

Package ‘ecoregime’

September 10, 2023

Title Analysis of Ecological Dynamic Regimes

Version 0.1.3

Description A toolbox for implementing the Ecological Dynamic Regime framework (Sánchez-Pinillos et al., 2023 <[doi:10.1002/ecm.1589](https://doi.org/10.1002/ecm.1589)>) to characterize and compare groups of ecological trajectories in multidimensional spaces defined by state variables. The package includes the RETRA-EDR algorithm to identify representative trajectories, functions to generate, summarize, and visualize representative trajectories, and several metrics to quantify the distribution and heterogeneity of trajectories in an ecological dynamic regime and quantify the dissimilarity between two or more ecological dynamic regimes.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

URL <https://mspinillos.github.io/ecoregime/>,
<https://github.com/MSPinillos/ecoregime>

BugReports <https://github.com/MSPinillos/ecoregime/issues>

Depends R (>= 3.4.0)

LazyData true

Imports ape, data.table, ecotraj, GDAtools, graphics, methods, shape, smacof, stats, stringr

Suggests knitr, primer, RColorBrewer, rmarkdown, testthat (>= 3.0.0), vegan

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Martina Sánchez-Pinillos [aut, cre, cph]
(<<https://orcid.org/0000-0002-1499-4507>>)

Maintainer Martina Sánchez-Pinillos <martina.sanchez.pinillos@gmail.com>

Repository CRAN

Date/Publication 2023-09-10 17:10:02 UTC

R topics documented:

define_retra	2
dist_edr	6
EDR_data	8
EDR_metrics	9
plot.RETRA	12
retra_edr	15
summary.RETRA	19
Index	21

define_retra	<i>Define representative trajectories from trajectory features</i>
--------------	--

Description

Generate an object of class RETRA from a data frame containing trajectory states to define representative trajectories in Ecological Dynamic Regimes (EDR).

Usage

```
define_retra(data, d = NULL, trajectories = NULL, states = NULL, retra = NULL)
```

Arguments

data	A data frame of four columns indicating identifiers for the new representative trajectories, the individual trajectories or sites to which the states belong, the order of the states in the individual trajectories, and the identifier of the representative trajectory to which the states belong (only if <code>!is.null(retra)</code>). Alternatively, 'data' can be a vector or a list of character vectors including the sequence of segments forming the new representative trajectory. See Details for further clarifications to define data.
d	Either a symmetric matrix or an object of class <code>dist</code> containing the dissimilarities between each pair of states of all trajectories in the EDR. If NULL (default), the length (<code>Length</code>) of the new representative trajectories and the distances between states of different trajectories or sites (<code>Link_distance</code>) are not calculated.
trajectories	Only needed if <code>!is.null(d)</code> . Vector indicating the trajectory or site to which each state in <code>d</code> belongs.
states	Only needed if <code>!is.null(d)</code> . Vector of integers indicating the order of the states in <code>d</code> for each trajectory.
retra	Object of class RETRA returned from <code>retra_edr()</code> . If NULL (default), <code>minSegs</code> and <code>Seg_density</code> are not provided for the new representative trajectories.

Details

Each representative trajectory returned by the function `retra_edr()` corresponds to the longest sequence of representative segments that can be linked according to the criteria defined in the RETRA-EDR algorithm (Sánchez-Pinillos et al., 2023). One could be interested in splitting the obtained trajectories, considering only a fraction of the returned trajectories, or defining representative trajectories following different criteria than those in RETRA-EDR. The function `define_retra()` allows generating an object of class RETRA that can be used in other functions of `ecoregime` (e.g., `plot()`).

For that, it is necessary to provide information about the set of segments or trajectory states that form the new representative trajectory through the argument `data`:

- `data` can be defined as a **data frame** with as many rows as the number of states in all representative trajectories and the following columns:
 - RT A string indicating the identifier of the new representative trajectories. Each identifier needs to appear as many times as the number of states forming each representative trajectory.
 - RT_traj A vector indicating the individual trajectories in the EDR to which each state of the new representative trajectory belongs.
 - RT_states A vector of integers indicating the identifier of the states forming the new representative trajectories. Each integer must refer to the order of the states in the individual trajectories of the EDR to which they belong.
 - RT_retra Only if the new trajectories are defined from representative trajectories returned by `retra_edr()` (when `!is.null(retra)`). A vector of strings indicating the representative trajectory in `retra` to which each state belongs.
- Alternatively, `data` can be defined as either a **vector** (if there is one representative trajectory) or a **list of character vectors** (with as many elements as the number of representative trajectories desired) containing the sequence of segments of the representative trajectories. In any case, each segment needs to be specified in the form `traj[st1-st2]`, where `traj` is the identifier of the original trajectory to which the segment belongs and `st1` and `st2` are identifiers of the initial and final states defining the segment. If only one state of an individual trajectory is considered to form the representative trajectory, the corresponding segment needs to be defined as `traj[st-st]`.

