Package 'NLMR'

November 30, 2017

Type Package

Title Simulating Neutral Landscape Models

Version 0.1.0

Description Provides neutral landscape models
(Gardner et al. 1987 <doi:10.1007/BF02275262>,
With 1997 <doi:10.1046/j.1523-1739.1997.96210.x>) that can easily extend in
existing landscape analyses. Neutral landscape models range from ``hard''
neutral models (only random functions) to ``soft'' ones (with parameters) and
generate landscape patterns that are not grounded in ecological reasoning.
Thus, these patterns can be used as null models in landscape ecology. 'NLMR'
combines a large number of algorithms from published software (Saura &
Martínez 2000 <doi:10.1023/A:1008107902848>, Etherington et
al. 2015 <doi:10.1111/2041-210X.12308>) for simulating neutral landscapes and
includes utility functions to classify and combine the landscapes. The
simulation results are obtained in a geospatial 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

Encoding UTF-8

LazyData true

ByteCompile true

Depends R (>= 3.1.0)

RoxygenNote 6.0.1

Imports checkmate, dismo, dplyr, ggplot2, gstat, igraph, lemon,
magrittr, maptools, purrr, RandomFields, raster, rasterVis,
R.utils, sp, spatstat, stats, tibble, viridis

URL https://marcosci.github.io/NLMR/

BugReports https://github.com/marcosci/NLMR/issues/

Suggests testthat, covr, knitr, rmarkdown, lintr

VignetteBuilder knitr

NeedsCompilation no

Author Marco Sciaini [aut, cre] (Department of Ecosystem Modelling, University of Goettingen),
Matthias Fritsch [aut] (Department of Ecosystem Modelling, University of Goettingen),
Cédric Scherer [aut] (Leibniz Institute for Zoo and Wildlife Research, Berlin),
Craig Simpkins [aut] (Department of Ecosystem Modelling, University of Goettingen)

Maintainer Marco Sciaini <sciaini.marco@gmail.com>

Repository CRAN

Date/Publication 2017-11-30 10:11:05 UTC

R topics documented:

`metric_area` *metric_area*

Description

Report area for each of the constituent values making up a matrix

Usage

```
metric_area(x, poi = NA)
```

Arguments

`x`	[Raster* object]
`poi`	[numerical] Vector of numeric values indicating classes to be reported.

Details

Function reports the number of cells and the proportion of total cells made up by each unique class value

Value

List of tibbles

Examples

```
x <- nlm_random(100, 100)
y <- c(0.5, 0.25, 0.25)
z <- util_classify(x, y)

metric_area(z)
```

`NLMR` *NLMR*

Description

NLMR is an R package for simulating neutral landscape models (NLM).

Details

This package contains vignettes that introduce NLM and basic usage of the *NLMR* package. The vignettes in this package are listed below.

NLM Concepts + Terminology Background information on the concepts and terminology that underpin the simulation of neutral landscape models.

NLMR Bestiary Bestiary of the available neutral landscape models.

Quickstart Guide Short walk-through of the *NLMR* package and how to handle the simulations.

References

Gardner RH, Milne BT, Turnei MG, O'Neill R V. 1987. Neutral models for the analysis of broad-scale landscape pattern. *Landscape Ecology* 1:19–28.

`nlm_curds` *nlm_curds*

Description

Simulates a curdled neutral landscape model.

Usage

```
nlm_curds(p, s, ext = 1)
```

Arguments

`p`	`[numerical(x)]` Vector with percentage(s) to cut of (fill with matrix (zeroes)).
`s`	`[numerical(x)]` Vector of successive cutting steps for the blocks (split 1 block into x blocks).
`ext`	`[numerical(1)]` Extent of the resulting raster (0,x,0,x).

Details

Random curdling recursively subdivides the plane into blocks. At each level of the recursion, a fraction of the this block is declared as habitat (1) while the remaining stays matrix (0).

Value

raster

References

Keitt TH. 2000. Spectral representation of neutral landscapes. *Landscape Ecology* 15:479-493.

