Package: virtualspecies (via r-universe)

August 23, 2024

Type Package

Title Generation of Virtual Species Distributions

Version 1.6

Author Boris Leroy [cre, aut], Christine N. Meynard [ctb], Celine Bellard [ctb], Franck Courchamp [ctb], Robin Delsol [ctb], Willson Gaul [ctb]

Maintainer Boris Leroy <leroy.boris@gmail.com>

Description Provides a framework for generating virtual species distributions, a procedure increasingly used in ecology to improve species distribution models. This package integrates the existing methodological approaches with the objective of generating virtual species distributions with increased ecological realism.

Encoding UTF-8

License GPL (>=2.0)

Depends R (>= 3.5), terra

Imports ade4, graphics, grDevices, rnaturalearth, raster, stats, utils, viridis

URL https://borisleroy.com/virtualspecies/

BugReports https://github.com/Farewe/virtualspecies/issues

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/farewe/virtualspecies

RemoteRef HEAD

RemoteSha 2263b8138f6f74567c0ec8889e59f585d80bf5e1

Contents

virtualspecies-package

Generation of virtual species

Description

This package allows generating virtual species distributions, for example for testing species distribution modelling protocols. For a complete tutorial, see borisleroy.com/virtualspecies

Details

The process of generating a virtual species distribution is divided into four major steps.

1. Generate a virtual species distributions from environmental variables. This can be done by
   - defining partial response functions to each environmental variable, and then combining them to compute the overall environmental suitability, with generateSpFromFun
   - computing a PCA among environmental variables, and simulating the response of the species along the two first axes of the PCA with generateSpFromPCA

   This step can be randomised with generateRandomSp

2. Convert the virtual species distribution into presence-absence, with convertToPA

3. Facultatively, introduce a distribution bias with limitDistribution

4. Sample occurrence points (presence only or presence-absence) inside the virtual species distribution, either randomly or with biases, with sampleOccurrences

There are other useful functions in the package:

- formatFunctions: this is a helper function to format and illustrate the response functions as a correct input for generateSpFromFun

- plotResponse: to visualise the species-environment relationship of the virtual species
- removeCollinearity: this function can be used to remove collinearity among variables of a stack by selecting a subset of non-colinear variables
- synchroniseNA: this function can be used to synchronise NA values among layers of a stack

This packages makes use of different other packages:

- This package makes extensive use of the package terra to obtain spatialised environmental variables, and produce spatialised virtual species distributions.
- ade4 is used to generate species with a PCA approach.
- rnaturalearth is used to obtain free world shapefiles, in order to create dispersal limitations and sampling biases.

Author(s)

Boris Leroy <leroy.boris@gmail.com>

with help from C. N. Meynard, C. Bellard & F. Courchamp

Maintainer: Boris Leroy <leroy.boris@gmail.com>

References

Leroy, B. et al. 2016. virtualspecies, an R package to generate virtual species distributions. Ecography. 39(6):599-607

---

betaFun   *Beta response function*

Description

Generation of a beta response curve (see references) according to the equation:

$$k * (x - p1)^\alpha * (p2 - x)^\gamma$$

k is automatically estimated to have a maximum value of P equal to 1.

Usage

```
betaFun(x, p1, p2, alpha, gamma)
```

Arguments

x	a numeric value or vector. The input environmental variable.
p1	a numeric value or vector. Lower tolerance bound for the species
p2	a a numeric value or vector. Upper tolerance bound for the species
alpha	a numeric value or vector. Parameter controlling the shape of the curve (see details)
gamma	a numeric value or vector. Parameter controlling the shape of the curve (see details)

Details

p1 and p2 can be seen as the upper and lower critical threshold of the curve. `alpha` and `gamma` control the shape of the curve near p1 and p2, respectively. When `alpha = gamma`, the curve is symmetric. Low values of `alpha` and `gamma` result in smooth (< 1) to plateau (< 0.01) curves. Higher values result in peak (> 10) curves.

When `alpha < gamma`, the curve is skewed to the right. When `gamma < alpha`, the curve is skewed to the left.

Value

a numeric value or vector resulting from the function

Author(s)

Boris Leroy <leroy.boris@gmail.com>

Maintainer: Boris Leroy <leroy.boris@gmail.com>

References

Oksanen, J. & Minchin, P.R. (2002). Continuum theory revisited: what shape are species responses along ecological gradients? *Ecological Modelling* **157**:119-129.

See Also

linearFun, quadraticFun, custnorm

Examples

```
temp <- seq(-10, 40, length = 100)
# A curve similar to a thermal performance curve
P <- betaFun(x = temp, p1 = 0, p2 = 35, alpha = 0.9, gamma = 0.08)
plot(P ~ temp, type = "l")
```

| convertToPA | *Convert a virtual species distribution (or a suitability raster) into presence-absence* |

Description

This functions converts the probabilities of presence from the output of generateSpFromFun, generateSpFromPCA, generateRandomSp or a suitability raster into a presence-absence raster. The conversion can be threshold-based, or based on a probability of conversion (see details).

Usage

```
convertToPA(
  x,
  PA.method = "probability",
  prob.method = "logistic",
  beta = "random",
  alpha = -0.05,
  a = NULL,
  b = NULL,
  species.prevalence = NULL,
  plot = TRUE
)
```

Arguments

x	the output from functions generateSpFromFun, generateSpFromPCA or generateRandomSp, or a suitability SpatRaster
PA.method	"threshold" or "probability". If "threshold", then occurrence probabilities are simply converted into presence-absence according to the threshold beta. If "probability", then probabilities are converted according to a logistic function of threshold beta and slope alpha.
prob.method	"logistic" or "linear". Defines how the initial environmental suitability is translated into probabilities of presence/absence.
beta	"random", a numeric value in the range of your probabilities or NULL. This is the threshold of conversion into presence-absence (if PA.method = "probability" and prob.method = "logistic", then beta = the inflexion point). If "random", a numeric value will be randomly generated within the range of x.
alpha	NULL or a negative numeric value. Only useful if PA.method = "probability" and proba.method = "logistic". The value of alpha will shape the logistic function transforming occurrences into presence-absences. See logisticFun and examples therein for the choice of alpha
a	NULL or a numeric value. Only useful if PA.method = "probability" and proba.method = "linear". Slope of the linear conversion of environmental suitability.
b	NULL or a numeric value. Only useful if PA.method = "probability" and proba.method = "linear". Intercept of the linear conversion of environmental suitability.
species.prevalence	
	NULL or a numeric value between 0 and 1. The species prevalence is the proportion of sites actually occupied by the species.
plot	TRUE or FALSE. If TRUE, maps of probabilities of occurrence and presence-absence will be plotted.

Details

Online tutorial for this function

The conversion of environmental suitability into presence - absence used to be performed by selecting a threshold above which presence always occurs, and never below. However, this approach may is unrealistic because species may sometime be present in areas with a low probability of occurrence, or be absent from areas with a high probability of occurrence. In addition, when using a threshold you erase the previously generated response shapes: it all becomes threshold. Thus, this threshold approach should be avoided.

A more realistic conversion consists in converting environmental suitability into presence - absence with a probability function (see references). Such a probability conversion can be performed with two different methods here:

1. Using a logistic transformation of environmental suitability (see [`logisticFun`](#)). A logistic function on the other hand, will ensure that the simulated probability is within the 0-1 range and allow easy control of species prevalence. However, the logistic function will also flatten out the relationship at the extreme suitability values, and narrow or broaden the intermediate probability values depending on the slope of the logistic curve

2. Using a linear transformation of environmental suitability. A linear transformation will preserve the shape of the originally simulated occurrence-environment relationships, uniformly increasing or decreasing the probabilities of occurrence across the landscape.

— note —

If the Virtual Species study aims at comparing simulated and predicted probability values, it is important to recover the correct simulated probability instead of directly using the initial suitability function.

Therefore, the function stores the probability of occurrence in the output list, under the object `probability.of.occurrence`. The initial suitability function (before logistic or linear conversion) will still be stored in the output list as `suitab.raster`.

_____

PROBABILISTIC CONVERSION - LOGISTIC METHOD

To perform the logistic transformation of environmental suitability you have to define two of the three following parameters:

- `beta`: the 'threshold' of the logistic function (i.e. the inflexion point. It should normaly be in the range of values of your environmental suitability.)

