdacov_global_alphacov_global

Beverton-Holt model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

---

Description

Beverton-Holt model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

Usage

```
BH_project_alpha_pairwise_lambdacov_global_alphacov_global(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	single numeric value.
alpha_inter	numeric vector with interspecific alpha values.
lambda_cov	numeric vector with effects of covariates over lambda.
alpha_cov	named list of numeric values with effects of each covariate over alpha.
abundance	named numeric vector of abundances in the previous timestep.
covariates	matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation.

Value

numeric abundance projected one timestep

BH_project_alpha_pairwise_lambdacov_global_alphacov_pairwise

Beverton-Holt model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

---

Description

Beverton-Holt model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

Usage

```
BH_project_alpha_pairwise_lambdacov_global_alphacov_pairwise(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	named numeric lambda value.
alpha_intra	single numeric value.
alpha_inter	numeric vector with interspecific alpha values.
lambda_cov	numeric vector with effects of covariates over lambda.
alpha_cov	named list of named numeric vectors with effects of each covariate over alpha values.

abundance	named numeric vector of abundances in the previous timestep.
covariates	matrix with observations in rows and covariates in named columns. Each cell is the value of a covariate in a given observation.

Value

numeric abundance projected one timestep

`BH_project_alpha_pairwise_lambdacov_none_alphacov_none`
Beverton-Holt model for projecting abundances, with specific alpha values and no covariate effects

---

Description

Beverton-Holt model for projecting abundances, with specific alpha values and no covariate effects

Usage

```
BH_project_alpha_pairwise_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	single numeric value.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

`competitive_ability` *Competitive ability among pairs of species*

Description

Computes the competitive ability among two species, as defined by Hart et al. (2018). This metric, as others in MCT, is model-specific; the formulation for a series of Lotka-Volterra-like models is given in table A1 of Hart et al. (2018). We include in `cxr` by default the formulation for Beverton-Holt, Ricker, Law-Watkinson, and Lotka-Volterra families.

Usage

```
competitive_ability(
  cxr_multifit = NULL,
  cxr_sp1 = NULL,
  cxr_sp2 = NULL,
  lambda = NULL,
  pair_matrix = NULL,
  model_family = NULL
)
```

Arguments

`cxr_multifit`	cxr_pm_multifit object, with parameters for a series of species.
`cxr_sp1`	cxr_pm_fit object giving the parameters from the first species.
`cxr_sp2`	cxr_pm_fit object giving the parameters from the second species.
`lambda`	numeric lambda value of the focal species.
`pair_matrix`	2x2 matrix with intra and interspecific interaction coefficients between the focal and competitor species.
`model_family`	model family for which to calculate competitive ability.

Details

The function, as in `avg_fitness_diff` and `niche_overlap`, accepts three different parameterizations:

- A cxr_pm_multifit object, from which competitive ability of a focal species relative to a given competitor will be computed across all species pairs.
- two cxr_pm_fit objects, one for a focal species and one for a competitor.
- explicit lambda and alpha values, as well as the model family from which these parameters were obtained.

If the third parameterization is used, the function will try to find a model-specific function for obtaining the competitive ability, by looking at the 'model_family' parameter. If this specific function is not found, it will resort to the standard Lotka-Volterra formulation (lambda - 1 in the numerator term, Hart et al. 2018). Overall, we strongly suggest that you use the standard formulation ONLY if you are completely confident that the model from which you obtained your parameters is consistent with it. Otherwise, you should include your own formulation of competitive ability (see vignette 4).

Value

data frame with variable number of rows and three columns, specifying taxa identity and the competitive ability of focal species (sp1) relative to the competitor (sp2).

Examples

```
competitive_ability(lambda = runif(1,1,10),
                              pair_matrix = matrix(runif(4,0,1),nrow = 2),
                              model_family = "BH")
```

---

cxr *cxr*

Description

Tools and functions for evaluating multi-species coexistence.

---

cxr_er_bootstrap *standard error estimates for effect and response parameters*

Description

Computes bootstrap standard errors for a given effect/response function. This function is provided for completeness, but error calculation is integrated in the function `cxr_er_fit`.

Usage

```
cxr_er_bootstrap(
  fitness_model,
  optimization_method,
  data,
  covariates,
  init_par,
  lower_bounds,
  upper_bounds,
  fixed_parameters,
  bootstrap_samples
)
```

Arguments

`fitness_model`	effect/response function, see `cxr_er_fit`
`optimization_method`	numerical optimization method.
`data`	either a list of dataframes or a single dataframe. if 'data' is a list, each element is a dataframe with the following columns: • fitness: fitness metric for each observation • neighbours: named columns giving the number of neighbours of each column the names of the list elements are taken to be the names of the focal species. If 'data' is a dataframe, it also needs a 'focal' column. Regardless of the data structure, all focal species need to have the same number of observations (i.e. same number of rows), and the set of neighbour species needs to be the same as the set of focal species, so that the neighbours columns correspond to the names of the list elements or, if 'data' is a dataframe, to the values of the 'focal' column. Future versions will relax this requirement.
`covariates`	a data structure equivalent to 'data', in which each column are the values of a covariate.
`init_par`	initial values for parameters
`lower_bounds`	optional list with single values for "lambda", "effect","response", and optionally "lambda_cov", "effect_cov", "response_cov".
`upper_bounds`	optional list with single values for "lambda", "effect","response", and optionally "lambda_cov", "effect_cov", "response_cov".
`fixed_parameters`	list with values for fixed parameters, or NULL.
`bootstrap_samples`	number of bootstrap samples for error calculation. Defaults to 0, i.e. no error is calculated.

Value

1d vector, the standard error of each parameter in init_par

`cxr_er_fit` *General optimization for effect-response models*

Description

Estimates parameters of user-specified models of competitive effects and responses.

Usage

