Package 'voluModel'

May 23, 2026

Title Modeling Species Distributions in Three Dimensions

Version 0.2.4

Maintainer Hannah L. Owens <hannah.owens@gmail.com>

Description Facilitates modeling species' ecological niches and geographic distributions based on occurrences and environments that have a vertical as well as horizontal component, and projecting models into three-dimensional geographic space. Working in three dimensions is useful in an aquatic context when the organisms one wishes to model can be found across a wide range of depths in the water column. The package also contains functions to automatically generate marine training model training regions using machine learning, and interpolate and smooth patchily sampled environmental rasters using thin plate splines.
Davis Rabosky AR, Cox CL, Rabosky DL, Title PO, Holmes IA, Feldman A, McGuire JA (2016) <doi:10.1038/ncomms11484>.
Nychka D, Furrer R, Paige J, Sain S (2021) <doi:10.5065/D6W957CT>.
Pateiro-Lopez B, Rodriguez-Casal A (2022) <https://CRAN.R-project.org/package=alphahull>.

License GPL-3

URL https://hannahlowens.github.io/voluModel/, https://github.com/hannahlowens/voluModel

BugReports https://github.com/hannahlowens/voluModel/issues

Encoding UTF-8

Depends R (>= 4.0.0)

Imports dplyr, fields, ggplot2, ggtext, grDevices, methods, metR, predicts, rangeBuilder (>= 2.0), rnaturalearth, terra, viridisLite, sf

Suggests testthat (>= 3.0.0), nlme, knitr, covr, gridExtra, lattice, rJava, rmarkdown, rnaturalearthdata, tibble

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Hannah L. Owens [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0003-0071-1745>),
Emmaline Sheahan [aut] (ORCID: <https://orcid.org/0000-0003-3358-9758>),
Carsten Rahbek [aut] (ORCID: <https://orcid.org/0000-0003-4585-0300>)

Repository CRAN

Date/Publication 2026-05-23 11:00:02 UTC

Contents

`bottomRaster` *Bottom raster generation*

Description

Samples deepest depth values from a `SpatVector` data frame and generates a `SpatRaster`.

Usage

```
bottomRaster(rawPointData)
```

Arguments

rawPointData	A SpatVector object from which bottom variables will be sampled. See Details for more about format.

Details

rawPointData is a SpatVector object that contains measurements of a single environmental variable (e.g. salinity, temperature, etc.) with x, y, and z coordinates. The measurements in the data.frame should be organized so that each column is a depth slice, increasing in depth from left to right. The function was designed around the oceanographic data shapefiles supplied by the World Ocean Atlas (https://www.ncei.noaa.gov/access/world-ocean-atlas-2018/). The function selects the "deepest" (rightmost) measurement at each x, y coordinate pair that contains data. These measurements are then rasterized at the resolution and extent of the x,y coordinates, under the assumption that x and y intervals are equal and represent the center of a cell.

Value

A SpatRaster designed to approximate sea bottom measurements for modeling species' distributions and/or niches.

Examples

```
library(terra)

# Create point grid
coords <- data.frame(x = rep(seq(1:5), times = 5),
                     y = unlist(lapply(1:5, FUN = function(x) {
                       rep(x, times = 5)})))

# Create data and add NAs to simulate uneven bottom depths
dd <- data.frame(SURFACE = 1:25,
                 d5M = 6:30,
                 d10M = 11:35,
                 d25M = 16:40)
dd$d25M[c(1:5, 18:25)] <- NA
dd$d10M[c(3:5, 21:23)] <- NA
dd$d5M[c(4, 22)] <- NA

dd[,c("x","y")] <- coords

# Create SpatialPointsDataFrame
sp <- vect(dd, geom = c("x", "y"))

# Here's the function
result <- bottomRaster(rawPointData = sp)
plot(result)
```

centerPointRasterTemplate *Center Point Raster Template*

Description

Creates a SpatRaster template from a SpatVector point object in which the raster cells are centered on the vector points.

Usage

```
centerPointRasterTemplate(rawPointData)
```

Arguments

rawPointData	A SpatVector object with points that will represent the center of each cell in the output template.

Details

rawPointData is a SpatVector object that contains x and y coordinates.

Value

An empty SpatRaster designed to serve as a template for rasterizing SpatVector objects.

See Also

rasterize

Examples

```
library(terra)

# Create point grid
coords <- data.frame(x = rep(seq(1:5), times = 5),
                     y = unlist(lapply(1:5, FUN = function(x) {
                       rep(x, times = 5)})))

# Create data and add NAs to simulate uneven bottom depths
dd <- data.frame(SURFACE = 1:25,
                 d5M = 6:30,
                 d10M = 11:35,
                 d25M = 16:40)
dd$d25M[c(1:5, 18:25)] <- NA
dd$d10M[c(3:5, 21:23)] <- NA
dd$d5M[c(4, 22)] <- NA

dd[,c("x","y")] <- coords
```

```
# Create SpatialPointsDataFrame
sp <- vect(dd, geom = c("x", "y"))

# Here's the function
template <- centerPointRasterTemplate(rawPointData = sp)
class(template)
```

---

cleanDepth *Clean and flag coordinates based on intersection with a land polygon, intersection with a bathymetry layer, and/or by a given depth range*

Description

function to remove or flag coordinates on land, coordinates which are deeper than the known bathymetry at a given xy location, and/or coordinates outside of a specified depth range when cleaning coordinates for species in marine environments

Usage

```
cleanDepth(
  occs,
  bathy,
  land_poly,
  depth_range,
  flag = FALSE,
  bottom_correct = FALSE
)
```

Arguments

occs	dataframe containing columns named "longitude", "latitude", and "depth," where depth values are positive
bathy	a spatRaster of bathymetry in the same units as the occurrence depth, where values are negative
land_poly	optional polygon of continents in same projection as occurrence data
depth_range	optional range of depth values in the form of c(upper, lower) in which all points should fall between. depth value should be positive, as in the case of the occs
flag	default is F, if T, points are not removed, but flagged if they intersect, and a column is added to the output indicating what points have been flagged
bottom_correct	default is F, but if T, points are not removed, and a column is added to the output displaying the bathymetry value at the flagged point for corrective purposes

Details

assumes depths are listed as positive in the occs and depth_range, while the bathymetry layer lists them as negative when below the sea surface

Value

an object of class dataframe containing cleaned or flagged occurrences

Examples

```
library(terra)
library(dplyr)

# Create sample bathymetry
r1 <- rast(ncol=10, nrow=10)
values(r1) <- c(-100:0)

# Create test occurrences
set.seed(0)
longitude <- sample(ext(r1)[1]:ext(r1)[2], size = 10, replace = FALSE)
set.seed(0)
latitude <- sample(ext(r1)[3]:ext(r1)[4], size = 10, replace = FALSE)
set.seed(0)
depth <- sample(1:100, size = 10, replace = TRUE)
occs <- data.frame(longitude, latitude, depth)

#Heres the function
result <- cleanDepth(bathy = r1, occs = occs)
```

---

depthMatch *Match the depth values given in an occurrence dataset to the depth slice values of a user provided spatRaster stack*

Description

assigns coordinates to depth slices given a user provided raster template. if any points are deeper than the lowest depth slice, they are assigned to the lowest depth slice.

