Package: gecko (via r-universe)

September 18, 2024

Type Package

Title Geographical Ecology and Conservation Knowledge Online

Version 1.0.0

Depends R (>= 4.1.0)

Imports terra, sp, grDevices, graphics, stats, utils, geosphere, methods, red, biomod2, kernlab

BugReports `https://github.com/VascoBranco/gecko/issues`

Author Vasco V. Branco [cre, aut] (<`https://orcid.org/0000-0001-7797-3183`>), Pedro Cardoso [aut] (<`https://orcid.org/0000-0001-8119-9960`>), Luís Correia [ctb] (<`https://orcid.org/0000-0003-2439-1168`>)

Maintainer Vasco V. Branco <`vasco.branco@helsinki.fi`>

Description Includes a collection of geographical analysis functions aimed primarily at ecology and conservation science studies, allowing processing of both point and raster data. Future versions will integrate species threat datasets developed by the authors.

License GPL-2

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

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

RemoteUrl https://github.com/vascobranco/gecko

RemoteRef HEAD

RemoteSha e9990cbdc81b637e957927c8530db0504b0222e3

Contents

`clean` *Uniformize raster layers.*

Description

Crop raster layers to minimum size possible and uniformize `NA` values across layers.

Usage

```
clean(layers)
```

Arguments

`layers`	SpatRaster. As defined in package terra, see `terra::rast()`.

Details

Excludes all marginal rows and columns with only `NA` values and change values to `NA` if they are `NA` in any of the layers.

Value

SpatRaster. Same class as layers.

Examples

```
region = gecko.data("layers")
terra::plot(clean(region))
```

`confusion.matrix` *Create a confusion matrix*

Description

Create a confusion matrix for any multiclass set of predicted vs observed labels in a classification problem.

Usage

```
confusion.matrix(actual, predicted)
```

Arguments

`actual`	dataframe. Original labels.
`predicted`	dataframe. Predicted labels.

Value

data.frame. Predicted labels (rows) x Observed labels (cols).

Examples

```
x = c("FALSE", "TRUE", "FALSE", "TRUE", "TRUE")
y = c("TRUE", "TRUE", "TRUE", "TRUE", "TRUE")
confusion.matrix(x, y)
```

`create.east` *Create eastness layer.*

Description

Create a layer depicting eastness based on an elevation layer.

Usage

```
create.east(layers)
```

Arguments

`layers`	SpatRaster. A layer of elevation (a digital elevation model - DEM). As defined in package terra, see `terra::rast()`.

Details

Using elevation, aspect can be calculated. Yet, it is a circular variable (0 = 360) and has to be converted to northness and eastness to be useful for modelling.

Value

SpatRaster.

Examples

```
region = gecko.data("layers")
terra::plot(create.east(region[[3]]))
```

`create.lat` *Create latitude layer.*

Description

Create a layer depicting latitude based on any other.

Usage

```
create.lat(layers)
```

Arguments

`layers`	SpatRaster. As defined in package terra, see `terra::rast()`.

Details

Using latitude (and longitude) in models may help limiting the extrapolation of the predicted area much beyond known areas.

Value

SpatRaster.

Examples

```
region = gecko.data("layers")
terra::plot(create.lat(region[[1]]))
```

`create.long` *Create longitude layer.*

Description

Create a layer depicting longitude based on any other.

Usage

```
create.long(layers)
```

Arguments

`layers`	SpatRaster. As defined in package terra, see `terra::rast()`.

Details

Using longitude (and latitude) in models may help limiting the extrapolation of the predicted area much beyond known areas.

Value

SpatRaster.

Examples

```
region = gecko.data("layers")
terra::plot(create.long(region))
```

`create.north` *Create northness layer.*

Description

Create a layer depicting northness based on an elevation layer.

Usage

```
create.north(layers)
```

Arguments

`layers`	SpatRaster. A layer of elevation (a digital elevation model - DEM). As defined in package terra, see `terra::rast()`.

Details

Using elevation, aspect can be calculated. Yet, it is a circular variable (0 = 360) and has to be converted to northness and eastness to be useful for modelling.

Value

SpatRaster.

Examples

```
region = gecko.data("layers")
terra::plot(create.north(region[[3]]))
```

`distance` *Create distance layer.*

Description

Creates a layer depicting distances to records using the minimum, average, distance to the minimum convex polygon or distance taking into account a cost surface.