```
cxr_er_fit(
  data,
  model_family = c("BH"),
  covariates = NULL,
 optimization_method = c("Nelder-Mead", "BFGS", "CG", "ucminf", "L-BFGS-B", "nlm",
  "nlminb", "Rcgmin", "Rvmmin", "spg", "bobyqa", "nmkb", "hjkb", "nloptr_CRS2_LM",
    "nloptr_ISRES", "nloptr_DIRECT_L_RAND", "DEoptimR", "hydroPSO", "GenSA"),
  lambda_cov_form = c("none", "global"),
  effect_cov_form = c("none", "global"),
  response_cov_form = c("none", "global"),
  initial_values = list(lambda = 1, effect = 1, response = 1, lambda_cov = 0,
    effect_cov = 0, response_cov = 0),
  lower_bounds = NULL,
  upper_bounds = NULL,
  fixed_terms = NULL,
  bootstrap_samples = 0
)
```

Arguments

`data` either a list of dataframes or a single dataframe. if 'data' is a list, each element is a dataframe with the following columns:

- fitness: fitness metric for each observation
- neighbours: named columns giving the number of neighbours of each column the names of the list elements are taken to be the names of the focal species.

If 'data' is a dataframe, it also needs a 'focal' column. Regardless of the data structure, all focal species need to have the same number of observations (i.e. same number of rows), and the set of neighbour species needs to be the same as the set of focal species, so that the neighbours columns correspond to the names of the list elements or, if 'data' is a dataframe, to the values of the 'focal' column. Future versions will relax this requirement.

`model_family` family of model to use. Available families are BH (Beverton-Holt), LV (Lotka-Volterra), RK (Ricker), and LW (Law-Watkinson). Users may also define their own families and models (see vignette 4).

`covariates` a data structure equivalent to 'data', in which each column are the values of a covariate.

`optimization_method` numerical optimization method.

`lambda_cov_form` form of the covariate effects on lambda. Either "none" (no covariate effects) or "global" (one estimate per covariate).

`effect_cov_form` form of the covariate effects on competitive effects. Either "none" (no covariate effects) or "global" (one estimate per covariate)

`response_cov_form`
: form of the covariate effects on competitive responses. Either "none" (no covariate effects) or "global" (one estimate per covariate)

`initial_values`
: list with components "lambda","effect","response", and optionally "lambda_cov", "effect_cov", "response_cov", specifying the initial values for numerical optimization. Single values are allowed.

`lower_bounds`
: optional list with single values for "lambda", "effect","response", and optionally "lambda_cov", "effect_cov", "response_cov".

`upper_bounds`
: optional list with single values for "lambda", "effect","response", and optionally "lambda_cov", "effect_cov", "response_cov".

`fixed_terms`
: optional list specifying which model parameters are fixed.

`bootstrap_samples`
: number of bootstrap samples for error calculation. Defaults to 0, i.e. no error is calculated.

Value

an object of class 'cxr_er_fit' which is a list with the following components:

- model_name: string with the name of the fitness model
- model: model function
- data: data supplied
- taxa: names of the taxa fitted
- covariates: covariate data supplied
- optimization_method: optimization method used
- initial_values: list with initial values
- fixed_terms: list with fixed terms
- lambda: fitted values for lambdas, or NULL if fixed
- effect: fitted values for competitive effects, or NULL if fixed
- response: fitted values for competitive responses, or NULL if fixed
- lambda_cov: fitted values for effect of covariates on lambdas, or NULL if fixed
- effect_cov: fitted values for effect of covariates on competitive effects, or NULL if fixed
- response_cov: fitted values for effect of covariates on competitive responses, or NULL if fixed
- lambda_standard_error: standard errors for lambdas, if calculated
- effect_standard_error: standard errors for competitive effects, if calculated
- response_standard_error: standard errors for competitive responses, if calculated
- lambda_cov_standard_error: standard errors for effect of covariates on lambdas, if calculated
- effect_cov_standard_error: standard errors for effect of covariates on competitive effects, if calculated
- response_cov_standard_error: standard errors for effect of covariates on competitive responses, if calculated
- log_likelihood: log-likelihood of the fits

Examples

```
# fit three species at once
data("neigh_list")
# these species all have >250 observations
example_sp <- c("BEMA","LEMA","HOMA")
sp.pos <- which(names(neigh_list) %in% example_sp)
data <- neigh_list[sp.pos]
n.obs <- 250
# keep only fitness and neighbours columns
for(i in 1:length(data)){
  data[[i]] <- data[[i]][1:n.obs,c(2,sp.pos+2)]#2:length(data[[i]])]
}

# covariates: salinity
data("salinity_list")
salinity <- salinity_list[example_sp]
# keep only salinity column
for(i in 1:length(salinity)){
  salinity[[i]] <- salinity[[i]][1:n.obs,2:length(salinity[[i]])]
}

initial_values = list(lambda = 1,
                      effect = 1,
                      response = 1
                      # lambda_cov = 0,
                      # effect_cov = 0,
                      # response_cov = 0
)
lower_bounds = list(lambda = 0,
                    effect = 0,
                    response = 0
                    # lambda_cov = 0,
                    # effect_cov = 0,
                    # response_cov = 0
)
upper_bounds = list(lambda = 100,
                     effect = 10,
                     response = 10
                    # lambda_cov = 0,
                    # effect_cov = 0,
                    # response_cov = 0
)

er_3sp <- cxr_er_fit(data = data,
                     model_family = "BH",
                     # fit without covariates,
                     # as it may be very computationally expensive
                     # covariates = salinity,
                     optimization_method = "bobyqa",
                     lambda_cov_form = "none",
                     effect_cov_form = "none",
```

```
                      response_cov_form = "none",
                      initial_values = initial_values,
                      lower_bounds = lower_bounds,
                      upper_bounds = upper_bounds,
                      # syntaxis for fixed values
                      # fixed_terms = list("response"),
                      bootstrap_samples = 3)
# brief summary
summary(er_3sp)
```

`cxr_generate_test_data` *Generate simulated interaction data*

Description

Model fitness responses to neighbours and covariates using a Beverton-Holt functional form. This function is fairly restricted and under development, but can be used to generate simple test data to run the main functions of `cxr`.

Usage

```
cxr_generate_test_data(
  focal_sp = 1,
  neigh_sp = 1,
  covariates = 0,
  observations = 10,
  alpha_form = c("pairwise", "none", "global"),
  lambda_cov_form = c("none", "global"),
  alpha_cov_form = c("none", "global", "pairwise"),
  focal_lambda = NULL,
  min_lambda = 0,
  max_lambda = 10,
  alpha = NULL,
  min_alpha = 0,
  max_alpha = 1,
  alpha_cov = NULL,
  min_alpha_cov = -1,
  max_alpha_cov = 1,
  lambda_cov = NULL,
  min_lambda_cov = -1,
  max_lambda_cov = 1,
  min_cov = 0,
  max_cov = 1
)
```

Arguments

focal_sp	number of focal species, defaults to 1.
neigh_sp	number of neighbour species, defaults to 1.
covariates	number of covariates, defaults to 0.
observations	number of observations, defaults to 10.
alpha_form	what form does the alpha parameter take? one of "none" (no alpha in the model), "global" (a single alpha for all pairwise interactions), or "pairwise" (one alpha value for every interaction).
lambda_cov_form	form of the covariate effects on lambda. Either "none" (no covariate effects) or "global" (one estimate per covariate).
alpha_cov_form	form of the covariate effects on alpha. One of "none" (no covariate effects), "global" (one estimate per covariate on every alpha), or "pairwise" (one estimate per covariate and pairwise alpha).
focal_lambda	optional 1d vector with lambdas of the focal sp.
min_lambda	if no focal_lambda is provided, lambdas are taken from a uniform distribution with min_lambda and max_lambda as minimum and maximum values.
max_lambda	if no focal_lambda is provided, lambdas are taken from a uniform distribution with min_lambda and max_lambda as minimum and maximum values.
alpha	optional interaction matrix, neigh_sp x neigh_sp
min_alpha	if no focal_alpha is provided, alphas are taken from a uniform distribution with min_alpha and max_alpha as minimum and maximum values.
max_alpha	if no focal_alpha is provided, alphas are taken from a uniform distribution with min_alpha and max_alpha as minimum and maximum values.
alpha_cov	————-Under development————-
min_alpha_cov	if no focal_alpha_cov is provided, alpha_covs are taken from a uniform distribution with min_alpha_cov and max_alpha_cov as minimum and maximum values.
max_alpha_cov	if no focal_alpha_cov is provided, alpha_covs are taken from a uniform distribution with min_alpha and max_alpha as minimum and maximum values.
lambda_cov	optional matrix of neigh_sp x covariates giving the effect of each covariate over the fecundity (lambda) of each species.
min_lambda_cov	if no focal_lambda_cov is provided, lambda_covs are taken from a uniform distribution with min_lambda_cov and max_lambda_cov as minimum and maximum values.
max_lambda_cov	if no focal_lambda_cov is provided, lambda_covs are taken from a uniform distribution with min_lambda and max_lambda as minimum and maximum values.
min_cov	minimum value for covariates
max_cov	maximum value for covariates

Value

list with two components: 'observations' is a list with as many components as focal species. Each component of 'observations' is a dataframe with stochastic number of neighbours and associated fitness. The second component, 'covariates', is again a list with one component per focal species. Each component of 'covariates' is a dataframe with the values of each covariate for each associated observation.

Examples

