

Package: NLMR (via r-universe)

August 4, 2026

Type Package

Title Simulating Neutral Landscape Models

Version 1.2.0

Description Provides neutral landscape models

([<doi:10.1007/BF02275262>](https://doi.org/10.1007/BF02275262),
[<https://sci-hub.in/10.1007/bf02275262>](https://sci-hub.in/10.1007/bf02275262)). Neutral landscape models range from ``hard" neutral models (completely random distributed), to ``soft" neutral models (definable spatial characteristics) and generate landscape patterns that are independent of ecological processes. Thus, these patterns can be used as null models in landscape ecology. 'NLMR' combines a large number of algorithms from other published software for simulating neutral landscapes. The simulation results are obtained in a spatial data format (raster* objects from the 'raster' package) and can, therefore, be used in any sort of raster data operation that is performed with standard observation data.

License GPL-3

URL <https://docs.ropensci.org/NLMR/>

BugReports <https://github.com/ropensci/NLMR/issues/>

Depends R (>= 3.1.0)

Imports checkmate, dplyr, fasterize, raster, Rcpp, sf,
spatstat.random, spatstat.geom, stats, tibble

Suggests igrph, knitr, kableExtra, landscapemetrics, purrr,
rmarkdown, testthat

LinkingTo Rcpp

VignetteBuilder knitr

ByteCompile true

Encoding UTF-8

Roxygen list(roclets = c("`rd", "`namespace", "`collate"))

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev
Repository <https://ropensci.r-universe.dev>
Date/Publication 2026-07-05 11:07:37 UTC
RemoteUrl <https://github.com/ropensci/NLMR>
RemoteRef main
RemoteSha 7d5434d6c50b571d95c742f6e73f678c3f26636f

Contents

nlm_curds	2
nlm_distancegradient	4
nlm_edgegradient	5
nlm_fbm	6
nlm_gaussianfield	8
nlm_mosaicfield	9
nlm_mosaicgibbs	11
nlm_mosaictess	12
nlm_mpd	14
nlm_neigh	15
nlm_percolation	17
nlm_planargradient	18
nlm_random	20
nlm_randomcluster	21
nlm_randomrectangularcluster	22
util_classify	24
Index	26

nlm_curds	<i>nlm_curds</i>
-----------	------------------

Description

Simulates a random curd neutral landscape model with optional wheys.

Usage

```
nlm_curds(  
  curds,  
  recursion_steps,  
  wheyes = NULL,  
  resolution = 1,  
  user_seed = NULL  
)
```

Arguments

curds	[numerical(x)] Vector with percentage/s to fill with curds (fill with habitat (value == TRUE)).
recursion_steps	[numerical(x)] Vector of successive cutting steps for the blocks (split 1 block into x blocks).
wheyes	[numerical(x)] Vector with percentage/s to fill with wheys, which fill matrix in an additional step with habitat.
resolution	[numerical(1)] Resolution of the resulting raster.
user_seed	[numerical(1)] Set random seed for the simulation.

Details

Random curdling recursively subdivides the plane into blocks. At each level of the recursion, a fraction of the blocks are declared as habitat (value == TRUE) while the remaining blocks continue to be defined as matrix (value == FALSE) and enter the next recursive cycle.

The optional argument (wheyes) allows wheys to be added, in which a set proportion of cells that were declared matrix (value == FALSE) during recursion, are now set as habitat cells (value == TRUE).

If

$$curds_1 = curds_2 = recursion_steps_2 = \dots = curds_n = recursion_steps_n$$

the models resembles a binary random map.

Note that you can not set ncol and nrow with this landscape algorithm. The amount of cells and hence dimension of the raster is given by the vector product of the recursive steps.

Value

raster

References

- Keitt TH. 2000. Spectral representation of neutral landscapes. *Landscape Ecology* 15:479-493.
- Szaro, Robert C., and David W. Johnston, eds. Biodiversity in managed landscapes: theory and practice. *Oxford University Press*, USA, 1996.

