Package 'maskRangeR'

May 8, 2026

Title Mask Species Geographic Ranges

Version 1.1

Description Mask ranges based on expert knowledge or remote sensing layers. These tools can be combined to quantitatively and reproducibly generate a new map or to update an existing map. Methods include expert opinion and data-driven tools to generate thresholds for binary masks.

License GPL-3

Encoding UTF-8

Depends R (>= 3.4.0)

Imports dplyr, e1071, lubridate, magrittr, raster, sp, stats, utils,

Suggests dismo, testthat, knitr, rmarkdown, wallace

RoxygenNote 7.1.2

VignetteBuilder knitr

NeedsCompilation no

Author Cory Merow [aut, cre],
Peter J. Galante [aut],
Jamie M. Kass [aut],
Cecina Babich Morrow [aut],
Valentina Grisales Betancur [aut]

Maintainer Cory Merow <cory.merow@gmail.com>

Repository CRAN

Date/Publication 2022-05-11 11:50:02 UTC

Contents

`annotate` *Annotate point data with rasters based on matching dates.*

Description

Annotate point data with rasters based on matching dates associated with points to dates associated with rasters. Specifically, we're thinking of the points as species observations and the rasters as remotely sensed environmental layers, but they can represent any points and rasters with dates.

Usage

```
annotate(datedOccs, env, envDates, dateScale)
```

Arguments

`datedOccs`	a 'SpatialPointsDataFrame' of occurrence localities (generally longitude and latitude in decimal degrees) paired with dates. One column must be labeled 'date' and have class 'POSIXct', e.g., as obtained from using 'lubridate::parse_date_time'
`env`	a raster stack
`envDates`	a vector of dates the same length as 'env'. The vector should have class 'POSIXct', e.g., as obtained from using 'lubridate::parse_date_time'
`dateScale`	string: 'year', 'month', or 'day'

Details

See Examples.

Value

a SpatialPointsDataFrame

Author(s)

Cory Merow <cory.merow@gmail.com>,

Examples

```
r1 <- raster::raster(nrows=50, ncols=50, xmn=-50, xmx=50)
raster::values(r1)<- runif(n = (50*50))
r2 <-  raster::raster(nrows=50, ncols=50, xmn=-50, xmx=50)
raster::values(r2)<- runif(n = (50*50))
env <-  raster::stack(r1,r2)
names(env) <- c("1995","1996")
datedOccs <- data.frame(cbind(c(0,10), c(-10,15)))
colnames(datedOccs) <- c("long", "lat")
datedOccs$date <- c("1995", "1996")
datedOccs$date <- lubridate::parse_date_time(datedOccs$date, orders = c("Y", "Ym"))
sp::coordinates(datedOccs) <- c("long", "lat")
raster::projection(datedOccs) <-  raster::projection(env)
dateScale = "year"
envDates <- c("1995","1996")
annotate(datedOccs = datedOccs, env = env, envDates = envDates, dateScale = dateScale)
```

`continuousMask` *Update a binary raster (species range map) to continuous values that describe the proportion of cell that is suitable or the quality of the cell.*

Description

The use case envision is updating a binary map to continuous values that describe the proportion of the cell that is suitable, based on land use/land cover classes

Usage

```
continuousMask(contStack, suitable, binaryRange, maskValue = NA, ...)
```

Arguments

`contStack`	a stack of layers with continuous values.
`suitable`	a vector of names of suitable layers of ‘contStack‘. These can be substrings of the layer names that can be ‘grep‘ped from ‘names(contStack)‘
`binaryRange`	a binary raster
`maskValue`	numeric. The value in ‘binaryRange‘ that indicates the unsuitable cell
`...`	arguments to be passed to ‘raster::mask‘

Details

See Examples.

Value

a raster

Author(s)

Cory Merow <cory.merow@gmail.com>,

`cropResampleTrim` *Line two rasters or stacks or lists of rasters*

Description

Obtain the same extents and resample to the finest resolution layer.