Value

An object of class RETRA, which is a list of length equal to the number of representative trajectories defined. For each trajectory, the following information is returned:

- `minSegs` Value of the `minSegs` parameter used in `retra_edr()`. If `retra` is NULL, `minSegs = NA`.
- `Segments` Vector of strings including the sequence of segments forming the representative trajectory. Each segment is identified by a string of the form `traj[st1-st2]`, where `traj` is the identifier of the original trajectory to which the segment belongs and `st1` and `st2` are identifiers of the initial and final states defining the segment. The same format `traj[st1-st2]` is maintained when only one state of an individual trajectory is considered (`st1 = st2`). `traj`, `st1`, and `st2` are recycled from `data`.
- `Size` Integer indicating the number of states forming the representative trajectory.

Length Numeric value indicating the length of the representative trajectory, calculated as the sum of the dissimilarities in *d* between every pair of consecutive states. If *d* is NULL, Length = NA.

Link_distance Data frame of two columns indicating artificial links between two segments (**Link**) and the dissimilarity between the connected states (**Distance**). When two representative segments are linked by a common state or by two consecutive states of the same trajectory, the link distance is zero or equal to the length of a real segment, respectively. In both cases, the link is not considered in the returned data frame. If *d* is NULL, Link_distance = NA.

Seg_density Data frame of two columns and one row for each representative segment. Density contains the number of segments in the EDR that is represented by each segment of the representative trajectory. kdTree_depth contains the depth of the k-d tree for each leaf represented by the corresponding segment. That is, the number of partitions of the ordination space until finding a region with minSegs segments or less. If *retra* is NULL, Seg_density = NA.

Author(s)

Martina Sánchez-Pinillos

See Also

[retra_edr\(\)](#) for identifying representative trajectories in EDRs through RETRA-EDR.

[summary\(\)](#) for summarizing the characteristics of the representative trajectories.

[plot\(\)](#) for plotting representative trajectories in an ordination space representing the state space of the EDR.

Examples

```
# Example 1 -----
# Define representative trajectories from the outputs of retra_edr().

# Identify representative trajectories using retra_edr()
d <- EDR_data$EDR1$state_dissim
trajectories <- EDR_data$EDR1$abundance$traj
states <- EDR_data$EDR1$abundance$state
old_retra <- retra_edr(d = d, trajectories = trajectories, states = states,
                      minSegs = 5)

# retra_edr() returns three representative trajectories
old_retra

# Keep the last five segments of trajectories "T2" and "T3"
selected_segs <- old_retra$T2$Segments[4:length(old_retra$T2$Segments)]

# Identify the individual trajectories for each state...
selected_segs
selected_traj <- rep(c(15, 4, 4, 1, 14), each = 2)

# ...and the states (in the same order as the representative trajectory).
selected_states <- c(1, 2, 2, 3, 3, 4, 1, 2, 2, 3)

# Generate the data frame with the format indicated in the documentation
```

```

df <- data.frame(RT = rep("A", length(selected_states)),
                 RT_traj = selected_traj,
                 RT_states = as.integer(selected_states),
                 RT_retra = rep("T2", length(selected_states)))

# Remove duplicates (trajectory 4, state 3)
df <- unique(df)

# Generate a RETRA object using define_retra()
new_retra <- define_retra(data = df,
                          d = d,
                          trajectories = trajectories,
                          states = states,
                          retra = old_retra)

# Example 2 -----
# Define representative trajectories from sequences of segments

# Select all segments in T1, split T2 into two new trajectories, and include
# a trajectory composed of states belonging to trajectories "5", "6", and "7"
data <- list(old_retra$T1$Segments,
             old_retra$T2$Segments[1:3],
             old_retra$T2$Segments[4:8],
             c("5[1-2]", "5[2-3]", "7[4-4]", "6[4-5]"))

# Generate a RETRA object using define_retra()
new_retra <- define_retra(data = data,
                          d = d,
                          trajectories = trajectories,
                          states = states,
                          retra = old_retra)

# Example 3 -----
# Define two representative trajectories from individual trajectories in EDR1.

# Define trajectory "A" from states in trajectories 3 and 4
data_A <- data.frame(RT = rep("A", 4),
                    RT_traj = c(3, 3, 4, 4),
                    RT_states = c(1:2, 4:5))

# Define trajectory "B" from states in trajectories 5, 6, and 7
data_B <- data.frame(RT = rep("B", 5),
                    RT_traj = c(5, 5, 7, 6, 6),
                    RT_states = c(1, 2, 4, 4, 5))

# Compile data for both trajectories in a data frame
df <- rbind(data_A, data_B)
df$RT_states <- as.integer(df$RT_states)

# Generate a RETRA object using define_retra()
new_retra <- define_retra(data = df, d = EDR_data$EDR1$state_dissim,
                          trajectories = EDR_data$EDR1$abundance$traj,
                          states = EDR_data$EDR1$abundance$state)

```

 dist_edr

Dissimilarities between Ecological Dynamic Regimes

Description

Generate a matrix containing dissimilarities between one or more pairs of Ecological Dynamic Regimes (EDR). `dist_edr()` computes different dissimilarity indices, all of them based on the dissimilarities between the trajectories of two EDRs.