Usage

```
depthMatch(occs, rasterTemplate)
```

Arguments

occs	dataframe of occurrence records with colnames "longitude," "latitude," and "depth." Depth values should be positive.
rasterTemplate	a spatRaster stack where the layer names correspond to the depth value, as a positive number

Details

rasterTemplate names and values in the depth column of occs should be positive. Points lower than the deepest depth slice are assigned to the deepest depth slice

Value

object of class dataframe, the same occs fed into the function but with the depth values adjusted to match the spatRaster stack depth slices

Examples

```
library(terra)
# Create test raster brick
r1 <- rast(ncol = 100, nrow = 100)
r2 <- rast(ncol = 100, nrow = 100)
r3 <- rast(ncol = 100, nrow = 100)
rbrick <- c(r1, r2, r3)
names(rbrick) <- c(1, 40, 90)

# Create test occurrences
set.seed(0)
longitude <- sample(ext(r1)[1]:ext(r1)[2], size = 10, replace = FALSE)
set.seed(0)
latitude <- sample(ext(r1)[3]:ext(r1)[4], size = 10, replace = FALSE)
set.seed(0)
depth <- sample(1:100, size = 10, replace = TRUE)
occs <- data.frame(longitude, latitude, depth)

# Here's the function
result <- depthMatch(occs = occs, rasterTemplate = rbrick)
```

---

diversityStack *Diversity stack*

Description

Takes list of rasters of species distributions (interpreted as 1 = presence, 0 = absence), which do not have to have the same extents, and stack them to create an estimate of species richness that matches the extent and resolution of a template.

Usage

```
diversityStack(rasterList, template)
```

Arguments

`rasterList`	A `list` of `SpatRaster` objects, which are interpreted as species distributions (1 = presence, 0 = absence).
`template`	A `SpatRaster` with the desired extent

Value

A `SpatRaster`

Examples

```
library(terra)
rast1 <- rast(ncol=10, nrow=10)
values(rast1) <- rep(0:1, 50)

rast2 <- rast(ncol=10, nrow=10)
values(rast2) <- c(rep(0, 50), rep(1,50))

rastList <- list(rast1, rast2)
result <- diversityStack(rasterList = rastList,
                         template = rast2)
result
plot(result)
```

`downsample` *Occurrence downsampling*

Description

Reduces number of occurrences to resolution of input raster

Usage

```
downsample(occs, rasterTemplate, verbose = TRUE)
```

Arguments

`occs`	A `data.frame` with at least two columns named "longitude" and "latitude" or that can be coerced into this format.
`rasterTemplate`	A `SpatRaster` object to serve as a template for the resolution at which `occs` should be downsampled.
`verbose`	`logical`. Switching to `FALSE` mutes message describing which columns in `occs` are interpreted as x and y coordinates.

Value

A `data.frame` with two columns named "longitude" and "latitude" or with names that were used when coercing input data into this format.

Examples

```
library(terra)
# Create sample raster
r <- rast(ncol=10, nrow=10)
values(r) <- 1:100

# Create test occurrences
```

```
set.seed(0)
longitude <- sample(ext(r)[1]:ext(r)[2],
                    size = 10, replace = FALSE)
set.seed(0)
latitude <- sample(ext(r)[3]:ext(r)[4],
                   size = 10, replace = FALSE)
occurrences <- as.data.frame(cbind(longitude,latitude))

# Here's the function
result <- downsample(occs = occurrences, rasterTemplate = r)
```

`env_stack_transform` *Coverts a list of 'SpatRaster' stacks where the elements are environmental variables and the layers are depths to a list of SpatRaster stacks where the elements are depths and the layers are environmental variables*

Description

A user will likely already have environmental data organized into a list of 'spatRaster' stacks where each element is a different environmental variable and each layer corresponds to a depth slice for background sampling and extracting. However, the maxent_3D function requires the data be input as a list of 'spatRaster' stacks where each element corresponds to a depth slice and each layer is an environmental variable for the purpose of projecting the model back into geographic space, and this function has been developed for ease of conversion.

Usage

```
env_stack_transform(envs_all, envs_names)
```

Arguments

`envs_all`	a list of 'spatRaster' stacks where each element is a different environmental variable and each layer is a depth slice.
`envs_names`	A 'character' vector of the names of the environmental variables to be applied to the layers of the resulting list of 'spatRaster' stacks

Details

The extents of each environmental variable layer should match on a depth slice by depth slice basis.

Value

A list of 'spatRaster' stacks where each list element is a depth slice and each layer is an environmental variable.

Examples

```
library(terra)

# creating a list of spatRaster stacks where each element is an environmental variable
# and each layer is a depth

env1_d1 <- rast(ncol = 50, nrow = 50)
values(env1_d1) <- sample(c(1:100), size = 2500, replace = TRUE)
env2_d1 <- rast(ncol = 50, nrow = 50)
values(env2_d1) <- sample(c(1:100), size = 2500, replace = TRUE)
env1_d2 <- rast(ncol = 50, nrow = 50)
values(env1_d2) <- sample(c(1:100), size = 2500, replace = TRUE)
env2_d2 <- rast(ncol = 50, nrow = 50)
values(env2_d2) <- sample(c(1:100), size = 2500, replace = TRUE)

env1 <- c(env1_d1, env1_d2)
env2 <- c(env2_d1, env2_d2)
envs <- list(env1, env2)
envnames <- c("env1", "env2")

# Here's the function
result <- env_stack_transform(envs_all = envs, envs_names = envnames)
```

---

interpolateRaster *Interpolate patchily sampled rasters*

Description

Uses thin plate spline regression from `fields` package to interpolate missing two-dimensional raster values.

Usage

```
interpolateRaster(inputRaster, fast = FALSE, ...)
```

Arguments

`inputRaster`	An object of class `SpatRaster`
`fast`	A logical operator. Setting to `TRUE` triggers use of `fastTps` instead of `Tps`.
`...`	For any additional arguments passed to `Tps` or `fastTps`

Details

Missing data values from original raster are replaced with interpolated values. User has the option of choosing `fastTps` to speed calculation, but be advised that this is only an approximation of a true thin plate spline.

