Package: ELAplus (via r-universe)

October 26, 2026

Language en-US

Type Package

Title Energy Landscape Analysis for Ecological Communities

Version 1.0.2

Date 2026-05-15

Author Kenta Suzuki [aut, cre, cph], Sotaro Takano [aut], Hiroaki Fujita [aut], Zeus Sato [ctb], Fumihiko Ayabe [ctb]

Maintainer Kenta Suzuki <kenta.suzuki.zk@riken.jp>

Encoding UTF-8

License BSD_3_clause + file LICENSE

Description Energy landscape analysis (ELA) is a systematic method for analyzing an energy landscape represented as a weighted network. This R package is especially designed to analyze ecological dynamics, i.e., transitions in community compositions. For details of the analysis framework, please visit Suzuki K et al. (2021) <doi:10.1002/ecm.1469>.

Depends R (>= 4.1.0)

Imports Rcpp (>= 1.0.11), doParallel, RColorBrewer, foreach, dplyr, magrittr, stringdist, gtools, ggplot2, purrr, rlang, tidyr, data.table, stringr, igraph

Suggests cluster, plot3D, fpc, quanteda, slam, stm, topicmodels, tidyverse, Matrix, glmnet, gsubfn, zoo, ggsci, tidygraph, vegan, knitr, rmarkdown

VignetteBuilder knitr

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

RoxygenNote 7.3.3

LazyData true

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

Date/Publication 2026-05-28 12:20:11 UTC

RemoteUrl https://github.com/cran/ELAplus

RemoteRef HEAD

RemoteSha f75446e862d64f25408098e8f6dcb5ac46d4238e

Contents

ELAplus-package . . . 3
baseabtable . . . 5
basemetadata . . . 5
Bi . . . 6
bin2id . . . 7
bootstrap_ELA . . . 7
bootstrap_SA . . . 8
deltaE_histogram . . . 9
DisconnectivityGraph . . . 10
ELA . . . 11
ELPruning . . . 12
Energy . . . 13
Findbp . . . 13
Formatting . . . 15
GELAObj . . . 16
GradELA . . . 16
GradientDescent . . . 17
gStability . . . 18
hb.paramgen . . . 19
HeatBath . . . 20
id2bin . . . 21
MinTippingPath . . . 21
OnestepHBS . . . 22
PCplot . . . 22
plotSAtest . . . 23
runSA . . . 23
sa2params . . . 25
saEndpoint . . . 25
showDG . . . 26
showGELA3D . . . 27
showIntrGraph . . . 27
showSSD . . . 28
showSSD_ggplot . . . 28
SSentropy . . . 29
sstable . . . 29
Stability . . . 30
SteepestDescent . . . 31
tptable . . . 31

Index **33**

ELAplus-package *Energy Landscape Analysis for Ecological Communities*

Description

Energy landscape analysis (ELA) is a systematic method for analyzing an energy landscape represented as a weighted network. This R package is especially designed to analyze ecological dynamics, i.e., transitions in community compositions. For details of the analysis framework, please visit Suzuki K et al. (2021) <doi:10.1002/ecm.1469>.

Details

This package provides tools for energy landscape analysis (ELA), a framework for analyzing dynamical systems represented as weighted networks. The main functions support construction of energy landscapes, by fitting the community composition data to a pairwise maximum entropy model (or its extension) and subsequent identification of stable states, and analysis of transitions between community compositions.

Author(s)

Kenta Suzuki [aut, cre, cph], Sotaro Takano [aut], Hiroaki Fujita [aut], Zeus Sato [ctb], Fumihiko Ayabe [ctb]

Maintainer: Kenta Suzuki <kenta.suzuki.zk@riken.jp>

References

Suzuki, K., Nakaoka, S., Fukuda, S., & Masuya, H. (2021). "Energy landscape analysis elucidates the multistability of ecological communities across environmental gradients." Ecological Monographs. doi:10.1002/ecm.1469 https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1002/ecm.1469

See Also

GitHub: https://github.com/kecosz/rELA, Portal: https://elaportal.org

Examples

```
# load sample dataset
data("baseabtable")
data("basemetadata")
# Formatting dataset
formatted <- Formatting(baseabtable, basemetadata, 0, c(0.05, 0.05, 0.95))
ocvecs <- formatted[[1]]
envecs <- formatted[[3]]
# Example (without environmental parameters)
# Fitting the data to the extended maximum pairwise entropy model
# without explicit environmental parameters
sa <- runSA(
```

```
  ocvecs,
  enmat   = NULL,
  rep     = 1,
  getall  = FALSE,
  lambda  = 0.01,
  we      = 0.001,
  maxlr = 0.005,
  totalit = 1000
)

# Using the fitted model, normal ELA is run.
ela <- ELA(sa, env=NULL,
           threads=1,SS.itr=2000,
           FindingTip.itr=1000,
           reporting=TRUE)
# Pruning stable states
elap <- ELPruning(ela, th=0.1)
# show disconnectivity graph
showDG(elap[[1]], ocvecs, "test")
# show species interaction graph
showIntrGraph(elap[[1]], sa, th=0.01,
              annot_adj=c(0.75, 2.00))

# Example (with environmental parameters)
# Find best parameter sets for the machine learning (with envecs)

bpresult <- Findbp(ocvecs,enmat=envecs,rep=1,threads=1,nrepeat=1,cvmode="5fold")
bp <- bpresult[[1]]
bpm <- as.numeric(unlist(strsplit(names(bp)[1], split = "_")))
# Fitting the data to the extended maximum pairwise entropy model
# with explicit environmental parameters
sa <- runSA(
  ocvecs,
  enmat   = envecs,
  rep     = 16,
  getall  = FALSE,
  lambda  = bpm[[1]],
  we      = bpm[[2]],
  maxlr = bpm[[3]],
  totalit = bpm[[4]]
)
# Using the fitted model, gradient ELA is run.
gela <- GradELA(
  sa      = sa,
  eid     = "factor.1",
  enmat   = envecs,
  env     = NULL,
  range   = c(0, 1),
  steps   = 32,
  th      = 0.05,
  threads = 2
)
showSSD(gela)
```