```
example_obs <- cxr_generate_test_data(focal_sp = 2,
                                      neigh_sp = 2,
                                      alpha_form = "pairwise",
                                      lambda_cov_form = "global",
                                      alpha_cov_form = "global",
                                      covariates = 1)
```

---

cxr_pm_bootstrap *Standard error estimates for model parameters*

Description

Computes bootstrap standard errors for a given population dynamics model. This function is provided for completeness, but error calculation is integrated in the function `cxr_pm_fit`.

Usage

```
cxr_pm_bootstrap(
  fitness_model,
  optimization_method,
  data,
  focal_column,
  covariates,
  init_par,
  lower_bounds,
  upper_bounds,
  fixed_parameters,
  bootstrap_samples
)
```

Arguments

`fitness_model`	function returning a single value to minimize, given a set of parameters and a fitness metric
`optimization_method`	numerical optimization method
`data`	dataframe with observations in rows and two sets of columns:

- fitness: fitness metric for the focal individual
- neighbours: columns with user-defined names with number of neighbours for each group

`focal_column` optional integer value giving the position, or name, of the column with neighbours from the same species as the focal one. This is necessary if "alpha_intra" is specified.

`covariates` optional matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation.

`init_par` 1d vector of initial parameters

`lower_bounds` 1d vector of lower bounds

`upper_bounds` 1d vector of upper bounds

`fixed_parameters` optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter" and "alpha_cov".

`bootstrap_samples` how many bootstrap samples to compute.

Value

1d vector, the standard error of each parameter in init_par

`cxr_pm_fit` *General optimization for population models*

Description

Estimates parameters of user-specified population dynamics models.

Usage

```
cxr_pm_fit(
  data,
  focal_column = NULL,
  model_family,
  covariates = NULL,
 optimization_method = c("Nelder-Mead", "BFGS", "CG", "ucminf", "L-BFGS-B", "nlm",
  "nlminb", "Rcgmin", "Rvmmin", "spg", "bobyqa", "nmkb", "hjkb", "nloptr_CRS2_LM",
    "nloptr_ISRES", "nloptr_DIRECT_L_RAND", "DEoptimR", "hydroPSO", "GenSA"),
  alpha_form = c("none", "global", "pairwise"),
  lambda_cov_form = c("none", "global"),
  alpha_cov_form = c("none", "global", "pairwise"),
 initial_values = list(lambda = 0, alpha_intra = 0, alpha_inter = 0, lambda_cov = 0,
    alpha_cov = 0),
  lower_bounds = NULL,
  upper_bounds = NULL,
```

```
  fixed_terms = NULL,
  bootstrap_samples = 0
)
```

Arguments

data	dataframe with observations in rows and two sets of columns: • fitness: fitness metric for the focal individual • neighbours: numeric columns with user-defined names, giving number of neighbours for each group
focal_column	optional integer or character giving the column with neighbours from the same species as the focal one. This field is necessary if "alpha_intra" is specified in initial_values, lower_bounds, upper_bounds, or fixed_terms.
model_family	family of model to use. Available families are BH (Beverton-Holt), LV (Lotka-Volterra), RK (Ricker), and LW (Law-Watkinson). Users may also define their own families and models (see vignette 4).
covariates	optional named matrix or dataframe with observations (rows) of any number of environmental covariates (columns).
optimization_method	numerical optimization method.
alpha_form	what form does the alpha parameter take? one of "none" (no alpha in the model), "global" (a single alpha for all pairwise interactions), or "pairwise" (one alpha value for every interaction).
lambda_cov_form	form of the covariate effects on lambda. Either "none" (no covariate effects) or "global" (one estimate per covariate).
alpha_cov_form	form of the covariate effects on alpha. One of "none" (no covariate effects), "global" (one estimate per covariate on every alpha), or "pairwise" (one estimate per covariate and pairwise alpha)
initial_values	list with components "lambda","alpha_intra","alpha_inter","lambda_cov", "alpha_cov", specifying the initial values for numerical optimization. Single values are allowed.
lower_bounds	optional list with single values for "lambda","alpha_intra","alpha_inter","lambda_cov", "alpha_cov".
upper_bounds	optional list with single values for "lambda","alpha_intra","alpha_inter","lambda_cov", "alpha_cov".
fixed_terms	optional list of numeric vectors specifying the value of fixed model parameters, among "lambda","alpha_intra","alpha_inter","lambda_cov", and "alpha_cov".
bootstrap_samples	number of bootstrap samples for error calculation. Defaults to 0, i.e. no error is calculated.

Value

an object of class 'cxr_pm_fit' which is a list with the following components:

- model_name: string with the name of the fitness model
- model: model function
- data: data supplied
- focal_ID: name/ID of the focal taxa, if provided in 'focal_column'
- covariates: covariate data supplied
- optimization_method: optimization method used
- initial_values: list with initial values
- fixed_terms: list with fixed terms
- lambda: fitted value for lambda, or NULL if fixed
- alpha_intra: fitted value for intraspecific alpha, or NULL if fixed
- alpha_inter: fitted value for interspecific alpha, or NULL if fixed
- lambda_cov: fitted value(s) for lambda_cov, or NULL if fixed.
- alpha_cov: fitted value(s) for alpha_cov, or NULL if fixed. These are structured as a list with one element for each covariate.
- lambda_standard_error: standard error for lambda, if computed
- alpha_intra_standard_error: standard error for intraspecific alpha, if computed
- alpha_inter_standard_error: standard error for interspecific alpha, if computed
- lambda_cov_standard_error: standard error for lambda_cov, if computed
- alpha_cov_standard_error: standard error for alpha_cov, if computed
- log_likelihood: log-likelihood of the fit

Examples

```
data("neigh_list")
my.sp <- "BEMA"
# data for a single species, keep only fitness and neighbours columns
sp_data <- neigh_list[[my.sp]][2:ncol(neigh_list[[1]])]

  sp_fit <- cxr_pm_fit(data = sp_data,
                       focal_column = my.sp,
                       optimization_method = "bobyqa",
                       model_family = "BH",
                       alpha_form = "pairwise",
                       lambda_cov_form = "none",
                       alpha_cov_form = "none",
                     initial_values = list(lambda = 1,alpha_intra = 0.1,alpha_inter = 0.1),
                       lower_bounds = list(lambda = 0,alpha_intra = 0,alpha_inter = 0),
                      upper_bounds = list(lambda = 100,alpha_intra = 1,alpha_inter = 1),
                       bootstrap_samples = 3)
  summary(sp_fit)
```

cxr_pm_multifit *Multi-species parameter optimization*

Description

This function is a wrapper for estimating parameters for several focal species, instead of making separate calls to `cxr_pm_fit`.

Usage

