Package: rasterdiv (via r-universe)

November 7, 2024

Type Package

Title Diversity Indices for Numerical Matrices

Version 0.3.6

Date 2024-11-06

Maintainer Matteo Marcantonio <marcantoniomatteo@gmail.com>

Description Provides methods to calculate diversity indices on numerical matrices based on information theory, as described in Rocchini, Marcantonio and Ricotta (2017) <doi:10.1016/j.ecolind.2016.07.039>, and Rocchini et al. (2021) <doi:10.1101/2021.01.23.427872>.

Depends R (>= 4.0.0)

Imports doParallel, foreach, ggforce, ggplot2, methods, proxy, progress, terra, twdtw, viridis

Suggests knitr, rmarkdown, rasterVis, RColorBrewer, gridExtra, gstat, latticeExtra, COVID19

License GPL (>= 2)

Encoding UTF-8

URL https://mattmar.github.io/rasterdiv/

BugReports https://github.com/mattmar/rasterdiv/issues

VignetteBuilder knitr

RoxygenNote 7.3.2

Language en-GB

NeedsCompilation no

Author Matteo Marcantonio [aut, cre], Martina Iannacito [aut, ctb], Elisa Marchetto [ctb], Elisa Thouverai [aut, ctb], Michele Torresani [aut, ctb], Daniele Da Re [aut], Clara Tattoni [aut], Giovanni Bacaro [aut], Saverio Vicario [aut, ctb], Carlo Ricotta [aut], Duccio Rocchini [aut, ctb]

Repository CRAN

Date/Publication 2024-11-06 11:20:03 UTC

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libgdal-dev
gdal-bin libgeos-dev libproj-dev libsqlite3-dev

Contents

`.CRE_` *Cumulative Residual Entropy*

Description

This function calculates the Cumulative Residual Entropy (CRE) for a given set of values.

Usage

```
.CRE_(B, base = exp(1))
```

Arguments

`B`	A numeric vector or matrix representing the values for which CRE is to be calculated.
`base`	The base of the logarithm used in the calculation. The default is the natural logarithm (e).

Value

A numeric value representing the CRE.

Examples

```
B <- c(1, 2, 3, 4)
.CRE_(B)
```

`.CumRes` *Calculate Cumulative Residual Probability*

Description

This function computes the cumulative residual probability for a given set of probabilities.

Usage

```
.CumRes(a)
```

Arguments

`a`	A numeric vector or matrix representing probabilities.

Value

A numeric vector or matrix of cumulative residual probabilities.

Examples

```
a <- data.frame(V1= c(0.2, 0.3, 0.5), V2 =c(0.2, 0.3, 0.5))
.CumRes(a)
```

.Deltas *Calculate Differences Among Values*

Description

This function computes the differences among values of a table, used in probability calculations.

Usage

```
.Deltas(P, first = 0)
```

Arguments

P	A numeric vector or matrix representing probabilities.
first	The starting value for difference calculation.

Value

A vector or matrix of differences.

Examples

```
P <- c(0.2, 0.3, 0.5)
.Deltas(P)
```

.Prob *Calculate Point Probability*

Description

This function computes the probability of each point in a given vector or matrix.

Usage

```
.Prob(C)
```

Arguments

C	A numeric vector or matrix.

Value

A vector of probabilities corresponding to each point in ‘C‘.

Examples

```
C <- c(1, 1, 2, 2, 3)
.Prob(C)
```

`.Reorder` *Additional supporting functions like ‘.Reorder‘, ‘.Cumsum‘, ‘.Rev‘*

Description

These functions provide utility operations like reordering dimensions, computing cumulative sums, and reversing order along a specific dimension.

Usage

```
.Reorder(a, ax)
```

Arguments

`a`	ax Additional parameters specific to each function.
`ax`	Additional parameters specific to each function.

Value

Output varies depending on the function.

`BergerParker` *Berger-Parker’s diversity index*

Description

Computes Berger-Parker’s diversity index on different classes of numeric matrices using a moving window algorithm.