Usage

```
cropResampleTrim(expertMap, maskListRaw)
```

Arguments

`expertMap`	A binary map, either as a polygon or a raster.
`maskListRaw`	A list of rasters, each corresponding to layers from which masks will eventually be made (in another function).

Details

See Examples.

Value

a list

Author(s)

Cory Merow <cory.merow@gmail.com>,

Examples

```
r1 <- raster::raster(raster::extent(c(-72, -64, 41, 50)), res = c(0.008333333, 0.008333333))
raster::values(r1) <- (1:raster::ncell(r1))^2
coords <- dismo::randomPoints(r1, 4)
polyg <- sp::SpatialPolygons(list(sp::Polygons(list(sp::Polygon(coords)),1)))
r2 <- raster::raster(raster::extent(c(-72, -64, 41, 50)), res = c(0.008333333, 0.008333333))
raster::values(r2) <- (1:raster::ncell(r2))^3
r3 <- raster::raster(raster::extent(c(-72, -64, 41, 50)), res = c(0.008333333, 0.008333333))
raster::values(r3) <- (1:raster::ncell(r3))^0.5
maskListRaw <- list(r1, r2, r3)
cropResampleTrim(expertMap = polyg, maskListRaw = maskListRaw)
```

`focalCompare` *Generate layers based on different focal windows*

Description

Aids in exploring how different focal regions may affect masks.

Usage

```
focalCompare(layer, windowSizes, fun, mc.cores = 1)
```

Arguments

`layer`	A single raster layer to be summarized.
`windowSizes`	A vector of the number of cell in each direction to buffer for the focal summary. E.g., a value of 1 indicates the 8 cells immediately surrounding the focal cell, i.e., which are 1 cell away.
`fun`	The function fun should take multiple numbers, and return a single number. For example mean, modal, min or max. It should also accept a na.rm argument (or ignore it, e.g. as one of the 'dots' arguments. For example, length will fail, but function(x, ...)na.omit(length(x)) works. (Specifications from 'raster::focal')
`mc.cores`	Number of cores for (optional) parallelization

Details

See Examples.

Value

Raster object

Note

This may be particularly useful if for mobile species when their movement patterns cover a much larger extent than the single cell in which they were observed.

Author(s)

Cory Merow <cory.merow@gmail.com>,

Examples

```
r <- raster::raster(ncols=36, nrows=18, xmn=0)
r[] <- runif(raster::ncell(r))
r15 <- focalCompare(r, windowSizes = c(1:5),mc.cores=1,fun=mean)
```

`lotsOfMasks` *Generate and apply multiple masks to a map*

Description

Based on a potential distribution, environmental rasters, and bounds for suitable habitat on the environmental rasters

Usage

```
lotsOfMasks(expertRaster, maskStack, maskBounds)
```

Arguments

`expertRaster`	The binary expert map (1s and 0s), rasterized with the same projection as 'maskStack'
`maskStack`	A stack of *named* layers from which masks will be made
`maskBounds`	A data.frame with columns indicating the layer name (matching the names in maskStack), and the min and max values of that layer to be used for masking.

Details

See Examples.

Value

a RasterStack

Author(s)

Cory Merow <cory.merow@gmail.com>,

Examples

```
r1 <- raster::raster(nrows=108, ncols=21, xmn=0, xmx=10)
raster::values(r1)<- sort(runif(n = (108*21)))
r1[r1>0.5] <- 1
r1[r1<0.5] <- 0
r2 <- raster::raster(nrows=108, ncols=21, xmn=0, xmx=10)
raster::values(r2) <- runif(n=(108*21))
r3 <- raster::raster(nrows=108, ncols=21, xmn=0, xmx=10)
raster::values(r3) <- runif(n=(108*21))
maskStack <- raster::stack(r2, r3)
names(maskStack) <- c("r2", "r3")
minbounds <- c(0.3, 0.4)
maxbounds <- c(0.4, 0.5)
maskBounds <- data.frame(cbind(c("r2", "r3"), minbounds, maxbounds))
colnames(maskBounds)<- c("Layer", "Min Value", "Max Value")
maskBounds[,2] <- as.numeric(as.character(maskBounds[,2]))
```

```
maskBounds[,3] <- as.numeric(as.character(maskBounds[,3]))
out <- lotsOfMasks(expertRaster = r1, maskStack = maskStack, maskBounds = maskBounds)
```