Usage

```
dist_edr(
  d,
  d.type,
  trajectories = NULL,
  states = NULL,
  edr,
  metric = "dDR",
  symmetrize = NULL,
  ...
)
```

Arguments

<code>d</code>	Symmetric matrix or object of class <code>dist</code> containing the dissimilarities between each pair of states of all trajectories in the EDR or the dissimilarities between each pair of trajectories.
<code>d.type</code>	One of "dStates" (if <code>d</code> contains state dissimilarities) or "dTraj" (if <code>d</code> contains trajectory dissimilarities).
<code>trajectories</code>	Only if <code>d.type = "dStates"</code> . Vector indicating the trajectory or site corresponding to each entry in <code>d</code> .
<code>states</code>	Only if <code>d.type = "dStates"</code> . Vector of integers indicating the order of the states in <code>d</code> for each trajectory.
<code>edr</code>	Vector indicating the EDR to which each trajectory/state in <code>d</code> belongs.
<code>metric</code>	A string indicating the dissimilarity index to be used: "dDR" (default), "minDist", "maxDist".
<code>symmetrize</code>	String naming the function to be called to symmetrize the resulting dissimilarity matrix ("mean", "min", "max", "lower", "upper"). If NULL (default), the matrix is not symmetrized.
<code>...</code>	Only if <code>d.type = "dStates"</code> . Further arguments to calculate trajectory dissimilarities. See <code>ecotraj::trajectoryDistances()</code> .

Details

The implemented metrics are:

"dDR" $d_{DR}(R_1, R_2) = \frac{1}{n} \sum_{i=1}^n d_{TR}(T_{1i}, R_2)$

"minDist" $d_{DRmin}(R_1, R_2) = \min_{i=1}^n \{d_{TR}(T_{1i}, R_2)\}$

"maxDist" $d_{DRmax}(R_1, R_2) = \max_{i=1}^n \{d_{TR}(T_{1i}, R_2)\}$

where R_1 and R_2 are two EDRs composed of n and m ecological trajectories, respectively, and $d_{TR}(T_{1i}, R_2)$ is the dissimilarity between the trajectory T_{1i} of R_1 and the closest trajectory of R_2 :

$$d_{TR}(T_{1i}, R_2) = \min\{d_T(T_{1i}, T_{21}), \dots, d_T(T_{1i}, T_{2m})\}$$

The metrics calculated are not necessarily symmetric. That is, $d_{DR}(R_1, R_2)$ is not necessarily equal to $d_{DR}(R_2, R_1)$. It is possible to symmetrize the returned matrix by indicating the name of the function to be used in `symmetrize`:

"mean" $d_{DRsym} = \frac{d_{DR}(R_1, R_2) + d_{DR}(R_2, R_1)}{2}$

"min" $d_{DRsym} = \min\{d_{DR}(R_1, R_2), d_{DR}(R_2, R_1)\}$

"max" $d_{DRsym} = \max\{d_{DR}(R_1, R_2), d_{DR}(R_2, R_1)\}$

"lower" The lower triangular part of the dissimilarity matrix is used.

"upper" The upper triangular part of the dissimilarity matrix is used.

Value

Matrix including the dissimilarities between every pair of EDRs.

Author(s)

Martina Sánchez-Pinillos

References

Sánchez-Pinillos, M., Kéfi, S., De Cáceres, M., Dakos, V. 2023. Ecological Dynamic Regimes: Identification, characterization, and comparison. *Ecological Monographs*. doi:10.1002/ecm.1589

Examples

```
# Load species abundances and compile in a data frame
abun1 <- EDR_data$EDR1$abundance
abun2 <- EDR_data$EDR2$abundance
abun3 <- EDR_data$EDR3$abundance
abun <- data.frame(rbind(abun1, abun2, abun3))

# Define row names in abun to keep the reference of the EDR, trajectory, and
# state
row.names(abun) <- paste0(abun$EDR, "_", abun$traj, "_", abun$state)

# Calculate dissimilarities between every pair of states
# For example, Bray-Curtis index
```

```

dStates <- vegan::vegdist(abun[, -c(1, 2, 3)], method = "bray")

# Use the labels in dStates to define the trajectories to which each state
# belongs
id_traj <- vapply(strsplit(labels(dStates), "_"), function(x){
  paste0(x[1], "_", x[2])
}, character(1))
id_state <- vapply(strsplit(labels(dStates), "_"), function(x){
  as.integer(x[3])
}, integer(1))
id_edr <- vapply(strsplit(labels(dStates), "_"), function(x){
  paste0("EDR", x[1])
}, character(1))

# Calculate dissimilarities between every pair of trajectories
dTraj <- ecotraj::trajectoryDistances(d = dStates, sites = id_traj,
  surveys = id_state,
  distance.type = "DSPD")

# Use labels in dTraj to identify EDRs
id_edr_traj <- vapply(strsplit(labels(dTraj), "_"), function(x){
  paste0("EDR", x[1])
}, character(1))

# Compute dissimilarities between EDRs:
# 1.1) without symmetrizing the matrix and using state dissimilarities
dEDR <- dist_edr(d = dStates, d.type = "dStates",
  trajectories = id_traj, states = id_state, edr = id_edr,
  metric = "dDR", symmetrize = NULL)

# 1.2) without symmetrizing the matrix and using trajectory dissimilarities
dEDR <- dist_edr(d = dTraj, d.type = "dTraj", edr = id_edr_traj,
  metric = "dDR", symmetrize = NULL)

# 2) symmetrizing by averaging elements on and below the diagonal
dEDR <- dist_edr(d = dTraj, d.type = "dTraj", edr = id_edr_traj,
  metric = "dDR", symmetrize = "mean")