- `alpha`: the slope of the logistic function. It should generally be in value equal to something like 1/20 or 1/10 of your environmental suitability range

- `species.prevalence`: the proportion of sites in which the species occur

If you provide `beta` and `alpha`, the `species.prevalence` is calculated immediately calculated after conversion into presence-absence.

On the other hand, if you provide `species.prevalence` and either `beta` or `alpha`, the function will try to determine `alpha` (if you provided `beta`) or `beta` (if you provided `alpha`).

The relationship between species prevalence, alpha and beta is dependent on the available range of environmental conditions (see Meynard and Kaplan, 2011 and especially the Supporting Information). As a consequence, the desired species prevalence may not be available for the defined `alpha` or `beta`. In these conditions, the function will retain the `alpha` or `beta` which provides the closest prevalence to your `species.prevalence`, but you may also provide another value of `alpha` or `beta` to obtain a closer prevalence.

———————————————————————————————

PROBABILISTIC CONVERSION - LINEAR METHOD

To perform the linear transformation of environmental suitability you have to define *one* of the following:

- nothing - in which case your input environmental suitability will be used as the probability of occurrence for the Bernoulli trial (it is equivalent to defining a slope a of 1 and intercept b of 0.)
- the coefficients of the linear regression: slope a and intercept b. The transformed environmental suitability will be used as the probability of occurrence for the Bernoulli trial.
- `species.prevalence`: the proportion of sites in which the species occur. In this case, the function will try to find coefficients of a linear regression which results in the requested `species.prevalence` (see below).

Method used to find coefficients of a linear regression which results in the requested `species.prevalence`:

1. The simplest linear transformation of habitat suitability would be to just multiply the raw suitability by a constant. For example, if the raw average suitability in the area is 0.04, it means an expected prevalence of 0.40. To to go from this expected prevalence of 0.04 to an expected prevalence of 0.4, we can just multiply the raw suitability by 10. It is the default choice, unless it results in probabilities superior to 1 or raw suitability have values below 0, in which case the function proceeds to method 2.
2. If it does not work, then we look at the line that passes through (min suitability, 0) and (mean suitability, desired prevalence). For this line, we only need to ensure that the maximum probability of occurence is lower than 1. Otherwise, the function proceeds to method 3.
3. If method 2 fails, then we test the line going through (mean suitability, desired prevalence) and (max suitability, 1). If the minimum probability resulting from this line is greater than 0, then this method is correct.

One of these 3 lines should always work. In fact, one of the last two has to work, and it does not hurt to try the first one which is simpler.

———————————————————————————————

In all cases, the `species.prevalence` indicated in the output is the prevalence measured on the output presence-absence map.

Value

a `list` containing 6 elements:

- `approach`: the approach used to generate the species, *i.e.*, `"response"`
- `details`: the details and parameters used to generate the species
- `suitab.raster`: the environmental suitability of your virtual species, as a Raster object
- `probability.of.occurrence`: the probability of occurrence of your species, based on the chosen transformation of environmental suitability, as a Raster object
- `PA.conversion`: the parameters used to convert the suitability into presence-absence
- `pa.raster`: the presence-absence map, as a Raster object containing 0 (absence) / 1 (presence) / NA

The structure of the virtualspecies object can be seen using `str()`

Note

The approximation of `alpha` or `beta` to the chosen `species.prevalence` may take time if you work on very large rasters.

Author(s)

Boris Leroy <leroy.boris@gmail.com>

with help from C. N. Meynard, D.M. Kaplan, C. Bellard & F. Courchamp

References

Meynard C.N. & Kaplan D.M. 2013. Using virtual species to study species distributions and model performance. *Journal of Biogeography* **40**:1-8

Meynard C.N. & Kaplan D.M. 2011. The effect of a gradual response to the environment on species distribution model performance. *Ecography* **35**:499-509

Examples

```
# Create an example stack with two environmental variables
a <- matrix(rep(dnorm(1:100, 50, sd = 25)),
            nrow = 100, ncol = 100, byrow = TRUE)
env <- c(rast(a * dnorm(1:100, 50, sd = 25)),
         rast(a * 1:100))
names(env) <- c("variable1", "variable2")

# Creation of the parameter list
parameters <- formatFunctions(variable1 = c(fun = 'dnorm', mean = 0.00012,
                                             sd = 0.0001),
                              variable2 = c(fun = 'linearFun', a = 1, b = 0))
sp1 <- generateSpFromFun(env, parameters, plot = FALSE)

# Conversion into presence-absence with a threshold-based approach
convertToPA(sp1, PA.method = "threshold", beta = 0.2,  plot = TRUE)
convertToPA(sp1, PA.method = "threshold", beta = "random", plot = TRUE)

# Conversion into presence-absence with a probability approach using logistic
# method
convertToPA(sp1, PA.method = "probability", beta = 0.4,
                 alpha = -0.05, plot = TRUE)
convertToPA(sp1, PA.method = "probability", beta = "random",
                 alpha = -0.1, plot = TRUE)

# Conversion into presence-absence with a probability approach using linear
# method
convertToPA(sp1, PA.method = "probability", prob.method = "linear",
            a = 1, b = 0, plot = TRUE)


# Conversion into presence-absence by choosing the prevalence
# Threshold method
convertToPA(sp1, PA.method = "threshold",
```

```
                    species.prevalence = 0.3, plot = TRUE)
# Logistic method, with alpha provided
convertToPA(sp1, PA.method = "probability", alpha = -0.1,
                 species.prevalence = 0.2, plot = TRUE)
# Logistic method, with beta provided
convertToPA(sp1, PA.method = "probability", beta = 0.5,
                 species.prevalence = 0.2, alpha = NULL,
               plot = TRUE)
# Linear method
convertToPA(sp1, PA.method = "probability", prob.method = "linear",
              species.prevalence = 0.2,
              plot = TRUE)
convertToPA(sp1, PA.method = "probability", prob.method = "linear",
              species.prevalence = 0.5,
              plot = TRUE)
convertToPA(sp1, PA.method = "probability", prob.method = "linear",
              species.prevalence = 0.8,
              plot = TRUE)

# Plot the output Presence-Absence raster only
sp1 <- convertToPA(sp1, plot = FALSE)
plot(sp1$pa.raster)
```

---

custnorm	*Normal function defined by extremes*

Description

A modified version of the normal function based on three parameters:

- the mean
- the absolute difference between the mean and extreme values
- the percentage of area under the curve between the specified extreme values

See the example for an easier understanding.

Usage

```
custnorm(x, mean, diff, prob)
```

Arguments

x	a numeric value or vector. The input environmental variable.
mean	a numeric value or vector. The optimum (mean) of the normal curve
diff	a numeric value or vector. The absolute difference between the mean and extremes.
prob	a numeric value or vector. The percentage of the area under the curve between the chosen extreme values

Value

a numeric value or vector resulting from the function

Author(s)

Boris Leroy <leroy.boris@gmail.com>, Florian David

Maintainer: Boris Leroy <leroy.boris@gmail.com>

Examples

```
# Let's define the response of a species to temperature which
#  - has an optimum at 20 degrees C
#  - occurs 99% of the time between 13 and 27 degrees C.
# In that case, mean = 20, diff = 7, and prob = 0.99

# First, we generate an arbitrary temperature variable
# between 0 and 30 degrees C
temp <- seq(0, 30, length = 1000)


# Then, we calculate the response to this variable with the chosen values
response <- custnorm(x = temp, mean = 20, diff = 7, prob = .99)

plot(response ~ temp, type = "l")
```

---

formatFunctions   *Format and visualise functions used to generate virtual species with*
                                 generateSpFromFun

Description

This function is a helper function to simplify the formatting of functions for generateSpFromFun

Usage

```
formatFunctions(x = NULL, rescale = TRUE, ...)
```

Arguments

x	NULL or a RasterStack. If you want to visualise the functions, provide your RasterStack here.
rescale	TRUE or FALSE. If TRUE, individual response plots are rescaled between 0 and 1 with the formula (val - min) / (max - min).
...	the parameters to be formatted. See details.

Details

This function formats the `parameters` argument of `generateSpFromFun`. For each environmental variable, provide a vector containing the function name, and its arguments.

For example, assume we want to generate a species responding to two environmental variables bio1 and bio2.

- The response to bio1 is a normal response (`dnorm`), of mean 1 and standard deviation 0.5.
- The response to bio2 is a linear response (`linearFun`), of slope (a) 2 and intercept (b) 5.

The correct writing is:

`formatFunctions( bio1 = c(fun = "dnorm", mean = 1, sd = 0.5),bio2 = c(fun = "linearFun", a = 2, b = 5))`

Warning

Do not use 'x' as a name for your environmental variables.