Examples

```
nlm_curds(c(0.25, 0.25, 0.5), c(64, 8, 1))
```

`nlm_distancegradient` *nlm_distancegradient*

Description

Simulate a distance gradient neutral landscape model..

Usage

```
nlm_distancegradient(nCol, nRow, resolution = 1, origin, rescale = TRUE)
```

Arguments

`nCol`	[`numerical(1)`] Number of columns for the raster.
`nRow`	[`numerical(1)`] Number of rows for the raster.
`resolution`	[`numerical(1)`] Resolution of the raster.
`origin`	[`numerical(4)`] Edge coordinates of the origin 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

Examples

```
nlm_distancegradient(nCol = 100, nRow = 100, origin = c(20, 30, 10, 15))
```

nlm_edgegradient *nlm_edgegradient*

Description

Simulates an edge gradient neutral landscape model.

Usage

```
nlm_edgegradient(nCol, nRow, resolution = 1, direction = NA,
  rescale = TRUE)
```

Arguments

nCol	[numerical(1)] Number of columns for the raster.
nRow	[numerical(1)] Number of rows for 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.
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 that runs 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.

Examples

```
nlm_edgegradient(nCol = 100, nRow = 100, direction = 80)
```

nlm_fBm *nlm_fBM*

Description

Simulate two-dimensional fractional brownian motion model.

Usage

```
nlm_fBm(nCol, nRow, resolution = 1, H = 0.5, user_seed = NULL,
  rescale = TRUE)
```

Arguments

nCol	[numerical(1)] Number of columns for the raster.
nRow	[numerical(1)] Number of rows for the raster.
resolution	[numerical(1)] Resolution of the raster.
H	[numerical(1)] Hurst coefficient
user_seed	[numerical(1)] Set Seed for simulation
rescale	[numeric(1)] If TRUE (default), the values are rescaled between 0-1.

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 the Hurst coefficient H. An H of 1 produces a relatively smooth surface while an H of 0 produces a rough, uncorrelated, surface. Implementation of this method is limited to landscapes with extents less than 90 by 90 cells.

Value

RasterLayer

Examples

```
nlm_fBm(nCol = 40, nRow = 40, H = 0.5)
```

nlm_gaussianfield *nlm_gaussianfield*

Description

Simulate spatially correlated random fields (Gaussian random fields) model.

Usage

```
nlm_gaussianfield(nCol, nRow, resolution = 1, autocorr_range = 10,
  mag_var = 0.025, beta = c(1, 0.01, 0.005), nug = 1,
  direction = "random", angle = 1, rescale = TRUE)
```

Arguments

nCol	[numerical(1)] Number of columns for the raster.
nRow	[numerical(1)] Number of rows for the raster.
resolution	[numerical(1)] Resolution of the raster.
autocorr_range	[numerical(1)] Maximal distance of spatial autocorrelation
mag_var	[numerical(1)] Magnitude of variation in the field
beta	[numerical(1)] Mean value over the field.
nug	[numerical(1)] Small-scale variations in the field.
direction	[character("random" | "linear")] Direction of the gradient. Either random, or with a linear trend (default).
angle	[numerical(1)] Maximal distance of spatial autocorrelation
rescale	[numeric(1)] If TRUE (default), the values are rescaled between 0-1.

Value

RasterLayer

Examples

```
nlm_gaussianfield(nCol = 100, nRow = 100, 5)
```

nlm_mosaicfield *nlm_mosaicfield*

Description

Create a mosaic random field neutral landscape model.

Usage

```
nlm_mosaicfield(nCol, nRow, resolution = 1, n = 20, mosaic_mean = 0.5,
  mosaic_sd = 0.5, collect = FALSE, infinit = FALSE, rescale = TRUE)
```

Arguments

nCol	[numerical(1)] Number of columns for the raster.
nRow	[numerical(1)] Number of rows for the raster.
resolution	[numerical(1)] Resolution of the raster.
n	[numerical(1)] Number of steps that the mosaic random field algorithm is run
mosaic_mean	[numerical(1)] Mean value of the mosaic displacement distribution
mosaic_sd	[numerical(1)] Standard deviation of the mosaic displacement distribution
collect	[logical(1)] return RasterBrick of all steps 1:n
infinit	[logical(1)] return raster of the random mosaic field algorithm with infinite steps
rescale	[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

```
nlm_mosaicfield(nCol = 200,
                nRow = 200,
                n = NA,
                infinit = TRUE,
                collect = FALSE)
```

---

nlm_mpd *nlm_mpd*

Description

Create a midpoint displacement neutral landscape model.