```

EDR_data

Ecological Dynamic Regime data

Description

Example datasets to characterize and compare EDRs, including abundance data, state, segment, and trajectory dissimilarity matrices for 90 artificial communities belonging to three different EDRs.

Usage

EDR_data

Format

List of three nested sublists ("EDR1", "EDR2", and "EDR3"), each associated with one EDR, including the following elements:

- abundance: Data table with 15 columns and one row for each community state:
 - EDR: Integer indicating the identifier of the EDR.
 - traj: Integer containing the identifier of the trajectory for each artificial community in the corresponding EDR. Each trajectory represents a different sampling unit.
 - state: Integer indicating the observations or states of each community. The sequence of states of a given community forms a trajectory.
 - sp1, ..., sp12: Vectors containing species abundances for each community state.
- state_dissim: Object of class `dist` containing Bray-Curtis dissimilarities between every pair of states in abundance (see Details).
- segment_dissim: Object of class `dist` containing the dissimilarities between every pair of trajectory segments in abundance (see Details).
- traj_dissim: Object of class `dist` containing the dissimilarities between every pair of community trajectories in abundance (see Details).

Details

Artificial data was generated following the procedure explained in Box 1 in Sánchez-Pinillos et al. (2023) In particular, the initial state of each community was defined using a hypothetical environmental space with optimal locations for 12 species. Community dynamics were simulated using a general Lotka-Volterra model.

State dissimilarities were calculated using the Bray-Curtis metric. Segment and trajectory dissimilarities were calculated using the package `'ecotraj'`.

References

Sánchez-Pinillos, M., Kéfi, S., De Cáceres, M., Dakos, V. 2023. Ecological Dynamic Regimes: Identification, characterization, and comparison. *Ecological Monographs*. doi:10.1002/ecm.1589

Description

Set of metrics to analyze the distribution and variability of trajectories in Ecological Dynamic Regimes (EDR), including dynamic dispersion (dDis), dynamic beta diversity (dBD), and dynamic evenness (dEve).

Usage

```

dDis(
  d,
  d.type,
  trajectories,
  states = NULL,
  reference,
  w.type = "none",
  w.values,
  ...
)

dBD(d, d.type, trajectories, states = NULL, ...)

dEve(d, d.type, trajectories, states = NULL, w.type = "none", w.values, ...)

```

Arguments

<code>d</code>	Symmetric matrix or object of class <code>dist</code> containing the dissimilarities between each pair of states of all trajectories in the EDR or the dissimilarities between each pair of trajectories. To compute <code>dDis</code> , <code>d</code> needs to include the dissimilarities between all states/trajectories and the states/trajectory of reference.
<code>d.type</code>	One of "dStates" (if <code>d</code> contains state dissimilarities) or "dTraj" (if <code>d</code> contains trajectory dissimilarities).
<code>trajectories</code>	Vector indicating the trajectory or site corresponding to each entry in <code>d</code> .
<code>states</code>	Only if <code>d.type = "dStates"</code> . Vector of integers indicating the order of the states in <code>d</code> for each trajectory.
<code>reference</code>	Vector of the same class as <code>trajectories</code> and length equal to one, indicating the reference trajectory to compute <code>dDis</code> .
<code>w.type</code>	Method used to weight individual trajectories: <ul style="list-style-type: none"> "none": All trajectories are considered equally relevant (default). "length": Trajectories are weighted by their length, calculated as the sum of the dissimilarities between every pair of consecutive states. <code>d</code> must contain dissimilarities between trajectory states and <code>d.type = "dStates"</code>. "size": Trajectories are weighted by their size, calculated as the number of states forming the trajectory. <code>d</code> must contain dissimilarities between trajectory states and <code>d.type = "dStates"</code>. "precomputed": Trajectories weighted according to different criteria.
<code>w.values</code>	Only if <code>w.type = "precomputed"</code> . Numeric vector of length equal to the number of trajectories containing the weight of each trajectory.
<code>...</code>	Only if <code>d.type = "dStates"</code> . Further arguments to calculate trajectory dissimilarities. See <code>ecotraj::trajectoryDistances()</code> .