baseabtable *Abundance table for 16 taxa across 256 samples*

Description

'baseabtable' is a data frame containing abundance for 16 taxa across 256 samples.

Usage

```
baseabtable
```

Format

A data frame with 256 rows and 16 variables:

Details

Each row corresponds to a sample, and each column represents the abundance of a specific taxon.

The values represent abundance measurements for each taxon in each sample. The exact definition of "abundance" (e.g., counts, relative abundance, or normalized values) depends on the data preprocessing pipeline used.

Source

Generated for example purposes (replace with actual source if applicable)

Examples

```
data(baseabtable)
head(baseabtable)
formatted <- Formatting(baseabtable = baseabtable)
```

basemetadata *Environmental metadata for 256 samples*

Description

'basemetadata' is a data frame containing environmental parameters associated with the same 256 samples as in `baseabtable`.

Usage

```
basemetadata
```

Format

A data frame with 256 rows and 2 variables:

Details

Each row corresponds to a sample, and each column represents an environmental variable.

The rows in this data frame correspond to the same samples and ordering as in `baseabtable`. These variables can be used for ELA with environmental parameters such as `gradELA`

Source

Generated for example purposes (replace with actual source if applicable)

Examples

```
data(baseabtable)
data(basemetadata)
head(basemetadata)
formatted <- Formatting(baseabtable = baseabtable, basemetadata = basemetadata)
```

Bi *Bi*

Description

Identify basin (stable state) and its energy for a given state.

Applies SteepestDescent to the input state and returns the basin (stable state) reached, together with its energy. The stable state is encoded as an integer ID using bin2id.

Usage

```
Bi(xi, h, j)
```

Arguments

`xi`	Binary vector representing the initial state.
`h`	vector of implicit/explicit environmental effects on species.
`j`	Matrix of species interactions.

Value

A list with two elements: (1) integer stable state ID (basin assignment), (2) energy of the corresponding stable state.

`bin2id` *bin2id: map a binary vector to a 64base characters*

Description

Encode a binary vector into a base-64 id string index.

Usage

```
bin2id(bin)
```

Arguments

`bin`	A binary vector (0/1, community composition) to encode.

Value

A character scalar representing the encoded id.

`bootstrap_ELA` *Energy landscape analysis with bootstrap resampled SA results*

Description

Running energy landscape analysis with bootstrap sampling of estimated saresults

Performs ELA and ELPruning using SA fitting results based on the bootstrap resampled occurrence and environmental data using bootstrap_SA.

Basically, representative ELA results with original occurrence and environmental data should be given by `ela=ela`, alternatively the function can select a representative parameter where commonly appearing stable states are most frequent.

The function computes bootstrap energy distributions for all stable states and tipping points appearing in the specified pruned landscape. Energies that are not present in a given trial are stored as `NaN` in the summary table.

Usage

```
bootstrap_ELA(
  bs_params,
  ocmat,
  ela = NULL,
  env = NULL,
  enmat = NULL,
  threads = 1,
  reporting = FALSE
)
```

Arguments

bs_params	output of bootstrap_SA. list of estimated parameters using bootstrap resampled data set.
ocmat	community composition matrix.
ela	representative ELA result (output of ELA). If NULL, the function returns the most representative one.
env	target environmental factor.
enmat	array of the vectors of environmental factors for each sample. Normalization required.
threads	number of cores (threads) used for calculation.
reporting	enable/disable the reporting function.

Value

A list with two elements: (1) representative pruned ELA result (stable states, energies, tipping points, tipping energies), (2) a data frame of bootstrap energies (rows = state IDs, columns = trials) with NaN for missing states.

---

bootstrap_SA *Parameter fitting with bootstrap sampling of datasets*

Description

Perform runSA algorithm and estimate parameters using bootstrap resampled data set (ocmat and enmat). The basic configuration is same as runSA.

Usage

```
bootstrap_SA(
  ocmat,
  enmat = NULL,
  bootstrap.it = 2000,
  rep = 128,
  threads = 1,
  we = 0.001,
  totalit = 1000,
  lambda = 0.01,
  runadamW = TRUE,
  maxlr = 0.005,
  sparse = TRUE,
  reporting = TRUE
)
```

Arguments

ocmat	Numeric matrix (0/1). Rows are samples and columns are species.
enmat	Optional numeric matrix. Environmental variables for each sample. Should be normalized/standardized beforehand.
bootstrap.it	number of bootstrap trials (default: 1000).
rep	Integer. Number of parallel runs (replicates) for optimization.
threads	Integer. Number of cores used when running in parallel.
we	Numeric. Weight decay parameter for AdamW.
totalit	Integer. Total number of optimization iterations.
lambda	Numeric. Sparsity penalty parameter (used when sparse = TRUE).
runadamW	Logical. If TRUE, use AdamW-style optimization.
maxlr	Numeric. Maximum learning rate.
sparse	Logical. If TRUE, apply sparsity constraint.
reporting	Logical. If TRUE, print progress messages.