Usage

```
distance(longlat, layers, type = "minimum")
```

Arguments

`longlat`	matrix. Matrix of longitude and latitude or eastness and northness (two columns in this order) of species occurrence records.
`layers`	SpatRaster. As defined in package terra, see `terra::rast()`. To serve as model to create distance layer.
`type`	character. text string indicating whether the output should be the "minimum", "average" or "mcp" distance to all records. "mcp" means the distance to the minimum convex polygon encompassing all records.

Details

Using distance to records in models may help limiting the extrapolation of the predicted area much beyond known areas.

Value

SpatRaster.

Examples

```
userpar <- par(no.readonly = TRUE)
region = gecko.data("layers")
alt = region[[3]]
localities = gecko.data("records")
par(mfrow=c(3,2))
terra::plot(alt)
points(localities)
terra::plot(distance(localities, alt))
terra::plot(distance(localities, alt, type = "average"))
par(userpar)
```

gecko.data *Example data packaged with gecko*

Description

Load data included in the package. This includes **records**, a matrix of longitude and latitude (two columns) occurrence records for Hogna maderiana (Walckenaer, 1837); **range**, a SpatRaster object, as defined by package terra, of the geographic range of Hogna maderiana (Walckenaer, 1837); **layers**, a SpatRaster object with layers representing the average annual temperature, total annual precipitation, altitude and landcover for Madeira Island (Fick & Hijmans 2017, Tuanmu & Jetz 2014); **threat**, a layer of mean fire occurence in Madeira between 2006 and 2016; and **worldborders** is a simplified version of the vector of world country borders created by Victor Cazalis.

Usage

```
gecko.data(data = NULL)
```

Arguments

`data`	character. String of one of the data names mentioned in the description, e.g.: `"gecko.records"`. If `NULL`, the example files will be listed.

Source

This function is inspired by `palmerpanguins::path_to_file()` which in turn is based on `readxl::readxl_example()`.

Examples

```
## Not run:
gecko.data()
gecko.data("range")

## End(Not run)
```

`gecko.getDir` *Read GIS directory.*

Description

Read directory where GIS files are stored.

Usage

```
gecko.getDir()
```

Details

Reads a txt file pointing to where the world GIS files are stored.

`gecko.setDir` *Setup GIS directory.*

Description

Setup directory where GIS files are stored.

Usage

```
gecko.setDir(gisPath = NULL)
```

Arguments

`gisPath`	Path to the directory where the gis files are stored.

Details

Writes a txt file in the red directory allowing the package to always access the world GIS files directory.

`gecko.worldclim` *Download worldclim files.*

Description

Download the latest version of worldclim to your gecko work directory. If you have not yet setup a work directory, it will be be setup as if running `gecko::gecko.setDir()` with `gisPath = NULL`. This is a large dataset that is prone to fail by timeout if downloaded through R. Instead of using this function you can run gecko.setDir() (if you haven't yet) and download the files at https://geodata.ucdavis.edu/climate/worldclim/2_1/base/wc2.1_30s_bio.zip or https://geodata.ucdavis.edu/climate/worldclim
Unzip their contents correspondingly to the folders "./worldclim/1 km" or "./worldclim/10 km" inside the folder returned by gecko.getDir().

Usage

```
gecko.worldclim(res)
```

Arguments

`res`	character. Specifies the resolution of environmental data used.

Details

Reads a txt file pointing to where the world GIS files are stored.

Examples

```
## Not run:
gecko.worldclim("10 km")

## End(Not run)
```

`move` *Move records to closest non-NA cell.*

Description

Identifies and moves presence records to cells with environmental values.

Usage

```
move(longlat, layers, buffer = 0)
```

Arguments

longlat	matrix. Matrix of longitude and latitude or eastness and northness (two columns in this order) of species occurrence records.
layers	SpatRaster. As defined in package terra, see `terra::rast()`.
buffer	numeric. Maximum distance in map units that a record will move. If 0 all NA records will be changed.

Details

Often records are in coastal or other areas for which no environmental data is available. This function moves such records to the closest cells with data so that no information is lost during modelling.

Value

A matrix with new coordinate values.