Author(s)

Boris Leroy <leroy.boris@gmail.com>

with help from C. N. Meynard, C. Bellard & F. Courchamp

Examples

```
my.parameters <- formatFunctions(variable1 = c(fun = 'dnorm',
                                                mean = 0.00012, sd = 0.0001),
                                 variable2 = c(fun = 'linearFun', a = 1, b = 0))


my.parameters <- formatFunctions(bio1 = c(fun = "logisticFun",
                                          alpha = -12.7, beta = 68),
                                 bio2 = c(fun = "linearFun",
                                          a = -0.03, b = 191.2),
                                 bio3 = c(fun = "dnorm",
                                          mean = 86.4, sd = 19.1),
                                 bio4 = c(fun = "logisticFun",
                                          alpha = 2198.5, beta = 11381.4))
## Not run:
# An example using worldclim data
bio1.4 <- getData('worldclim', var='bio', res=10)[[1:4]]
my.parameters <- formatFunctions(x = bio1.4,
                                 bio1 = c(fun = "logisticFun",
                                          alpha = -12.7, beta = 68),
                                 bio2 = c(fun = "linearFun",
                                          a = -0.03, b = 191.2),
                                 bio3 = c(fun = "dnorm",
                                          mean = 86.4, sd = 19.1),
                                 bio4 = c(fun = "logisticFun",
                                          alpha = 2198.5, beta = 11381.4))

## End(Not run)
```

---

generateRandomSp   *Generate a random virtual species distribution from environmental variables*

Description

This function generates randomly a virtual species distribution.

Usage

```
generateRandomSp(
  raster.stack,
  approach = "automatic",
  rescale = TRUE,
  convert.to.PA = TRUE,
  relations = c("gaussian", "linear", "logistic", "quadratic"),
  rescale.each.response = TRUE,
  realistic.sp = TRUE,
  species.type = "multiplicative",
  niche.breadth = "any",
  sample.points = FALSE,
  nb.points = 10000,
  PA.method = "probability",
  alpha = -0.1,
  adjust.alpha = TRUE,
  beta = "random",
  species.prevalence = NULL,
  plot = TRUE
)
```

Arguments

raster.stack	a SpatRaster object, in which each layer represent an environmental variable.
approach	"automatic", "random", "response" or "pca". This parameters defines how species will be generated. "automatic": If less than 6 variables in raster.stack, a response approach will be used, otherwise a pca approach will be used. "random": the approach will be randomly picked. Otherwise choose "response" or "pca". See details.
rescale	TRUE or FALSE. If TRUE, the final probability of presence is rescaled between 0 and 1.
convert.to.PA	TRUE or FALSE. If TRUE, the virtual species distribution will also be converted into Presence-Absence.
relations	[response approach] a vector containing the possible types of response function. The implemented type of relations are "gaussian", "linear", "logistic" and "quadratic".

rescale.each.response

TRUE or FALSE. If TRUE, the individual responses to each environmental variable are rescaled between 0 and 1

realistic.sp   [response approach] TRUE or FALSE. If TRUE, the function will try to define responses that can form a viable species. If FALSE, the responses will be randomly generated (may result in environmental conditions that do not co-exist).

species.type   [response approach] "additive" or "multiplicative". Defines how the final probability of presence is calculated: if "additive", responses to each variable are summed; if "multiplicative", responses are multiplied. See generateSpFromFun

niche.breadth   [pca approach] "any", "narrow" or "wide". This parameter defines how tolerant is the species regarding environmental conditions by adjusting the standard deviations of the gaussian functions. See generateSpFromPCA

sample.points   [pca approach] TRUE of FALSE. If you have a large raster file then use this parameter to sample a number of points equal to nb.points.

nb.points   [pca approach] a numeric value. Only useful if sample.points = TRUE. The number of sampled points from the raster, to perform the PCA. A too small value may not be representative of the environmental conditions in your raster.

PA.method   "threshold" or "probability". If "threshold", then occurrence probabilities are simply converted into presence-absence according to the threshold beta. If "probability", then probabilities are converted according to a logistic function of threshold beta and slope alpha.

alpha   NULL or a negative numeric value. Only useful if PA.method = "probability". The value of alpha will shape the logistic function transforming occurrences into presence-absences. See logisticFun and examples therein for the choice of alpha

adjust.alpha   TRUE or FALSE. Only useful if rescale = FALSE. If adjust.alpha = TRUE, then the value of alpha will be adjusted to an appropriate value for the range of suitabilities.

beta   "random", a numeric value in the range of your probabilities or NULL. This is the threshold of conversion into presence-absence (= the inflexion point if PA.method = "probability"). If "random", a numeric value will be randomly generated within the range of probabilities of occurrence. See convertToPA

species.prevalence

NULL or a numeric value between 0 and 1. The species prevalence is the proportion of sites actually occupied by the species. See convertToPA

plot   TRUE or FALSE. If TRUE, the generated virtual species will be plotted.

Details

Online tutorial for this function

This function generate random virtual species, either using a PCA approach, or using a response approach. In case of a response approach, only four response functions are currently used: gaussian, linear, logistic and quadratic functions.

Note that in case of numerous predictor variables, the "response" approach will not work well because it will often generate contradicting response functions (e.g., mean annual temperature optimum at degrees C and temperature of the coldest month at 10 degrees C). In these case, it is advised to use the PCA approach (by default, a PCA approach will be used if there are more than 6 predictor variables).

If `rescale.each.response = TRUE`, then the probability response to each variable will be normalised between 0 and 1 according to the following formula: P.rescaled = (P - min(P)) / (max(P) - min (P)). Similarly, if `rescale = TRUE`, the final environmental suitability will be rescaled between 0 and 1 with the same formula.

By default, the function will perform a probabilistic conversion into presence- absence, with a randomly chosen beta threshold. If you want to customise the conversion parameters, you have to define **two** of the three following parameters:

- `beta`: the 'threshold' of the logistic function (i.e. the inflexion point)
- `alpha`: the slope of the logistic function
- `species.prevalence`: the proportion of sites in which the species occur

If you provide `beta` and `alpha`, the `species.prevalence` is calculated immediately calculated after conversion into presence-absence.

As explained in [convertToPA](#), if you choose choose a precise `species.prevalence`, it may not be possible to reach this particular value because of the availability of environmental conditions. Several runs may be necessary to reach the desired `species.prevalence`.

Value

a `list` with 3 to 5 elements (depending if the conversion to presence-absence was performed):

- `approach`: the approach used to generate the species, *i.e.*, `"response"`
- `details`: the details and parameters used to generate the species
- `suitab.raster`: the virtual species distribution, as a SpatRaster object containing the environmental suitability)
- `PA.conversion`: the parameters used to convert the suitability into presence-absence
- `pa.raster`: the presence-absence map, as a SpatRaster object containing 0 (absence) / 1 (presence) / NA

The structure of the virtualspecies can object be seen using `str()`

Author(s)

Boris Leroy <leroy.boris@gmail.com>

with help from C. N. Meynard, C. Bellard & F. Courchamp

Examples

```
# Create an example stack with six environmental variables
a <- matrix(rep(dnorm(1:100, 50, sd = 25)),
            nrow = 100, ncol = 100, byrow = TRUE)
env <- c(rast(a * dnorm(1:100, 50, sd = 25)),
```

```
        rast(a * 1:100),
        rast(a * logisticFun(1:100, alpha = 10, beta = 70)),
        rast(t(a)),
        rast(exp(a)),
        rast(log(a)))
names(env) <- paste("Var", 1:6, sep = "")

# More than 6 variables: by default a PCA approach will be used
generateRandomSp(env)

# Manually choosing a response approach: this may fail because it is hard
# to find a realistic species with six distinct responses to six variables

generateRandomSp(env, approach = "response")


# Randomly choosing the approach
generateRandomSp(env, approach = "random")
```

---

generateSpFromBCA	*Generate a virtual species distribution from a Between Component Analysis of environmental variables*

Description

A Between Component Analysis is similar to a PCA, except that two sets of environmental conditions (e.g. current and future) will be used. This function is useful to generate species designed to test the extrapolation capacity of models, e.g. for climate change extrapolations

Usage

```
generateSpFromBCA(
  raster.stack.current,
  raster.stack.future,
  rescale = TRUE,
  niche.breadth = "any",
  means = NULL,
  sds = NULL,
  bca = NULL,
  sample.points = FALSE,
  nb.points = 10000,
  plot = TRUE
)
```