Value

An object of class raster

See Also

Tps, fastTps

Examples

```
library(terra)
library(fields)
# Create sample raster
r <- rast(ncol=50, nrow=50)
values(r) <- 1:2500

# Introduce a "hole"
values(r)[c(117:127, 167:177, 500:550)] <- NA
plot(r)

# Patch hole with interpolateRaster
interpolatedRaster <- interpolateRaster(r)
plot(interpolatedRaster)
fastInterp <- interpolateRaster(r, fast = TRUE, aRange = 3.0)
plot(fastInterp)
```

---

marineBackground *Marine background shapefile generation*

Description

Automatically generates background shapefiles for sampling pseudoabsences and/or background points for niche modeling or species distribution modeling. Delineating background sampling regions can be one of the trickiest parts of generating a good model. Automatically generated background shapefiles should be inspected carefully prior to model use.

Useful references, among others:

- Barve N, Barve V, Jiménez-Valverde A, Lira-Noriega A, Maher SP, Peterson AT, Soberón J, Villalobos F. 2011. The crucial role of the accessible area in ecological niche modeling and species distribution modeling. *Ecological modelling* 222:1810-9.
- Merow, C, Smith MJ, Silander JA. 2013. A practical guide to MaxEnt for modeling species' distributions: what it does, and why inputs and settings matter." *Ecography* 36: 1058-69.
- Murphy SJ. 2021. Sampling units derived from geopolitical boundaries bias biodiversity analyses. *Global Ecology and Biogeography* 30: 1876-88.

Usage

```
marineBackground(occs, clipToOcean = TRUE, verbose = TRUE, ...)
```

Arguments

occs	A data.frame with at least two columns named "longitude" and "latitude" or that can be coerced into this format.
clipToOcean	logical. Clips shapefile to oceans where species occurs. Useful in cases where buffers jump over narrow peninsulas (e.g. Isthmus of Panama). Can be quite artificial at ocean boundaries.
verbose	logical. Switching to FALSE mutes message describing which columns in occs are interpreted as x and y coordinates.
...	Additional optional arguments to pass to getDynamicAlphaHull.

Details

The meat of this function is a special-case wrapper around getDynamicAlphaHull() from the rangeBuilder package. The function documented here is especially useful in cases where one wants to automatically generate training regions that overlap the international date line. Regions that exceed the line are cut and pasted into the appropriate hemisphere instead of being deleted.

If the argument buff is not supplied, a buffer is calculated by taking the mean between the 10th and 90th percentile of horizontal distances between occurrence points.

If getDynamicAlphaHull() cannot satisfy the provided conditions, the occurrences are buffered and then a minimum convex hull is drawn around the buffer polygons.

Value

A SpatVector

See Also

getDynamicAlphaHull

Examples

```
library(terra)
# Create sample raster
r <- rast(ncol=10, nrow=10)
values(r) <- 1:100

# Create test occurrences
set.seed(0)
longitude <- sample(-50:50,
                    size = 20, replace = FALSE)
set.seed(0)
latitude <- sample(-30:30,
                   size = 20, replace = FALSE)
occurrences <- as.data.frame(cbind(longitude,latitude))

# Here's the function
result <- marineBackground(occs = occurrences, buff = 100000,
                           fraction = .9, partCount = 2, clipToOcean = FALSE)
```

maxent_3D *Create 3D Ecological Niche Models with Maxent*

Description

Uses MaxEnt from `predicts` package to test multiple models with different feature class combinations, regularization multipliers, and a user supplied partitioning scheme for training and testing. The function outputs model objects, model results, as well as prediction SpatRasters of each model projected back onto geographic space using the supplied environmental SpatRaster stacks. Note: you will need to install `rJava` to successfully run this function.

Usage

```
maxent_3D(
  maxent_df,
  wanted_fc,
  wanted_rm,
  wanted_partition = NULL,
  projection_layers,
  occs,
  depth_list
)
```

Arguments

`maxent_df`	'data.frame' where the first column is a vector of presences named "p" containing 1's and 0's. Each row represents a cell in the spatRaster volume with an x, y, z coordinate, and 1's are presences while 0's are absences, or background points. Other columns are environmental variable values extracted at the occurrence and background points, and should have the same names as the names of the environmental layers in the projection_layers list of SpatRaster stacks.
`wanted_fc`	a character vector giving what feature class combinations should be tried. "L" refers to linear, "Q" refers to quadratic, "H" refers to hinge, and "P" refers to product. Should be in the format c("L", "Q", "LQ") etc.
`wanted_rm`	regularization multipliers to be tried in format c(1:4)
`wanted_partition`	optional, should be the output of 'partition_3D'. if no partition is supplied, all points will be used for training
`projection_layers`	`list` of SpatRaster stacks by depth for predictions, must all be cropped to the depth slice with the largest extent and masked to the accessible area of the matching depth slice. Each element is a depth slice, and each stack should contain all of the environmental variables used in the model with the same names as that used in the model.

occs	data.frame of longitude, latitude, and depth occurrences with columns named "longitude", "latitude", and "depth".
depth_list	vector of depths corresponding to depth slices of list elements of projection_layers. Should be positive and go from shallowest depth to deepest depth

Details

The names of the projection_layers should be the same as the column names of the environmental variables in maxent_df. The number of models output will be the number of feature classes multiplied by the number of regularization multipliers. For example, a wanted_fc of c("L", "Q", "P") and a wanted_rm of c(1:3) will output 9 total models.

Value

An object of class list with four components:

$models, a list containing each model object produced.

$results, a data.frame where each row is a model corresponding to the list element in $models and $predictions. If there was no partition supplied for training and testing, each column will report the feature class, regularization multiplier, AUC, total coefficients, nonzero coefficients, AICc, and delta AICc. if a partition is used, it will report the average of these statistics across all partitions.

$predictions, a list of spatRaster stacks where each list element is a model corresponding to the rows of $results and elements of $models projected onto the supplied projection_layers. Each layer in the stack is a depth slice.

$partition_results, a list object of the same length as the number of partition groups containing a data.frame of results for each model for each partition. Only produced if a partition is supplied.