Examples

```
region <- terra::rast(matrix(c(rep(NA,100), rep(1,100), rep(NA,100)), ncol = 15))
presences <- cbind(runif(100, 0, 0.55), runif(100, 0, 1))
terra::plot(region)
points(presences)
presences <- move(presences, region)
terra::plot(region)
points(presences)
```

---

normalize *Normalize raster.*

Description

Normalize a raster file according to one three methods, 'standard', 'range' or 'rank'.

Usage

```
normalize(layer, method = "standard", filepath = NULL)
```

Arguments

layer	SpatRaster. Object with a single layer as defined by package terra.
method	character. Specifying `'standard'`, `'range'` or `'rank'`.
filepath	character. Optional, specifies a path to the output file.

Details

The three options, "standard" standardizes data to a mean = 0 and sd = 1, "range" standardizes to a range of 0 to 1, and "rank" similarly standardizes to a range of 0 to 1 but does so after ranking all points.

Value

A raster layer.

Examples

```
## Not run:
region = gecko.data("layers")[[1]]
ranked_region = normalize(region, method = "rank")

## End(Not run)
```

`outliers.detect` *Detect outliers in a set of geographical coordinates*

Description

This function generates pseudo-abscences from an input data.frame containing latitude and longitude coordinates by using environmental data and then uses both presences and pseudo-absences to train a SVM model used to flag possible outliers for a given species.

Usage

```
outliers.detect(
  longlat,
  training = NULL,
  hi_res = TRUE,
  crop = FALSE,
  threshold = 0.05,
  method = "all"
)
```

Arguments

`longlat`	data.frame. With two columns containing latitude and longitude, describing the locations of a species, which may contain outliers.
`training`	data.frame. With the same formatting as `longlat`, indicating only known locations where a target species occurs. Used exclusively as training data for method 'svm'.
`hi_res`	logical. Specifies if 1 KM resolution environmental data should be used. If `FALSE` 10 KM resolution data is used instead.

`crop`	logical. Indicates whether environmental data should be cropped to an extent similar to what is given in `longlat` and `training`. Useful to avoid large processing times of higher resolutions.
`threshold`	numeric. Value indicating the threshold for classifying outliers in methods `"geo"` and `"env"`. E.g.: under the default of 0.05, points that are at an average distance greater than the 95 of the average distances of all points, will be classified as outliers.
`method`	A string specifying the outlier detection method. `"geo"` calculates the euclidean distance between point coordinates and classifies as outliers those outside the 0 `"env"` performs the same calculation but instead uses the environmental data extracted from those points. `"svm"` will use the dataset given to `"longlat"` and it corresponding extracted environmental data to train a support vector machine model that then predicts outliers.

Details

Environmental data used is WorldClim and requires a long download, see `gecko::gecko.setDir()` This function is heavily based on the methods described in Liu et al. (2017). There the authors describe SVM_pdSDM, a pseudo-SDM method similar to a two-class presence only SVM that is capable of using pseudo-absence points, implemented with the ksvm function in the R package kernlab. It is suggested that, for each set of `"n"` occurence records, `"2 * n"` pseudo-absences points are generated. Whilst using it keep in mind works highlighting limitations such as such as Meynard et al. (2019). See References section.

Value

list if `method = "all"`, containing whether or not a given point was classified as `TRUE` or `FALSE` along with the confusion matrix for the training data. If `method = "geo"` or `method = "env"` a data.frame is returned.

References

Liu, C., White, M. and Newell, G. (2017) ‘Detecting outliers in species distribution data’, Journal of Biogeography, 45(1), pp. 164–176. doi:10.1111/jbi.13122.

Meynard, C.N., Kaplan, D.M. and Leroy, B. (2019) ‘Detecting outliers in species distribution data: Some caveats and clarifications on a virtual species study’, Journal of Biogeography, 46(9), pp. 2141–2144. doi:10.1111/jbi.13626.

Examples

```
## Not run:
new_occurences = gecko.data("records")
old_occurences = data.frame(X = runif(10, -17.1, -17.05), Y = runif(10, 32.73, 32.76))
outliers.detect(new_occurences, old_occurences)

## End(Not run)
```

`outliers.visualize` *Visual detection of outliers.*

Description

Draws plots of sites in geographical (longlat) and environmental (2-axis PCA) space.

Usage

```
outliers.visualize(longlat, layers)
```

Arguments

`longlat`	matrix. Matrix of longitude and latitude or eastness and northness (two columns in this order) of species occurrence records.
`layers`	SpatRaster. As defined in package terra, see `terra::rast()`. It can be any set of environmental layers thought to allow the identification of environmental outliers.