Details

Dynamic dispersion (dDis())

$dDis$ is calculated as the average dissimilarity between each trajectory in an EDR and a target trajectory taken as reference (Sánchez-Pinillos et al., 2023).

$$dDis = \frac{\sum_{i=1}^m d_{i\alpha}}{m}$$

where $d_{i\alpha}$ is the dissimilarity between trajectory i and the trajectory of reference α , and m is the number of trajectories.

Alternatively, it is possible to calculate a weighted mean of the dissimilarities by assigning a weight to each trajectory.

$$dDis = \frac{\sum_{i=1}^m w_i d_{i\alpha}}{\sum_{i=1}^m w_i}$$

where w_i is the weight assigned to trajectory i .

Dynamic beta diversity (dBD())

dBD quantifies the overall variation of the trajectories in an EDR and is equivalent to the average distance to the centroid of the EDR (De Cáceres et al., 2019).

$$dBD = \frac{\sum_{i=1}^{m-1} \sum_{j=i+1}^m d_{ij}^2}{m(m-1)}$$

Dynamic evenness (dEve())

$dEve$ quantifies the regularity with which an EDR is filled by the individual trajectories (Sánchez-Pinillos et al., 2023).

$$dEve = \frac{\sum_{l=1}^{m-1} \min\left(\frac{d_{ij}}{\sum_{l=1}^{m-1} d_{ij}}, \frac{1}{m-1}\right) - \frac{1}{m-1}}{1 - \frac{1}{m-1}}$$

where d_{ij} is the dissimilarity between trajectories i and j linked in a minimum spanning tree by the link l .

Optionally, it is possible to weight the trajectories of the EDR. In that case, $dEve$ becomes analogous to the functional evenness index proposed by Villéger et al. (2008).

$$dEve_w = \frac{\sum_{l=1}^{m-1} \min\left(\frac{EW_{ij}}{\sum_{l=1}^{m-1} EW_{ij}}, \frac{1}{m-1}\right) - \frac{1}{m-1}}{1 - \frac{1}{m-1}}$$

where EW_{ij} is the weighted evenness:

$$EW_{ij} = \frac{d_{ij}}{w_i + w_j}$$

Value

- $dDis()$ returns the value of dynamic dispersion for a given trajectory taken as a reference.
- $dBD()$ returns the value of dynamic beta diversity.
- $dEve()$ returns the value of dynamic evenness.

Author(s)

Martina Sánchez-Pinillos

References

- De Cáceres, M, Coll L, Legendre P, Allen RB, Wisser SK, Fortin MJ, Condit R & Hubbell S. (2019). Trajectory analysis in community ecology. *Ecological Monographs*.
- Sánchez-Pinillos, M., Kéfi, S., De Cáceres, M., Dakos, V. 2023. Ecological Dynamic Regimes: Identification, characterization, and comparison. *Ecological Monographs*. doi:10.1002/ecm.1589
- Villéger, S., Mason, N.W.H., Mouillot, D. (2008) New multidimensional functional diversity indices for a multifaced framework in functional ecology. *Ecology*.

Examples

```
# Data to compute dDis, dBD, and dEve
dStates <- EDR_data$EDR1$state_dissim
dTraj <- EDR_data$EDR1$traj_dissim
trajectories <- paste0("T", EDR_data$EDR1$abundance$traj)
states <- EDR_data$EDR1$abundance$state

# Dynamic dispersion taking the first trajectory as reference
dDis(d = dTraj, d.type = "dTraj", trajectories = unique(trajectories),
     reference = "T1")

# Dynamic dispersion weighting trajectories by their length
dDis(d = dStates, d.type = "dStates", trajectories = trajectories, states = states,
     reference = "T1", w.type = "length")

# Dynamic beta diversity using trajectory dissimilarities
dBD(d = dTraj, d.type = "dTraj", trajectories = unique(trajectories))

# Dynamic evenness
dEve(d = dStates, d.type = "dStates", trajectories = trajectories, states = states)

# Dynamic evenness considering that the 10 first trajectories are three times
# more relevant than the rest
w.values <- c(rep(3, 10), rep(1, length(unique(trajectories))-10))
dEve(d = dTraj, d.type = "dTraj", trajectories = unique(trajectories),
     w.type = "precomputed", w.values = w.values)
```

plot.RETRA

Plot representative trajectories of Ecological Dynamic Regimes

Description

Plot representative trajectories of an Ecological Dynamic Regime (EDR) in the state space distinguishing between the segments belonging to real trajectories of the EDR and the artificial links between segments.