Value

a list of SA parameters estimated with bootstrap resampling (length bootstrap.it).

---

deltaE_histogram *Plot histograms of bootstrap deltaE values*

Description

For each stable state, this function computes the difference between the energy values of the corresponding tipping point and the stable state across bootstrap samples, then draws a histogram of the resulting deltaE values.

Usage

```
deltaE_histogram(bootstrap_ene, grobj_to_plot, getData = TRUE)
```

Arguments

bootstrap_ene	A data frame containing energies in stable states and tipping points, output of bootstrap_ELA.
grobj_to_plot	A data frame containing stable state - tipping point relationship, output of showDG
getData	Logical. If 'TRUE', return the computed deltaE values as a list. If 'FALSE', return 'NULL'.

Value

If 'getData = TRUE', a list of numeric vectors, one per deltaE for each stable state. If 'getData = FALSE', 'NULL'.

`DisconnectivityGraph` *DisconnectivityGraph: Generate a disconnectivity graph*

Description

Draws a disconnectivity graph from the data frame returned by `GraphObj`. Nodes are plotted at their assigned x-positions and energy levels, and are connected to their parent components by paths.

If `show_pvalue = TRUE`, approximate p-values for the energy gaps of stable states are displayed. The p-values are shown on a log10 scale using the values specified in `scale_labels`.

Usage

```
DisconnectivityGraph(
  grobj,
  s,
  DG_sample = "DG_sample",
  annot_adj = c(0.75, 2),
  min_pval = 0.05,
  scale_labels = c(0.05, 0.1, 0.2, 0.5),
  show_pvalue = FALSE,
  getGraphObj = FALSE,
  plot = TRUE
)
```

Arguments

`grobj`	A data frame returned by `GraphObj`.
`s`	Integer; number of species/features (used for labeling by id2bin).
`DG_sample`	Character; title of the plot.
`annot_adj`	Numeric vector of length 2; horizontal and vertical adjustment for labels.
`min_pval`	Numeric (must be > 0); minimum p-value shown in the visualization. This is usually set to the minimum detection level in bootstrap sampling (e.g., if the number of bootstrap iterations is 1000, the minimum level is 1/1000 = 0.001).
`scale_labels`	Numeric vector of values between 0 and 1; labels for the p-value scale. These values are converted to log-scaled values for plotting.
`show_pvalue`	Logical; if `TRUE`, plots approximate p-values based on the energy gaps between each stable state and its corresponding tipping point estimated by bootstrap re-sampling.
`getGraphObj`	Logical; if `TRUE`, returns the data frame used for plotting the disconnectivity graph, so that users can manually modify the graphical settings.
`plot`	Logical; if TRUE, show disconnectivity graph

Value

If `getGraphObj = TRUE`, returns a data frame containing energy values, x-positions, and p-values for the energy gaps of the identified stable states and tipping points. Otherwise, returns `NULL`.

ELA *Energy landscape analysis in parallel mode*

Description

Identifying stable states and tipping points.

This function estimates stable states by repeated steepest-descent searches `SSestimate` and then checks tipping points between all pairs of stable states using `FindingTippingpoint_cpp`. The result is summarized and converted by `elaconvert` into a list of stable state IDs, their energies, tipping point IDs, and tipping energies.

Usage

```
ELA(
  sa = sa,
  env = NULL,
  SS.itr = 20000,
  FindingTip.itr = 10000,
  threads = 1,
  reporting = TRUE,
  fork = TRUE
)
```

Arguments

`sa`	output of runSA function.
`env`	vector of environmental factors for specifying a single environmental condition (optional).
`SS.itr`	number of iterations of the process to identify the stable states.
`FindingTip.itr`	number of iterations of the process to identify the tipping points.
`threads`	number of cores (threads) used for calculation.
`reporting`	enable/disable the reporting function.
`fork`	logical; if TRUE and `threads > 1`, a cluster is created internally for parallel execution.

Value

A list returned by `elaconvert`: (1) stable state IDs, (2) stable state energies, (3) tipping point ID matrix/list, and (4) tipping energies matrix/list.

ELPruning *ELPruning*

Description

Pruning shallow stable states of energy landscape.

Iteratively prunes stable states (merges basins). If `type="relative"`, stable state whose energy depth is smaller than `th` times the deepest basin depth are pruned and merged to the deeper basins. If `type="bootstrap"`, stable state whose bootstrap confidence interval (default: 95 cross zero is pruned. The function updates stable states and corresponding tipping structures, and returns both the pruned ELA object and a mapping table from stable states before pruning to those after pruning.