Usage

```
nlm_mpd(nCol, nRow, resolution = 1, roughness = 0.5, rand_dev = 1,
  rescale = TRUE, verbose = TRUE)
```

Arguments

nCol	[numerical(1)] Number of columns for the raster.
nRow	[numerical(1)] Number of rows for the raster.
resolution	[numerical(1)] Resolution of the raster.
roughness	[numerical(1)] Controls the level of spatial autocorrelation (!= hurst index)
rand_dev	[numerical(1)] Initial standard deviation for the displacement step (default == 1)
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(nCol, nRow) in *n^2 + 1* and assign value to n. Setup matrix of size (n^2 + 1)*(n^2 + 1). Afterwards, assign a random value to the four corners of the matrix.

- Diamond 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 index, is reduced.

The image below shows the steps involved in running the diamond-square algorithm on a 5×5 matrix:

(By Christopher Ewin - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=42510593)

Value

RasterLayer

References

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

Examples

```
nlm_mpd(nCol = 100, nRow = 100, roughness = 0.2)
```

`nlm_neigh` *nlm_neigh*

Description

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

Usage

```
nlm_neigh(nCol, nRow, resolution = 1, p_neigh, p_empty, categories = 3,
  neighborhood = "Von-Neumann", proportions = NA, rescale = TRUE)
```

Arguments

`nCol`	[numerical(1)] Number of columns for the raster.
`nRow`	[numerical(1)] Number of rows for the raster.
`resolution`	[numerical(1)] Resolution of the raster.
`p_neigh`	[numerical(1)] Probability to turn 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.

`neighborhood` [string(1)]
The neighborhood used to determined adjacent cells: ‘"Moore"‘ takes the eight surrounding cells, while ‘"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.

`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- or Von-Neumann-neighborhood, 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 categorial value. If the proportion of cells with this value is reached, the categorial 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

```
nlm_neigh(nCol = 10, nRow = 10, p_neigh = 0.1, p_empty = 0.3,
                    categories = 5, neighborhood = "Von-Neumann")
```

`nlm_percolation` *nlm_percolation*

Description

Create a random percolation neutral landscape model.

Usage

```
nlm_percolation(nCol, nRow, resolution = 1, prob = 0.5)
```

Arguments

`nCol`	[`numerical(1)`] Number of columns for the raster.
`nRow`	[`numerical(1)`] Number of rows for the raster.
`resolution`	[`numerical(1)`] Resolution of the raster.
`prob`	[`numerical(1)`] Probability value for setting a cell either to 0 or 1.

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 1 - if it is higher the cell is set to 0.

The proportion of 0 and 1 is thus controlled with the argument `prob`.

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

```
nlm_percolation(nCol = 100, nRow = 100, prob=0.5)
```

`nlm_planargradient` *nlm_planargradient*

Description

Create a planar gradient neutral landscape model.

Usage

```
nlm_planargradient(nCol, nRow, resolution = 1, direction = NA,
  rescale = TRUE)
```

Arguments

`nCol`	[`numerical(1)`] Number of columns for the raster.
`nRow`	[`numerical(1)`] Number of rows for 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.
`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.

Examples

```
nlm_planargradient(nCol = 100, nRow = 100)
```

nlm_polylands *nlm_polylands*

Description

Simulate the NLM introduced in Gaucherel (2008).