`manyMaskSensitivity` *Sensitivity testing for masks*

Description

Compare how masks of climate tolerances affect predicted area

Usage

```
manyMaskSensitivity(crt, rasProj = NULL, maskBounds, expertRaster)
```

Arguments

`crt`	A raster stack; the output from the function maskRangeR::cropResampleTrim
`rasProj`	(optional) a character string: a proj4string showing the projection of the environmental layers. If NULL, areas will be estimated using the raster package.
`maskBounds`	A data.frame with columns indicating the layer name (matching the names in maskStack), and the min and max values of that layer to be used for masking.
`expertRaster`	The binary expert map (1s and 0s), rasterized with the same projection as 'maskStack'

Details

See Examples.

Value

returns a data.frame where row names are the environmental layer name combinations, and Area is expressed in square km, unless a projection is supplied

Author(s)

Peter Galante <pgalante@amnh.org>

Examples

```
#See lotsOfMasks and maskRanger examples
```

maskRanger *Make a matrix of modeling decisions to be used to specify clipping rules*

Description

Performs data driven masking of potential species distributions.

Usage

```
maskRanger(
  potentialDist,
  initialDist = NULL,
  maskLayers,
  logicString,
  method = "mask"
)
```

Arguments

potentialDist	A raster stack of binary or continuous values. Supplying more than one layer will be interepreted as different time periods. Layers should follow the naming convention 'Y2000', 'Y2001', etc. Must have same extent and resolution as maskLayers.
initialDist	A raster showing a previously created optimally tuned SDM. Must have same extent and resolution as maskLayers.
maskLayers	A single raster or a raster stack. If a single raster, the same mask will be applied to each layer of 'potentialDist'. If a stack it must have the same number of layers as potentialDist, and each layer corresponds to a different time period. Must have same extent and resolution as initialDist.
logicString	a character indicating the logical conditions to use for masking.
method	A list of strings defining methods to be used, in the same order as 'rsList'. If a single value is provided it will be applied to all rasters in 'rsList'. Options currently include only 'mask' to mask cells with values outside the bounds.

Details

See Examples.

Value

a raster stack

Note

To apply multiple masks, e.g., elevation and forest cover, use separate calls to maskRS.

Author(s)

Cory Merow <cory.merow@gmail.com>,

Examples

```
# Multiple Expert Maps
# Generate random polygon
env1 <- raster::raster(nrows=108, ncols=21, xmn=0, xmx=10)
env2 <- raster::raster(nrows=108, ncols=21, xmn=0, xmx=10)
env3 <- raster::raster(nrows=108, ncols=21, xmn=0, xmx=10)
raster::values(env1)<- sort(runif(n = (108*21)))
raster::values(env2)<- runif(n = (108*21))
raster::values(env3)<- runif(n = (108*21))
sdm <- raster::raster(nrows=108, ncols=21, xmn=0, xmx=10)
raster::values(sdm)<- sort(runif(n = (108*21)))
coords <- dismo::randomPoints(sdm, 3)
polyg <- sp::Polygon(coords)
polyg <- sp::SpatialPolygons(list(sp::Polygons(list(polyg), ID = "a")))
expertRaster <- raster::rasterize(polyg, sdm)
maskStack <-raster:: stack(env1, env2, env3)
names(maskStack) <- c("env1", "env2", "env3")
# Get list of tolerances for environmental data
env1Vals <- quantile(raster::values(env1), prob = c(0, 0.025, 0.25, 0.5, 0.75, 0.975, 1),
                     na.rm = TRUE)
env2Vals <- quantile(raster::values(env2), prob = c(0, 0.025, 0.25, 0.5, 0.75, 0.975, 1),
                     na.rm = TRUE)
env3Vals <- quantile(raster::values(env3), prob = c(0, 0.025, 0.25, 0.5, 0.75, 0.975, 1),
                     na.rm = TRUE)
maskBounds <- data.frame(rbind(cbind(env1Vals[[3]], env1Vals[[5]]),
                               cbind(env2Vals[[3]], env2Vals[[5]]),
                               cbind(env3Vals[[3]], env3Vals[[5]])))
maskBounds <- cbind(names(maskStack), maskBounds)
colnames(maskBounds) <- c("Layer", "min", "max")
# mask range by these tolerance masks
realized <- lotsOfMasks(expertRaster, maskStack, maskBounds)
```