Examples

```
library(dplyr)
library(predicts)
library(terra)

# creating list of spatraster stacks where each element is a depth slice
r1_d1 <- rast(ncol = 100, nrow = 100)
set.seed(0)
values(r1_d1) <- as.numeric(sample(c(1:100), size = 1000, replace = TRUE))
r2_d1 <- rast(ncol = 100, nrow = 100)
set.seed(0)
values(r2_d1) <- as.numeric(sample(c(1:1000), size = 1000, replace = FALSE))
r1_d2 <- r1_d1
values(r1_d2) <- as.numeric(values(r1_d1)+10)
r2_d2 <- r2_d1
values(r2_d2) <- as.numeric(values(r2_d1)+10)
d1 <- c(r1_d1, r2_d1)
names(d1) <- c("valsr1", "valsr2")
d2 <- c(r1_d2, r2_d2)
names(d2) <- c("valsr1", "valsr2")
envlist <- list(d1, d2)
```

```
# creating occs and bgs
set.seed(0)
occs <- sample(c(1:nrow(crds(envlist[[1]][[1]]))), size = 50, replace = FALSE)
bgs <- sample(c(1:nrow(crds(envlist[[1]][[1]]))), size = 500, replace = FALSE)

occ_indices <- sample(c(1:nrow(crds(envlist[[1]][[1]]))), size = 50, replace = FALSE)
bg_indices  <- sample(c(1:nrow(crds(envlist[[1]][[1]]))), size = 500, replace = FALSE)

occs_d1 <- crds(envlist[[1]][[1]])[occ_indices[1:25],]
occs_d2 <- crds(envlist[[2]][[1]])[occ_indices[26:50],]
bg_d1   <- crds(envlist[[1]][[1]])[bg_indices[1:250],]
bg_d2   <- crds(envlist[[2]][[1]])[bg_indices[251:500],]

# extracting at occs and bgs
occ_valsr1_d1 <- extract(envlist[[1]][[1]], occs_d1)
occ_valsr1_d2 <- extract(envlist[[2]][[1]], occs_d2)
occ_valsr2_d1 <- extract(envlist[[1]][[2]], occs_d1)
occ_valsr2_d2 <- extract(envlist[[2]][[2]], occs_d2)

occ_valsr1 <- rbind(occ_valsr1_d1, occ_valsr1_d2)
occ_valsr2 <- rbind(occ_valsr2_d1, occ_valsr2_d2)

bg_valsr1_d1 <- extract(envlist[[1]][[1]], bg_d1)
bg_valsr1_d2 <- extract(envlist[[2]][[1]], bg_d2)
bg_valsr2_d1 <- extract(envlist[[1]][[2]], bg_d1)
bg_valsr2_d2 <- extract(envlist[[2]][[2]], bg_d2)

bg_valsr1 <- rbind(bg_valsr1_d1, bg_valsr1_d2)
bg_valsr2 <- rbind(bg_valsr2_d1, bg_valsr2_d2)

valsr1 <- rbind(occ_valsr1, bg_valsr1)
valsr2 <- rbind(occ_valsr2, bg_valsr2)

p1 <- rep(1, times = 50)
p0 <- rep(0, times = 500)
p <- c(p1, p0)

maxdf <- data.frame(p, valsr1, valsr2)

coords <- rbind(occs_d1, occs_d2)
colnames(coords) <- c("longitude", "latitude")
depth_vector     <- c(rep(1, times = 25), rep(2, times = 25))

# Use data.frame instead of cbind so $depth is completely valid inside maxent_3D
occs_dataframe <- data.frame(coords, depth = depth_vector)


# Pass the clean data.frame to the function
if(requireNamespace("rJava", quietly = TRUE)){
  result <- maxent_3D(maxent_df = maxdf, wanted_fc = c("L", "Q"),
                      wanted_rm = c(1:2), projection_layers = envlist,
                      occs = occs_dataframe, depth_list = c(1,2))
}
```

MESS3D *Calculate MESS*

Description

Calculates multivariate environmental similarity surface based on model calibration and projection data

Usage

```
MESS3D(calibration, projection)
```

Arguments

`calibration`	A `data.frame` of environmental variables used to calibrate an ecological niche model, one row for measurements from each voxel included in the data used to calibrate the model. Columns with names not corresponding to `projection list` items are ignored.
`projection`	A named `list` of `SpatRaster` objects for projection; names correspond to `calibration` column names. Each `SpatRaster` should have the same number of layers, corresponding to vertical depth slices.

Details

`MESS3D` is a wrapper around `MESS` from the `modEvA` package. It calculates MESS for each depth layer. Negative values indicate areas of extrapolation which should be interpreted with caution (see Elith *et al*, 2010 in *MEE*).

Value

A `SpatRaster` vector with MESS scores for each voxel; layer names correspond to layer names of first `SpatRaster` vector in `projection list`.

Note

The calibration dataset should include both presences and background/pseudoabsence points used to calibrate an ecological niche model.

References

Elith J, Kearney M, and Phillips S. 2010. The art of modelling range-shifting species. *Methods in Ecology and Evolution*, 1, 330-342.

See Also

`mess`

Examples

```
library(terra)
library(dplyr)

# Create sample rasterBricks
r1 <- rast(ncol=10, nrow=10)
values(r1) <- 1:100
r2 <- rast(ncol=10, nrow=10)
values(r2) <- c(rep(20, times = 50), rep(60, times = 50))
r3 <- rast(ncol=10, nrow=10)
values(r3) <- 8
envBrick1 <- c(r1, r2, r3)
names(envBrick1) <- c(0, 10, 30)

r1 <- rast(ncol=10, nrow=10)
values(r1) <- 100:1
r2 <- rast(ncol=10, nrow=10)
values(r2) <- c(rep(10, times = 50), rep(20, times = 50))
r3 <- rast(ncol=10, nrow=10)
values(r3) <- c(rep(c(10,20,30,25), times = 25))
envBrick2 <- c(r1, r2, r3)
names(envBrick2) <- c(0, 10, 30)

rastList <- list("temperature" = envBrick1, "salinity" = envBrick2)

# Create test reference set
set.seed(0)
longitude <- sample(ext(envBrick1)[1]:ext(envBrick1)[2],
                    size = 10, replace = FALSE)
set.seed(0)
latitude <- sample(ext(envBrick1)[3]:ext(envBrick1)[4],
                   size = 10, replace = FALSE)
set.seed(0)
depth <- sample(0:35, size = 10, replace = TRUE)
occurrences <- as.data.frame(cbind(longitude,latitude,depth))

# Sample data at occurrences to characterize calibration region
cal_temp <- xyzSample(occurrences, rastList$temperature)
cal_sal  <- xyzSample(occurrences, rastList$salinity)

calibration <- data.frame(
  temperature = cal_temp,
  salinity    = cal_sal)

# Run the function
messStack <- MESS3D(calibration = calibration, projection = rastList)
plot(messStack)
```

---

mSampling2D *2D background sampling*

Description

Samples in 2D at resolution of raster

Usage

```
mSampling2D(occs, rasterTemplate, mShp, verbose = TRUE)
```

Arguments

occs	A dataframe with at least two columns named "longitude" and "latitude", or that can be coerced into this format.
rasterTemplate	A SpatRaster object to serve as a template for generating background sampling coordinates.
mShp	A shapefile defining the area from which background points should be sampled.
verbose	logical. Switching to FALSE mutes message describing which columns in occs are interpreted as x and y coordinates.