Usage

```
BergerParker(
  x,
  window = 3,
  rasterOut = TRUE,
  np = 1,
  na.tolerance = 1,
  cluster.type = "SOCK",
  debugging = FALSE
)
```

Arguments

`x`	Input data may be a matrix, a Spatial Grid Data Frame, a SpatRaster, or a list of these objects. In the latter case, only the first element of the list will be considered.
`window`	The side of the square moving window, it must be an odd numeric value greater than 1 to ensure that the target pixel is in the centre of the moving window. Default value is 3.
`rasterOut`	Boolean, if TRUE, output will be in SpatRaster format with *x* as a template.
`np`	The number of processes (cores) which will be spawned. Default value is 1.
`na.tolerance`	A numeric value (0.0-1.0) which indicates the proportion of NA values that will be tolerated to calculate Berger-Parker's index in each moving window over *x*. If the relative proportion of NA's in a moving window is bigger than na.tolerance, then the value of the window will be set as NA, otherwise, Rao's index will be calculated considering the non-NA values. Default values are 1.0 (i.e., no tolerance for NA's).
`cluster.type`	The type of cluster which will be created. The options are `"MPI"` (calls "makeMPIcluster"), `"FORK"`, and `"SOCK"` (call "makeCluster"). Default type is `"SOCK"`.
`debugging`	A boolean variable set to FALSE by default. If TRUE, additional messages will be printed. For de-bugging only.

Details

Berger-Parker's index is the relative abundance of the most abundant category (i.e., unique numerical values in the considered numerical matrix). Berger-Parker's index equals the logarithm of the inverse Renyi's index of order infinity, $log(1/^{\infty}H)$ or the inverse of Hill's index of order infinity, $1/^{\infty}D$.

Value

A numerical matrix with dimensions as `dim(x)`.

Note

Linux users need to install libopenmpi for MPI parallel computing. Linux Ubuntu users may try: apt-get update; apt-get upgrade; apt-get install mpi; apt-get install libopenmpi-dev; apt-get install r-cran-rmpi

Microsoft Windows users may need some additional work to use "MPI", see:
`https://bioinfomagician.wordpress.com/2013/11/18/installing-rmpi-mpi-for-r-on-mac-and-windows/`

Author(s)

Marcantonio Matteo <`marcantoniomatteo@gmail.com`>, Martina Iannacito <`martina.iannacito@inria.fr`>, Duccio Rocchini <`duccio.rocchini@unibo.it`>

References

Berger, W.H., Parker, F.L. (1970). Diversity of planktonic foraminifera in deep-sea sediments". Science, 168: 1345-1347.

Examples

```
## Not run:
# Minimal example; compute Renyi's index with alpha 1:5
a <- matrix(c(10,10,10,20,20,20,20,30,30),ncol=3,nrow=3)
berpar <- BergerParker(x=a, window=3)

## End(Not run)
```

BergerParkerP *Calculate Berger-Parker Index on a Matrix*

Description

This function computes Berger-Parker Index for each cell of a matrix, using a parallelized approach and considering a specified moving window.

Usage

```
BergerParkerP(x, window = 1, na.tolerance = 1, debugging = FALSE, np = 1)
```

Arguments

`x`	A numeric matrix representing the data on which the index is to be calculated.
`window`	The width of the moving window to consider for each cell. The actual window size will be '(2 * window + 1) x (2 * window + 1)'. Default is 1.
`na.tolerance`	The tolerance level for missing data within the moving window. A window will be processed only if the proportion of non-missing data is above this threshold. Value should be between 0 and 1. Default is 1.
`debugging`	Boolean flag to enable or disable debugging messages. Default is FALSE.
`np`	The number of processes (cores) which will be spawned. Default value is 2.

Value

A matrix of the same dimensions as 'x', where each cell contains the Berger-Parker Index calculated for the window around the cell.

Examples

```
## Not run:
data <- matrix(runif(100), nrow = 10)
bp_index <- BergerParkerP(data, window = 1, np=2)

## End(Not run)
```

`BergerParkerS` *Sequential Berger-Parker's diversity index*

Description

This function calculates the Berger-Parker's diversity index for each cell in a matrix, considering a specified moving window around each cell.

Usage

```
BergerParkerS(x, window = 1, na.tolerance = 1, debugging = FALSE)
```

Arguments

`x`	A numeric matrix representing the data on which the index is to be calculated.
`window`	The width of the moving window to consider for each cell. The actual window size will be '(2 * window + 1) x (2 * window + 1)'. Default is 1.
`na.tolerance`	The tolerance level for missing data within the moving window. A window will be processed only if the proportion of non-missing data is above this threshold. Value should be between 0 and 1. Default is 1.
`debugging`	Boolean flag to enable or disable debugging messages. Default is FALSE.