```
cxr_pm_multifit(
  data,
  model_family = c("BH"),
  focal_column = NULL,
  covariates = NULL,
 optimization_method = c("BFGS", "CG", "Nelder-Mead", "ucminf", "L-BFGS-B", "nlm",
  "nlminb", "Rcgmin", "Rvmmin", "spg", "bobyqa", "nmkb", "hjkb", "nloptr_CRS2_LM",
    "nloptr_ISRES", "nloptr_DIRECT_L_RAND", "DEoptimR", "hydroPSO", "GenSA"),
  alpha_form = c("none", "global", "pairwise"),
  lambda_cov_form = c("none", "global"),
  alpha_cov_form = c("none", "global", "pairwise"),
  initial_values = NULL,
  lower_bounds = NULL,
  upper_bounds = NULL,
  fixed_terms = NULL,
  bootstrap_samples = 0
)
```

Arguments

`data`	named list in which each component is a dataframe with a fitness column and a number of columns representing neighbours
`model_family`	family of model to use. Available families are BH (Beverton-Holt), LV (Lotka-Volterra), RK (Ricker), and LW (Law-Watkinson). Users may also define their own families and models (see vignette 4).
`focal_column`	character vector with the same length as data, giving the names of the columns representing intraspecific observations for each species, or numeric vector giving the position of such columns.
`covariates`	optional named list in which each component is a dataframe with values of each covariate for each observation. The ith component of `covariates` are the covariate values that correspond to the ith component of `data`, so they must have the same number of observations.
`optimization_method`	numerical optimization method.

`alpha_form` what form does the alpha parameter take? one of "none" (no alpha in the model), "global" (a single alpha for all pairwise interactions), or "pairwise" (one alpha value for every interaction).

`lambda_cov_form` form of the covariate effects on lambda. Either "none" (no covariate effects) or "global" (one estimate per covariate).

`alpha_cov_form` form of the covariate effects on alpha. One of "none" (no covariate effects), "global" (one estimate per covariate on every alpha), or "pairwise" (one estimate per covariate and pairwise alpha)

`initial_values` list with components "lambda","alpha_intra","alpha_inter","lambda_cov", "alpha_cov", specifying the initial values for numerical optimization. Single values are allowed.

`lower_bounds` optional list with single values for "lambda","alpha_intra","alpha_inter","lambda_cov", "alpha_cov".

`upper_bounds` optional list with single values for "lambda","alpha_intra","alpha_inter","lambda_cov", "alpha_cov".

`fixed_terms` optional named list in which each component is itself a list containing fixed terms for each focal species.

`bootstrap_samples` number of bootstrap samples for error calculation. Defaults to 0, i.e. no error is calculated.

Value

an object of class 'cxr_pm_multifit' which is a list with the following components:

- model_name: string with the name of the fitness model
- model: model function
- data: data supplied
- taxa: names of the taxa fitted
- covariates: covariate data supplied
- optimization_method: optimization method used
- initial_values: list with initial values
- fixed_terms: list with fixed terms
- lambda: fitted values for lambda, or NULL if fixed
- alpha_intra: fitted values for alpha_intra, or NULL if fixed
- alpha_inter: fitted values for alpha_inter, or NULL if fixed
- lambda_cov: fitted values for lambda_cov, or NULL if fixed
- alpha_cov: fitted values for alpha_cov, or NULL if fixed
- lambda_standard_error: standard errors for lambda, if computed
- alpha_standard_error: standard errors for alpha, if computed
- lambda_cov_standard_error: standard errors for lambda_cov, if computed
- alpha_cov_standard_error: standard errors for alpha_cov, if computed
- log_likelihood: log-likelihoods of the fits

Examples

```
# fit three species at once
data("neigh_list")
data <- neigh_list[1:3]
# keep only fitness and neighbours columns
for(i in 1:length(data)){
  data[[i]] <- data[[i]][,2:length(data[[i]])]
}
# covariates: salinity
data("salinity_list")
salinity <- salinity_list[1:3]
# keep only salinity column
for(i in 1:length(salinity)){
  salinity[[i]] <- salinity[[i]][,2:length(salinity[[i]])]
}

  fit_3sp <- cxr_pm_multifit(data = data,
                             optimization_method = "bobyqa",
                             model_family = "BH",
                             covariates = salinity,
                             alpha_form = "pairwise",
                             lambda_cov_form = "global",
                             alpha_cov_form = "global",
                             initial_values = list(lambda = 1,
                                                   alpha_intra = 0.1,
                                                   alpha_inter = 0.1,
                                                   lambda_cov = 0.1,
                                                   alpha_cov = 0.1),
                             lower_bounds = list(lambda = 0.01,
                                                 alpha_intra = 0,
                                                 alpha_inter = 0,
                                                 lambda_cov = 0,
                                                 alpha_cov = 0),
                             upper_bounds = list(lambda = 100,
                                                 alpha_intra = 1,
                                                 alpha_inter = 1,
                                                 lambda_cov = 1,
                                                 alpha_cov = 1),
                             bootstrap_samples = 3)
  # brief summary
  summary(fit_3sp)
  # interaction matrix
  fit_3sp$alpha
```

`fitness_ratio` *Fitness ratio among two or more species*

Description

Fitness ratio among two or more species

Usage

```
fitness_ratio(
  effect_response_fit = NULL,
  fitness_sp1 = NULL,
  fitness_sp2 = NULL
)
```

Arguments

effect_response_fit	cxr_er_fit object
fitness_sp1	numeric value representing the fitness (a.k.a. competitive ability) of the first taxa
fitness_sp2	numeric value representing the fitness (a.k.a. competitive ability) of the second taxa

Value

either a matrix with fitness ratios for all pairs of fitted species, or a single numeric value. The matrix elements represent the ratios of species in columns over species in rows, and conversely, the numeric value represents the ratio of sp1 over sp2.

Examples

```
fitness_ratio(fitness_sp1 = 0.6, fitness_sp2 = 0.3)
```

LV_er_lambdacov_global_effectcov_global_responsecov_global

Effect response Lotka-Volterra model with covariate effects on lambda, effect, and response

---

Description

Note that, as e and r are not pair-specific, all species parameters are fit in the same function.

Usage

```
LV_er_lambdacov_global_effectcov_global_responsecov_global(
  par,
  fitness,
  target,
  density,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector with initial parameters in the order: lambda,lambda_cov,effect,effect_cov,response,response_co
fitness	1d vector with fitness observations
target	matrix with species in rows, observations in columns. Value is 1 if a species is focal for a given observation, 0 otherwise.
density	matrix with species in rows, observations in columns. Value is density of each sp as neighbour for each observation.
covariates	numeric dataframe or matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","lambda_cov","effect","eff "response","response_cov".