`rangeSVM` *Classify species ranges based on occurrence coordinates and SDM scores.*

Description

`rangeSVM()` returns a tuned support vector machine (SVM) model that predicts species identity based on predictors that are solely spatial, based on occurrence coordinates, or a combination of spatial and environmental, based on both occurrence coordinates and environmental suitability values. Suitability values can be predicted with species distribution models (SDMs; a.k.a. ecological niche models).

Usage

```
rangeSVM(xy1, xy2, ..., sdm = NULL, nrep = 100, weight = FALSE, mc.cores = 1)
```

Arguments

xy1	Matrix or data frame of occurrence coordinates for species 1.
xy2	Matrix or data frame of occurrence coordinates for species 2.
...	Other matrices or data frames of occurrence coordinates for additional species.
sdm	Raster or RasterStack representing environmental suitability (can be predictions from SDMs). These must have the same extent as both species' occurrence points. Default is NULL.
nrep	Numeric for number of SVM tuning iterations. Default is 100.
weight	Boolean. If TRUE, species with fewer occurrence records are weighted higher in the SVM. Default is FALSE.
mc.cores	Number of cores to use for parallel processing. Default is 1.

Details

The tuning operation uses `tune.svm()` from the e1071 package, which performs 10-fold cross validation and selects the best model based on classification error. Ranges of the cost and gamma parameters are explored in the tuning exercise. The tuning function is iterated `nrep` times, and the parameter combination used most frequently across all iterations is used to build a final SVM model.

When `sdm = NULL`, the SVM is purely spatial, based only on the occurrence coordinates of each species. Otherwise, the SVM is fit with both a spatial predictor and any additional ones added as rasters. These extra predictors can be based on predictions from a species distribution model (SDM; a.k.a. ecological niche model), and in this case would represent environmental or climatic suitability, depending on the variables used in the SDM.

Value

The tuned SVM model.

Examples

```
r1.sdm <- raster::raster(raster::extent(c(-72, -64, 41, 50)), res = c(0.008333333, 0.008333333))
raster::values(r1.sdm) <- (1:raster::ncell(r1.sdm))^2
r2.sdm <- raster::raster(raster::extent(c(-72, -64, 41, 50)), res = c(0.008333333, 0.008333333))
raster::values(r2.sdm) <- (raster::ncell(r2.sdm):1)^2
r3.sdm <- raster::raster(raster::extent(c(-72, -64, 41, 50)), res = c(0.008333333, 0.008333333))
r3.sdm [1] <- 10
r3.sdm <- raster::distance(r3.sdm)
sp1.xy <- data.frame(dismo::randomPoints(r1.sdm, 15, prob = TRUE))
colnames(sp1.xy) <- c("longitude", "latitude")
sp2.xy <- data.frame(dismo::randomPoints(r2.sdm, 15, prob = TRUE))
colnames(sp2.xy) <- c("longitude", "latitude")
sp3.xy <- data.frame(dismo::randomPoints(r3.sdm, 15, prob = TRUE))
colnames(sp3.xy) <- c("longitude", "latitude")
```

```
# Spatial SVMs (this can take about a minute to run)
svm.SP <- rangeSVM(sp1.xy, sp2.xy, sp3.xy, nrep=5) # more reps are recommended
```

`rangeSVM_predict` *Generate a raster based on predictions of SVM model with values corresponding to the species.*

Description

`rangeSVM_predict()` returns a raster representing the ranges of the species predicted by the fitted SVM tuned with `rangeSVM()`.