Details

This function is designed to sample background points for distributional modeling in two dimensions. The returned data.frame contains all points from across the designated background. It is up to the user to determine how to appropriately sample from those background points.

Value

A data.frame with 2D coordinates of points for background sampling.

Examples

```
library(terra)

# Create sample raster
r <- rast(ncol=10, nrow=10)
values(r) <- 1:100

# Create test occurrences
set.seed(0)
longitude <- sample(ext(r)[1]:ext(r)[2],
                    size = 10, replace = FALSE)
set.seed(0)
latitude <- sample(ext(r)[3]:ext(r)[4],
                   size = 10, replace = FALSE)
occurrences <- data.frame(longitude,latitude)

# Generate background sampling buffer
buffPts <- vect(occurrences,
                c("longitude", "latitude"))
crs(buffPts) <- crs(r)
mShp <- aggregate(buffer(buffPts, width = 1000000))
```

```
# Here's the function
result <- mSampling2D(occs = occurrences, rasterTemplate = r, mShp = mShp)
```

`mSampling3D` *3D background sampling*

Description

Samples XYZ coordinates from a shapefile from maximum to minimum occurrence depth at XYZ resolution of envBrick.

Usage

```
mSampling3D(occs, envBrick, mShp, depthLimit = "all", verbose = TRUE)
```

Arguments

`occs`	A `data.frame` with at least three columns named "longitude", "latitude", and "depth", or that can be coerced into this format.
`envBrick`	A `SpatRaster` vector object to serve as a template for generating background sampling coordinates.
`mShp`	A shapefile defining the area from which background points should be sampled.
`depthLimit`	An argument controlling the depth extent of sampling. Refer to `Details` for more information.
`verbose`	`logical`. Switching to `FALSE` mutes message describing which columns in `occs` are interpreted as x, y, and z coordinates.

Details

This function is designed to sample background points for distributional modeling in three dimensions. If a voxel (3D pixel) in the `SpatRaster` vector intersects with an occurrence from `occs`, it is removed. Note that this function returns points representing every voxel in the background area within the specified depth range. It is up to the user to downsample from these data as necessary, depending on the model type being used.

`depthLimit` argument options:

- `occs` Samples background from the full depth extent of `occs`.
- `all` Samples background from the full depth extent of `envBrick`.
- A `vector` of length 2 with maximum and minimum depth values from which to sample.

Value

A `data.frame` with 3D coordinates of points for background sampling.

Examples

```
library(terra)

# Create test raster
r1 <- rast(ncol=10, nrow=10)
values(r1) <- 1:100
r2 <- rast(ncol=10, nrow=10)
values(r2) <- c(rep(20, times = 50), rep(60, times = 50))
r3 <- rast(ncol=10, nrow=10)
values(r3) <- 8
envBrick <- c(r1, r2, r3)
names(envBrick) <- c(0, 10, 30)

# Create test occurrences
set.seed(0)
longitude <- sample(ext(envBrick)[1]:ext(envBrick)[2],
                    size = 10, replace = FALSE)
set.seed(0)
latitude <- sample(ext(envBrick)[3]:ext(envBrick)[4],
                   size = 10, replace = FALSE)
set.seed(0)
depth <- sample(0:35, size = 10, replace = TRUE)
occurrences <- data.frame(longitude,latitude,depth)

# Generate background sampling buffer
buffPts <- vect(occurrences,
                c("longitude", "latitude"))
crs(buffPts) <- crs(envBrick)
mShp <- aggregate(buffer(buffPts, width = 1000000))

# Here's the function
occSample3d <- mSampling3D(occs = occurrences,
                           envBrick = envBrick,
                           mShp = mShp,
                           depthLimit = "occs")
```

`oneRasterPlot` *Single raster plot*

Description

A convenient wrapper around `ggplot` to generate a formatted plot of a single raster.

Usage

```
oneRasterPlot(
  rast,
  land = NA,
```

```
  landCol = "black",
  scaleRange = NA,
  graticule = TRUE,
  title = "A Raster",
  verbose = TRUE,
  ...
)
```

Arguments

rast	A single SpatRaster layer on a continuous scale.
land	An optional coastline polygon shapefile of types sf or SpatRaster to provide geographic context for the occurrence points.
landCol	Color for land on map.
scaleRange	Optional numeric vector containing maximum and minimum values for color scale. Helpful when making multiple plots for comparison. Defaults to minimum and maximum of input rast.
graticule	logical. Do you want a grid of lon/lat lines?
title	A title for the plot.
verbose	logical. Switching to FALSE mutes message alerting user if input rast values exceed a specified scaleRange.
...	Additional optional arguments to pass to plot initial plot object or viridis.

Value

A plot of mapping the values of the input raster layer

See Also

viridis ggplot

Examples

```
library(terra)
rast <- rast(ncol=10, nrow=10)
values(rast) <- seq(0,99, 1)

oneRasterPlot(rast = rast)
```

`partition_3D` *Create 3D partitions*

Description

Creates partition schemes in 3D for model training and testing. Can create block or kfold partitions, with output as a `vector` or `list`

Usage

```
partition_3D(
  maxent_df,
  coord_df,
  which_partition = "k.fold",
  kfolds = NULL,
  orientation = "lon_lat",
  return_format = "vector",
  ensure_all_folds = TRUE,
  max_attempts = 100,
  na_strategy = "NA"
)
```

Arguments

`maxent_df`	Data frame with column 'p' (1=presence, 0=absence).
`coord_df`	Data frame with 'longitude','latitude','depth' aligned with maxent_df.
`which_partition`	"k.fold" (default) or "block".
`kfolds`	Integer >= 2 for k.fold.
`orientation`	For "block": "lon_lat" (default) or "lat_lon".
`return_format`	"vector" (default) or "list".
`ensure_all_folds`	(k.fold) Ensure all folds appear among presences with known depth (default TRUE).
`max_attempts`	(k.fold) Retry cap when ensure_all_folds=TRUE (default 100).
`na_strategy`	(k.fold) How to handle NA depth rows: "NA" (default) leaves them NA; "random" assigns a random fold.

Details

maxent_df and coord_df should be the same dataframes to be provided to `maxent_3D()` for model production. The spatial block partition, similarly to a traditional 2D partition, will separate the occurrences into 4 groups of equal (or about equal) size, with two groups making up two blocks on the upper half of the depth distribution and two groups on the lower half. Background points are assigned to spatial groups according to how they fall into the spatial partitions delimited by the occurrences. Note that this means the number of background points in each partition may not be as close to equal as the occurrences.