Usage

```
ELPruning(
  ela,
  type = "relative",
  th = 0.05,
  lower_qr = 0.05,
  bootstrap_ene = NULL,
  threads = 1,
  reporting = TRUE,
  fork = TRUE
)
```

Arguments

`ela`	output of ELA function.
`type`	the type of pruning `"relative"`(use with `th`) or `"bootstrap"` (use with `lower_qr` and `bootstrap_ene`)
`th`	ratio of the depth of the basin to be pruned with respect to the deepest basin.
`lower_qr`	Numeric value. Quantile used for constructing confidence intervals from bootstrap samples to check whether the estimated energy gaps include zero. Default is `lower_qr = 0.05`.
`bootstrap_ene`	a data frame of bootstrap energies generated by bootstrap_ELA. Necessary when `type="quantile"`.
`threads`	number of cores (threads) used for calculation.
`reporting`	enable/disable the reporting function.
`fork`	logical; if TRUE and `threads > 1`, a cluster is created internally for parallel execution.

Value

A list of two elements: (1) pruned ELA tuple (same format as ELA output), and (2) a data frame with columns `ss.before.pruning` and `ss.after.pruning`.

Energy *Energy*

Description

Calculate energy of a given state.

Computing energy in `state` using given parameters,`alpha` and `beta` based on the (extended) maximum pairwise entropy model.

Usage

```
Energy(state, alpha, beta)
```

Arguments

`state`	Binary vector representing a state (community composition).
`alpha`	vector of implicit/explicit environmental effects on species.
`beta`	Matrix of species interactions.

Value

A numeric scalar giving the energy of the state.

Findbp *Search best hyperparameters for SA fitting (lambda and weight decay) by cross-validation*

Description

Repeating cross-validation over a grid of hyperparameters for simulated annealing-based fitting, evaluates prediction/fitting errors on held-out samples, and returns candidate best parameter/timepoint combinations.

Usage

```
Findbp(
  ocmat,
  enmat = NULL,
  cvmode = c("10fold", "5fold", "loo"),
  nrepeat = 5,
  we = c(0.001),
  maxlrs = c(0.005),
  totalit = 4000,
  lmd = c(5e-04, 0.001, 0.0025, 0.005, 0.0075),
  rep = 8,
```

```
  intv = 100,
  threads = 1,
  tol = 0.03,
  runadamW = TRUE,
  sparse = TRUE,
  fastfitting = TRUE,
  reporting = TRUE
)
```

Arguments

ocmat	Numeric matrix (0/1). occurrence matrix of the communities.
enmat	Optional environmental parameter matrix.
cvmode	Cross-validation mode. One of ‘"10fold"‘, ‘"5fold"‘, or ‘"loo"‘. ‘"loo"‘ performs leave-one-out cross-validation. Default is ‘"10fold"‘.
nrepeat	Number of repeated cross-validation runs for k-fold CV. Ignored when ‘cvmode = "loo"‘. Default is ‘10‘.
we	Numeric vector. Candidate weight decay values.
maxlrs	Numeric vector. Maximum learning rate.
totalit	Integer. Total iterations for SA.
lmd	Numeric vector. Candidate `lambda` values.
rep	Integer. Number of replicates for runSA.
intv	Integer. Recording interval.
threads	Integer. Number of cores.
tol	Numeric. Error tolerance for finding the best hyperparameters (default: 0.03)
runadamW	Logical. Use AdamW.
sparse	Logical. Use sparsity penalty.
fastfitting	Logical. If `TRUE`, returns early iterations within the range specified by the parameter `tol` (default: 0.03) of best error.
reporting	Logical. Print progress.

Value

A list with:

best_candidates A named list of best hyperparameter settings and scores.

allresults All validation results for each grid point.

`Formatting` *Formatting*

Description

Generate occurrence(binary)/abundance/factor matrices and labels from input abundance/metadata tables.

Usage

```
Formatting(
  baseabtable,
  basemetadata = NULL,
  normalize = 1,
  parameters = c(0.05, 0.05, 0.95),
  grouping = 0,
  grouping_th = 0,
  reporting = TRUE
)
```

Arguments

`baseabtable`	Abundance table (samples x species) as data.frame/matrix.
`basemetadata`	Optional environmental metadata table (samples x factors); NULL/empty for none.
`normalize`	Integer 0 or 1. If 1, normalize within each sample to sum to 1.
`parameters`	Numeric vector/list of thresholds (following 3 parameters). 1. "0/1 threshold" 2. "min occurrence threshold" 3. "max occurrence threshold" (e.g., c(0.05, 0.05, 0.95) : species abundance > 0.05 is regarded as present, and species occurring in more than 0.05 and less than 0.95 in the all samples are kept.)
`grouping`	Integer 0 or 1. If 1, group similar occurrence patterns.
`grouping_th`	Numeric threshold for grouping (relative Hamming distance).
`reporting`	Logical. If TRUE, print processing summary.

Value

A list: (ocmatrix, abmatrix, fctmatrix, samplelabel, specieslabel, factorlabel).

GELAObj *GELAObj*

Description

Construct smoothed energy landscape surface objects from GELA output and parameters.

Usage

```
GELAObj(gela, sa, threads = 1)
```

Arguments

gela	GELA object/list
sa	Parameter list generated by runSA
threads	Number of parallel workers.

Value

A list containing surface grid data and state-specific surface points.

GradELA *GradELA*

Description

Estimate energy landscape across environmental gradients.