Value

log-likelihood value

LV_er_lambdacov_none_effectcov_none_responsecov_none

Effect response Lotka-Volterra model without covariate effects

---

Description

Note that, as e and r are not pair-specific, all species parameters are fit in the same function.

Usage

```
LV_er_lambdacov_none_effectcov_none_responsecov_none(
  par,
  fitness,
  target,
  density,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector with initial parameters in the order: lambda,effect,response,sigma.
fitness	1d vector with fitness observations.
target	matrix with species in rows, observations in columns. Value is 1 if a species is focal for a given observation, 0 otherwise.
density	matrix with species in rows, observations in columns. Value is density of each sp as neighbour for each observation.
covariates	included for compatibility, not used in this model.
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","effect","response".

Value

log-likelihood value

`LV_pm_alpha_global_lambdacov_none_alphacov_none` *Lotka-Volterra model with a global alpha and no covariate effects*

Description

Lotka-Volterra model with a global alpha and no covariate effects

Usage

```
LV_pm_alpha_global_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: lambda, alpha, and sigma.
`fitness`	1d vector of fitness observations, in log scale.
`neigh_intra_matrix`	included for compatibility, not used in this model.
`neigh_inter_matrix`	matrix of arbitrary columns, number of neighbours for each observation. As in this model there is a single alpha argument, do not distinguish neighbour identity
`covariates`	included for compatibility, not used in this model.
`fixed_parameters`	optional list specifying values of fixed parameters, with components "lambda","alpha_inter".

Value

log-likelihood value

LV_pm_alpha_none_lambdacov_none_alphacov_none

Lotka-Volterra model with no alphas and no covariate effects

Description

This model, in all families, is simply given by lambda.

Usage

```
LV_pm_alpha_none_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: lambda and sigma
`fitness`	1d vector of fitness observations, in log scale
`neigh_intra_matrix`	included for compatibility, not used in this model.
`neigh_inter_matrix`	included for compatibility, not used in this model.
`covariates`	included for compatibility, not used in this model
`fixed_parameters`	included for compatibility, not used in this model

Value

log-likelihood value

LV_pm_alpha_pairwise_lambdacov_global_alphacov_global

Lotka-Volterra model with pairwise alphas and global covariate effects on lambda and alpha

Description

Lotka-Volterra model with pairwise alphas and global covariate effects on lambda and alpha

Usage

```
LV_pm_alpha_pairwise_lambdacov_global_alphacov_global(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector of initial parameters: lambda, lambda_cov, alpha, alpha_cov, and sigma
fitness	1d vector of fitness observations, in log scale
neigh_intra_matrix	optional matrix of one column, number of intraspecific neighbours for each observation
neigh_inter_matrix	matrix of arbitrary columns, number of interspecific neighbours for each observation
covariates	optional matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

LV_pm_alpha_pairwise_lambdacov_global_alphacov_pairwise

Lotka-Volterra model with pairwise alphas, covariate effects on lambda, and pairwise covariate effects on alpha

---

Description

Lotka-Volterra model with pairwise alphas, covariate effects on lambda, and pairwise covariate effects on alpha

Usage

```
LV_pm_alpha_pairwise_lambdacov_global_alphacov_pairwise(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector of initial parameters: lambda, lambda_cov, alpha, alpha_cov, and sigma
fitness	1d vector of fitness observations, in log scale
neigh_intra_matrix	optional matrix of one column, number of intraspecific neighbours for each observation
neigh_inter_matrix	matrix of arbitrary columns, number of interspecific neighbours for each observation
covariates	optional matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

LV_pm_alpha_pairwise_lambdacov_none_alphacov_none

Lotka-Volterra model with pairwise alphas and no covariate effects

---

Description

Lotka-Volterra model with pairwise alphas and no covariate effects

Usage

```
LV_pm_alpha_pairwise_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector of initial parameters: 'lambda', 'alpha_intra' (optional), 'alpha_inter', and 'sigma'
fitness	1d vector of fitness observations, in log scale
neigh_intra_matrix	optional matrix of one column, number of intraspecific neighbours for each observation
neigh_inter_matrix	matrix of arbitrary columns, number of interspecific neighbours for each observation
covariates	included for compatibility, not used in this model
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

LV_project_alpha_global_lambdacov_none_alphacov_none

Lotka-Volterra model for projecting abundances, with a global alpha and no covariate effects

---

Description

Lotka-Volterra model for projecting abundances, with a global alpha and no covariate effects

Usage

```
LV_project_alpha_global_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	single numeric value.
lambda_cov	included for compatibility, not used in this model.

alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

`LV_project_alpha_none_lambdacov_none_alphacov_none` *Model for projecting abundances, with no alpha and no covariate effects*

Description

Model for projecting abundances, with no alpha and no covariate effects

Usage

```
LV_project_alpha_none_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	included for compatibility, not used in this model.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

`LV_project_alpha_pairwise_lambdacov_global_alphacov_global` *Lotka-Volterra model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda*

Description

Lotka-Volterra model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

Usage

```
LV_project_alpha_pairwise_lambdacov_global_alphacov_global(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

`lambda`	numeric lambda value.
`alpha_intra`	single numeric value.
`alpha_inter`	numeric vector with interspecific alpha values.
`lambda_cov`	numeric vector with effects of covariates over lambda.
`alpha_cov`	named list of numeric values with effects of each covariate over alpha.
`abundance`	named numeric vector of abundances in the previous timestep.
`covariates`	matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation.

Value

numeric abundance projected one timestep

`LV_project_alpha_pairwise_lambdacov_global_alphacov_pairwise` *Lotka-Volterra model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda*

Description

Lotka-Volterra model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

Usage

```
LV_project_alpha_pairwise_lambdacov_global_alphacov_pairwise(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

`lambda`	named numeric lambda value.
`alpha_intra`	single numeric value.
`alpha_inter`	numeric vector with interspecific alpha values.
`lambda_cov`	numeric vector with effects of covariates over lambda.
`alpha_cov`	named list of named numeric vectors with effects of each covariate over alpha values.
`abundance`	named numeric vector of abundances in the previous timestep.
`covariates`	matrix with observations in rows and covariates in named columns. Each cell is the value of a covariate in a given observation.

Value

numeric abundance projected one timestep

LV_project_alpha_pairwise_lambdacov_none_alphacov_none

Lotka-Volterra model for projecting abundances, with specific alpha values and no covariate effects

Description

Lotka-Volterra model for projecting abundances, with specific alpha values and no covariate effects

Usage

```
LV_project_alpha_pairwise_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	single numeric value.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

LW_er_lambdacov_global_effectcov_global_responsecov_global

Effect response Law-Watkinson model with covariate effects on lambda, effect, and response

Description

Note that, as e and r are not pair-specific, all species parameters are fit in the same function.

Usage

```
LW_er_lambdacov_global_effectcov_global_responsecov_global(
  par,
  fitness,
  target,
  density,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector with initial parameters in the order: lambda,lambda_cov,effect,effect_cov,response,response_co
fitness	1d vector with fitness observations
target	matrix with species in rows, observations in columns. Value is 1 if a species is focal for a given observation, 0 otherwise.
density	matrix with species in rows, observations in columns. Value is density of each sp as neighbour for each observation.
covariates	numeric dataframe or matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","lambda_cov","effect","eff "response","response_cov".