Value

If return_format="vector": integer vector (1..k or 1..4), length == nrow(maxent_df). Unassignable rows get NA. If "list": list(occ_partitions, bg_partitions).

Examples

```
# create test dataframe
occ <- rep(1, times = 10)
bg <- rep(0, times = 1000)
env1 <- sample(c(1:100), size = 1010, replace = TRUE)
env2 <- sample(c(1:1000), size = 1010, replace = TRUE)
p <- c(occ, bg)
testdf <- data.frame(p, env1, env2)

# create test coord data
r <- terra::rast(ncol = 100, nrow = 100)
set.seed(0)
longitude <- sample(terra::ext(r)[1]:terra::ext(r)[2], size = 1010, replace = TRUE)
set.seed(0)
latitude <- sample(terra::ext(r)[3]:terra::ext(r)[4], size = 1010, replace = TRUE)
depth <- sample(c(0, 5, 10, 15, 20, 25, 30, 35, 40, 45), size = 1010, replace = TRUE)
test_coords <- data.frame(longitude, latitude, depth)

# Here's the function
result_kfold <- partition_3D(maxent_df = testdf, coord_df = test_coords,
which_partition = 'k.fold', kfolds = 3)

result_block <- partition_3D(maxent_df = testdf, coord_df = test_coords,
which_partition = 'block', orientation = 'lat_lon')
```

---

plotLayers *Plotting 3D model in 2D*

Description

This script plots a semitransparent layer of suitable habitat for each depth layer. The redder the color, the shallower the layer, the bluer, the deeper. The more saturated the color, the more layers with suitable habitat.

Usage

```
plotLayers(
  rast,
  land = NA,
  landCol = "black",
  title = NULL,
  graticule = TRUE,
```

```
  ...
)
```

Arguments

rast	A SpatRaster vector with the 3D presence/absence distribution of a species (interpreted as 1 = presence, 0 = absence).
land	An optional coastline polygon shapefile of types sf or SpatRaster to provide geographic context for the occurrence points.
landCol	Color for land on map.
title	A title for the plot. If not title is supplied, the title "Suitability from (MINIMUM DEPTH) to (MAXIMUM DEPTH)" is inferred from names of rast.
graticule	Do you want a grid of lon/lat lines?
...	Additional optional arguments.

Value

A plot of class recordedplot

Note

Only include the depth layers that you actually want to plot.

See Also

viridis

Examples

```
library(terra)

rast1 <- rast(ncol=10, nrow=10)
values(rast1) <- rep(0:1, 50)

rast2 <- rast(ncol=10, nrow=10)
values(rast2) <- c(rep(0, 50), rep(1,50))

rast3 <- rast(ncol=10, nrow=10)
values(rast3) <- rep(c(1,0,0,1), 25)

distBrick <- c(rast1, rast2, rast3)

plotLayers(distBrick)
```

pointCompMap *Comparative point mapping*

Description

A convenient wrapper around ggplot to generate formatted plots comparing two sets of occurrence point plots.

Usage

```
pointCompMap(
  occs1,
  occs2,
  spName,
  land = NA,
  occs1Col = "#bd0026",
  occs2Col = "#fd8d3c",
  agreeCol = "black",
  occs1Name = "Set 1",
  occs2Name = "Set 2",
  landCol = "gray",
  waterCol = "steelblue",
  ptSize = 1,
  verbose = TRUE,
  ...
)
```

Arguments

occs1	A data.frame with at least two columns named "longitude" and "latitude" or that can be coerced into this format.
occs2	A data.frame with at least two columns named "longitude" and "latitude" or that can be coerced into this format.
spName	A character string with the species name to be used in the plot title.
land	An optional coastline polygon shapefile of types sf or SpatRaster to provide geographic context for the occurrence points.
occs1Col	Color for occurrence points on map
occs2Col	Color for occurrence points on map
agreeCol	Color for occurrence points shared between occs1 and occs2.
occs1Name	An optional name for the first set of occurrences, which will be color-coded to occs1Col in the resulting plot.
occs2Name	An optional name for the first set of occurrences, which will be color-coded to occs2Col in the resulting plot.
landCol	Color for land on map

waterCol	Color for water on map
ptSize	numeric value for cex; size of occurrence points on map.
verbose	logical. Switching to FALSE mutes message describing which columns in occs1 and occs2 are interpreted as x and y coordinates.
...	Additional optional arguments to pass to ggplot initial plot object.

Value

A ggplot plot object.

Note

The x and y column names of occs1 and occs2 must match.

See Also

ggplot

Examples

```
set.seed(5)
occs <- data.frame(cbind(decimalLatitude = sample(seq(7,35), 24),
                         decimalLongitude = sample(seq(-97, -70), 24)))

set.seed(0)
occs1 <- occs[sample(1:nrow(occs),
                     size = 12, replace = FALSE),]
set.seed(10)
occs2 <- occs[sample(1:nrow(occs),
                     size = 12, replace = FALSE),]

pointCompMap(occs1 = occs1, occs2 = occs2,
             occs1Col = "red", occs2Col = "orange",
             agreeCol = "purple",
             occs1Name = "2D",
             occs2Name = "3D",
             waterCol = "steelblue",
             spName = "Steindachneria argentea",
             ptSize = 2,
             verbose = FALSE)
```

---

pointMap *Point mapping*

Description

A convenient wrapper around ggplot to generate formatted occurrence point plots.

Usage

```
pointMap(
  occs,
  spName,
  land = NA,
  ptCol = "#bd0026",
  landCol = "gray",
  waterCol = "steelblue",
  ptSize = 1,
  verbose = TRUE,
  ...
)
```

Arguments

occs	A data.frame with at least two columns named "longitude" and "latitude" or that can be coerced into this format.
spName	A character string with the species name to be used in the plot title.
land	An optional coastline polygon shapefile of types sf or SpatRaster to provide geographic context for the occurrence points.
ptCol	Color for occurrence points on map
landCol	Color for land on map
waterCol	Color for water on map
ptSize	numeric value for cex; size of occurrence points on map.
verbose	logical. Switching to FALSE mutes message describing which columns in occs are interpreted as x and y coordinates.
...	Additional optional arguments to pass to ggplot initial plot object.

Value

A ggplot plot object.