Details

Erroneous data sources or errors in transcriptions may introduce outliers that can be easily detected by looking at simple graphs of geographical or environmental space.

Value

data.frame. Contains coordinate values and distance to centroid in pca. Two plots are drawn for visual inspection. The environmental plot includes row numbers for easy identification of possible outliers.

Examples

```
localities = gecko.data("records")
region = gecko.data("layers")
outliers.visualize(localities, region[[1:3]])
```

`performance.metrics` *Performance of model predictions*

Description

Calculate the performance of a model through a comparison between predicted and observed labels. Available metrics are `accuracy`, `F1` and `TSS`.

Usage

```
performance.metrics(actual, predicted, metric)
```

Arguments

`actual`	dataframe. Same formatting as `y`, containg some sort of classification data.
`predicted`	dataframe. Same formatting as `x`, containg the predicted classifications of a model trained over the data in `x`.
`metric`	character. String specifying the metric used, one of `accuracy`, `F1` and `TSS`.

Details

The F-score or F-measure (F1) is:

$F1 = 2\frac{Precision * Recall}{Precision + Recall}$, with
$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$
$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$

Accuracy is:

$$\frac{100 * (TruePostives + TrueNegatives)}{TruePostives + TrueNegatives + FalsePositives + FalseNegatives}$$

The Pierce's skill score (PSS), Bookmaker's Informedness (BM) or True Skill Statistic (TSS) is:

$TSS = TPR + TNR - 1$,
with TPR being the True Positive Rate, positives correctly labelled as such and TNR, the True Negative Rate, the rate of negatives correctly labelled, such that:

$TPR = \frac{TruePositives}{TruePositives + FalseNegatives}$
$TNR = \frac{TrueNegatives}{TrueNegatives + FalsePositives}$
Take in consideration the fact that the F1 score is not a robust metric in datasets with class imbalances.

Value

numeric.

References

PSS: Peirce, C. S. (1884). The numerical measure of the success of predictions. Science, 4, 453–454.

Examples

```
observed = c("FALSE", "TRUE", "FALSE", "TRUE", "TRUE")
predicted = c("TRUE", "TRUE", "TRUE", "TRUE", "TRUE")
performance.metrics(observed, predicted, "TSS")
```

`reduce` *Reduce dimensionality of raster layers.*

Description

Reduce the number of layers by either performing a PCA on them or by eliminating highly correlated ones.

Usage

```
reduce(layers, method = "pca", n = NULL, thres = NULL)
```

Arguments

`layers`	SpatRaster. As defined in package terra, see `terra::rast()`.
`method`	character. Either Principal Components Analysis ("pca", default) or Pearson's correlation ("cor").
`n`	numeric. Number of layers to reduce to.
`thres`	numeric. Value for pairwise Pearson's correlation above which one of the layers (randomly selected) is eliminated.

Details

Using a large number of explanatory variables in models with few records may lead to overfitting. This function allows to avoid it as much as possible. If both n and thres are given, n has priority. If method is not recognized and layers come from read function, only landcover is reduced by using only the dominating landuse of each cell.

Value

SpatRaster.

`spectre.area` *Get SPECTRE raster segments.*

Description

Downloads SPECTRE segments according to a bounding box selection.

Usage

```
spectre.area(
  index,
  ext = c(-180, 180, -60, 90),
  normalize = FALSE,
  filepath = NULL
)
```

Arguments

`index`	numeric. A vector of integers specifying the layers. Refer to the list.
`ext`	numeric or SpatExtent. A vector of `xmin`, `xmax`, `ymin`, `ymax` or a `terra` spatial extent object (See `terra::ext()`). If no input is given, an extent of `xmin` = `-180`, `xmax` = `180`, `ymin` = `-60`, `ymax` = `90` is selected.
`normalize`	character or logical. Either logical on whether data should be normalized for the given interval or a character specifying a type of normalization. Type default to "standard". Check `gecko::normalize()` for more info.
`filepath`	character. An optional user defined path for the final output. If `NULL`, requested files are left in the current temp directory.

Value

SpatRaster.