Arguments

raster.stack.current

a SpatRaster object, in which each layer represent an environmental variable from the "current" time horizon.

raster.stack.future

a SpatRaster object, in which each layer represent an environmental variable from a "future" time horizon.

rescale   TRUE of FALSE. Should the output suitability raster be rescaled between 0 and 1?

niche.breadth   "any", "narrow" or "wide". This parameter defines how tolerant is the species regarding environmental conditions by adjusting the standard deviations of the gaussian functions. See details.

means   a vector containing two numeric values. Will be used to define the means of the gaussian response functions to the axes of the BCA.

sds   a vector containing two numeric values. Will be used to define the standard deviations of the gaussian response functions to the axes of the BCA.

bca   a bca object. You can provide a bca object that you already computed yourself with generateSpFromBCA

sample.points   TRUE of FALSE. If you have large raster files then use this parameter to sample a number of points equal to nb.points. However the representation of your environmental variables will not be complete.

nb.points   a numeric value. Only useful if sample.points = TRUE. The number of sampled points from the raster, to perform the PCA. A too small value may not be representative of the environmental conditions in your rasters.

plot   TRUE or FALSE. If TRUE, the generated virtual species will be plotted.

Details

This function generates a virtual species distribution by computing a Between Component Analysis based on two different stacks of environmental variables. The response of the species is then simulated along the two first axes of the BCA with gaussian functions in the same way as in generateSpFromPCA.

A Between Component Analysis is used to separate two sets of environmental conditions. This function proceeds in 4 steps:

1. A Principal Component Analysis is generated based on both set of environmental conditions

2. A BCA of this PCA is generated using the function bca from package ade4. Note that at this step we choose one random point from raster.stack.future, and we use this single point as if it was a third set of environmental conditions for the BCA. This trick allows us to subtly change the shape of the bca in order to generate different types of conditions.

3. Gaussian responses to the first two axes are computed

4. These responses are multiplied to obtain the final environmental suitability

If rescale = TRUE, the final environmental suitability is rescaled between 0 and 1, with the formula (val - min) / (max - min).

The shape of gaussian responses can be randomly generated by the function or defined manually by choosing means and sds. The random generation is constrained by the argument niche.breadth, which controls the range of possible standard deviation values. This range of values is based on a fraction of the axis:

- "any": the standard deviations can have values from 1% to 50% of axes' ranges. For example if the first axis of the PCA ranges from -5 to +5, then sd values along this axis can range from 0.1 to 5.

- "narrow": the standard deviations are limited between 1% and 10% of axes' ranges. For example if the first axis of the PCA ranges from -5 to +5, then sd values along this axis can range from 0.1 to 1.

- "wide": the standard deviations are limited between 10% and 50% of axes' ranges. For example if the first axis of the PCA ranges from -5 to +5, then sd values along this axis can range from 1 to 5.

If a bca object is provided, the output bca object will contain the new environments coordinates along the provided bca axes.

Value

a list with 4 elements:

- approach: the approach used to generate the species, *i.e.*, "bca"

- details: the details and parameters used to generate the species

- suitab.raster.current: the virtual species distribution, as a SpatRaster object containing the current environmental suitability

- suitab.raster.future: the virtual species distribution, as a SpatRaster object containing the future environmental suitability

The structure of the virtualspecies object can be seen using str()

Note

To perform the BCA, the function has to transform rasters into matrices. This may not be feasible if the chosen rasters are too large for the computer's memory. In this case, you should run the function with sample.points = TRUE and set the number of points to sample with nb.points.

Author(s)

Robin Delsol, Boris Leroy

Maintainer: Boris Leroy <leroy.boris@gmail.com>

See Also

[generateSpFromFun](#) to generate a virtual species with the responses to each environmental variables.[generateSpFromPCA](#) to generate a virtual species with the PCA of environmental variables.

Examples

```
# Create two example stacks with four environmental variables each
a <- matrix(rep(dnorm(1:100, 50, sd = 25)),
            nrow = 100, ncol = 100, byrow = TRUE)

env1 <- c(rast(a * dnorm(1:100, 50, sd = 25)),
          rast(a * 1:100),
          rast(a),
          rast(t(a)))
names(env1) <- c("var1", "var2", "var3", "var4")
plot(env1) # Illustration of the variables

b <- matrix(rep(dnorm(1:100, 25, sd = 50)),
            nrow = 100, ncol = 100, byrow = TRUE)

env2 <- c(rast(b * dnorm(1:100, 50, sd = 25)),
          rast(b * 1:100),
          rast(b),
          rast(t(b)))

names(env2) <- c("var1", "var2", "var3", "var4")
plot(env2) # Illustration of the variables

# Generating a species with the BCA

generateSpFromBCA(raster.stack.current = env1, raster.stack.future = env2)

# The left part of the plot shows the BCA and the response functions along
# the two axes.
# The top-right part shows environmental suitability of the virtual
# species in the current environment.
# The bottom-right part shows environmental suitability of the virtual
# species in the future environment.


# Defining manually the response to axes

generateSpFromBCA(raster.stack.current = env1, raster.stack.future = env2,
          means = c(-2, 0),
          sds = c(0.6, 1.5))
```

---

generateSpFromFun   *Generate a virtual species distributions with responses to environmental variables*

Description

This function generates a virtual species distribution from a stack of environmental variables and a defined set of responses to each environmental parameter.

Usage

```
generateSpFromFun(
  raster.stack,
  parameters,
  rescale = TRUE,
  formula = NULL,
  species.type = "multiplicative",
  rescale.each.response = TRUE,
  plot = FALSE
)
```

Arguments

raster.stack	a SpatRaster object, in which each layer represent an environmental variable.
parameters	a list containing the functions of response of the species to environmental variables with their parameters. See details.
rescale	TRUE or FALSE. If TRUE, the final probability of presence is rescaled between 0 and 1.
formula	a character string or NULL. The formula used to combine partial responses into the final environmental suitability value (e.g., "layername1 + 2 * layername2 + layername3 * layername4 etc."). If NULL then partial responses will be added or multiplied according to species.type
species.type	"additive" or "multiplicative". Only used if formula = NULL. Defines how the final environmental suitability is calculated: if "additive", responses to each variable are summed; if "multiplicative", responses are multiplied.
rescale.each.response	
	TRUE or FALSE. If TRUE, the individual responses to each environmental variable are rescaled between 0 and 1 (see details).
plot	TRUE or FALSE. If TRUE, the generated virtual species will be plotted.

Details

[Online tutorial for this function](#)

This function proceeds in two steps:

1. The response to each environmental variable is calculated with the functions provided in parameters. This results in a suitability of each variable.

   By default, each response is rescaled between 0 and 1. Disable with rescale.each.response = FALSE

2. The final environmental suitability is calculated according to the chosen species.type.

   By default, the final suitability is rescaled between 0 and 1. Disable with rescale = FALSE

The SpatRaster stack containing environmental variables must have consistent names, because they will be checked with the parameters. For example, they can be named var1, var2, etc. Names can be checked and set with names(my.stack).

The parameters have to be carefully created, otherwise the function will not work:

- Either see [formatFunctions](#) to easily create your list of parameters
- Or create a list defined according to the following template:
  list(var1 = list(fun = 'fun1', args = list(arg1 = ..., arg2 = ..., etc.)), var2 = list(fun = 'fun2', args = list(arg1 = ..., arg2 = ..., etc.)))
  It is important to keep the same names in the parameters as in the stack of environmental variables. Similarly, argument names must be identical to argument names in the associated function (e.g., if you use fun = 'dnorm', then args should look like list(mean = 0, sd = 1)).
  See the example section below for more examples.

Any response function that can be applied to the environmental variables can be chosen here. Several functions are proposed in this package: [linearFun](#), [logisticFun](#) and [quadraticFun](#). Another classical example is the normal distribution: [stats::dnorm()](#). Users can also create and use their own functions very easily.

If rescale.each.response = TRUE, then the probability response to each variable will be normalised between 0 and 1 according to the following formula: P.rescaled = (P - min(P)) / (max(P) - min (P)) This rescaling has a strong impact on response functions, so users may prefer to use rescale.each.response = FALSE and apply their own rescaling within their response functions.