Usage

```
nlm_polylands(nCol, nRow, resolution = 1, option = 1, germs, g, R,
  patch_classes, rescale = TRUE)
```

Arguments

nCol	[numerical(1)] Number of columns for the raster.
nRow	[numerical(1)] Number of rows for the raster.
resolution	[numerical(1)] Resolution of the raster.
option	[numerical(1)] If 1 (default), the Tessellation method is used to simulate the NLM. If 2, the Gibbs algorithm method is used to simulate the NLM.
germs	[numerical(1)] Intensity parameter (non-negative integer).
g	[numerical(1)] Interaction parameter (a value between 0 and 1, inclusive - only used when option = 2).
R	[numerical(1)] Interaction radius (non-negative integer).
patch_classes	[numerical(1)] Number of classes for germs.
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1. Otherwise, the distance in raster units is calculated.

Details

The function offers 2 of the 3 NLM described in Gaucherel (2008). The first one (option = 1) is a tessellation method. It generates a random point pattern (the germs) with an independent distribution and uses the voronoi tessellation to simulate the patchy landscapes. The second one (option = 2) is the Gibbs algorithm method. The method works in principal like the tessellation method, but instead of a random point pattern one 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 - poission 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

```
nlm_polylands(nCol = 50, nRow = 50, germs = 20)
```

`nlm_random` *nlm_random*

Description

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

Usage

```
nlm_random(nCol, nRow, resolution = 1, rescale = TRUE)
```

Arguments

`nCol`	[`numerical(1)`] Number of columns for the raster.
`nRow`	[`numerical(1)`] Number of rows for the raster.
`resolution`	[`numerical(1)`] Resolution of the raster.
`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

```
nlm_random(nCol = 100, nRow = 100)
```

`nlm_randomcluster` *nlm_randomcluster*

Description

Simulates a random cluster nearest-neighbour neutral landscape.

Usage

```
nlm_randomcluster(nCol, nRow, resolution = 1, neighbourhood = 4, p,
  rescale = TRUE)
```

Arguments

`nCol`	[`integer(1)`] Number of columns in the raster.
`nRow`	[`integer(1)`] Number of rows in the raster.
`resolution`	[`numerical(1)`] Resolution of the raster.
`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).
`p`	[`numerical(1)`] The p value defines the proportion of elements randomly selected to form clusters.
`rescale`	[`logical(1)`] If `TRUE` (default), the values are rescaled between 0-1.

Details

The implemented algorithm has been adopted from Etherington et al. 2014 and is itself an adaptation of the MRC algorithm by Saura & Martínez-Millán (2000). The algorithm simulates a percolation map, which defines random clusters by running a connected labelling algorithm which detects clusters and gives each a unique ID. The algorithm controls the size and directional bias of the cluster with the proportion of the matrix that is within a cluster and with specifying a specific neighbourhood rule. Each cluster is than given a random value and non-cluster cells are assigned values by performing a nearest neighbour interpolation.

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.

Etherington TR, Holland EP, O'Sullivan D. 2015. NLMpy: A python software package for the creation of neutral landscape models within a general numerical framework. *Methods in Ecology and Evolution* 6:164 – 168.

Examples

```
nlm_randomcluster(nCol = 10, nRow = 10, resolution = 10, neighbourhood = 4, p = 0.4)
```

`nlm_randomelement` *nlm_randomelement*

Description

Simulates a random rectangular cluster neutral landscape model.

Usage

```
nlm_randomelement(nCol, nRow, resolution = 1, n, rescale = TRUE)
```

Arguments

`nCol`	[`numerical(1)`] Number of columns for the raster.
`nRow`	[`numerical(1)`] Number of rows for the raster.
`resolution`	[`numerical(1)`] Resolution of the raster.
`n`	[`numerical(1)`] The number of elements randomly selected to form the basis of nearest-neighbour clusters.
`rescale`	[`logical(1)`] If `TRUE` (default), the values are rescaled between 0-1.

Details

The algorithm selects `n` random cell of the matrix and assigns them a uniformly distributed random value. The cells are then converted to a point pattern and interpolated by voronoi tessellation.

Value

Raster layer with values between 0 and 1

References

Etherington TR, Holland EP, O'Sullivan D. 2015. NLMpy: A python software package for the creation of neutral landscape models within a general numerical framework. *Methods in Ecology and Evolution* 6:164 – 168.