Examples

```
## Not run:
regional_threats = spectre.area(3, terra::ext(-17.3,-16.6,32.6,32.9), normalize = FALSE)
terra::plot(regional_threats[[1]], main = "Human Density")

## End(Not run)
```

`spectre.citations` *Get in text citations for SPECTRE layers*

Description

Generate in-text citations for a selection of SPECTRE layers.

Usage

```
spectre.citations(index)
```

Arguments

`index`	numeric. A vector of integers specifying the layers. Refer to the Details section.

Details

The current layers in SPECTRE are:

1. **MINING_AREA**. Mining density based on the number of known mining properties (pre-operational, operational, and closed) in a 50-cell radius (1x1 km cells).
2. **HAZARD_POTENTIAL**. Number of significant hazards (earthquakes, volcanoes, landslides, floods, drought, cyclones) potentially affecting cells based on hazard frequency data.

3. **HUMAN_DENSITY** Continuous metric of population density.
4. **BUILT_AREA** Percentage metric indicating the built-up presence.
5. **ROAD_DENSITY**. Continuous metric of road density.
6. **FOOTPRINT_PERC**. Percentage metric indicating anthropogenic impacts on the environment.
7. **IMPACT_AREA**. Classification of land into very low impact areas (1), low impact areas (2) and non-low impact areas (3).
8. **MODIF_AREA**. Continuous 0-1 metric that reflects the proportion of a landscape that has been modified.
9. **HUMAN_BIOMES**. Classification of land cover into different anthropogenic biomes of differing pressure such as dense settlements, villages and cropland.
10. **FIRE_OCCUR**. Continuous metric of mean fire occurrence during the years of 2006 and 2016.
11. **CROP_PERC_UNI**. Percentage metric indicating the proportion of cropland in each cell.
12. **CROP_PERC_IIASA**. Percentage metric indicating the proportion of cropland in each cell.
13. **LIVESTOCK_MASS**. Estimated total amount of livestock wet biomass based on global livestock head counts.
14. **FOREST_LOSS_PERC**. Continuous -100 to 100 metric of forest tree cover loss between 2007 and 2017.
15. **FOREST_TREND**. Classification metric of 0 (no loss) or a discrete value from 1 to 17, representing loss (a stand-replacement disturbance or change from a forest to non-forest state) detected primarily in the year 2001-2019, respectively.
16. **NPPCARBON_GRAM**. Quantity of carbon needed to derive food and fiber products (HANPP).
17. **NPPCARBON_PERC**. HANNP as a percentage of local Net Primary Productivity.
18. **LIGHT_MCDM2**. Continuous simulated zenith radiance data.
19. **FERTILIZER_LGHA**. Continuous metric of kilograms of fertilizer used per hectare.
20. **TEMP_TRENDS**. Continuous metric of temperature trends, based on the linear regression coefficients of mean monthly temperature for the years of 1950 to 2019.
21. **TEMP_SIGNIF**. Continuous metric of temperature trend significance, the temperature trends divided by its standard error.
22. **CLIM_EXTREME**. Continuous metric calculated as whatever is the largest of the absolute of the trend coefficients of the months with the lowest or highest mean temperatures.
23. **CLIM_VELOCITY**. Continuous metric of the velocity of climate change, the ratio between TEMP_TRENDS and a local spatial gradient in mean temperature calculated as the slope of a plane fitted to the values of a 3x3 cell neighbourhood centered on each pixel.
24. **ARIDITY_TREND**. Continuous metric of aridity trends, based on the linear regression coefficients of aridity for the years of 1990 to 2019, i.e: MPET/(MPRE+1).

Value

list. Contains two elements, both characters: the first a single character containing the in-text citations, the second a character of length `x` with the bibliographic citations.

Examples

```
sources = c(2,3)
out = spectre.citations(sources)
```

`spectre.points` *Get SPECTRE data from points.*

Description

Downloads SPECTRE layer data according to a selection of points.

Usage

```
spectre.points(index, points)
```

Arguments

`index`	numeric. A vector of integers specifying the layers. Refer to the documentation of `gecko::spectre.citations()` for a list of available layers.
`points`	data.frame or matrix. Containing point data coordinates, organized in longitude, latitude (longlat).

Value

data.frame or matrix. Contains both the points given as well as their respective values for each layer specified.