Usage

```
## S3 method for class 'RETRA'
plot(
  x,
  d,
  trajectories,
  states,
  select_RT = NULL,
  traj.colors = NULL,
  RT.colors = NULL,
  sel.color = NULL,
  link.color = NULL,
  link.lty = 2,
  axes = c(1, 2),
  ...
)
```

Arguments

x	Object of class RETRA.
d	Symmetric matrix or dist object containing the dissimilarities between each pair of states of all trajectories in the EDR or data frame containing the coordinates of all trajectory states in an ordination space.
trajectories	Vector indicating the trajectory or site to which each state in d belongs.
states	Vector of integers indicating the order of the states in d for each trajectory.
select_RT	Optional string indicating the name of a representative trajectory that must be highlighted in the plot. By default (select_RT = NULL), all representative trajectories are represented with the same color.
traj.colors	Specification for the color of all individual trajectories (defaults "grey") or a vector with length equal to the number of trajectories indicating the color for each individual trajectory.
RT.colors	Specification for the color of representative trajectories (defaults "black").
sel.color	Specification for the color of the selected representative trajectory (defaults "red"). Only if !is.null(select_RT).
link.color	Specification for the color of the links between trajectory segments forming representative trajectories. By default, the same color than RT.colors is used.
link.lty	The line type of the links between trajectory segments forming representative trajectories. Defaults 2 = "dashed" (See graphics::par).
axes	An integer vector indicating the pair of axes in the ordination space to be plotted.
...	Arguments for generic plot() .

Value

The function plot() plots a set of individual trajectories and the representative trajectories in an ordination space defined through d or calculated by applying metric multidimensional scaling (mMDS; Borg and Groenen, 2005) to d.

Author(s)

Martina Sánchez-Pinillos

References

Borg, I., & Groenen, P. J. F. (2005). *Modern Multidimensional Scaling* (2nd ed.). Springer.

Sánchez-Pinillos, M., Kéfi, S., De Cáceres, M., Dakos, V. 2023. Ecological Dynamic Regimes: Identification, characterization, and comparison. *Ecological Monographs*. doi:10.1002/ecm.1589

See Also

[retra_edr\(\)](#) for identifying representative trajectories in EDRs applying RETRA-EDR.

[define_retra\(\)](#) for defining representative trajectories from a subset of segments or trajectory features.

[summary\(\)](#) for summarizing representative trajectories in EDRs.

Examples

```
# Example 1 -----

# d contains the dissimilarities between trajectory states
d <- EDR_data$EDR1$state_dissim

# trajectories and states are defined according to `d` entries.
trajectories <- EDR_data$EDR1$abundance$traj
states <- EDR_data$EDR1$abundance$state

# x defined from retra_edr(). We obtain three representative trajectories.
RT <- retra_edr(d = d, trajectories = trajectories, states = states, minSegs = 5)
summary(RT)

# Plot individual trajectories in blue and representative trajectories in orange,
# "T2" will be displayed in green. Artificial links will be displayed with a
# dotted line.
plot(x = RT, d = d, trajectories = trajectories, states = states, select_RT = "T2",
     traj.colors = "lightblue", RT.colors = "orange", sel.color = "darkgreen",
     link.lty = 3, main = "Representative trajectories in EDR1")

# Example 2 -----

# d contains the coordinates in an ordination space. For example, we use
# the coordinates of the trajectory states after applying a principal component
# analysis (PCA) to an abundance matrix.
abun <- EDR_data$EDR1$abundance
pca <- prcomp(abun[, -c(1:3)])
coord <- data.frame(pca$x)

# trajectories and states are defined according to the abundance matrix
# used in the PCA
```

```

trajectories <- EDR_data$EDR1$abundance$traj
states <- EDR_data$EDR1$abundance$state

# Instead of using the representative trajectories obtained from `retra_edr()`,
# we will define the set of trajectories that we want to highlight. For example,
# we can select the trajectories whose initial and final states are in the
# extremes of the first axis.
T1 <- trajectories[which.max(coord[, 1])]
T2 <- trajectories[which.min(coord[, 1])]
RT_traj <- c(trajectories[trajectories %in% T1],
            trajectories[trajectories %in% T2])
RT_states <- c(states[which(trajectories %in% T1)],
              states[which(trajectories %in% T2)])

# Create a data frame to generate a RETRA object using define_retra
RT_df <- data.frame(RT = c(rep("T1", sum(trajectories %in% T1)),
                        rep("T2", sum(trajectories %in% T2))),
                  RT_traj = RT_traj,
                  RT_states = as.integer(RT_states))
RT_retra <- define_retra(data = RT_df)

# Plot the defined trajectories with the default graphic values
plot(x = RT_retra, d = coord, trajectories = trajectories, states = states,
     main = "Extreme trajectories in EDR1")

```

retra_edr	<i>Representative trajectories in Ecological Dynamic Regimes (RETRA-EDR)</i>
-----------	--

Description

`retra_edr()` applies the algorithm RETRA-EDR (Sánchez-Pinillos et al., 2023) to identify representative trajectories summarizing the main dynamical patterns of an Ecological Dynamic Regime (EDR).