Details

Berger-Parker's diversity index calculated sequentially over a raster matrix.

Value

A matrix of the same dimensions as '`x`', where each cell contains the Berger-Parker's diversity index calculated for the window around the cell.

Examples

```
data <- matrix(runif(100), nrow = 10)
bp_index <- BergerParkerS(data, window = 1)
```

`copNDVI` *Copernicus Long Term (1999-2017) NDVI Overview (5km)*

Description

A ‘SpatRaster‘ (EPSG: 4326) of the global average NDVI value per pixel for the 21st of June over the period 1999-2017.

Format

A ‘SpatRaster‘ containing the following elements:

NDVI Normalised Difference Vegetation Index value (0-255) for each 5 km pixel. This index provides an indication of the presence of live green vegetation in the area.

Details

This dataset provides a long-term overview of the Normalised Difference Vegetation Index (NDVI) across the globe. Each pixel represents a 5 km area, with NDVI values ranging from 0 to 255.

Source

https://land.copernicus.eu/en/products/vegetation

References

https://land.copernicus.eu/en/products/vegetation

Examples

```
copNDVI <- load_copNDVI()
```

`CRE` *Cumulative Residual Entropy (CRE) Function*

Description

Computes the Cumulative Residual Entropy (CRE) for spatial raster data. This function can be used with either a single raster layer or a list of raster layers. It supports both classic and multidimensional methods for CRE computation.

Usage

```
CRE(
  x,
  window = 3,
  method = "classic",
  rasterOut = TRUE,
  rescale = FALSE,
  na.tolerance = 1,
  simplify = 2,
  np = 1,
  cluster.type = "SOCK",
  progBar = TRUE,
  debugging = FALSE
)
```

Arguments

`x`	A matrix, SpatRaster, or a list of SpatRaster objects.
`window`	The size of the moving window, must be an odd integer.
`method`	The method for CRE computation, either "classic" or "multidimensional".
`rasterOut`	Logical, if TRUE, returns a SpatRaster, else returns a matrix.
`rescale`	Logical, if TRUE, rescales the data before processing.
`na.tolerance`	A numeric value between 0 and 1, indicating the tolerance level for NA values.
`simplify`	Integer, the number of decimal places for data rounding in case of float numbers.
`np`	The number of parallel processes to use.
`cluster.type`	The type of parallel cluster to use, options are "SOCK", "FORK", or "MPI".
`progBar`	logical. If TRUE a progress bar is shown.
`debugging`	Logical, if TRUE, provides additional debugging information during execution.

Value

Depending on the ’rasterOut’ parameter, this function returns either a SpatRaster or a matrix.

Examples

```
## Not run:
# For a matrix input:
result <- CRE(matrix_data, window=3, method="classic")

# For a SpatRaster input:
result <- CRE(raster_data, window=3, method="classic", rasterOut=TRUE)

## End(Not run)
```

heliPlot — *Create a Helical Plot for Time Series Data*

Description

Creates a helical plot to visualize time series data, emphasizing both the magnitude and rate of change over time.

Usage

```
heliPlot(
  data,
  group = NULL,
  facet = FALSE,
  xlabel = "Rate of Change",
  ylabel = "Values",
  arrow = TRUE,
  dateFont = 3,
  dateInterval = FALSE,
  sizeRange = c(1, 3),
  facetScales = "free",
  dateFormat = "%d %b %y",
  n = nrow(data),
  ...
)
```

Arguments

`data`	A data frame containing the time series data with required columns: "values_avg", "change_rate", and "date".
`group`	(Optional) A string specifying the column name in ‘data‘ to use for grouping data in the plot. If NULL, no grouping is applied.
`facet`	Logical indicating whether to facet the plot based on the ‘group‘ variable. If TRUE and ‘group‘ is NULL, an error is raised.
`xlabel`	Label for the x-axis, defaults to "Rate of Change".
`ylabel`	Label for the y-axis, defaults to "Values".
`arrow`	Logical indicating whether to add an arrow to the end of each line, defaults to TRUE.
`dateFont`	Numeric specifying the size of the date font, defaults to 3.
`dateInterval`	Numeric specifying the interval at which date labels should be displayed. If FALSE, no date labels are shown.
`sizeRange`	Numeric vector of length 2 specifying the range of line widths.
`facetScales`	Character string indicating whether scales should be "fixed", "free_x", "free_y", or "free".