Value

a list with 3 elements:

- approach: the approach used to generate the species, *i.e.*, "response"
- details: the details and parameters used to generate the species
- suitab.raster: the raster containing the environmental suitability of the virtual species

The structure of the virtualspecies object can be seen using str()

Author(s)

Boris Leroy <leroy.boris@gmail.com>

with help from C. N. Meynard, C. Bellard & F. Courchamp

See Also

[generateSpFromPCA](#) to generate a virtual species with a PCA approach

Examples

```
# Create an example stack with two environmental variables
a <- matrix(rep(dnorm(1:100, 50, sd = 25)),
            nrow = 100, ncol = 100, byrow = TRUE)
env <- c(rast(a * dnorm(1:100, 50, sd = 25)),
         rast(a * 1:100))
names(env) <- c("variable1", "variable2")
plot(env) # Illustration of the variables

# Easy creation of the parameter list:
# see in real time the shape of the response functions
parameters <- formatFunctions(variable1 = c(fun = 'dnorm', mean = 1e-04,
```

```
                                               sd = 1e-04),
                          variable2 = c(fun = 'linearFun', a = 1, b = 0))

# If you provide env, then you can see the shape of response functions:
parameters <- formatFunctions(x = env,
                          variable1 = c(fun = 'dnorm', mean = 1e-04,
                                        sd = 1e-04),
                          variable2 = c(fun = 'linearFun', a = 1, b = 0))

# Generation of the virtual species
sp1 <- generateSpFromFun(env, parameters)
sp1
par(mfrow = c(1, 1))
plot(sp1)


# Manual creation of the parameter list
# Note that the variable names are the same as above
parameters <- list(variable1 = list(fun = 'dnorm',
                                    args = list(mean = 0.00012,
                                                sd = 0.0001)),
                   variable2 = list(fun = 'linearFun',
                                    args = list(a = 1, b = 0)))
# Generation of the virtual species
sp1 <- generateSpFromFun(env, parameters, plot = TRUE)
sp1
plot(sp1)
```

---

generateSpFromPCA	*Generate a virtual species distribution with a PCA of environmental variables*

Description

This functions generates a virtual species distribution by computing a PCA among environmental variables, and simulating the response of the species along the two first axes of the PCA. The response to axes of the PCA is determined with gaussian functions.

Usage

```
generateSpFromPCA(
  raster.stack,
  rescale = TRUE,
  niche.breadth = "any",
  axes = c(1, 2),
  means = NULL,
  sds = NULL,
  pca = NULL,
  sample.points = FALSE,
```

```
  nb.points = 10000,
  plot = TRUE
)
```

Arguments

raster.stack	a SpatRaster object, in which each layer represent an environmental variable.
rescale	TRUE or FALSE. Should the output suitability raster be rescaled between 0 and 1?
niche.breadth	"any", "narrow" or "wide". This parameter defines how tolerant is the species regarding environmental conditions by adjusting the standard deviations of the gaussian functions. See details.
axes	a vector of values. Which axes would you like to keep in your PCA? At least 2 axes should be included (Only 1 axis currently not supported)
means	a vector containing as many numeric values as axes. Will be used to define the means of the gaussian response functions to the axes of the PCA.
sds	a vector containing as many numeric values as axes. Will be used to define the standard deviations of the gaussian response functions to the axes of the PCA.
pca	a dudi.pca object. You can provide a pca object that you computed yourself with dudi.pca
sample.points	TRUE of FALSE. If you have a large raster file then use this parameter to sample a number of points equal to nb.points.
nb.points	a numeric value. Only useful if sample.points = TRUE. The number of sampled points from the raster, to perform the PCA. A too small value may not be representative of the environmental conditions in your raster.
plot	TRUE or FALSE. If TRUE, the generated virtual species will be plotted.

Details

Online tutorial for this function

This function proceeds in 3 steps:

1. A PCA of environmental conditions is generated
2. Gaussian responses to the first two axes are computed
3. These responses are multiplied to obtain the final environmental suitability

If rescale = TRUE, the final environmental suitability is rescaled between 0 and 1, with the formula (val - min) / (max - min).

The shape of gaussian responses can be randomly generated by the function or defined manually by choosing means and sds. The random generation is constrained by the argument niche.breadth, which controls the range of possible standard deviation values. This range of values is based on a fraction of the axis:

- "any": the standard deviations can have values from 1% to 50% of axes' ranges. For example if the first axis of the PCA ranges from -5 to +5, then sd values along this axis can range from 0.1 to 5.

- "narrow": the standard deviations are limited between 1% and 10% of axes' ranges. For example if the first axis of the PCA ranges from -5 to +5, then sd values along this axis can range from 0.1 to 1.

- "wide": the standard deviations are limited between 10% and 50% of axes' ranges. For example if the first axis of the PCA ranges from -5 to +5, then sd values along this axis can range from 1 to 5.

Value

a `list` with 3 elements:

- approach: the approach used to generate the species, *i.e.*, "pca"
- details: the details and parameters used to generate the species
- suitab.raster: the virtual species distribution, as a SpatRaster object containing the environmental suitability

The structure of the virtualspecies object can be seen using `str()`

Note

To perform the PCA, the function has to transform the raster into a matrix. This may not be feasible if the raster is too large for the computer's memory. In this case, you should perform the PCA on a sample of your raster with set `sample.points = TRUE` and choose the number of points to sample with `nb.points`.

Author(s)

Boris Leroy <leroy.boris@gmail.com>

with help from C. N. Meynard, C. Bellard & F. Courchamp

See Also

[generateSpFromFun](#) to generate a virtual species with the responses to each environmental variables.

Examples

```
# Create an example stack with four environmental variables
a <- matrix(rep(dnorm(1:100, 50, sd = 25)),
            nrow = 100, ncol = 100, byrow = TRUE)
env <- c(rast(a * dnorm(1:100, 50, sd = 25)),
             rast(a * 1:100),
             rast(a * logisticFun(1:100, alpha = 10, beta = 70)),
             rast(t(a)))
names(env) <- c("var1", "var2", "var3", "var4")
plot(env) # Illustration of the variables
```

```
# Generating a species with the PCA

generateSpFromPCA(raster.stack = env)

# The top part of the plot shows the PCA and the response functions along
# the two axes.
# The bottom part shows the probabilities of occurrence of the virtual
# species.




# Defining manually the response to axes

generateSpFromPCA(raster.stack = env,
          means = c(-2, 0),
          sds = c(0.6, 1.5))

# This species can be seen as occupying intermediate altitude ranges of a
# conic mountain.


# Beyond the first two axes
generateSpFromPCA(raster.stack = env,
                axes = c(1, 3))

sp <- generateSpFromPCA(raster.stack = env,
                axes = 1:3)
plotResponse(sp, axes = c(1, 2))
plotResponse(sp, axes = c(1, 3))
plotResponse(sp, axes = c(2, 3))
```

---

limitDistribution   *Limit a virtual species distribution to a defined area*

Description

This function is designed to limit species distributions to a subsample of their total distribution range. It will thus generate a species which is not at the equilibrium with its environment (i.e., which did not occupy the full range of suitable environmental conditions).

This function basically takes any type of raster and will limit values above 0 to areas where the species is allowed to disperse.

Usage

```
limitDistribution(x, geographical.limit = "extent", area = NULL, plot = TRUE)
```

Arguments

x   a SpatRaster object composed of 0, 1 and NA, or the output list from generateSpFromFun,
                   generateSpFromPCA or generateRandomSp

geographical.limit
                   "country", "region", "continent", "polygon", "raster" or "extent". The
                   method used to limit the distribution range: see details.

area   NULL, a character string, a polygon, a raster or an extent object. The area in
                   which the distribution range will be limited: see details. If NULL and geographical.limit
                   = "extent", then you will be asked to draw an extent on the map.

plot   TRUE or FALSE. If TRUE, the resulting limited distribution will be plotted.

Details

Online tutorial for this function

How the function works:

The function will remove occurrences of the species outside the chosen area:

- NA are kept unchanged
- 0 are kept unchanged
- values > 0 within the limits of area are kept unchanged
- values > 0 outside the limits of area are set to 0

How to define the area in which the range is limited:

You can choose to limit the distribution range of the species to:

1. a particular country, region or continent (assuming your raster has the WGS84 projection):

   Set the argument geographical.limit to "country", "region" or "continent", and pro-
   vide the name(s) of the associated countries, regions or continents to area (see examples).