Examples

```
# simulate random curdling
(random_curdling <- nlm_curds(curds = c(0.5, 0.3, 0.6),
                             recursion_steps = c(32, 6, 2)))

# simulate wheyed curdling
(wheyed_curdling <- nlm_curds(curds = c(0.5, 0.3, 0.6),
```

```

recursion_steps = c(32, 6, 2),
wheyes = c(0.1, 0.05, 0.2)))

## Not run:
# Visualize the NLMs
raster::plot(random_curdling)
raster::plot(wheyed_curdling)

## End(Not run)

```

nlm_distancegradient *nlm_distancegradient*

Description

Simulates a distance-gradient neutral landscape model.

Usage

```
nlm_distancegradient(ncol, nrow, resolution = 1, origin, rescale = TRUE)
```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
origin	[numerical(4)] Edge coordinates of the origin (raster::extent with xmin, xmax, ymin, ymax) of the distance measurement.
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1. Otherwise, the distance in raster units is calculated.

Details

The function takes the number of columns and rows as input and creates a RasterLayer with the same extent. Origin is a numeric vector of xmin, xmax, ymin, ymax for a rectangle inside the raster from which the distance is measured.

Value

RasterLayer

See Also

[nlm_edgegradient](#), [nlm_planargradient](#)

Examples

```
# simulate a distance gradient
distance_gradient <- nlm_distancegradient(ncol = 100, nrow = 100,
                                         origin = c(20, 30, 10, 15))

## Not run:
# visualize the NLM
raster::plot(distance_gradient)

## End(Not run)
```

nlm_edgegradient	<i>nlm_edgegradient</i>
------------------	-------------------------

Description

Simulates an edge-gradient neutral landscape model.

Usage

```
nlm_edgegradient(
  ncol,
  nrow,
  resolution = 1,
  direction = NA,
  user_seed = NULL,
  rescale = TRUE
)
```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
direction	[numerical(1)] Direction of the gradient (between 0 and 360 degrees), if unspecified the direction is randomly determined.
user_seed	[numerical(1)] Set random seed for the simulation.
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1.

Details

Simulates a linear gradient orientated on a specified or random direction that has a central peak running perpendicular to the gradient direction.

Value

RasterLayer

References

Travis, J.M.J. & Dytham, C. (2004) A method for simulating patterns of habitat availability at static and dynamic range margins. *Oikos*, 104, 410–416.

See Also

[nlm_distancegradient](#), [nlm_planargradient](#)

Examples

```
# simulate random curdling
edge_gradient <- nlm_edgegradient(ncol = 100, nrow = 100, direction = 80)

## Not run:
# visualize the NLM
raster::plot(edge_gradient)

## End(Not run)
```

nlm_fbm

nlm_fbm

Description

Creates a two-dimensional fractional Brownian motion neutral landscape model.

Usage

```
nlm_fbm(
  ncol,
  nrow,
  resolution = 1,
  fract_dim = 1,
  user_seed = NULL,
  rescale = TRUE,
  ...
)
```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
frac_dim	[numerical(1)] The fractal dimension of the process (0,2)
user_seed	[numerical(1)] Set random seed for the simulation
rescale	[numeric(1)] If TRUE (default), the values are rescaled between 0-1.
...	Unused

Details

Neutral landscapes are generated using fractional Brownian motion, an extension of Brownian motion in which the amount of correlation between steps is controlled by `frac_dim`. A high value of `frac_dim` produces a relatively smooth, correlated surface while a low value produces a rough, uncorrelated one.

Value

character(1) Temporary maintenance message.

References

Travis, J.M.J. & Dytham, C. (2004). A method for simulating patterns of habitat availability at static and dynamic range margins. *Oikos*, 104, 410–416.

Martin Schlather, Alexander Malinowski, Peter J. Menck, Marco Oesting, Kirstin Strokorb (2015). `nlm_fBm`. *Journal of Statistical Software*, 63(8), 1-25. URL <http://www.jstatsoft.org/v63/i08/>.

Examples

```
# simulate fractional brownian motion
fbm_raster <- nlm_fbm(ncol = 20, nrow = 30, fract_dim = 0.8)

## Not run:

# visualize the NLM
raster::plot(fbm_raster)

## End(Not run)
```

nlm_gaussianfield *nlm_gaussianfield*

Description

Simulates a spatially correlated random fields (Gaussian random fields) neutral landscape model.