See Also

ggplot

Examples

```
occs <- read.csv(system.file("extdata/Steindachneria_argentea.csv",
                             package='voluModel'))
spName <- "Steindachneria argentea"
pointMap(occs = occs, spName = spName,
         land = rnaturalearth::ne_countries(scale = "small",
                                            returnclass = "sf")[1])
```

`rasterComp` *Comparative raster mapping*

Description

A convenient wrapper around `terra::plot` to generate formatted plots comparing two rasters. This is used in the context of voluModel to overlay semi-transparent distributions (coded as 1) in two different `RasterLayers`.

Usage

```
rasterComp(
  rast1 = NULL,
  rast2 = NULL,
  col1 = "#1b9e777F",
  col2 = "#7570b37F",
  rast1Name = "Set 1",
  rast2Name = "Set 2",
  land = NA,
  landCol = "black",
  title = "A Raster Comparison",
  graticule = TRUE,
  ...
)
```

Arguments

`rast1`	A single `SpatRaster` showing the distribution of the species corresponding to `rast1Name`. Should have values of 0 (absence) and 1 (presence). Can also be `NULL`.
`rast2`	A single `SpatRaster` showing the distribution of the species corresponding to `rast2Name`. Should have values of 0 (absence) and 1 (presence). Must match the extent and resolution of `rast1`. Can also be `NULL`.
`col1`	Color for `rast1` presences
`col2`	Color for `rast2` presences
`rast1Name`	An optional name for the first set of occurrences, which will be color-coded to `occs1Col` in the resulting plot.
`rast2Name`	An optional name for the first set of occurrences, which will be color-coded to `occs2Col` in the resulting plot.
`land`	An optional coastline polygon shapefile of types `sf` or `SpatRaster` to provide geographic context for the occurrence points.
`landCol`	Color for land on map.
`title`	A title for the plot.
`graticule`	Do you want a grid of lon/lat lines?
`...`	Additional optional arguments to pass to `terra::plot()`.

Value

A plot of class `recordedplot` overlaying mapped, semitransparent extents of the input rasters

Note

The extents of `rast1` and `rast2` must match.

See Also

`plot`

Examples

```
library(terra)
rast1 <- rast(ncol=10, nrow=10)
values(rast1) <- rep(0:1, 50)

rast2 <- rast(ncol=10, nrow=10)
values(rast2) <- c(rep(0, 50), rep(1,50))

rasterComp(rast1 = rast1, rast2 = rast2)
```

---

smoothRaster *Smooth rasters*

Description

Uses thin plate spline regression from `fields` package to smooth raster values.

Usage

```
smoothRaster(inputRaster, fast = FALSE, ...)
```

Arguments

`inputRaster`	An object of class `SpatRaster`
`fast`	A logical operator. Setting to `TRUE` triggers use of `fastTps` instead of `Tps`.
`...`	For any additional arguments passed to `Tps` or `fastTps`

Details

Original raster is smoothed using a thin plate spline. This may be desirable in cases where the user has a reasonable expectation of spatial autocorrelation, but observes putative measurement errors in a raster. The user has the option of choosing `fastTps` to speed calculation, but be advised that this is only an approximation of a true thin plate spline.

Value

An object of class SpatRaster

See Also

Tps, fastTps

Examples

```
library(terra)
library(fields)
# Create sample raster
r <- rast(ncol=100, nrow=100)
values(r) <- 1:10000

# Introduce a "bubble"
values(r)[720:725] <- 9999
plot(r)

# Smooth bubble with smoothRaster
fastSmooth <- smoothRaster(r, fast = TRUE, aRange = 10.0)
plot(fastSmooth)
```

---

threshold_3D *Test different threshold levels and produce presence/absence layers for 3D models*

Description

Creates 3D presence/absence layers by setting all values in a suitability layer above a certain threshold to 1 and all values below that threshold to 0. The threshold value is determined by the sensitivity given by the user, where the sensitivity is the percentage of occurrences to be counted as present in the resulting presence/absence layer. For example, with a sensitivity of 0.9 or 90%, The threshold is the suitability value where 90% of the occurrence points fall on grid cells with suitability scores above that value.

The function can try multiple different sensitivity levels, and for each will output a presence/absence spatRaster stack, as well as the specificity (proportion of psuedoabsences correctly considered absent), the true skill statistic (TSS), which is a measure of how well the threshold balances commission and omission error, and the suitability value of the threshold.

The user can also downweight sensitivity when calculating TSS with the "weights" parameter. This may be important to do as 3D niche models often have a higher ratio of psuedoabsences to occurrences than 2D models.

Usage

```
threshold_3D(
  predicted_layers,
  thresholding_vals,
  maxent_df,
  coord_df,
  weights = NULL
)
```

Arguments

predicted_layers	A spatRaster stack of suitability layers where each layer corresponds to a depth slice
thresholding_vals	A vector of sensitivity levels for creating different thresholds and presence/absence layers. If one wanted to test sensitivities of 90%, and 95%, this would be input as c(0.9, 0.95)
maxent_df	'data.frame' where first column is a vector of presences named "p" containing 1's and 0's. Each row represents a cell in the spatRaster volume with an x, y, z coordinate, and 1's are presences while 0's are absences, or background points. Other columns are environmental variable values extracted at the occurrence and background points.
coord_df	A dataframe containing the longitude, latitude, and depth for each cell in maxent_df, named "longitude," "latitude," and "depth"
weights	a numeric giving what the sensitivity should be downweighted by. If no value is given, TSS will be calculated according to its original formula

Value

a 'list' with two components:

$threshold_layers, a spatRaster stack of presence absence rasters thresholded at the sensitivity with the highest TSS

$tss_results, a 'data.frame' containing the specificity, TSS, and suitability score for each sensitivity given in thresholding_vals

References

Allouche O, Tsoar A, and Kadmon R. 2006. Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). *Journal of Applied Ecology*, 43, 1223-1232.