Runs ELA and ELPruning comprehensively across specified environmental parameters. One environmental parameter can be changed by specifying `eid` over `steps` within `range`. If `env` is not given, the mean vector of `enmat` is used as the reference environmental condition.

Usage

```
GradELA(
  sa = sa,
  pruning_type = "relative",
  eid = NULL,
  enmat = enmat,
  env = NULL,
  range = NULL,
  steps = 16,
  th = 0.05,
  threads = 1,
  SS.itr = 20000,
  FindingTip.itr = 10000,
```

```
  lower_qr = 0.05,
  bs_params = NULL,
  ocmat = NULL
)
```

Arguments

sa	output of runSA function.
pruning_type	the type of pruning "relative"(use with th) or "bootstrap" (use with lower_qr,ocmat and bs_params)
eid	target environmental factor: the specified environmental factor to be changed.
enmat	array/matrix of environmental factors for each sample (normalization required).
env	array of environmental factor values. For the environmental parameters that are not specified for eid, these values are used as references. If not specified, then the mean of values in enmat are used as references.
range	range over which the environmental factor specified by eid is changed.
steps	number of steps in the range.
th	ratio of the depth of the basin to be pruned with respect to the deepest basin.
threads	number of cores (threads) used for calculation.
SS.itr	number of iterations to identify stable states.
FindingTip.itr	number of iterations to identify tipping points.
lower_qr	Numeric value. Quantile used for constructing confidence intervals from bootstrap samples to check whether the estimated energy gaps include zero. Default is lower_qr = 0.05,necessary when type="bootstrap".
bs_params	output of bootstrap_SA. list of estimated parameters using bootstrap resampled data set. necessary when type="bootstrap".
ocmat	community composition matrix.

Value

A list of three elements: (1) ELA results for each step (pruned), (2) the sequence of environmental values, and (3) the reference env vector.

GradientDescent *Gradient descent estimation of model parameters*

Description

Estimates parameters h and J by iterative gradient updates to match first and second moments of the observed binary data under an maximum pairwise entropy model.

Usage

```
GradientDescent(data, maxit)
```

Arguments

`data`	Numeric matrix (typically 0/1). Rows are samples and columns are species or taxa.
`maxit`	Integer. Maximum number of gradient descent iterations.

Details

This function calls `Energy()` to compute energies for candidate states. Ensure that `Energy` is available in the package namespace.

Value

A list with elements:

h.est Numeric vector of estimated implicit environmental parameters.

j.est Numeric matrix of estimated interaction matrix.

`gStability` *gstability*

Description

Calculate energy gap, energy barrier and stable state entropy.

This function evaluates stability indices for both (i) the pruned and (ii) the non-pruned energy landscape. For each sample, it assigns the basin by Bi and computes: - energy gap (sample energy minus basin energy), - energy barrier (minimum tipping energy connected to the basin minus basin energy), - stable state entropy (`SSentropy`).

For basin connectivity: when only one stable state exists, tipping energy is treated as `Inf` and therefore energy barrier is returned as `Inf`.

Usage

```
gStability(
  sa,
  ocmat,
  enmat = NULL,
  seitr = 1000,
  seconvTime = 10000,
  pruning_type = "relative",
  th = 0.1,
  lower_qr = 0.05,
  threads = 1,
  reporting = TRUE,
  bs_params = NULL
)
```

Arguments

sa	: output of runSA function.
ocmat	: matrix of presence/absence status for each sample.
enmat	: matrix of environmental factors for each sample. Normalization required.
seitr	: Number of stochastic search iterations.
seconvTime	: Maximum number of iterations for each search.
pruning_type	the type of pruning "relative"(use with th) or "bootstrap" (use with lower_qr and bootstrap_ene)
th	: Numerical (0~1). Ratio of the depth of the basin to be pruned with respect to the deepest basin in each pair.
lower_qr	Numeric value. Quantile used for constructing confidence intervals from bootstrap samples to check whether the estimated energy gaps include zero. Default is lower_qr = 0.05,necessary when type="bootstrap".
threads	number of cores (threads) used for calculation.#' @param threads : number of cores (threads) used for calculation.
reporting	: enable/disable the reporting function.
bs_params	output of bootstrap_SA. list of estimated parameters using bootstrap resampled data set, necessary when type="bootstrap".

Value

A list of four elements: (1) pruned stability data frame, (2) non-pruned stability data frame, (3) auxiliary object for pruned landscape (parameters + stable states table), and (4) auxiliary object for sample-wise evaluation (inputs and ELA objects).

---

hb.paramgen *Randomly generate model parameters for heat-bath sampling*

Description

Generates random implicit environmental parameters h and a interaction matrix J. Optionally generates explicit environmental coefficients g. The j_connectivity parameter controls the fraction of non-zero off-diagonal interactions.

Usage

```
hb.paramgen(nn, ne = 0, j_connectivity = 1)
```

Arguments

nn	Integer. Number of units (species, taxa, or members).
ne	Integer. Number of environmental factors. If ne > 0, g is generated.
j_connectivity	Numeric in [0, 1]. Fraction of possible pairwise interactions set to non-zero.