Usage

```
nlm_gaussianfield(
  ncol,
  nrow,
  resolution = 1,
  autocorr_range = 10,
  mag_var = 5,
  nug = 0.2,
  mean = 0.5,
  user_seed = NULL,
  rescale = TRUE
)
```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
autocorr_range	[numerical(1)] Maximum range (raster units) of spatial autocorrelation.
mag_var	[numerical(1)] Magnitude of variation over the entire landscape.
nug	[numerical(1)] Magnitude of variation in the scale of autocorr_range, smaller values lead to more homogeneous landscapes.
mean	[numerical(1)] Mean value over the field.
user_seed	[numerical(1)] Set random seed for the simulation
rescale	[numeric(1)] If TRUE (default), the values are rescaled between 0-1.

Details

Gaussian random fields are a collection of random numbers on a spatially discrete set of coordinates (landscape raster). Natural sciences often apply them with spatial autocorrelation, meaning that objects which distant are more distinct from one another than they are to closer objects.

Value

character(1) Temporary maintenance message.

References

Kéry & Royle (2016) *Applied Hierarchical Modeling in Ecology* Chapter 20

Examples

```
# simulate random gaussian field
gaussian_field <- nlm_gaussianfield(ncol = 90, nrow = 90,
                                   autocorr_range = 60,
                                   mag_var = 8,
                                   nug = 5)

## Not run:
# visualize the NLM
raster::plot(gaussian_field)

## End(Not run)
```

nlm_mosaicfield	<i>nlm_mosaicfield</i>
-----------------	------------------------

Description

Simulates a mosaic random field neutral landscape model.

Usage

```
nlm_mosaicfield(
  ncol,
  nrow,
  resolution = 1,
  n = 20,
  mosaic_mean = 0.5,
  mosaic_sd = 0.5,
  user_seed = NULL,
  collect = FALSE,
  infinit = FALSE,
  rescale = TRUE
)
```

Arguments

<code>ncol</code>	[numerical(1)] Number of columns forming the raster.
<code>nrow</code>	[numerical(1)] Number of rows forming the raster.
<code>resolution</code>	[numerical(1)] Resolution of the raster.
<code>n</code>	[numerical(1)] Number of steps over which the mosaic random field algorithm is run
<code>mosaic_mean</code>	[numerical(1)] Mean value of the mosaic displacement distribution
<code>mosaic_sd</code>	[numerical(1)] Standard deviation of the mosaic displacement distribution
<code>user_seed</code>	[numerical(1)] Set random seed for the simulation.
<code>collect</code>	[logical(1)] return RasterBrick of all steps 1:n
<code>infinite</code>	[logical(1)] return raster of the random mosaic field algorithm with infinite steps
<code>rescale</code>	[logical(1)] If TRUE (default), the values are rescaled between 0-1.

Value

RasterLayer or List with RasterLayer/s and/or RasterBrick

References

Schwab, Dimitri, Martin Schlather, and Jürgen Potthoff. "A general class of mosaic random fields." arXiv preprint arXiv:1709.01441 (2017).
 Baddeley, Adrian, Ege Rubak, and Rolf Turner. Spatial point patterns: methodology and applications with R. CRC Press, 2015.

Examples

```
# simulate mosaic random field
mosaic_field <- nlm_mosaicfield(ncol = 100,
                               nrow = 200,
                               n = NA,
                               infinite = TRUE,
                               collect = FALSE)

## Not run:
# visualize the NLM
raster::plot(mosaic_field)

## End(Not run)
```

nlm_mosaicgibbs	<i>nlm_mosaicgibbs</i>
-----------------	------------------------

Description

Simulate a neutral landscape model using the Gibbs algorithm introduced in Gaucherel (2008).

Usage

```
nlm_mosaicgibbs(
  ncol,
  nrow,
  resolution = 1,
  germs,
  R,
  patch_classes,
  user_seed = NULL,
  rescale = TRUE
)
```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
germs	[numerical(1)] Intensity parameter (non-negative integer).
R	[numerical(1)] Interaction radius (non-negative integer) for the fitting of the spatial point pattern process - the min. distance between germs in map units.
patch_classes	[numerical(1)] Number of classes for germs.
user_seed	[numerical(1)] Set random seed for the simulation.
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1.

