Package 'voluModel'

August 21, 2024

Title Modeling Species Distributions in Three Dimensions

Version 0.2.2

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/

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

Encoding UTF-8

Depends R (>= 4.0.0)

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

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

VignetteBuilder knitr

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

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

Repository CRAN

Date/Publication 2024-08-20 22:50:01 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)
```

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)
```

`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)
```

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))

# Calibration
calibration <- lapply(rastList, FUN = function(x) xyzSample(occurrences, x)) %>% bind_rows

# 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)
```

`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)
```

`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