   List of possible area names:

   - Countries: type unique(rnaturalearth::ne_countries(returnclass ='sf')$sovereignt)
     in the console
   - Regions: "Africa", "Antarctica", "Asia", "Oceania", "Europe", "Americas"
   - Continents: "Africa", "Antarctica", "Asia", "Europe", "North America", "Oceania", "South
     America"

2. a polygon:

   Set geographical.limit to "polygon", and provide your polygon to area.

3. a raster:

   Set geographical.limit to "raster", and provide your raster to area. Your raster values
   should be 1 (suitable area), 0 (unsuitable area) or NA (outside your mask).

4. an extent object:

   Set geographical.limit to "extent", and either provide your extent object to area, or leave
   it NULL to draw an extent on the map.

Value

a `list` containing 7 elements:

- `approach`: the approach used to generate the species, *i.e.*, `"response"`
- `details`: the details and parameters used to generate the species
- `suitab.raster`: the virtual species distribution, as a Raster object containing the environmental suitability)
- `PA.conversion`: the parameters used to convert the suitability into presence-absence
- `pa.raster`: the presence-absence map, as a Raster object containing 0 (absence) / 1 (presence) / NA
- `geographical.limit`: the method used to limit the distribution and the area in which the distribution is restricted
- `occupied.area`: the area occupied by the virtual species as a Raster of presence-absence

The structure of the virtualspecies object can be seen using `str()`

Author(s)

Boris Leroy <leroy.boris@gmail.com>

with help from C. N. Meynard, C. Bellard & F. Courchamp

Examples

```
# Create an example stack with six environmental variables
a <- matrix(rep(dnorm(1:100, 50, sd = 25)),
            nrow = 100, ncol = 100, byrow = TRUE)
env <- c(rast(a * dnorm(1:100, 50, sd = 25)),
         rast(a * 1:100),
         rast(a * logisticFun(1:100, alpha = 10, beta = 70)),
         rast(t(a)),
         rast(exp(a)),
         rast(log(a)))
names(env) <- paste("Var", 1:6, sep = "")

# More than 6 variables: by default a PCA approach will be used
sp <- generateRandomSp(env)

# limiting the distribution to a specific extent
limit <- ext(1, 50, 1, 50)

limitDistribution(sp, area = limit)


# Example of a raster of habitat patches
habitat.raster <- setValues(sp$pa.raster,
                            sample(c(0, 1), size = ncell(sp$pa.raster),
                            replace = TRUE))

plot(habitat.raster) # 1 = suitable habitat; 0 = unsuitable habitat
```

```
sp <- limitDistribution(sp, geographical.limit = "raster", area = habitat.raster)
par(mfrow = c(2, 1))
plot(sp$pa.raster)
plot(sp$occupied.area) # Species could not occur in many cells because
# habitat patches were unsuitable
```

---

linearFun *Linear function*

Description

A simple linear function of the form

$$ax + b$$

Usage

```
linearFun(x, a, b)
```

Arguments

x	a numeric value or vector
a	a numeric value or vector
b	a numeric value or vector

Value

a numeric value or vector resulting from the function

Author(s)

Boris Leroy <leroy.boris@gmail.com>

Maintainer: Boris Leroy <leroy.boris@gmail.com>

See Also

logisticFun, quadraticFun

Examples

```
x <- 1:100
y <- linearFun(x, a = 0.5, b = 0)
plot(y ~ x, type = "l")
```

---

logisticFun   *Logistic function*

Description

A simple logistic function of the form

$$\frac{1}{1 + e^{\frac{x - \beta}{\alpha}}}$$

Usage

```
logisticFun(x, alpha, beta)
```

Arguments

x	a numeric value or vector
alpha	a numeric value or vector
beta	a numeric value or vector

Details

The value of beta determines the 'threshold' of the logistic curve (i.e. the inflexion point).

The value of alpha determines the slope of the curve (see examples):

- alpha very close to 0 will result in a threshold-like response.
- Values of alpha with the same order of magnitude as the range of x (e.g., the range of x / 10) will result in a logistic function.
- alpha very far from 0 will result in a linear function.

Value

a numeric value or vector resulting from the function

Author(s)

Boris Leroy <leroy.boris@gmail.com>

Maintainer: Boris Leroy <leroy.boris@gmail.com>

See Also

[linearFun](), [quadraticFun]()

Examples

```
x <- 1:100
y <- logisticFun(x, alpha = -10, b = 50)
plot(y ~ x, type = "l")

# The effect of alpha:
y1 <- logisticFun(x, alpha = -0.01, b = 50)
y2 <- logisticFun(x, alpha = -10, b = 50)
y3 <- logisticFun(x, alpha = -1000, b = 50)

par(mfrow = c(1, 3))
plot(y1 ~ x, type = "l", main = expression(alpha %->% 0))
plot(y2 ~ x, type = "l", main = expression(alpha %~~% range(x)/10))
plot(y3 ~ x, type = "l", main = expression(alpha %->% infinity))
```

---

plotResponse   *Visualise the response of the virtual species to environmental variables*

Description

This function plots the relationships between the virtual species and the environmental variables. It requires either the output from generateSpFromFun, generateSpFromPCA, generateRandomSp, or a manually defined set of environmental variables and response functions.

Usage

```
plotResponse(
  x,
  parameters = NULL,
  approach = NULL,
  rescale = NULL,
  axes.to.plot = NULL,
  no.plot.reset = FALSE,
  rescale.each.response = NULL,
  ...
)
```

Arguments

x	the output from generateSpFromFun, generateSpFromPCA, generateRandomSp, or a raster layer/stack of environmental variables (see details for the latter).
parameters	in case of manually defined response functions, a list containing the associated parameters. See details.
approach	in case of manually defined response functions, the chosen approach: either "response" for a per-variable response approach, or "pca" for a PCA approach.
rescale	TRUE or FALSE. If TRUE, individual response plots are rescaled between 0 and 1.

axes.to.plot   a vector of 2 values listing the two axes of the PCA to plot. Only useful for a
                  PCA species.

no.plot.reset   TRUE or FALSE. If FALSE, the plot window will be reset to its initial state after
                  the response has been plotted.

rescale.each.response

                  TRUE or FALSE. If TRUE, the individual responses to each environmental variable
                  are rescaled between 0 and 1.

...   further arguments to be passed to plot. See plot and [par](par).

Details

If you provide the output from generateSpFromFun, generateSpFromPCA or generateRandomSp
then the function will automatically make the appropriate plots.

Otherwise, you can provide a raster layer/stack of environmental variables to x and a list of functions
to parameters to perform the plot. In that case, you have to specify the approach: ″reponse″ or
″PCA″:

- if approach = ″response″: Provide to parameters a list exactly as defined in generateSpFromFun:
  list(var1 = list(fun = 'fun1', args = list(arg1 = ..., arg2 = ..., etc.)), var2 = list(fun
  = 'fun2', args = list(arg1 = ..., arg2 = ..., etc.)))

- if approach = ″PCA″: Provide to parameters a list containing the following elements:
  - pca: a dudi.pca object computed with dudi.pca
  - means: a vector containing two numeric values. Will be used to define the means of the
    gaussian response functions to the axes of the PCA.
  - sds a vector containing two numeric values. Will be used to define the standard deviations
    of the gaussian response functions to the axes of the PCA.

Author(s)

Boris Leroy <leroy.boris@gmail.com>

with help from C. N. Meynard, C. Bellard & F. Courchamp

Examples

```
# Create an example stack with four environmental variables
a <- matrix(rep(dnorm(1:100, 50, sd = 25)),
            nrow = 100, ncol = 100, byrow = TRUE)
env <- c(rast(a * dnorm(1:100, 50, sd = 25)),
         rast(a * 1:100),
         rast(a * logisticFun(1:100, alpha = 10, beta = 70)),
         rast(t(a)))
names(env) <- c(″var1″, ″var2″, ″var3″, ″var4″)

# Per-variable response approach:
parameters <- formatFunctions(var1 = c(fun = 'dnorm', mean = 0.00012,
                                       sd = 0.0001),
                              var2 = c(fun = 'linearFun', a = 1, b = 0),
```

```
                                    var3 = c(fun = 'quadraticFun', a = -20, b = 0.2,
                                             c = 0),
                                    var4 = c(fun = 'logisticFun', alpha = -0.001,
                                             beta = 0.002))
sp1 <- generateSpFromFun(env, parameters, plot = TRUE)
plotResponse(sp1)

# PCA approach:
sp2 <- generateSpFromPCA(env, plot = FALSE)
par(mfrow = c(1, 1))
plotResponse(sp2)
```

---

plotSuitabilityToProba

> *Visualise the function that was used to transform environmental suitability into probability of occurrence*

Description

This function plots the relationships between the environmental suitability and the probability of occurrence, which is used to generate the presence- absence distribution. It requires the output from convertToPA.