Details

nlm_mosaicgibbs offers the second option of simulating a neutral landscape model described in Gaucherel (2008). The method works in principal like the tessellation method (nlm_mosaictess), but instead of a random point pattern the algorithm fits a simulated realization of the Strauss process.

The Strauss process starts with a given number of points and uses a minimization approach to fit a point pattern with a given interaction parameter (0 - hardcore process; 1 - Poisson process) and interaction radius (distance of points/germs being apart).

Value

RasterLayer

References

Gaucherel, C. (2008) Neutral models for polygonal landscapes with linear networks. *Ecological Modelling*, 219, 39 - 48.

Examples

```
# simulate polygonal landscapes
mosaicgibbs <- nlm_mosaicgibbs(ncol = 40,
                              nrow = 30,
                              germs = 20,
                              R = 0.02,
                              patch_classes = 12)

## Not run:
# visualize the NLM
raster::plot(mosaicgibbs)

## End(Not run)
```

nlm_mosaictess	<i>nlm_mosaictess</i>
----------------	-----------------------

Description

Simulate a neutral landscape model using the tessellation approach introduced in Gaucherel (2008).

Usage

```
nlm_mosaictess(
  ncol,
  nrow,
  resolution = 1,
  germs,
  user_seed = NULL,
  rescale = TRUE
)
```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
germs	[numerical(1)] Intensity parameter (non-negative integer).
user_seed	[numerical(1)] Set random seed for the simulation.
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1.

Details

nlm_mosaictess offers the first option of simulating a neutral landscape model described in Gaucherel (2008). It generates a random point pattern (germs) with an independent distribution and uses the Voronoi tessellation to simulate mosaic landscapes.

Value

RasterLayer

References

Gaucherel, C. (2008) Neutral models for polygonal landscapes with linear networks. *Ecological Modelling*, 219, 39 - 48.

Examples

```
# simulate polygonal landscapes
mosaictess <- nlm_mosaictess(ncol = 30, nrow = 60, germs = 200)

## Not run:
# visualize the NLM
raster::plot(mosaictess)

## End(Not run)
```

nlm_mpd

*nlm_mpd***Description**

Simulates a midpoint displacement neutral landscape model.

Usage

```
nlm_mpd(
  ncol,
  nrow,
  resolution = 1,
  roughness = 0.5,
  rand_dev = 1,
  user_seed = NULL,
  torus = FALSE,
  rescale = TRUE,
  verbose = TRUE
)
```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
roughness	[numerical(1)] Controls the level of spatial autocorrelation (!= Hurst exponent)
rand_dev	[numerical(1)] Initial standard deviation for the displacement step (default == 1), sets the scale of the overall variance in the resulting landscape.
user_seed	[numerical(1)] Set random seed for the simulation.
torus	[logical(1)] Logical value indicating whether the algorithm should be simulated on a torus (default FALSE)
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1.
verbose	[logical(1)] If TRUE (default), the user gets a warning that the functions changes the dimensions to an appropriate one for the algorithm.

Details

The algorithm is a direct implementation of the midpoint displacement algorithm. It performs the following steps:

Initialization: Determine the smallest fit of $\max(\text{ncol}, \text{nrow})$ in $n^2 + 1$ and assign value to n . Setup matrix of size $(n^2 + 1) \times (n^2 + 1)$. Afterwards, assign a random value to the four corners of the matrix.

Square Step: For each square in the matrix, assign the average of the four corner points plus a random value to the midpoint of that square.

Diamond Step: For each diamond in the matrix, assign the average of the four corner points plus a random value to the midpoint of that diamond.

At each iteration the roughness, an approximation to common Hurst exponent, is reduced.