Value

log-likelihood value

LW_er_lambdacov_none_effectcov_none_responsecov_none

Effect response Law-Watkinson model without covariate effects

---

Description

Note that, as e and r are not pair-specific, all species parameters are fit in the same function.

Usage

```
LW_er_lambdacov_none_effectcov_none_responsecov_none(
  par,
  fitness,
  target,
  density,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector with initial parameters in the order: lambda,effect,response,sigma.
fitness	1d vector with fitness observations.
target	matrix with species in rows, observations in columns. Value is 1 if a species is focal for a given observation, 0 otherwise.
density	matrix with species in rows, observations in columns. Value is density of each sp as neighbour for each observation.
covariates	included for compatibility, not used in this model.
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","effect","response".

Value

log-likelihood value

LW_pm_alpha_global_lambdacov_none_alphacov_none

Law-Watkinson model with a global alpha and no covariate effects

---

Description

Law-Watkinson model with a global alpha and no covariate effects

Usage

```
LW_pm_alpha_global_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector of initial parameters: lambda, alpha, and sigma.
fitness	1d vector of fitness observations, in log scale.
neigh_intra_matrix	included for compatibility, not used in this model.
neigh_inter_matrix	matrix of arbitrary columns, number of neighbours for each observation. As in this model there is a single alpha argument, do not distinguish neighbour identity
covariates	included for compatibility, not used in this model.
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","alpha_inter".

Value

log-likelihood value

`LW_pm_alpha_none_lambdacov_none_alphacov_none` *Law-Watkinson model with no alphas and no covariate effects*

Description

This model, in all families, is simply given by lambda.

Usage

```
LW_pm_alpha_none_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: lambda and sigma
`fitness`	1d vector of fitness observations, in log scale
`neigh_intra_matrix`	included for compatibility, not used in this model.
`neigh_inter_matrix`	included for compatibility, not used in this model.
`covariates`	included for compatibility, not used in this model
`fixed_parameters`	included for compatibility, not used in this model

Value

log-likelihood value

`LW_pm_alpha_pairwise_lambdacov_global_alphacov_global` *Law-Watkinson model with pairwise alphas and global covariate effects on lambda and alpha*

Description

Law-Watkinson model with pairwise alphas and global covariate effects on lambda and alpha

Usage

```
LW_pm_alpha_pairwise_lambdacov_global_alphacov_global(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: lambda, lambda_cov, alpha, alpha_cov, and sigma
`fitness`	1d vector of fitness observations, in log scale
`neigh_intra_matrix`	optional matrix of one column, number of intraspecific neighbours for each observation
`neigh_inter_matrix`	matrix of arbitrary columns, number of interspecific neighbours for each observation
`covariates`	optional matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation
`fixed_parameters`	optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

`LW_pm_alpha_pairwise_lambdacov_global_alphacov_pairwise` *Law-Watkinson model with pairwise alphas, covariate effects on lambda, and pairwise covariate effects on alpha*

Description

Law-Watkinson model with pairwise alphas, covariate effects on lambda, and pairwise covariate effects on alpha

Usage

```
LW_pm_alpha_pairwise_lambdacov_global_alphacov_pairwise(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: lambda, lambda_cov, alpha, alpha_cov, and sigma
`fitness`	1d vector of fitness observations, in log scale
`neigh_intra_matrix`	optional matrix of one column, number of intraspecific neighbours for each observation
`neigh_inter_matrix`	matrix of arbitrary columns, number of interspecific neighbours for each observation
`covariates`	optional matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation
`fixed_parameters`	optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

`LW_pm_alpha_pairwise_lambdacov_none_alphacov_none` *Law-Watkinson model with pairwise alphas and no covariate effects*

Description

Law-Watkinson model with pairwise alphas and no covariate effects

Usage

```
LW_pm_alpha_pairwise_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: 'lambda', 'alpha_intra' (optional), 'alpha_inter', and 'sigma'
`fitness`	1d vector of fitness observations, in log scale
`neigh_intra_matrix`	optional matrix of one column, number of intraspecific neighbours for each observation
`neigh_inter_matrix`	matrix of arbitrary columns, number of interspecific neighbours for each observation
`covariates`	included for compatibility, not used in this model
`fixed_parameters`	optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

LW_project_alpha_global_lambdacov_none_alphacov_none

Law-Watkinson model for projecting abundances, with a global alpha and no covariate effects

Description

Law-Watkinson model for projecting abundances, with a global alpha and no covariate effects

Usage

```
LW_project_alpha_global_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	single numeric value.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

LW_project_alpha_none_lambdacov_none_alphacov_none

Model for projecting abundances, with no alpha and no covariate effects

Description

Model for projecting abundances, with no alpha and no covariate effects

Usage

```
LW_project_alpha_none_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	included for compatibility, not used in this model.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

LW_project_alpha_pairwise_lambdacov_global_alphacov_global

Law-Watkinson model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

---

Description

Law-Watkinson model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

Usage

```
LW_project_alpha_pairwise_lambdacov_global_alphacov_global(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	single numeric value.
alpha_inter	numeric vector with interspecific alpha values.
lambda_cov	numeric vector with effects of covariates over lambda.
alpha_cov	named list of numeric values with effects of each covariate over alpha.
abundance	named numeric vector of abundances in the previous timestep.
covariates	matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation.

Value

numeric abundance projected one timestep

LW_project_alpha_pairwise_lambdacov_global_alphacov_pairwise

Law-Watkinson model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

---

Description

Law-Watkinson model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

Usage

```
LW_project_alpha_pairwise_lambdacov_global_alphacov_pairwise(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	named numeric lambda value.
alpha_intra	single numeric value.
alpha_inter	numeric vector with interspecific alpha values.
lambda_cov	numeric vector with effects of covariates over lambda.
alpha_cov	named list of named numeric vectors with effects of each covariate over alpha values.

abundance	named numeric vector of abundances in the previous timestep.
covariates	matrix with observations in rows and covariates in named columns. Each cell is the value of a covariate in a given observation.

Value

numeric abundance projected one timestep

`LW_project_alpha_pairwise_lambdacov_none_alphacov_none`
Law-Watkinson model for projecting abundances, with specific alpha values and no covariate effects

---

Description

Law-Watkinson model for projecting abundances, with specific alpha values and no covariate effects

Usage

```
LW_project_alpha_pairwise_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	single numeric value.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

neigh_list *neighbours and fitness observations*

Description

A dataset containing fitness and neighbours for plant individuals of 17 species. The dataset is a named list with 16 elements, each of which is a dataframe with the following columns:

- obs_ID: unique identifier for each observation
- fitness: number of viable seeds of the focal individual
- 17 columns indicating the number of neighbours from each plant sp. in a radius of 7.5 cm from the focal individual

Usage

```
data(neigh_list)
```

Format

A list with 17 elements, each of which a dataframe of variable number of rows and 18 columns

Note

For details, see Lanuza et al. 2018 Ecology Letters.

niche_overlap *Niche overlap between two species*

Description

quoting Godoy et al. (2014): reflects the average degree to which species limit individuals of their own species relative to competitors. Low niche overlap causes species to have greater per capita growth rates when rare than when common. If species limit individuals of their own species and their competitors equally, then niche overlap is 1, and coexistence is not possible unless species are otherwise identical. At the other extreme, if species have no interspecific effects, then niche overlap is 0.

Usage