Usage

```
plotSuitabilityToProba(sp, add = FALSE, ...)
```

Arguments

sp	the output from convertToPA.
add	TRUE or FALSE. If TRUE, the relationship will be added to the currently active graph.
...	further arguments to be passed to plot. See plot and par.

Author(s)

Boris Leroy <leroy.boris@gmail.com>

Examples

```
# Create an example stack with two environmental variables
a <- matrix(rep(dnorm(1:100, 50, sd = 25)),
            nrow = 100, ncol = 100, byrow = TRUE)
env <- c(rast(a * dnorm(1:100, 50, sd = 25)),
         rast(a * 1:100))
names(env) <- c("variable1", "variable2")
```

```
parameters <- formatFunctions(variable1 = c(fun = 'dnorm', mean = 1e-04,
                                             sd = 1e-04),
                              variable2 = c(fun = 'linearFun', a = 1, b = 0))
# Generation of the virtual species
sp1 <- generateSpFromFun(env, parameters)
sp1


# Converting to presence-absence, probablistic method, logistic conversion
# A species with a low prevalence:

sp1.lowprev <- convertToPA(sp1, species.prevalence = 0.1)
plotSuitabilityToProba(sp1.lowprev)

# A species with a high prevalence:

sp1.highprev <- convertToPA(sp1, species.prevalence = 0.9)
plotSuitabilityToProba(sp1.lowprev)

# Converting to presence-absence, probablistic method, linear conversion
# A species with a low prevalence:

sp1.lowprev <- convertToPA(sp1, species.prevalence = 0.1,
                           prob.method = "linear")
plotSuitabilityToProba(sp1.highprev)

# A species with a high prevalence:

sp1.highprev <- convertToPA(sp1, species.prevalence = 0.9,
                            prob.method = "linear")
plotSuitabilityToProba(sp1.highprev)
```

---

quadraticFun   *Quadratic function*

Description

A simple quadratic function of the form

$$ax^2 + bx + c$$

Usage

```
quadraticFun(x, a, b, c)
```

Arguments

x	a numeric value or vector
a	a numeric value or vector

b   a numeric value or vector

c   a numeric value or vector

Value

a numeric value or vector resulting from the function

Author(s)

Boris Leroy <leroy.boris@gmail.com>

Maintainer: Boris Leroy <leroy.boris@gmail.com>

See Also

[linearFun](), [quadraticFun]()

Examples

```
x <- 1:100
y <- quadraticFun(x, a = 2, b = 2, c = 3)
plot(y ~ x, type = "l")
```

---

removeCollinearity   *Remove collinearity among variables of a raster stack*

Description

This functions analyses the correlation among variables of the provided stack of environmental variables (using Pearson's R), and can return a vector containing names of variables that are not colinear, or a list containing grouping variables according to their degree of collinearity.

Usage

```
removeCollinearity(
  raster.stack,
  multicollinearity.cutoff = 0.7,
  select.variables = FALSE,
  sample.points = FALSE,
  nb.points = 10000,
  plot = FALSE,
  method = "pearson"
)
```

Arguments

raster.stack   a SpatRaster object, in which each layer represent an environmental variable.

multicollinearity.cutoff

         a numeric value corresponding to the cutoff of correlation above which to group variables.

select.variables

         TRUE or FALSE. If TRUE, then the function will choose one variable among each group to return a vector of non correlated variables (see details). If FALSE, the function will return a list containing the groups of correlated variables.

sample.points   TRUE or FALSE. If you have a large raster file then use this parameter to sample a number of points equal to nb.points.

nb.points   a numeric value. Only useful if sample.points = TRUE. The number of sampled points from the raster, to perform the PCA. A too small value may not be representative of the environmental conditions in your raster.

plot   TRUE or FALSE. If TRUE, the hierarchical ascendant classification used to group variables will be plotted.

method   "pearson", "spearman" or "kendall". The correlation method to be used. If your variables are skewed or have outliers (e.g. when working with precipitation variables) you should favour the Spearman or Kendall methods.

Details

This function uses the Pearson's correlation coefficient to analyse correlation among variables. This coefficient is then used to compute a distance matrix, which in turn is used it compute an ascendant hierarchical classification, with the '*complete*' method (see [hclust](#)). If at least one correlation above the multicollinearity.cutoff is detected, then the variables will be grouped according to their degree of correlation.

If select.variables = TRUE, then the function will return a vector containing variables that are not colinear. The variables not correlated to any other variables are automatically included in this vector. For each group of colinear variables, one variable will be randomly chosen and included in this vector.

Value

a vector of non correlated variables, or a list where each element is a group of non correlated variables.

Author(s)

Boris Leroy <leroy.boris@gmail.com>

with help from C. N. Meynard, C. Bellard & F. Courchamp

Examples

```
# Create an example stack with six environmental variables
a <- matrix(rep(dnorm(1:100, 50, sd = 25)),
            nrow = 100, ncol = 100, byrow = TRUE)
```

```
env <- c(rast(a * dnorm(1:100, 50, sd = 25)),
         rast(a * 1:100),
         rast(a * logisticFun(1:100, alpha = 10, beta = 70)),
         rast(t(a)),
         rast(exp(a)),
         rast(log(a)))
names(env) <- paste("Var", 1:6, sep = "")

# Defaults settings: cutoff at 0.7
removeCollinearity(env, plot = TRUE)

# Changing cutoff to 0.5
removeCollinearity(env, plot = TRUE, multicollinearity.cutoff = 0.5)

# Automatic selection of variables not intercorrelated
removeCollinearity(env, plot = TRUE, select.variables = TRUE)

# Assuming a very large raster file: selecting a subset of points
removeCollinearity(env, plot = TRUE, select.variables = TRUE,
                   sample.points = TRUE, nb.points = 5000)
```

---

sampleOccurrences   *Sample occurrences in a virtual species distribution*

Description

This function samples occurrences/records (presence only or presence-absence) within a species distribution, either randomly or with a sampling bias. The sampling bias can be defined manually or with a set of predefined biases.

Usage

```
sampleOccurrences(
  x,
  n,
  type = "presence only",
  extract.probability = FALSE,
  sampling.area = NULL,
  detection.probability = 1,
  correct.by.suitability = FALSE,
  error.probability = 0,
  bias = "no.bias",
  bias.strength = 50,
  bias.area = NULL,
  weights = NULL,
  sample.prevalence = NULL,
  replacement = FALSE,
```

```
    plot = TRUE
)
```

Arguments

x	a `SpatRaster` object or the output list from `generateSpFromFun`, `generateSpFromPCA`, `generateRandomSp`, `convertToPA` or `limitDistribution` The raster must contain values of 0 or 1 (or NA).
n	an integer. The number of occurrence points / records to sample.
type	`"presence only"` or `"presence-absence"`. The type of occurrence points to sample.
extract.probability	
	TRUE or FALSE. If TRUE, then true probability at sampled locations will also be extracted
sampling.area	a character string, a `polygon` or an `extent` object. The area in which the sampling will take place. See details.
detection.probability	
	a numeric value between 0 and 1, corresponding to the probability of detection of the species. See details.
correct.by.suitability	
	TRUE or FALSE. If TRUE, then the probability of detection will be weighted by the suitability, such that cells with lower suitabilities will further decrease the chance that the species is detected when sampled. NOTE: this will NOT increase likelihood of samplings in areas of high suitability. In this case look for argument weights.
error.probability	
	TRUE or FALSE. Probability to attribute an erroneous presence (False Positive) in cells where the species is actually absent.
bias	`"no.bias"`, `"country"`, `"region"`, `"extent"`, `"polygon"` or `"manual"`. The method used to generate a sampling bias: see details.
bias.strength	a positive numeric value. The strength of the bias to be applied in area (as a multiplier). Above 1, area will be oversampled. Below 1, area will be undersampled.
bias.area	NULL, a character string, a `polygon` or an `extent` object. The area in which the sampling will be biased: see details. If NULL and bias = `"extent"`, then you will be asked to draw an extent on the map.
weights	NULL or a raster layer. Only used if bias = `"manual"`. The raster of bias weights to be applied to the sampling of occurrences. Higher weights mean a higher probability of sampling. For example, species suitability raster can be entered here to increase likelihood of sampling occurrences in areas with high suitability.
sample.prevalence	
	NULL or a numeric value between 0 and 1. Only useful if type = `"presence-absence"`. Defines the sample prevalence, i.e. the proportion of presences sampled. Note that the probabilities of detection and error are applied AFTER this parameter, so the final sample prevalence may not different if you apply probabilities of detection and/or error

| replacement | TRUE or FALSE. If TRUE, multiple samples can occur in the same cell. Can be useful to mimic real datasets where samplings can be duplicated or repeated in time. |
| plot | TRUE or FALSE. If TRUE, the sampled occurrence points will be plotted. |

Details

Online tutorial for this function

How the function works:

The function randomly selects n cells in which samples occur. If a bias is chosen, then the selection of these cells will be biased according to the type and strength of bias chosen. If the sampling is of type "presence only", then only cells where the species is present will be chosen. If the sampling is of type "presence-absence", then all non-NA cells can be chosen.