Value

RasterLayer

References

https://en.wikipedia.org/wiki/Diamond-square_algorithm

Examples

```
# simulate midpoint displacement
midpoint_displacement <- nlm_mpd(ncol = 100,
                                nrow = 100,
                                roughness = 0.3)

## Not run:
# visualize the NLM
raster::plot(midpoint_displacement)

## End(Not run)
```

nlm_neigh

nlm_neigh

Description

Create a neutral landscape model with categories and clustering based on neighborhood characteristics.

Usage

```
nlm_neigh(
  ncol,
  nrow,
  resolution = 1,
  p_neigh,
  p_empty,
  categories = 3,
  neighbourhood = 4,
  proportions = NA,
  user_seed = NULL,
  rescale = TRUE
)
```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
p_neigh	[numerical(1)] Probability of a cell will turning into a value if there is any neighbor with the same or a higher value.
p_empty	[numerical(1)] Probability a cell receives a value if all neighbors have no value (i.e. zero).
categories	[numerical(1)] Number of categories used.
neighbourhood	[numerical(1)] The neighbourhood used to determined adjacent cells: '8 ("Moore")' takes the eight surrounding cells, while '4 ("Von-Neumann")' takes the four adjacent cells (i.e. left, right, upper and lower cells).
proportions	[vector(1)] The algorithm uses uniform proportions for each category by default. A vector with as many proportions as categories and that sums up to 1 can be used for other distributions.
user_seed	[numerical(1)] Set random seed for the simulation.
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1.

Details

The algorithm draws a random cell and turns it into a given category based on the probabilities `p_neigh` and `p_empty`, respectively. The decision is based on the probability `p_neigh`, if there is

any cell in the Moore- (8 cells) or Von-Neumann-neighborhood (4 cells), otherwise it is based on `p_empty`. To create clustered neutral landscape models, `p_empty` should be (significantly) smaller than `p_neigh`. By default, the Von-Neumann-neighborhood is used to check adjacent cells. The algorithm starts with the highest categorical value. If the proportion of cells with this value is reached, the categorical value is reduced by 1. By default, a uniform distribution of the categories is applied.

Value

RasterLayer

References

Scherer, Cédric, et al. "Merging trait-based and individual-based modelling: An animal functional type approach to explore the responses of birds to climatic and land use changes in semi-arid African savannas." *Ecological Modelling* 326 (2016): 75-89.

Examples

```
# simulate neighborhood model
neigh_raster <- nlm_neigh(ncol = 50, nrow = 50, p_neigh = 0.7, p_empty = 0.1,
                        categories = 5, neighbourhood = 4)

## Not run:
# visualize the NLM
raster::plot(neigh_raster)

## End(Not run)
```

nlm_percolation	<i>nlm_percolation</i>
-----------------	------------------------

Description

Generates a random percolation neutral landscape model.

Usage

```
nlm_percolation(ncol, nrow, resolution = 1, prob = 0.5, user_seed = NULL)
```

Arguments

<code>ncol</code>	[numerical(1)] Number of columns forming the raster.
<code>nrow</code>	[numerical(1)] Number of rows forming the raster.

resolution	[numerical(1)] Resolution of the raster.
prob	[numerical(1)] Probability value for setting a cell to 1.
user_seed	[numerical(1)] Set random seed for the simulation.

Details

The simulation of a random percolation map is accomplished in two steps:

Initialization: Setup matrix of size (ncol*nrow).

Map generation: For each cell in the matrix a single uniformly distributed random number is generated and tested against a probability prob. If the random number is smaller than prob, the cell is set to TRUE; if it is higher the cell is set to FALSE.

Value

RasterLayer

References

1. Gardner RH, O'Neill R V, Turner MG, Dale VH. 1989. Quantifying scale-dependent effects of animal movement with simple percolation models. *Landscape Ecology* 3:217 - 227.
2. Gustafson, E.J. & Parker, G.R. (1992) Relationships between landcover proportion and indices of landscape spatial pattern. *Landscape Ecology* , 7, 101 - 110.

Examples

```
# simulate percolation model
percolation <- nlm_percolation(ncol = 100, nrow = 100, prob = 0.5)
## Not run:
# visualize the NLM
raster::plot(percolation)

## End(Not run)
```

nlm_planargradient	<i>nlm_planargradient</i>
--------------------	---------------------------

Description

Simulates a planar gradient neutral landscape model.