Usage

```

retra_edr(
  d,
  trajectories,
  states,
  minSegs,
  dSegs = NULL,
  coordSegs = NULL,
  traj_Segs = NULL,
  state1_Segs = NULL,
  state2_Segs = NULL,
  Dim = NULL,

```

```

    eps = 0
  )

```

Arguments

<code>d</code>	Either a symmetric matrix or an object of class <code>dist</code> containing the dissimilarities between each pair of states of all trajectories in the EDR.
<code>trajectories</code>	Vector indicating the trajectory or site to which each state in <code>d</code> belongs.
<code>states</code>	Vector of integers indicating the order of the states in <code>d</code> for each trajectory.
<code>minSegs</code>	Integer indicating the minimum number of segments in a region of the EDR represented by a segment of the representative trajectory.
<code>dSegs</code>	Either a symmetric matrix or an object of class <code>dist</code> containing the dissimilarities between every pair of trajectory segments (see Details).
<code>coordSegs</code>	Matrix containing the coordinates of trajectory segments (rows) in each axis (columns) of an ordination space (see Details).
<code>traj_Segs</code>	Vector indicating the trajectory to which each segment in <code>dSeg</code> and/or <code>coordSegs</code> belongs. Only required if <code>dSegs</code> or <code>coordSegs</code> are not <code>NULL</code> .
<code>state1_Segs</code>	Vector indicating the initial state of each segment in <code>dSegs</code> and/or <code>coordSegs</code> according to the values given in <code>states</code> . Only required if <code>dSegs</code> or <code>coordSegs</code> are not <code>NULL</code> .
<code>state2_Segs</code>	Vector indicating the final state of each segment in <code>dSegs</code> and/or <code>coordSegs</code> according to the values given in <code>states</code> . Only required if <code>dSegs</code> or <code>coordSegs</code> are not <code>NULL</code> .
<code>Dim</code>	Optional integer indicating the number of axes considered to partition the segment space and generate a k-d tree. By default (<code>Dim = NULL</code>), all axes are considered.
<code>eps</code>	Numeric value indicating the minimum length in the axes of the segment space to be partitioned when the k-d tree is generated. If <code>eps = 0</code> (default), partitions are made regardless of the size.

Details

The algorithm RETRA-EDR is based on a partition-and-group approach by which it identifies regions densely crossed by ecological trajectories in an EDR, selects a representative segment in each dense region, and joins the representative segments by a set of artificial Links to generate a network of representative trajectories. For that, RETRA-EDR splits the trajectories of the EDR into segments and uses an ordination space generated from a matrix containing the dissimilarities between trajectory segments. Dense regions are identified by applying a k-d tree to the ordination space.

By default, RETRA-EDR calculates segment dissimilarities following the approach by De Cáceres et al. (2019) and applies metric multidimensional scaling (mMDS, Borg and Groenen, 2005) to generate the ordination space. It is possible to use other dissimilarity metrics and/or ordination methods and reduce the computational time by indicating the dissimilarity matrix and the coordinates of the segments in the ordination space through the arguments `dSegs` and `coordSegs`, respectively.

- If `!is.null(dSegs)` and `is.null(coordSegs)`, RETRA-EDR is computed by applying mMDS to `dSegs`.

- If `!is.null(dSegs)` and `!is.null(coordSegs)`, RETRA-EDR is directly computed from the coordinates provided in `coordSegs` and representative segments are identified using `dSegs`. `coordSegs` should be calculated by the user from `dSegs`.
- If `is.null(dSegs)` and `!is.null(coordSegs)` (not recommended), RETRA-EDR is directly computed from the coordinates provided in `coordSegs`. As `dSegs` is not provided, `retra_edr()` assumes that the ordination space is metric and identifies representative segments using the Euclidean distance.

Value

The function `retra_edr()` returns an object of class `RETRA`, which is a list of length equal to the number of representative trajectories identified. For each trajectory, the following information is returned:

`minSegs` Value of the `minSegs` parameter.

`Segments` Vector of strings including the sequence of segments forming the representative trajectory. Each segment is identified by a string of the form `traj[st1-st2]`, where `traj` is the identifier of the original trajectory to which the segment belongs and `st1` and `st2` are identifiers of the initial and final states defining the segment.

`Size` Numeric value indicating the number of states forming the representative trajectory.

`Length` Numeric value indicating the length of the representative trajectory, calculated as the sum of the dissimilarities in `d` between every pair of consecutive states.

`Link_distance` Data frame of two columns indicating artificial links between representative segments (`Link`) and the dissimilarity between the connected states (`Distance`). When two representative segments are linked by a common state or by two consecutive states of the same trajectory, the link distance is zero or equal to the length of a real segment, respectively. In both cases, the link is not considered in the returned data frame.

`Seg_density` Data frame of two columns and one row for each representative segment. `Density` contains the number of segments in the EDR that is represented by each segment of the representative trajectory. `kdTree_depth` contains the depth of the k-d tree for each leaf represented by the corresponding segment. That is, the number of partitions of the ordination space until finding a region with `minSegs` segments or less.