Examples

```
## Not run:
localities = gecko.data("records")
local_threats = spectre.points(c(2,3), localities)

## End(Not run)
```

`spectre.template` *Download the SPECTRE template.*

Description

Download the raster template for SPECTRE layers to your gecko work directory. If you have not yet setup a work directory, it will be be setup as if running `gecko::gecko.setDir()` with `gisPath = NULL`. This is a large dataset that is prone to fail by timeout if downloaded through R. Instead of using this function you can run gecko.setDir() (if you haven't yet) and download the file at https://github.com/VascoBranco/spectre.content/raw/main/spectre.template.zip. Unzip its contents to a folder "./spectretemplate" inside the folder returned by gecko.getDir().

Usage

```
spectre.template()
```

Details

Reads a txt file pointing to where the world GIS files are stored.

Examples

```
## Not run:
spectre.template()

## End(Not run)
```

`spectrify` *Make a raster layer SPECTRE compatible*

Description

Transform a given raster object to the resolution, datum, projection and extent used in SPECTRE.

Usage

```
spectrify(layers, continuous = TRUE, filepath = NULL)
```

Arguments

`layers`	SpatRaster. A raster object that you would like to be SPECTRE compatible.
`continuous`	logical. Whether the data present in `layers` is continuous. If `TRUE` bilinear interpolation will be used in the case of resampling and reprojection. if `FALSE` nearest neighbour will be used instead. See `terra::resample()` for more information on interpolation methods.
`filepath`	character. Optional file path to where the final raster layer should be saved, in the format "folder/file.tif". If `filepath` is `NULL` your layer will be saved to your current working directory.

Value

SpatRaster.

Examples

```
## Not run:
# For the sake of demonstration we will transform our raster layer "range".
distribution = gecko.data("range")
standard_dist = spectrify(distribution)
terra::plot(standard_dist)

## End(Not run)
```

splitDataset *Split a dataset for model training*

Description

Split a dataset for model training while keeping class representativity.

Usage

```
splitDataset(data, proportion)
```

Arguments

data	dataframe. Containg some sort of classification data. The last column must contain the label data.
proportion	numeric. A value between 0 a 1 determining the proportion of the dataset split between training and testing.

Value

list. First element is the train data, second element is the test data.

Examples

```
# Binary label case
my_data = data.frame(X = runif(20), Y = runif(20), Z = runif(20), Label =
c(rep("presence", 10), rep("outlier", 10)) )
splitDataset(my_data, 0.8)

# Multi label case
my_data = data.frame(X = runif(60), Y = runif(60), Z = runif(60), Label =
c(rep("A", 20), rep("B", 30), rep("C", 10)) )
splitDataset(my_data, 0.8)
```

stats *Get a short summary of a given raster segment.*

Description

Return a set of descriptive statistics of the given layer, either a specific one (minimum, q1, median, q3, maximum, median absolute deviation (mad), mean, standard deviation (sd)) or all of them.

Usage

```
stats(layer, plot = FALSE)
```

Arguments

`layer`	SpatRaster. Raster object, as defined by package terra, with a single layer.
`plot`	logical. If TRUE, a histogram of raster values is drawn.

Value

data.frame. If plot is TRUE, also outputs a histogram of the layer.

Examples

```
region = gecko.data("layers")
stats(region[[1]])
```

`thin` *Spatial thinning of occurrence records.*

Description

Thinning of records with minimum distances either absolute or relative to the species range.

Usage

```
thin(longlat, distance = 0.01, relative = TRUE, runs = 100)
```

Arguments

`longlat`	matrix. Matrix of longitude and latitude or eastness and northness (two columns in this order) of species occurrence records.
`distance`	numeric. Distance either in relative terms (proportion of maximum distance between any two records) or in raster units.
`relative`	logical. If `TRUE`, represents the proportion of maximum distance between any two records. If `FALSE`, is in raster units.
`runs`	numeric. Number of runs

Details

Clumped distribution records due to ease of accessibility of sites, emphasis of sampling on certain areas in the past, etc. may bias species distribution models. The algorithm used here eliminates records closer than a given distance to any other record. The choice of records to eliminate is random, so a number of runs are made and the one keeping more of the original records is chosen.

Value

A matrix of species occurrence records separated by at least the given distance.

Examples

```
userpar <- par(no.readonly = TRUE)
occ_points <- matrix(sample(100), ncol = 2)
par(mfrow=c(1,2))
graphics::plot(occ_points)
occ_points <- thin(occ_points, 0.1)
graphics::plot(occ_points)
par(userpar)
```

Index