```
niche_overlap(
  cxr_multifit = NULL,
  cxr_sp1 = NULL,
  cxr_sp2 = NULL,
  pair_matrix = NULL
)
```

Arguments

cxr_multifit	cxr_pm_multifit object, with parameters for a series of species.
cxr_sp1	cxr_pm_fit object giving the parameters from the first species.
cxr_sp2	cxr_pm_fit object giving the parameters from the second species.
pair_matrix	2x2 matrix with intra and interspecific interaction coefficients between the two species.

Details

Niche overlap has a common functional form, in the context of Modern Coexistence Theory (MCT), for a series of models, including those specified in table A1 of Hart et al. (2018) Journal of Ecology 106, 1902-1909. Other model families may not adhere to the general definition.

Furthermore, the MCT definition only accounts for competitive interactions (i.e. positive alpha coefficients in these models). An alternative definition is given in Saavedra et al. (2017) Ecological Monographs 87,470-486. In this 'structural approach', positive interactions are allowed. Incidentally, both approaches yield qualitatively similar, but not equivalent, results for purely competitive matrices.

In all cases, these definitions only apply to models whose feasible equilibrium point can be described by a linear equation (see Saavedra et al. 2017, Hart et al. 2018 for details).

This function calculates niche overlap among two or more taxa, using both the MCT and the structural formulation. The function, as in `avg_fitness_diff` and `competitive_ability`, accepts three different parameterizations:

- A cxr_pm_multifit object, from which niche overlap will be computed across all species pairs.
- two cxr_pm_fit objects, one for each species.
- explicit lambda and alpha values, as well as the model family from which these parameters were obtained.

If negative interactions are present, the MCT niche overlap will be NA. The cxr objects may be calculated with user-defined model families. If this is the case, or if simply a 2x2 matrix is provided, the niche overlap metrics will be calculated and a warning will be raised.

Value

either a dataframe with as many rows as species, or a single named numeric vector, containing niche overlap values for the MCT (modern coexistence theory) and SA (structural approach) formulations.

Examples

```
niche_overlap(pair_matrix = matrix(c(0.33,0.12,0.2,0.4),nrow = 2))
```

`RK_er_lambdacov_global_effectcov_global_responsecov_global` *Effect response Beverton-Holt model with covariate effects on lambda, effect, and response*

Description

Note that, as e and r are not pair-specific, all species parameters are fit in the same function.

Usage

```
RK_er_lambdacov_global_effectcov_global_responsecov_global(
  par,
  fitness,
  target,
  density,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector with initial parameters in the order: lambda,lambda_cov,effect,effect_cov,response,response_co
`fitness`	1d vector with fitness observations
`target`	matrix with species in rows, observations in columns. Value is 1 if a species is focal for a given observation, 0 otherwise.
`density`	matrix with species in rows, observations in columns. Value is density of each sp as neighbour for each observation.
`covariates`	numeric dataframe or matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation
`fixed_parameters`	optional list specifying values of fixed parameters, with components "lambda","lambda_cov","effect","eff "response","response_cov".

Value

log-likelihood value

RK_er_lambdacov_none_effectcov_none_responsecov_none

Effect response Ricker model without covariate effects

Description

Note that, as e and r are not pair-specific, all species parameters are fit in the same function.

Usage

```
RK_er_lambdacov_none_effectcov_none_responsecov_none(
  par,
  fitness,
  target,
  density,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector with initial parameters in the order: lambda,effect,response,sigma.
fitness	1d vector with fitness observations.
target	matrix with species in rows, observations in columns. Value is 1 if a species is focal for a given observation, 0 otherwise.
density	matrix with species in rows, observations in columns. Value is density of each sp as neighbour for each observation.
covariates	included for compatibility, not used in this model.
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","effect","response".

Value

log-likelihood value

RK_pm_alpha_global_lambdacov_none_alphacov_none

Ricker model with a global alpha and no covariate effects

Description

Ricker model with a global alpha and no covariate effects

Usage

```
RK_pm_alpha_global_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector of initial parameters: lambda, alpha, and sigma.
fitness	1d vector of fitness observations, in log scale.
neigh_intra_matrix	included for compatibility, not used in this model.
neigh_inter_matrix	matrix of arbitrary columns, number of neighbours for each observation. As in this model there is a single alpha argument, do not distinguish neighbour identity
covariates	included for compatibility, not used in this model.
fixed_parameters	optional list specifying values of fixed parameters, with components "lambda","alpha_inter".

Value

log-likelihood value

RK_pm_alpha_none_lambdacov_none_alphacov_none

Ricker model with no alphas and no covariate effects

Description

This model, in all families, is simply given by lambda.

Usage

```
RK_pm_alpha_none_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector of initial parameters: lambda and sigma
fitness	1d vector of fitness observations, in log scale
neigh_intra_matrix	included for compatibility, not used in this model.
neigh_inter_matrix	included for compatibility, not used in this model.
covariates	included for compatibility, not used in this model
fixed_parameters	included for compatibility, not used in this model

Value

log-likelihood value

RK_pm_alpha_pairwise_lambdacov_global_alphacov_global

Ricker model with pairwise alphas and global covariate effects on lambda and alpha

---

Description

Ricker model with pairwise alphas and global covariate effects on lambda and alpha

Usage

```
RK_pm_alpha_pairwise_lambdacov_global_alphacov_global(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

par	1d vector of initial parameters: lambda, lambda_cov, alpha, alpha_cov, and sigma
fitness	1d vector of fitness observations, in log scale
neigh_intra_matrix	optional matrix of one column, number of intraspecific neighbours for each observation

neigh_inter_matrix
: matrix of arbitrary columns, number of interspecific neighbours for each observation

covariates
: optional matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation

fixed_parameters
: optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

RK_pm_alpha_pairwise_lambdacov_global_alphacov_pairwise

Ricker model with pairwise alphas, covariate effects on lambda, and pairwise covariate effects on alpha

---

Description

Ricker model with pairwise alphas, covariate effects on lambda, and pairwise covariate effects on alpha

Usage

```
RK_pm_alpha_pairwise_lambdacov_global_alphacov_pairwise(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

par
: 1d vector of initial parameters: lambda, lambda_cov, alpha, alpha_cov, and sigma

fitness
: 1d vector of fitness observations, in log scale

neigh_intra_matrix
: optional matrix of one column, number of intraspecific neighbours for each observation

neigh_inter_matrix
: matrix of arbitrary columns, number of interspecific neighbours for each observation

covariates
: optional matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation

fixed_parameters
: optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

`RK_pm_alpha_pairwise_lambdacov_none_alphacov_none`
Ricker model with pairwise alphas and no covariate effects

Description

Ricker model with pairwise alphas and no covariate effects

Usage