Author(s)

Martina Sánchez-Pinillos

References

- Borg, I., & Groenen, P. J. F. (2005). *Modern Multidimensional Scaling* (2nd ed.). Springer.
- De Cáceres, M, Coll L, Legendre P, Allen RB, Wiser SK, Fortin MJ, Condit R & Hubbell S. (2019). Trajectory analysis in community ecology. *Ecological Monographs*.
- Sánchez-Pinillos, M., Kéfi, S., De Cáceres, M., Dakos, V. 2023. Ecological Dynamic Regimes: Identification, characterization, and comparison. *Ecological Monographs*. doi:10.1002/ecm.1589

See Also

`summary()` for summarizing the characteristics of the representative trajectories.

`plot()` for plotting representative trajectories in an ordination space representing the state space of the EDR.

`define_retra()` for defining representative trajectories from a subset of segments or trajectory features.

Examples

```
# Example 1 -----
# Identify representative trajectories from state dissimilarities

# Calculate state dissimilarities (Bray-Curtis) from species abundances
abundance <- data.frame(EDR_data$EDR1$abundance)
d <- vegan::vegdist(abundance[, -c(1:3)], method = "bray")

# Identify the trajectory (or site) and states in d
trajectories <- abundance$traj
states <- as.integer(abundance$state)

# Compute RETRA-EDR
RT1 <- retra_edr(d = d, trajectories = trajectories, states = states,
                 minSegs = 5)

# Example 2 -----
# Identify representative trajectories from segment dissimilarities

# Calculate segment dissimilarities using the Hausdorff distance
dSegs <- ecotraj::segmentDistances(d = d, sites = trajectories,
                                   surveys = states,
                                   distance.type = "Hausdorff")

dSegs <- dSegs$Dseg

# Identify the trajectory (or site) and states in dSegs:
# Split the labels of dSegs (traj[st1-st2]) into traj, st1, and st2
seg_components <- strsplit(gsub("\\]", "", gsub("\\[", "-", labels(dSegs))), "-")
traj_Segs <- sapply(seg_components, "[", 1)
state1_Segs <- as.integer(sapply(seg_components, "[", 2))
state2_Segs <- as.integer(sapply(seg_components, "[", 3))

# Compute RETRA-EDR
RT2 <- retra_edr(d = d, trajectories = trajectories, states = states, minSegs = 5,
                 dSegs = dSegs, traj_Segs = traj_Segs,
                 state1_Segs = state1_Segs, state2_Segs = state2_Segs)
```

summary.RETRA	<i>Summarize representative trajectories</i>
---------------	--

Description

Summarize the properties of representative trajectories returned by [retra_edr\(\)](#) or [define_retra\(\)](#)

Usage

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

Arguments

object	An object of class RETRA.
...	(not used)

Value

Data frame with nine columns and one row for each representative trajectory in object. The columns in the returned data frame contain the following information:

ID Identifier of the representative trajectories.

Size Number of states forming each representative trajectory.

Length Sum of the dissimilarities in *d* between every pair of consecutive states forming the representative trajectories.

Avg_link Mean value of the dissimilarities between consecutive states of the representative trajectories that do not belong to the same ecological trajectory or site (i.e., artificial links).

Sum_link Sum of the dissimilarities between consecutive states of the representative trajectories that do not belong to the same ecological trajectory or site (i.e., artificial links).

Avg_density Mean value of the number of segments represented by each segment of the representative trajectory (excluding artificial links).

Max_density Maximum number of segments represented by at least one of the segments of the representative trajectory (excluding artificial links).

Avg_depth Mean value of the k-d tree depths, that is, the number of partitions of the ordination space until finding a region with `minSegs` segments or less.

Max_depth Maximum depth in the k-d tree, that is, the number of partitions of the ordination space until finding a region with `minSegs` segments or less.

See Also

[retra_edr\(\)](#) for identifying representative trajectories in EDRs applying RETRA-EDR.

[define_retra\(\)](#) for generating an object of class RETRA from trajectory features.

Examples

```
# Apply RETRA-EDR to identify representative trajectories
d = EDR_data$EDR1$state_dissim
trajectories = EDR_data$EDR1$abundance$traj
states = EDR_data$EDR1$abundance$state
RT <- retra_edr(d = d, trajectories = trajectories, states = states, minSegs = 5)

# Summarize the properties of the representative trajectories in a data frame
summary(RT)
```

Index

* datasets

EDR_data, 8

dBD (EDR_metrics), 9
dDis (EDR_metrics), 9
define_retra, 2
define_retra(), 14, 18, 19
dEve (EDR_metrics), 9
dist, 2, 6, 10, 16
dist_edr, 6

ecotraj::trajectoryDistances(), 6, 10
EDR_data, 8
EDR_metrics, 9

graphics::par, 13

plot(), 3, 4, 13, 18
plot.RETRA, 12

retra_edr, 15
retra_edr(), 2-4, 14, 19

summary(), 4, 14, 18
summary.RETRA, 19