Value

A list containing at least:

h Named numeric vector of length nn. Implicit environmental parameters

J Named numeric matrix (nn x nn), symmetric. Interaction matrix

If ne > 0, also includes:

g Named numeric matrix (nn x ne). Explicit environmental parameters

---

HeatBath *Heat-bath sampling of binary states*

Description

Generates samples of binary state vectors using heat-bath updates under an extended maximum pairwise entropy model parameterized by implicit environmental parameters h and interaction matrix J. Optionally includes environmental effects g, in which case an environmental matrix is generated internally.

Usage

```
HeatBath(steps, nproc, h, J, g = NULL)
```

Arguments

steps	Integer. Number of heat-bath iterations to run.
nproc	Integer. Number of parallel samples (rows) to maintain.
h	Numeric vector of length nn, implicit environmental parameters.
J	Numeric matrix of size nn x nn, interaction matrix.
g	Optional numeric matrix of size nn x ne, environmental coefficients.

Value

A list with components:

y Numeric matrix (nproc x nn) of sampled states (0/1).

enmat Numeric matrix (nproc x ne) of environmental values, or NULL if g is NULL.

Examples

```
set.seed(1)
nn <- 5
pars <- hb.paramgen(nn)
res <- HeatBath(steps = 10, nproc = 8, h = pars[[1]], J = pars[[2]])
str(res)
```

`id2bin` *id2bin: map a 64base characters to a binary vector*

Description

Decode a base-64 id string index to a binary vector of length s.

Usage

```
id2bin(id, s)
```

Arguments

`id`	A character scalar encoded by bin2id.
`s`	Integer number of bits to return.

Value

An integer vector of 0/1 bits of length s.

`MinTippingPath` *MinTippingPath*

Description

Compute a minimum "tipping" energy path between two states.

Usage

```
MinTippingPath(s1, s2, hg, j, tmax)
```

Arguments

`s1`	Another integer state ID
`s2`	Another integer state ID
`hg`	explicit/implicit environmental factors
`j`	species interaction matrix
`tmax`	Iteration limit.

Value

Numeric vector of energies along the selected path (or scalar if s1==s2).

OnestepHBS *Perform one heat-bath update step*

Description

Updates each row (sample) of a binary state matrix by choosing one random position per row and resampling it according to precomputed heat-bath probabilities.

Usage

```
OnestepHBS(y, logmo)
```

Arguments

y	Numeric matrix of size nproc x nn (typically 0/1), current states.
logmo	Numeric matrix of probabilities compatible with y. Each row corresponds to a row in y; entries are used to sample 0/1 updates.

Value

A numeric matrix with the same dimensions as y, updated states.

PCplot *PCplot:Scatter plot of community compositions colored by basin label*

Description

Performs PCA on the community composition matrix (ocmat) and plots the first two PCs. Points are colored by the stable state (basin) label estimated from Bi. If pruned = TRUE, colors correspond to the pruned/merged basin labels in ssrep (stablestates2), otherwise to the original labels (stablestates). When export = TRUE, the plotting data frame (stable-state labels + PCs + merged labels) is returned.

Usage

```
PCplot(ocmat, sa = sa, env = NULL, ssrep = NULL, pruned = TRUE, export = FALSE)
```

Arguments

ocmat	Data frame (or matrix) of community compositions (rows = samples, columns = species/features).
sa	Parameter set passed to sa2params.
env	Optional environment object passed to sa2params.
ssrep	Data frame-like object providing stable state label mapping with two columns: stablestates and stablestates2.
pruned	Logical; if TRUE, use stablestates2 for coloring.
export	Logical; if TRUE, return the data used for plotting.

Value

NULL invisibly (or a data frame when `export = TRUE`).

`plotSAtest` *plotSAtest: plot the results of SA fitting (Findbp output)*

Description

Plot mean validation scores over time for the results returned by Findbp, including `95%` confidence intervals.

Usage

```
plotSAtest(sa_results, ylim = c("auto", "auto"))
```

Arguments

`sa_results`	Output list from Findbp.
`ylim`	Numeric length-2 vector (ymin, ymax). Default: `c("auto","auto")`.

Details

This function uses **ggplot2** and tidyverse-style data manipulation. Ensure required packages are installed.

Value

A **ggplot2** object.

`runSA` *Stochastic approximation for parameter fitting*

Description

Estimate species interaction parameters and (optionally) environmental preference parameters by stochastic approximation (SA). The function runs multiple SA optimizations in parallel (when `threads > 1`) using **foreach**.

Usage

```
runSA(
  ocmat,
  enmat = NULL,
  rep = 128,
  threads = 1,
  we = 0.001,
  totalit = 1000,
  lambda = 0.01,
  intv = 100,
  runadamW = TRUE,
  maxlr = 0.005,
  sparse = TRUE,
  getall = FALSE,
  reporting = TRUE
)
```

Arguments

ocmat	Numeric matrix (0/1). Rows are samples and columns are species.
enmat	Optional numeric matrix. Environmental variables for each sample. Should be normalized/standardized beforehand.
rep	Integer. Number of parallel runs (replicates) for optimization.
threads	Integer. Number of cores used when running in parallel.
we	Numeric. Weight decay parameter for AdamW.
totalit	Integer. Total number of optimization iterations.
lambda	Numeric. Sparsity penalty parameter (used when sparse = TRUE).
intv	Integer. Interval of iterations at which intermediate results are recorded.
runadamW	Logical. If TRUE, use AdamW-style optimization.
maxlr	Numeric. Maximum learning rate.
sparse	Logical. If TRUE, apply sparsity constraint.
getall	Logical. If TRUE, return all replicate trajectories.
reporting	Logical. If TRUE, print progress messages.