The function then samples the species inside the chosen cells. In cells where the species is present the species will always be sampled unless the parameter detection.probability is lower than 1. In that case the species will be sampled with the associated probability of detection.

In cells where the species is absent (in case of a "presence-absence" sampling), the function will always assign absence unless error.probability is greater than 1. In that case, the species can be found present with the associated probability of error. Note that this step happens AFTER the detection step. Hence, in cells where the species is present but not detected, it can still be sampled due to a sampling error.

How to restrict the sampling area:

Use the argument sampling.area:

- Provide the name (s) (or a combination of names) of country(ies), region(s) or continent(s). Examples:
  - sampling.area = "Africa"
  - sampling.area = c("Africa", "North America", "France")
- Provide a polygon (SpatialPolygons or SpatialPolygonsDataFrame of package sp)
- Provide an extent object

How the sampling bias works:

The argument bias.strength indicates the strength of the bias. For example, a value of 50 will result in 50 times more samples within the bias.area than outside. Conversely, a value of 0.5 will result in half less samples within the bias.area than outside.

How to choose where the sampling is biased:

You can choose to bias the sampling in:

1. a particular country, region or continent (assuming your raster has the WGS84 projection):
   Set the argument bias to "country", "region" or "continent", and provide the name(s) of the associated countries, regions or continents to bias.area (see examples).
   List of possible bias.area names:
   - Countries: type unique(rnaturalearth::ne_countries(returnclass = 'sf')$sovereignt) in the console
   - Regions: "Africa", "Antarctica", "Asia", "Oceania", "Europe", "Americas"

    • Continents: "Africa", "Antarctica", "Asia", "Europe", "North America", "Oceania", "South America"

2. a polygon:

    Set bias to "polygon", and provide your polygon to area.

3. an extent object:

    Set bias to "extent", and either provide your extent object to bias.area, or leave it NULL to draw an extent on the map.

Otherwise you can enter a raster of sampling probability. It can be useful if you want to increase likelihood of samplings in areas of high suitability (simply enter the suitability raster in weights; see examples below), or if you want to define sampling biases manually, *e.g.* to to create biases along roads. In that case you have to provide to weights a raster layer in which each cell contains the probability to be sampled.

The .Random.seed and RNGkind are stored as attributes when the function is called, and can be used to reproduce the results as shown in the examples (though it is preferable to set the seed with set.seed before calling sampleOccurrences() and to then use the same value in set.seed to reproduce results later. Note that reproducing the sampling will only work if the same original distribution map is used.

Value

a list with 8 elements:

- type: type of occurrence sampled (presence-absences or presence-only)
- sample.points: data.frame containing the coordinates of samples, true and sampled observations (i.e, 1, 0 or NA), and, if asked, the true environmental suitability in sampled locations
- detection.probability: the chosen probability of detection of the virtual species
- error.probability: the chosen probability to assign presence in cells where the species is absent
- bias: if a bias was chosen, then the type of bias and the associated area will be included.
- replacement: indicates whether multiple samples could occur in the same cells
- original.distribution.raster: the distribution raster from which samples were drawn
- sample.plot: a recorded plot showing the sampled points overlaying the original distribution.

Note

Setting sample.prevalence may at least partly override bias, e.g. if bias is specified with extent to an area that contains no presences, but sample prevalence is set to > 0, then cells outside of the biased sampling extent will be sampled until the number of presences required by sample.prevalence are obtained, after which the sampling of absences will proceed according to the specified bias.

Author(s)

Boris Leroy <leroy.boris@gmail.com> Willson Gaul <wgaul@hotmail.com>

with help from C. N. Meynard, C. Bellard & F. Courchamp

Examples

```
# Create an example stack with six environmental variables
a <- matrix(rep(dnorm(1:100, 50, sd = 25)),
            nrow = 100, ncol = 100, byrow = TRUE)
env <- c(rast(a * dnorm(1:100, 50, sd = 25)),
         rast(a * 1:100),
         rast(a * logisticFun(1:100, alpha = 10, beta = 70)),
         rast(t(a)),
         rast(exp(a)),
         rast(log(a)))
names(env) <- paste("Var", 1:6, sep = "")

# More than 6 variables: by default a PCA approach will be used
sp <- generateRandomSp(env, niche.breadth = "wide")

# Sampling of 25 presences
sampleOccurrences(sp, n = 25)

# Sampling of 30 presences and absences
sampleOccurrences(sp, n = 30, type = "presence-absence")


# Reducing of the probability of detection
sampleOccurrences(sp, n = 30, type = "presence-absence",
                  detection.probability = 0.5)

# Further reducing in relation to environmental suitability
sampleOccurrences(sp, n = 30, type = "presence-absence",
                  detection.probability = 0.5,
                  correct.by.suitability = TRUE)


# Creating sampling errors (far too much)
sampleOccurrences(sp, n = 30, type = "presence-absence",
                  error.probability = 0.5)

# Introducing a sampling bias (oversampling)
biased.area <- ext(1, 50, 1, 50)
sampleOccurrences(sp, n = 50, type = "presence-absence",
                  bias = "extent",
                  bias.area = biased.area)
# Showing the area in which the sampling is biased
plot(biased.area, add = TRUE)

# Introducing a sampling bias (no sampling at all in the chosen area)
biased.area <- ext(1, 50, 1, 50)
sampleOccurrences(sp, n = 50, type = "presence-absence",
                  bias = "extent",
                  bias.strength = 0,
                  bias.area = biased.area)
# Showing the area in which the sampling is biased
plot(biased.area, add = TRUE)
```

```
samps <- sampleOccurrences(sp, n = 50,
                           bias = "manual",
                           weights = sp$suitab.raster)
plot(sp$suitab.raster)
points(samps$sample.points[, c("x", "y")])

# Create a sampling bias so that more presences are sampled in areas with
# higher suitability




# Reproduce sampling based on the saved .Random.seed from a previous result
samps <- sampleOccurrences(sp, n = 100,
                           type = "presence-absence",
                           detection.probability = 0.7,
                           bias = "extent",
                           bias.strength = 50,
                           bias.area = biased.area)
# Reset the random seed using the value saved in the attributes
.Random.seed <- attr(samps, "seed")
reproduced_samps <- sampleOccurrences(sp, n = 100,
                                      type = "presence-absence",
                                      detection.probability = 0.7,
                                      bias = "extent",
                                      bias.strength = 50,
                                      bias.area = biased.area)
identical(samps$sample.points, reproduced_samps$sample.points)
```

---

synchroniseNA   *Synchronise NA values among layers of a stack*

Description

This function ensures that cells containing NAs are the same among all the layers of a raster stack, i.e.that for any given pixel of the stack, if one layer has a NA, then all layers should be set to NA for that pixel.

Usage

```
synchroniseNA(x)
```

Arguments

x   a raster stack object which needs to be synchronised.

Details

This function can do that in two different ways; if your computer has enough RAM a fast way will be used; otherwise a slower but memory-safe way will be used.

Author(s)

Boris Leroy <leroy.boris@gmail.com>

with help from C. N. Meynard, C. Bellard & F. Courchamp

Examples

```
# Creation of a stack with different NAs across layers
m <- matrix(nr = 10, nc = 10, 1:100)
r1 <- rast(m)
r2 <- rast(m)
r1[sample(1:ncell(r1), 20)] <- NA
r2[sample(1:ncell(r2), 20)] <- NA
s <- c(r1, r2)


# Effect of the synchroniseNA() function
plot(s) # Not yet synchronised
s <- synchroniseNA(s)
plot(s) # Synchronised
```

Index