`dateFormat`	Format for the date labels, defaults to d-b-y.
`n`	Numeric specifying the number of points to interpolate along the spline, defaults to the number of rows in ‘data‘.
`...`	Additional arguments passed on to ‘ggplot2‘ layer functions.

Value

A ‘ggplot‘ object representing the helical plot.

Examples

```
# Assuming `dataPrep` is a data frame prepared with the required structure:

## Not run:
heliPlot(dataPrep, group = "myGroup", arrow = TRUE,
dateFont = 3, dateInterval = 30, sizeRange = c(1, 3))

## End(Not run)
```

`heliPrep` *Prepare Data for Helical Plotting*

Description

This function preprocesses a time series data for helical plotting by applying a moving average and smoothing the values and their rate of change. It also handles conversion of numeric dates to Date objects and ensures proper alignment of the time series for plotting.

Usage

```
heliPrep(dates, values, filterWidth = 7)
```

Arguments

`dates`	A vector of dates associated with the values; can be numeric or Date objects. If numeric, they are treated as days since a given start date.
`values`	A numeric vector of the time series values corresponding to the dates.
`filterWidth`	The size of the moving window to calculate the moving average. Defaults to 7

Value

A data frame suitable for helical plotting, containing the original dates, the smoothed values (‘ch_avg‘), the smoothed rate of change (‘ch_rate‘), and the endpoints for plotting (‘yend‘, ‘xend‘).

Examples

```
## Not run:
  # Assume 'dates' and 'values' are available time series data
  prepared_data <- heliPrep(dates, values)
  # Now 'prepared_data' can be used for helical plotting with 'heliPlot'

## End(Not run)
```

Hill   *Hill's index of diversity - Hill numbers (D)*

Description

Computes Hill's index of diversity (Hill numbers) on different classes of numeric matrices using a moving window algorithm.

Usage

```
Hill(
  x,
  window = 3,
  alpha = 1,
  base = exp(1),
  rasterOut = TRUE,
  np = 1,
  na.tolerance = 1,
  cluster.type = "SOCK",
  debugging = FALSE
)
```

Arguments

`x`	Input data may be a matrix, a Spatial Grid Data Frame, a SpatRaster, or a list of these objects. In the latter case, only the first element of the list will be considered.
`window`	The side of the square moving window. It must be an odd numeric value greater than 1 to ensure that the target pixel is in the centre of the moving window. Default value is 3.
`alpha`	Order of the Hill number to compute the index. If `alpha` is a vector with length greater than 1, then the index will be calculated over `x` for each value in the sequence.
`base`	The logarithm base for the calculation, default is natural logarithm.
`rasterOut`	Boolean; if TRUE, the output will be in SpatRaster format with `x` as the template.
`np`	The number of processes (cores) which will be spawned. Default value is 1.

`na.tolerance`	A numeric value between 0.0 and 1.0, which indicates the proportion of NA values that will be tolerated to calculate Hill's index in each moving window over `x`. If the relative proportion of NA's in a moving window is bigger than na.tolerance, then the value of the window will be set as NA; otherwise, Hill's index will be calculated considering the non-NA values. Default value is 1.0 (i.e., full tolerance for NA's).
`cluster.type`	The type of cluster which will be created. Options are "MPI" (calls "makeMPIcluster"), "FORK," and "SOCK" (call "makeCluster"). Default type is "SOCK".
`debugging`	A boolean variable set to FALSE by default. If TRUE, additional messages will be printed for debugging purposes.

Details

Hill numbers (${}^qD$) are calculated on numerical matrices as ${}^qD = (\sum_{i=1}^{R} {p^q}_i)^{1/(1-q)}$, where q is the order of the Hill number, R is the total number of categories (i.e., unique numerical values in a numerical matrix), and p is the relative abundance of each category. When q=1, Shannon.R is called to calculate $exp(H^1)$ instead of the indefinite ${}^1D$. If $q > 2 * 10^9$, BergerParker.R is called to calculate $1/{}^\infty D$. Hill numbers of low order weight more rare categories, whereas Hill numbers of higher order weight more dominant categories.

Value

A list of matrices of dimension `dim(x)` with length equal to the length of `alpha`.