Value

If getall = TRUE, a list of SA trajectories (length rep). Otherwise, a list with two elements, c(params, enmat) is returned. params is the concatenated matrix of the estimated parameter set:

h Implicit environmental effects on species occurrence

J Species-species interaction matrix

g (If enmat is not NULL) Explicit effects of the environmental factors.

sa2params — *Convert SA output to model parameter objects*

Description

Convert the output of runSA / saEndpoint into parameter objects in (extended) maximum pairwise entropy model: h, J, and optionally g when environmental parameter matrix is specified.

Usage

```
sa2params(sa, env = NULL)
```

Arguments

sa	A list in the form list(param_matrix, enmat) where param_matrix contains estimated parameters.
env	Optional numeric vector of environmental values (column names correspond to environmental factors).Default is NULL.

Value

A list with elements he (h), je (J), ge (g or NULL), and hge (combined field).

saEndpoint — *Extract endpoint parameters from runSA output*

Description

Summarize the final parameter estimates across SA replicates by averaging the last recorded block of parameters.

Usage

```
saEndpoint(fittingMat, ocmat, enmat = NULL)
```

Arguments

fittingMat	A list of SA trajectories returned by runSA(getall = TRUE).
ocmat	Numeric matrix (0/1). Rows are samples and columns are species.
enmat	Optional environmental parameter matrix used in fitting.

Value

A numeric matrix of median endpoint parameters across serials. Rownames correspond to samples and Colnames correspond to species list

showDG *showDG: draw DisconnectivityGraph*

Description

Computing graph objects from ela (and optionally bootstrap energies) and draw the disconnectivity graph for a given community composition matrix oc. More than two stable state are required.

Usage

```
showDG(
  ela,
  oc,
  label = "",
  bootstrap_ene = NULL,
  min_pval = 0.05,
  plot = TRUE,
  scale_labels = c(0.05, 0.1, 0.2, 0.5),
  getGraphObj = FALSE
)
```

Arguments

ela	ELA output tuple/list used by GraphObj.
oc	Community composition matrix/data frame (used to infer s = ncol(oc)).
label	Character; plot title.
bootstrap_ene	Optional bootstrap energy matrix/data frame for computing statistical significance levels of energy gaps.
min_pval	Numeric; minimum p-value shown in the visualization. This is usually set to the minimum detection level in bootstrap sampling (e.g., if the number of bootstrap iterations is 1000, the minimum level is 1/1000 = 0.001).
plot	Logical; if TRUE, show disconnectivity graph
scale_labels	Numeric vector of values between 0 and 1; labels for the p-value scale. These values are converted to log-scaled values for plotting.
getGraphObj	Logical; if TRUE, returns data.frame object used for plotting DisconnectivityGraph Users can manually change the graphical configurations.

Value

A data frame containing energy values, xpositions, pvalues for energy gaps in identified stable states and tipping points.

showGELA3D *GELA3D: draw 3D plot of energy landscape*

Description

Draws a 3D surface of the energy landscape and overlays line trajectories using **plot3D** function.

Usage

```
showGELA3D(gelsobj, theta = 25, phi = 50)
```

Arguments

gelsobj	List with two elements: surface data frame and line data frame.
theta	numeric value. plot3D option: Azimuth angle (in degrees) controlling rotation around the z-axis.
phi	numeric value. plot3D option: Elevation angle (in degrees) controlling the vertical viewing position.

Value

NULL (produces a 3D plot as side effect).

showIntrGraph *showIntrGraph: draw interaction matrix on a PCoA plot*

Description

Computes a 2D PCoA embedding from the interaction matrix (`jest`) and overlays interaction links above a threshold `th`. Positive and negative links are drawn separately, and species names are annotated at their coordinates.

Usage

```
showIntrGraph(ela, sa, env = NULL, th = 0, annot_adj = c(0.75, 2))
```

Arguments

ela	ELA output tuple/list (used to obtain stable states etc.; currently mainly used for context).
sa	Parameter set passed to sa2params.
env	target environmental factor.
th	Numeric threshold for drawing links (absolute value).
annot_adj	Numeric length-2; horizontal/vertical adjustment for labels.

Value

NULL.

---

showSSD *showSSD: draw stable state diagram*

Description

Plots energy values of stable states across a factor (e.g., environmental or control parameter) using the output from `gela`. Each unique stable state is drawn as a separate line with points, and a legend is added.

Usage

```
showSSD(gela)
```

Arguments

`gela`	The output of GradELA. Object containing stable state lists, energies, the factor, and environment labels as used in the code.

Value

NULL (produces a plot as side effect).

---

showSSD_ggplot *showSSD_ggplot: draw stable state diagram using ggplot*

Description

Plots energy values of stable states across a factor (e.g., environmental or control parameter) using the output from `gela`. Each unique stable state is drawn as a separate line with points, and a legend is added.

Usage

```
showSSD_ggplot(gela, plot = TRUE, getSSDobj = TRUE)
```

Arguments

`gela`	The output of GradELA. Object containing stable state lists, energies, the factor, and environment labels as used in the code.
`plot`	Logical; if TRUE, show SSD plot
`getSSDobj`	Logical; if TRUE, returns data.frame object used for plotting SSD diagram Users can manually change the graphical configurations.