```
RK_pm_alpha_pairwise_lambdacov_none_alphacov_none(
  par,
  fitness,
  neigh_intra_matrix = NULL,
  neigh_inter_matrix,
  covariates,
  fixed_parameters
)
```

Arguments

`par`	1d vector of initial parameters: 'lambda', 'alpha_intra' (optional), 'alpha_inter', and 'sigma'
`fitness`	1d vector of fitness observations, in log scale
`neigh_intra_matrix`	optional matrix of one column, number of intraspecific neighbours for each observation
`neigh_inter_matrix`	matrix of arbitrary columns, number of interspecific neighbours for each observation
`covariates`	included for compatibility, not used in this model
`fixed_parameters`	optional list specifying values of fixed parameters, with components "lambda","alpha_intra","alpha_inter"

Value

log-likelihood value

RK_project_alpha_global_lambdacov_none_alphacov_none

Ricker model for projecting abundances, with a global alpha and no covariate effects

Description

Ricker model for projecting abundances, with a global alpha and no covariate effects

Usage

```
RK_project_alpha_global_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	single numeric value.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

RK_project_alpha_none_lambdacov_none_alphacov_none

Model for projecting abundances, with no alpha and no covariate effects

Description

Model for projecting abundances, with no alpha and no covariate effects

Usage

```
RK_project_alpha_none_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	included for compatibility, not used in this model.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

RK_project_alpha_pairwise_lambdacov_global_alphacov_global

Ricker model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

---

Description

Ricker model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

Usage

```
RK_project_alpha_pairwise_lambdacov_global_alphacov_global(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	single numeric value.
alpha_inter	numeric vector with interspecific alpha values.
lambda_cov	numeric vector with effects of covariates over lambda.
alpha_cov	named list of numeric values with effects of each covariate over alpha.
abundance	named numeric vector of abundances in the previous timestep.
covariates	matrix with observations in rows and covariates in columns. Each cell is the value of a covariate in a given observation.

Value

numeric abundance projected one timestep

RK_project_alpha_pairwise_lambdacov_global_alphacov_pairwise

Ricker model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

---

Description

Ricker model for projecting abundances, with specific alpha values and global covariate effects on alpha and lambda

Usage

```
RK_project_alpha_pairwise_lambdacov_global_alphacov_pairwise(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	named numeric lambda value.
alpha_intra	single numeric value.
alpha_inter	numeric vector with interspecific alpha values.
lambda_cov	numeric vector with effects of covariates over lambda.
alpha_cov	named list of named numeric vectors with effects of each covariate over alpha values.

abundance	named numeric vector of abundances in the previous timestep.
covariates	matrix with observations in rows and covariates in named columns. Each cell is the value of a covariate in a given observation.

Value

numeric abundance projected one timestep

`RK_project_alpha_pairwise_lambdacov_none_alphacov_none` *Ricker model for projecting abundances, with specific alpha values and no covariate effects*

Description

Ricker model for projecting abundances, with specific alpha values and no covariate effects

Usage

```
RK_project_alpha_pairwise_lambdacov_none_alphacov_none(
  lambda,
  alpha_intra,
  alpha_inter,
  lambda_cov,
  alpha_cov,
  abundance,
  covariates
)
```

Arguments

lambda	numeric lambda value.
alpha_intra	included for compatibility, not used in this model.
alpha_inter	single numeric value.
lambda_cov	included for compatibility, not used in this model.
alpha_cov	included for compatibility, not used in this model.
abundance	named numeric vector of abundances in the previous timestep.
covariates	included for compatibility, not used in this model.

Value

numeric abundance projected one timestep

salinity_list *Salinity measurements*

Description

A list containing salinity values associated to the data from 'neigh_list'. The list has 17 elements, one for each focal species considered. Each element of the list is a dataframe with 2 columns:

- obs_ID: unique identifier of each observation
- salinity: salinity measurement for that observation, in accumulated microsiemens/m2

Usage

```
data(salinity_list)
```

Format

A list with 17 elements, each of which a dataframe of variable number of rows and 2 numeric columns

Note

For details, see Lanuza et al. 2018 Ecology Letters.

spatial_sampling *spatial arrangement of the observations*

Description

A dataset giving the spatial arrangement of observations. The dataset is a list of 16 elements following the structure of 'neigh_list'. Each list component is a dataframe with columns:

Usage

```
data(spatial_sampling)
```

Format

A list with 16 elements, each of which a dataframe of variable number of rows and 18 columns

Details

- obs_ID: unique identifier for each observation
- plot: one of 9 plots of 8.5 x 8.5 m
- subplot: one of 36 subplots of 1x1 m within each plot

Note

For details, see Lanuza et al. 2018 Ecology Letters.

`species_fitness` *Fitness of a species*

Description

Calculates the fitness of a species sensu Godoy et al. (2014).Note that its definition is model-specific, i.e. it depends on the model family from which interaction coefficients were estimated. The function given here assumes a community of n-species, so that species fitness is calculated according to a general competitive response (r) substituting the 2-sp denominator terms of table A1 of Hart et al. 2018. This competitive response can be calculated for a series of species with the function 'cxr_er_fit'.

Usage

```
species_fitness(
  effect_response_fit = NULL,
  lambda = NULL,
  competitive_response = NULL,
  model_family = NULL
)
```

Arguments

`effect_response_fit`	cxr_er_fit object with valid lambda and response terms.
`lambda`	per capita fecundity of the species in the absence of competition.
`competitive_response`	parameter reflecting the species' sensitivity to competition.
`model_family`	model family for which to calculate species fitness.

Details

Thus, the function accepts two sets of parameters. First, a 'cxr_er_fit' object returned from that function. In this case, species fitness will be calculated for all focal taxa included in the 'cxr_er_fit' object.

Otherwise, users may enter a specification of the model to use, as well as lambda and competitive response parameters of a single species.

If no model family is provided, or a model family for which there is no associated 'XX_species_fitness' function, the function resorts to the standard Lotka-Volterra formulation (Hart et al. 2018). Overall, we strongly suggest that you use the standard formulation ONLY if you are completely confident that the model from which you obtained your parameters is consistent with it. Otherwise, you should include your own formulation of species fitness (see vignette 4).

Value

single numeric value/vector, species fitness of one or several taxa

---

`species_rates` *Species germination and survival rates*

Description

A dataset containing germination and survival rates for 17 plant species. It includes columns with the scientific names and their associated codes.

Usage

```
data(species_rates)
```

Format

A data frame with 17 rows and 4 variables

Details

- species: binomial name
- code: four-letter code used in other datasets
- germination: germination rate
- seed.survival: annual survival of ungerminated seed in the soil

Note

For details, see Lanuza et al. 2018 Ecology Letters.

`summary.cxr_er_fit` *CXR summary method for effect response model fits*

Description

CXR summary method for effect response model fits

Usage

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

Arguments

`object`	a `cxr_er_fit` object, from the function with the same name
`...`	other arguments, not used

Value

console output

---

summary.cxr_pm_fit *CXR summary method for population model fits*

Description

CXR summary method for population model fits

Usage

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

Arguments

object	a cxr_pm_fit object, from the function with the same name
...	other arguments, not used

Value

console output

---

summary.cxr_pm_multifit *CXR summary method for multispecies fits*

Description

CXR summary method for multispecies fits

Usage

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

Arguments

object	a cxr_pm_multifit object, from the function with the same name
...	other arguments, not used

Value

console output

Index