Note

Linux users need to install libopenmpi for MPI parallel computing. Linux Ubuntu users may try: `apt-get update; apt-get upgrade; apt-get install mpi; apt-get install libopenmpi-dev; apt-get install r-cran-rmpi`

Microsoft Windows users may need some additional work to use "MPI". For more details, see: https://bioinfomagician.wordpress.com/2013/11/18/installing-rmpi-mpi-for-r-on-mac-and-windows/

References

Hill, M.O. (1973). Diversity and evenness: a unifying notation and its consequences. Ecology 54, 427-432.

See Also

`BergerParker`, `Shannon`

Examples

```
# Minimal example; compute Hill's index with alpha 1:5
a <- matrix(c(10,10,10,20,20,20,20,30,30),ncol=3,nrow=3)
hill <- Hill(x=a,window=3,alpha=1:5)
```

`load_copNDVI` *Load Copernicus NDVI data*

Description

This function loads and returns the Copernicus Long Term (1999-2017) NDVI Overview stored within the package.

Usage

```
load_copNDVI()
```

Value

A ‘SpatRaster‘ object representing the Copernicus NDVI data.

Examples

```
copNDVI <- load_copNDVI()
```

`mpaRaoAreaS` *Area-Based Sequential Parametric Rao’s index of quadratic entropy (Q)*

Description

Calculates an area-based sequential version of the parametric Rao’s index of quadratic entropy (Q). This function is designed for situations where the diversity index needs to consider geographical areas and works with raster data representing the distribution of species or other measures.

Usage

```
mpaRaoAreaS(rasterm, area, alpha, simplify, dist_m, rescale, lambda, window)
```

Arguments

`rasterm`	Raster; the input raster data representing variables across a geographic space.
`area`	Numeric; the input vector data representing the areas of interest.
`alpha`	Numeric; alpha value for order of diversity in Hill’s Index.
`simplify`	Numeric; the parameter that determines the rounding off of the calculations.
`dist_m`	Character; type of distance metric used (e.g., "euclidean", "manhattan", etc.).
`rescale`	Logical; whether to scale and centre the values in each element of the raster data.
`lambda`	Numeric; lambda parameter for Minkowski distance calculation.
`window`	Numeric; defines the size of the moving window for the analysis.

Value

A vector similar to the input, with additional columns representing Rao’s index values for each area.

Author(s)

Matteo Marcantonio <marcantoniomatteo@gmail.com>, Duccio Rocchini <duccio.rocchini@unibo.it>, Michele Torresani <michele.torresani@unibo.it>

See Also

paRao for a related function dealing with the parallel computation of Rao’s index.

---

mpaRaoS *Multidimensional sequential Parametric Rao’s index of quadratic entropy (Q)*

Description

This function calculates the multidimensional parametric Rao’s index of quadratic entropy (Q) using a sequential method. It is particularly useful in contexts where parallel computation is not feasible or desired. The function applies a moving window approach to the provided raster data stack.

Usage

```
mpaRaoS(
  x,
  alpha,
  window,
  dist_m,
  na.tolerance,
  rescale,
  lambda,
  diag,
  time_vector,
  stepness,
  midpoint,
  cycle_length,
  time_scale,
  debugging,
  isfloat,
  mfactor,
  np,
  progBar
)
```

Arguments

`x`	input list.
`alpha`	Numeric; alpha value for order of diversity in Hill's Index.
`window`	Numeric; half of the side of the square moving window used for calculation.
`dist_m`	Character; type of distance used in the analysis.
`na.tolerance`	Numeric; a threshold between 0.0 and 1.0 indicating the allowable proportion of NA values within each moving window. If the proportion of NA values exceeds this, the window's value is set as NA; otherwise, the computation uses the non-NA values.
`rescale`	Logical; if TRUE, scales and centres the values in each element of 'x'.
`lambda`	Numeric; lambda value used for Minkowski distance calculation.
`diag`	Logical; if TRUE, includes the diagonal of the distance matrix in computations.
`time_vector`	time;
`stepness`	numeric; steepness of the logistic function.
`midpoint`	numeric; midpoint of the logistic function
`cycle_length`	string; The length of the cycle. Can be a numeric value or a string specifying the units ('year', 'month', 'day', 'hour', 'minute', 'second'). When numeric, the cycle length is in the same units as time_scale. When a string, it specifies the time unit of the cycle.
`time_scale`	string; Specifies the time scale for the conversion. Must be one of 'year', 'month', 'day', 'hour', 'minute', 'second'. When cycle_length is a string, time_scale changes the unit in which the result is expressed. When cycle_length is numeric, time_scale is used to compute the elapsed time in seconds.
`debugging`	Logical; if TRUE, additional diagnostic messages are output, useful for debugging. Default is FALSE.
`isfloat`	Logical; specifies if the input data are floats.
`mfactor`	Numeric; multiplication factor applied if input data are float numbers.
`np`	Number of processes for parallel computation.
`progBar`	logical. If TRUE a progress bar is shown.