Examples

```
library(terra)
library(dplyr)

# creating list of spatRaster stacks where each element is a depth slice
r1_d1 <- rast(ncol = 100, nrow = 100)
```

```
set.seed(0)
values(r1_d1) <- sample(c(1:100), size = 1000, replace = TRUE)
r2_d1 <- rast(ncol = 100, nrow = 100)
set.seed(0)
values(r2_d1) <- sample(c(1:1000), size = 1000, replace = TRUE)
r1_d2 <- r1_d1
values(r1_d2) <- values(r1_d1)+10
r2_d2 <- r2_d1
values(r2_d2) <- values(r2_d1)+10
d1 <- c(r1_d1, r2_d1)
names(d1) <- c("valsr1", "valsr2")
d2 <- c(r1_d2, r2_d2)
names(d2) <- c("valsr1", "valsr2")
envlist <- list(d1, d2)

# creating occs and bgs
set.seed(0)
occs <- sample(c(1:nrow(crds(envlist[[1]][[1]]))), size = 50, replace = FALSE)
bgs <- sample(c(1:nrow(crds(envlist[[1]][[1]]))), size = 500, replace = FALSE)

occs_d1 <- crds(envlist[[1]][[1]])[occs[1:25],]
occs_d2 <- crds(envlist[[2]][[1]])[occs[26:50],]
bg_d1 <- crds(envlist[[1]][[1]])[bgs[1:250],]
bg_d2 <- crds(envlist[[1]][[1]])[bgs[251:500],]

# extracting at occs and bgs
occ_valsr1_d1 <- extract(envlist[[1]][[1]], occs_d1)
occ_valsr1_d2 <- extract(envlist[[2]][[1]], occs_d2)
occ_valsr2_d1 <- extract(envlist[[1]][[2]], occs_d1)
occ_valsr2_d2 <- extract(envlist[[2]][[2]], occs_d2)

occ_valsr1 <- rbind(occ_valsr1_d1, occ_valsr1_d2)
occ_valsr2 <- rbind(occ_valsr2_d1, occ_valsr2_d2)

bg_valsr1_d1 <- extract(envlist[[1]][[1]], bg_d1)
bg_valsr1_d2 <- extract(envlist[[2]][[1]], bg_d2)
bg_valsr2_d1 <- extract(envlist[[1]][[2]], bg_d1)
bg_valsr2_d2 <- extract(envlist[[2]][[2]], bg_d2)

bg_valsr1 <- rbind(bg_valsr1_d1, bg_valsr1_d2)
bg_valsr2 <- rbind(bg_valsr2_d1, bg_valsr2_d2)

valsr1 <- rbind(occ_valsr1, bg_valsr1)
valsr2 <- rbind(occ_valsr2, bg_valsr2)

p1 <- rep(1, times = 50)
p0 <- rep(0, times = 500)
p <- c(p1, p0)

maxdf <- data.frame(p, valsr1, valsr2)

# creating coord_df
coord_df <- rbind(occs_d1, occs_d2, bg_d1, bg_d2)
```

```
z <- c(rep(1, times = 25), rep(2, times = 25), rep(1, times = 250),
rep(2, times = 250))
coord_df <- cbind(coord_df, z)
colnames(coord_df) <- c("longitude", "latitude", "depth")

# creating suitability rasters
suitd1 <- d1[[1]]
values(suitd1) <- runif(min = 0, max = 1, n = length(values(suitd1)))
suitd2 <- d2[[1]]
values(suitd2) <- runif(min = 0, max = 1, n = length(values(suitd2)))
suit <- c(suitd1, suitd2)

# here's the function
result <- threshold_3D(predicted_layers = suit, thresholding_vals = c(0.9, 0.95),
maxent_df = maxdf, coord_df = coord_df, weights = 2/3)
```

`transectPlot` *Plot vertical sample*

Description

Plots cell values along a vertical transect

Usage

```
transectPlot(
  rast = NULL,
  sampleAxis = "lon",
  axisValue = NA,
  scaleRange = NA,
  plotLegend = TRUE,
  depthLim = as.numeric(max(names(rast))),
  transRange = c(-90, 90),
  transTicks = 20,
  verbose = FALSE,
  ...
)
```

Arguments

`rast`	A multilayer `SpatRaster` object, with names corresponding to the z coordinate represented by the layer. These names must be interpretable by `as.numeric`.
`sampleAxis`	Specifies whether a latitudinal ("lat") or longitudinal ("long") transect is desired.
`axisValue`	Numeric value specifying transect postion.
`scaleRange`	A numeric vector of length 2, specifying the range that should be used for the plot color scale.

plotLegend	logical, controls whether legend is plotted.
depthLim	A single vector of class numeric. How deep should the plot go?
transRange	A numeric vector of lenghth 2. How far along the transect should be plotted?
transTicks	numeric, spacing between breaks on x axis.
verbose	logical. Switching to FALSE mutes message alerting user if input rast values exceed specified scaleRange.
...	Additional optional arguments to pass to viridis.

Value

A ggplot showing a vertical slice through the SpatRaster.

Note

Only unprojected SpatRaster files are supported.

Examples

```
library(terra)

rast1 <- rast(ncol=10, nrow=10)
values(rast1) <- rep(0:3, 50)

rast2 <- rast(ncol=10, nrow=10)
values(rast2) <- c(rep(0, 50), rep(1,25), rep(2,25))

rast3 <- rast(ncol=10, nrow=10)
values(rast3) <- rep(c(1,3,2,1), 25)

distBrick <- c(rast1, rast2, rast3)
names(distBrick) <- c(0:2)

transectPlot(distBrick, depthLim = 3)
```

---

xyzSample *Sampling from a* SpatRaster *vector using 3D coordinates*

Description

Gets values at x,y,z occurrences from a given 3D environmental variable brick

Usage

```
xyzSample(occs, envBrick, verbose = TRUE)
```

Arguments

occs	A data.frame with at least three columns named "longitude", "latitude", and "depth", or that can be coerced into this format.
envBrick	A SpatRaster vector object with one environmental variable. Each layer represents a depth slice. See Details for more information.
verbose	logical. Switching to FALSE mutes message describing which columns in occs1 and occs2 are interpreted as x, y, and z coordinates.

Details

The SpatRaster vector object should have numeric names that correspond with the beginning depth of a particular depth slice. For example, one might have three layers, one from 0 to 10m, one from 10 to 30m, and one from 30 to 100m. You would name the layers in this brick names(envBrick) <- c(0, 10, 30. xyzSample identifies the layer name that is closest to the depth layer value at a particular X, Y coordinate, and samples the environmental value at that 3D coordinate.

Value

Vector of environmental values equal in length to number of rows of input occs data.frame.

Examples

```
library(terra)

# Create test raster
r1 <- rast(ncol=10, nrow=10)
values(r1) <- 1:100
r2 <- rast(ncol=10, nrow=10)
values(r2) <- c(rep(20, times = 50), rep(60, times = 50))
r3 <- rast(ncol=10, nrow=10)
values(r3) <- 8
envBrick <- c(r1, r2, r3)
names(envBrick) <- c(0, 10, 30)

# Create test occurrences
set.seed(0)
longitude <- sample(ext(envBrick)[1]:ext(envBrick)[2],
                    size = 10, replace = FALSE)
set.seed(0)
latitude <- sample(ext(envBrick)[3]:ext(envBrick)[4],
                   size = 10, replace = FALSE)
set.seed(0)
depth <- sample(0:35, size = 10, replace = TRUE)
occurrences <- as.data.frame(cbind(longitude,latitude,depth))

# Test function
occSample3d <- xyzSample(occurrences, envBrick)

# How to use
occurrences$envtValue <- occSample3d
```

Index