Usage

```
rangeSVM_predict(svm, r, sdm = NULL)
```

Arguments

`svm`	Model object for the SVM, returned by `rangeSVM()`.
`r`	Raster with the extent desired for the prediction. The values for cells used for predictions must have non-NA values, but the particular values do not matter.
`sdm`	Raster or RasterStack representing environmental suitability (can be predictions from SDMs). These rasters must match the predictor variables used in the SVM. Default is NULL.

Details

The values of the output raster are 1, 2, ..., corresponding to xy1, xy2, and any additional species used in `rangeSVM()`. These values represent the identities of the species.

Value

The Raster representing the SVM predictions.

Examples

```
r1.sdm <- raster::raster(raster::extent(c(-72, -64, 41, 50)), res = c(0.008333333, 0.008333333))
raster::values(r1.sdm) <- (1:raster::ncell(r1.sdm))^2
r2.sdm <- raster::raster(raster::extent(c(-72, -64, 41, 50)), res = c(0.008333333, 0.008333333))
raster::values(r2.sdm) <- (raster::ncell(r2.sdm):1)^2
r3.sdm <- raster::raster(raster::extent(c(-72, -64, 41, 50)), res = c(0.008333333, 0.008333333))
r3.sdm [1] <- 10
r3.sdm <- raster::distance(r3.sdm)
sp1.xy <- data.frame(dismo::randomPoints(r1.sdm, 15, prob = TRUE))
colnames(sp1.xy) <- c("longitude", "latitude")
sp2.xy <- data.frame(dismo::randomPoints(r2.sdm, 15, prob = TRUE))
```

```
colnames(sp2.xy) <- c("longitude", "latitude")
sp3.xy <- data.frame(dismo::randomPoints(r3.sdm, 15, prob = TRUE))
colnames(sp3.xy) <- c("longitude", "latitude")
# Spatial SVMs (this can take about a minute to run)
svm.SP <- rangeSVM(sp1.xy, sp2.xy, sp3.xy, nrep=5)
# Use SVM to create a raster of predicted regions
rand_svm.SP <- rangeSVM_predict(svm = svm.SP, r = r1.sdm)
```

`thresholdSensitivity` *Sensitivity testing for thresholds*

Description

Measure a number of reasonable thresholds and calculate areas based on these thresholds

Usage

```
thresholdSensitivity(
  datedOccs,
  maskLayer,
  maskClass,
  sdm,
  maskProjection = NULL,
  maskVal = NULL,
  selectedValue = NULL
)
```

Arguments

`datedOccs`	a ‘SpatialPointsDataFrame‘ where one column is labeled ‘date‘ and has class ‘POSIXct‘, e.g., as obtained from using ‘lubridate::parse_date_time‘
`maskLayer`	the layer from which you will build the mask. Usually the most recent satellite derived raster
`maskClass`	either top5, aroundSelected, quants, or userSpecified. "top5" ranks the environmental values at occurrences and returns the 5 highest observed values."aroundSelected" selects two observed values above and two below a user-specified threshold value. "quants" returns quantile values as thresholds. "userSpecified" will take a list of values defined by the user.
`sdm`	previously generated species distribution model
`maskProjection`	(optional) a proj4string showing the projection of the maskLayer. If NULL, areas will be estimated using the raster package.
`maskVal`	(optional) a user defined value for thresholding when using the "aroundSelected" maskClass, or if "userSpecified", a list of thresholds to use.
`selectedValue`	(optional) a user selected value around which masks will be selected using the nearest two threshold on either side

Details

See Examples.

Value

returns a list containing two items. The first is a rasterstack of the masked distributions. The second item is a table of thresholds and areas

Author(s)

Peter Galante <pgalante@amnh.org>

Index