Value

A list of matrices, each representing a layer of the input RasterStack, containing calculated Rao's index values. The dimensions correspond to those of the input, and the list length is equal to the length of 'alpha'.

Author(s)

Duccio Rocchini <`duccio.rocchini@unibo.it`>, Matteo Marcantonio <`marcantoniomatteo@gmail.com`>

See Also

`paRao` for the parallelized version of the Rao's index computation.

`ndviForestTS` *Simulated NDVI dataset*

Description

A `list` of 8-bit matrices.

Format

A `list` containing matrices:

ndviForestTS List of matrixes of 9 cells simulating NDVI of a patch of forests over 3 years. Each matrix represents a day in the time series.

Details

This list represents a time series of NDVI values of a patch of forest over 3 years. It is stored as a `list`, suitable for explaining how to make helical plots.

Examples

```
ndviForestTS <- readRDS(system.file("extdata", "ndviForestTS.rds", package = "rasterdiv"))
```

`openCluster` *Open a Parallel Cluster*

Description

Opens a parallel cluster for computation, registers it for parallel operations, and ensures its closure on script exit.

Usage

```
openCluster(cluster.type = "SOCK", np = 2, progBar = TRUE, debugging = FALSE)
```

Arguments

`cluster.type`	A character string specifying the type of cluster. Accepted values are "SOCK", "FORK", or "MPI".
`np`	An integer specifying the number of processes to be used in the parallel cluster.
`progBar`	logical. If TRUE a progress bar is shown.
`debugging`	logical. For developer use.

Value

An object representing the parallel cluster.

Examples

```
## Not run:
  # Open a SOCK cluster with 4 cores
  cls <- openCluster("SOCK", 4)
  # Your parallel computation code here
  # The cluster will automatically close when the script exits

## End(Not run)
```

---

paRao — *Parametric Rao's index of quadratic entropy (Q)*

Description

It computes the parametric version of Rao's index of quadratic entropy (Q) on different classes of numeric matrices using a moving window algorithm.

Usage

```
paRao(
  x,
  area = NULL,
  field = NULL,
  dist_m = "euclidean",
  window = 9,
  alpha = 1,
  method = "classic",
  rasterOut = TRUE,
  lambda = 0,
  na.tolerance = 1,
  rescale = FALSE,
  diag = TRUE,
  simplify = 0,
  np = 1,
  cluster.type = "SOCK",
  progBar = TRUE,
  debugging = FALSE,
  time_vector = NA,
  stepness = -0.5,
  midpoint = 35,
  cycle_length = "year",
  time_scale = "day"
)
```