Value

A data frame used for plotting SSD diagram.

`SSentropy` *SSentropy*

Description

Estimate stable state entropy for a given state.

Evaluating the entropy of transitions among stable states starting from a given community state.

The returned value includes the estimated entropy, the convergence time, and the number of successful convergences within the given iteration limits.

Usage

```
SSentropy(state, ss, alpha, beta, seitr = 1000, convTime = 10000)
```

Arguments

`state`	Binary vector (or matrix with one row) representing the current state.
`ss`	Matrix of stable states.
`alpha`	Vector of implicit or explicit environmental effects.
`beta`	Matrix of species interactions.
`seitr`	Number of stochastic search iterations.
`convTime`	Maximum number of iterations for each search.

Value

A numeric matrix with three columns: (1) stable state entropy, (2) convergence time, (3) number of converged searches.

`sstable` *sstable*

Description

Summary of stable states.

Builds a table of stable states and their energies from an ELA output. The returned data frame contains stable state IDs and the corresponding binary presence/absence vector (per species). This function is typically used for inspecting stable states after pruning (if `ela` is pruned) or before pruning.

Usage

```
sstable(ela, ocmat)
```

Arguments

ela	output of ELA or the first return value of ELPruning.
ocmat	community composition matrix.

Value

A data frame with columns ID, Energy.

Stability *stability*

Description

Calculate energy gap and stable state entropy as stability indices and returns a dataframe.

For each sample state, this function identifies the basin (stable state) by Bi, then computes (i) the energy gap and (ii) stable state entropy estimated by SSentropy. If enmat is provided and sa was estimated with environmental factors, the basin assignment is computed for each sample-specific env.

Note: the returned stable state id corresponds to the basin (stable state) the sample is assigned to.

Usage

```
Stability(
  sa,
  ocmat,
  enmat = NULL,
  seitr = 1000,
  seconvTime = 10000,
  threads = 1,
  reporting = TRUE
)
```

Arguments

sa	: output of runSA function.
ocmat	: array of the vectors of presence/absence status for each sample.
enmat	: array of the vectors of environmental factors for each sample. Normalization required.
seitr	: Number of stochastic search iterations.
seconvTime	: Maximum number of iterations for each search.
threads	: number of cores (threads) used for calculation.
reporting	: enable/disable the reporting function.

Value

A data frame with stability indices. Columns include at least `energy.gap`, `ss.entropy`, `e.realize`, `e.stable`, `state.id`, and `stable.state.id` (and may include `conv_time` and `convergence` depending on the branch).

`SteepestDescent` *SteepestDescent*

Description

Perform steepest descent on the energy landscape.

Starting from an initial binary state, the steepest descent algorithm will find a local minimum of the energy landscape. The computation is delegated to the C++ implementation `SteepestDescent_cpp`.

Usage

```
SteepestDescent(state, alpha, beta)
```

Arguments

`state`	Binary vector representing the initial state.
`alpha`	vector of implicit/explicit environmental effects.
`beta`	Matrix of species interactions.

Value

A numeric vector of the reached binary stable state and its energy.

`tptable` *tptable*

Description

Summary of tipping points.

Builds a table of tipping points (transition states) and their energies from an ELA output. This function lists only finite tipping points (i.e., entries not equal to `Inf`), and returns the corresponding binary state vectors.

Two stable states are required at least.

Usage

```
tptable(ela, ocmat)
```

Arguments

`ela`	output of ELA or the first return value of ELPruning.
`ocmat`	community composition matrix.

Value

A data frame with columns `TP`, `SS1`, `SS2`, `Energy`, and one column per species (binary state vector). If no tipping points exist, returns NULL (invisibly) after printing a message.

Index

∗ **datasets**
  baseabtable, 5
  basemetadata, 5
∗ **models**
  ELAplus-package, 3
∗ **multivariate**
  ELAplus-package, 3

baseabtable, 5
basemetadata, 5
Bi, 6, *18*, *22*, *30*
bin2id, *6*, 7, *21*
bootstrap_ELA, 7, *9*, *12*
bootstrap_SA, *7*, 8, *8*, *17*, *19*

deltaE_histogram, 9
DisconnectivityGraph, 10

ELA, *7*, *8*, 11, *12*, *16*, *27*, *32*
ELAplus *(*ELAplus-package*)*, 3
ELAplus-package, 3
ELPruning, *7*, 12, *16*, *32*
Energy, 13

Findbp, 13, *23*
Formatting, 15

GELAObj, 16
GradELA, 16, *28*
GradientDescent, 17
gStability, 18

hb.paramgen, 19
HeatBath, 20

id2bin, *10*, 21

MinTippingPath, 21

OnestepHBS, 22

PCplot, 22
plotSAtest, 23

runSA, *8*, *14*, *16*, 23, *25*

sa2params, *22*, 25, *27*
saEndpoint, 25, *25*
showDG, *9*, 26
showGELA3D, 27
showIntrGraph, 27
showSSD, 28
showSSD_ggplot, 28
SSentropy, 29, *30*
sstable, 29
Stability, 30
SteepestDescent, *6*, 31

tptable, 31