Usage

```
nlm_planargradient(  
  ncol,  
  nrow,  
  resolution = 1,  
  direction = NA,  
  user_seed = NULL,  
  rescale = TRUE  
)
```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
direction	[numerical(1)] Direction of the gradient in degrees, if unspecified the direction is randomly determined.
user_seed	[numerical(1)] Set random seed for the simulation.
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1.

Details

Simulates a linear gradient sloping in a specified or random direction.

Value

RasterLayer

References

Palmer, M.W. (1992) The coexistence of species in fractal landscapes. *The American Naturalist*, 139, 375 - 397.

See Also

[nlm_distancegradient](#), [nlm_edgegradient](#)

Examples

```
# simulate planar gradient
planar_gradient <- nlm_planargradient(ncol = 200, nrow = 200)

## Not run:
# visualize the NLM
raster::plot(planar_gradient)

## End(Not run)
```

nlm_random

nlm_random

Description

Simulates a spatially random neutral landscape model with values drawn a uniform distribution.

Usage

```
nlm_random(ncol, nrow, resolution = 1, user_seed = NULL, rescale = TRUE)
```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
user_seed	[numerical(1)] Set random seed for the simulation.
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1.

Details

The function takes the number of columns and rows as input and creates a RasterLayer with the same extent. Each raster cell is randomly assigned a value between 0 and 1 drawn from an uniform distribution (`runif(1, 0, 1)`).

Value

RasterLayer

Examples

```
# simulate spatially random model
random <- nlm_random(ncol = 200, nrow = 100)

## Not run:
# visualize the NLM
raster::plot(random)

## End(Not run)
```

nlm_randomcluster	<i>nlm_randomcluster</i>
-------------------	--------------------------

Description

Simulates a random cluster nearest-neighbour neutral landscape.

Usage

```
nlm_randomcluster(
  ncol,
  nrow,
  resolution = 1,
  p,
  ai = c(0.5, 0.5),
  neighbourhood = 4,
  user_seed = NULL,
  rescale = TRUE
)
```

Arguments

ncol	[integer(1)] Number of columns forming the raster.
nrow	[integer(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
p	[numerical(1)] Defines the proportion of elements randomly selected to form clusters.
ai	Vector with the cluster type distribution (percentages of occupancy). This directly controls the number of types via the given length.
neighbourhood	[numerical(1)] Clusters are defined using a set of neighbourhood structures, 4 (Rook's or von Neumann neighbourhood) (default), 8 (Queen's or Moore neighbourhood).

user_seed	[numerical(1)] Set random seed for the simulation.
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1.

Details

This is a direct implementation of steps A - D of the modified random clusters algorithm by Saura & Martínez-Millán (2000), which creates naturalistic patchy patterns.

Value

Raster with random values ranging from 0-1.

References

Saura, S. & Martínez-Millán, J. (2000) Landscape patterns simulation with a modified random clusters method. *Landscape Ecology*, 15, 661 – 678.

Examples

```
# simulate random clustering
random_cluster <- nlm_randomcluster(ncol = 30, nrow = 30,
                                   p = 0.4,
                                   ai = c(0.25, 0.25, 0.5))

## Not run:
# visualize the NLM
raster::plot(random_cluster)

## End(Not run)
```

```
nlm_randomrectangularcluster
      nlm_randomrectangularcluster
```

Description

Simulates a random rectangular clusters neutral landscape model with values ranging 0-1.

Usage

```
nlm_randomrectangularcluster(
  ncol,
  nrow,
  resolution = 1,
  minl,
  maxl,
```

```

    user_seed = NULL,
    rescale = TRUE
  )

```

Arguments

ncol	[numerical(1)] Number of columns forming the raster.
nrow	[numerical(1)] Number of rows forming the raster.
resolution	[numerical(1)] Resolution of the raster.
minl	[numerical(1)] The minimum possible width and height for each random rectangular cluster.
maxl	[numerical(1)] The maximum possible width and height for each random rectangular cluster.
user_seed	[numerical(1)] Set random seed for the simulation.
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1.