Examples

```
nlm_randomelement(nCol = 10, nRow = 10, n = 10)
```

`nlm_randomrectangularcluster`

nlm_randomrectangularcluster

Description

Create a random rectangular cluster neutral landscape model with values ranging 0-1.

Usage

```
nlm_randomrectangularcluster(nCol, nRow, resolution = 1, minL, maxL,
  rescale = TRUE)
```

Arguments

`nCol`	[`numerical(1)`] Number of columns for the raster.
`nRow`	[`numerical(1)`] Number of rows for 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.
`rescale`	[`logical(1)`] If `TRUE` (default), the values are rescaled between 0-1.

Value

RasterLayer with random values ranging from 0-1.

Examples

```
nlm_randomrectangularcluster(nCol = 100, nRow = 100, minL = 5, maxL = 40)
```

`util_binarize` *Binarize continuous raster values*

Description

Reclassify continuous raster values into binary map cells based upon given break(s).

Usage

```
util_binarize(x, breaks)
```

Arguments

`x`	`[Raster* object]`
`breaks`	a vector with one or more break percentages

Details

Breaks are considered to be habitat percentages (p). If more than one percentage is given multiple layers are written in the same brick.

Value

RasterBrick

Examples

```
rndMap <- nlm_random(10, 10)
breaks <- c(0.3, 0.5, 0.7, 0.9)
rnd_bin <- util_binarize(rndMap, breaks)
util_plot(rnd_bin)
```

util_classify *util_classify*

Description

Classify a raster into proportions based upon a vector of class weightings.

Usage

```
util_classify(x, weighting, level_names = NULL)
```

Arguments

x	[matrix(x,y)] 2D matrix of data values.
weighting	[numerical] Vector of numeric values.
level_names	[character] Vector of names for the factor levels.

Details

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 classification.

Value

RasterLayer

Examples

```
x <- nlm_random(10, 10)
y <- c(0.5, 0.25, 0.25)
util_classify(x, y, level_names = c("Land Use 1", "Land Use 2", "Land Use 3"))
```

util_merge *util_merge*

Description

Merge a primary raster with other rasters weighted by scaling factors.

Usage

```
util_merge(primary_nlm, secondary_nlm, scalingfactor = 1, rescale = TRUE)
```

Arguments

primary_nlm	[Raster* object] Primary Raster* object
secondary_nlm	[Raster* object or list] A list or stack of Raster* objects that are merged with the primary Raster* object
scalingfactor	[numerical(1)] Weight for the secondary Raster* objects
rescale	[logical(1)] If TRUE (default), the values are rescaled between 0-1.

Value

Rectangular matrix with values ranging from 0-1

Examples

```
util_merge(nlm_percolation(50, 50), nlm_random(50, 50))
```

util_plot *util_plot*

Description

Plot a Raster* object with the NLMR default theme

Usage

```
util_plot(x, scale = "A", discrete = FALSE, legendposition = "bottom",
  legendtitle = "Z")
```

Arguments

x	[Raster* object]
scale	[character(1)] Five options are available: "viridis - magma" (= "A", the default option), "viridis - inferno" (= "B"), "viridis - plasma" (= "C"), "viridis - viridis" (= "D")
discrete	[logical(1)] If TRUE, the function plots a raster with a discrete legend.
legendposition	[character(1)] The position of legends ("none", "left", "right", "bottom", "top")
legendtitle	[character(1)] Title for legend

Value

ggplot2 Object

Examples

```
# With continuous data
nlm_raster <- nlm_random(10,10)
util_plot(nlm_raster, scale = "D")

# With classified data
y <- c(0.5, 0.15, 0.25)
nlm_raster <- util_classify(nlm_raster, y)
util_plot(nlm_raster, scale = "D", discrete = TRUE)
```

---

util_rescale *util_rescale*

Description

Linearly rescale element values in a raster to a range between 0 and 1

Usage

```
util_rescale(x)
```

Arguments

x	[Raster* object]

Details

Rasters generated by nlm_ functions are scaled between 0 and 1 as default, this option can be set to FALSE if needed.

Value

Raster* object with values ranging from 0-1

Examples

```
util_rescale(nlm_random(10, 10,  rescale = FALSE))
```

Index