Details

The random rectangular cluster algorithm starts to fill a raster randomly with rectangles defined by minl and maxl until the surface of the landscape is completely covered. This is one type of realisation of a "falling/dead leaves" algorithm, for more details see Galerne & Gousseau (2012).

Value

RasterLayer

References

Gustafson, E.J. & Parker, G.R. (1992). Relationships between landcover proportion and indices of landscape spatial pattern. *Landscape ecology*, 7, 101–110. Galerne B. & Gousseau Y. (2012). The Transparent Dead Leaves Model. *Advances in Applied Probability, Applied Probability Trust*, 44, 1–20.

Examples

```

# simulate random rectangular cluster
randomrectangular_cluster <- nlm_randomrectangularcluster(ncol = 50,
  nrow = 30,
  minl = 5,
  maxl = 10)

## Not run:
# visualize the NLM
raster::plot(randomrectangular_cluster)

```

```
## End(Not run)
```

util_classify	<i>util_classify</i>
---------------	----------------------

Description

Classify continuous landscapes into landscapes with discrete classes

Usage

```
util_classify(x, n, weighting, level_names, real_land, mask_val)

## S3 method for class 'RasterLayer'
util_classify(
  x,
  n = NULL,
  weighting = NULL,
  level_names = NULL,
  real_land = NULL,
  mask_val = NULL
)
```

Arguments

x	raster
n	Number of classes
weighting	Vector of numeric values that are considered to be habitat percentages (see details)
level_names	Vector of names for the factor levels.
real_land	Raster with real landscape (see details)
mask_val	Value to mask (refers to real_land)

Details

Mode 1: Calculate the optimum breakpoints using Jenks natural breaks optimization, the number of classes is determined with 'n'. The Jenks optimization seeks to minimize the variance within categories, while maximizing the variance between categories.

Mode 2: The number of elements in the weighting vector determines the number of classes in the resulting matrix. The classes start with the value 1. If non-numerical levels are required, the user can specify a vector to turn the numerical factors into other data types, for example into character strings (i.e. class labels). If the numerical vector of weightings does not sum up to 1, the sum of the weightings is divided by the number of elements in the weightings vector and this is then used for the classificat#'. .

Mode 3: For a given 'real' landscape the number of classes and the weightings are extracted and used to classify the given landscape (any given weighting parameter is overwritten in this case!). If an optional mask value is given the corresponding class from the 'real' landscape is cut from the landscape beforehand.

Value

RasterLayer

Examples

```
## Not run:
# Mode 1
util_classify(fractal_landscape,
              n = 3,
              level_names = c("Land Use 1", "Land Use 2", "Land Use 3"))

# Mode 2
util_classify(fractal_landscape,
              weighting = c(0.5, 0.25, 0.25),
              level_names = c("Land Use 1", "Land Use 2", "Land Use 3"))

# Mode 3
real_land <- util_classify(gradient_landscape,
                          n = 3,
                          level_names = c("Land Use 1", "Land Use 2", "Land Use 3"))

fractal_landscape_real <- util_classify(fractal_landscape, real_land = real_land)
fractal_landscape_mask <- util_classify(fractal_landscape, real_land = real_land, mask_val = 1)

landscapes <- list(
  '1 nlm' = fractal_landscape,
  '2 real' = real_land,
  '3 result' = fractal_landscape_real,
  '4 result with mask' = fractal_landscape_mask
)

raster::plot(landscapes)

## End(Not run)
```

Index

nlm_curds, [2](#)
nlm_distancegradient, [4](#), [6](#), [19](#)
nlm_edgegradient, [5](#), [5](#), [19](#)
nlm_fbm, [6](#)
nlm_gaussianfield, [8](#)
nlm_mosaicfield, [9](#)
nlm_mosaicgibbs, [11](#)
nlm_mosaictess, [12](#)
nlm_mpd, [14](#)
nlm_neigh, [15](#)
nlm_percolation, [17](#)
nlm_planargradient, [5](#), [6](#), [18](#)
nlm_polylands (nlm_mosaictess), [12](#)
nlm_random, [20](#)
nlm_randomcluster, [21](#)
nlm_randomrectangularcluster, [22](#)

util_classify, [24](#)