Arguments

`x`	Input data may be a matrix, a Spatial Grid Data Frame, a SpatRaster, or a list of these objects.
`area`	Input vector area layer for area-based calculation.
`field`	Column name of the vector area layer to use to calculate the index.
`dist_m`	Define the type of distance to be calculated between numerical categories. ‘dist_m‘ can be a character string which defines the name of the distance to derive such as "euclidean". The distance names allowed are the same as for `proxy::dist`. Alternatively, ‘dist_m‘ can be a function which calculates a user-defined distance, (i.e., `function(x,y) {return(cos(y-x)-sin(y-x))}`) or a matrix of distances. If ‘method="multidimension"‘ then only "euclidean", "manhattan", "canberra", "minkowski" and "mahalanobis" can be used. Default value is "euclidean". If ‘dist_m‘ is a matrix, then the function will assume that the matrix contains the distances. Moreover *"twdtw"* (time weighted dynamic time warping) can be used as a way to calculate distances for time series in the ‘paRao‘ multidimensional mode.
`window`	The side of the square moving window, it must be a vector of odd numeric values greater than 1 to ensure that the target pixel is in the centre of the moving window. Default value is 3. ‘window‘ can be a vector with length greater than 1, in this case, Rao’s index will be calculated over ‘x‘ for each value in the vector.
`alpha`	Weight for the distance matrix. If ‘alpha = 0‘, distances will be averaged with a geometric average, if ‘alpha=1‘ with an arithmetic mean, if ‘alpha = 2‘ with a quadratic mean, ‘alpha = 3‘ with a cubic mean, and so on. if ‘alpha‘ tends to infinite (i.e., higher than the maximum integer allowed in R) or ‘alpha=Inf‘, then the maximum distance will be taken. ‘alpha‘ can be a vector with length greater than 1, in this case, Rao’s index will be calculated over ‘x‘ for each value in the vector.
`method`	Currently, there are two ways to calculate the parametric version of Rao’s index. If ‘method="classic"‘, then the normal parametric Rao’s index will be calculated on a single matrix. If ‘method="multidimension"‘ (experimental!), a list of matrices must be provided as input. In the latter case, the overall distance matrix will be calculated in a multi- or hyper-dimensional system by using the distance measure defined through the function argument ‘dist_m‘. Each pairwise distance is then multiplied by the inverse of the squared number of pixels in the considered moving window, and the Rao’s Q is finally derived by applying a summation. Default value is ‘"classic"‘.
`rasterOut`	Boolean, if TRUE the output will be a SpatRaster object with ‘x‘ as a template.
`lambda`	The value of the lambda of Minkowski’s distance. Considered only if ‘dist_m = "minkowski"‘ and ‘method="multidimension"‘. Default value is 0.
`na.tolerance`	Numeric value (0.0-1.0) which indicates the proportion of NA values that will be tolerated to calculate Rao’s index in each moving window over ‘x‘. If the relative proportion of NA’s in a moving window is bigger than ‘na.tolerance‘, then the value of the window will be set as NA, otherwise Rao’s index will be calculated considering the non-NA values. Default value is 1.0. In the rare event that a moving window has NA cells number < ‘na.tolerance‘ threshold but has only 1 non-NA value, then its resulting Rao value will always be 0.

`rescale`	Boolean. Considered only if ‘method="multidimension"‘. If TRUE, each element of ‘x‘ is rescaled and centred.
`diag`	Boolean. If TRUE then the diagonal of the distance matrix is filled with 0’s, otherwise with NA’s. If ‘diag=TRUE‘ and ‘alpha=0‘, the output matrix will inexorably be 0’s.
`simplify`	Number of decimal places to be retained to calculate distances in Rao’s index. Default ‘simplify=0‘.
`np`	The number of processes (cores) which will be spawned. Default value is 2.
`cluster.type`	The type of cluster which will be created. The options are ‘"MPI"‘ (which calls "makeMPIcluster"), ‘"FORK"‘, and ‘"SOCK"‘ (which call "makeCluster"). Default type is ‘"SOCK"‘.
`progBar`	logical. If TRUE a progress bar is shown.
`debugging`	A boolean variable set to FALSE by default. If TRUE, additional messages will be printed. For debugging only.
`time_vector`	time;
`stepness`	numeric; steepness of the logistic function.
`midpoint`	numeric; midpoint of the logistic function
`cycle_length`	string; The length of the cycle. Can be a numeric value or a string specifying the units (’year’, ’month’, ’day’, ’hour’, ’minute’, ’second’). When numeric, the cycle length is in the same units as time_scale. When a string, it specifies the time unit of the cycle.
`time_scale`	string; Specifies the time scale for the conversion. Must be one of ’year’, ’month’, ’day’, ’hour’, ’minute’, ’second’. When cycle_length is a string, time_scale changes the unit in which the result is expressed. When cycle_length is numeric, time_scale is used to compute the elapsed time in seconds.

Details

The parametric Rao’s Index (Q) is an extension of Rao’s Index which considers a generalized mean between distances. The general formula for the parametric Rao’s index is Q_a =

$$Q = \sum_{i,j} p_i p_j d_{ij}^{\alpha}$$

. Where ‘N‘ is the number of numerical categories, ‘i‘ and ‘j‘ are pair of numerical categories in the same moving window, and ‘alpha‘ is a weight given to distances. In the "multidimension" Rao’s index, first the distances among categories are calculated considering more than one feature, and then the overall Rao’s Q is derived by using these distances.

Value

A list of matrices of dimension ‘dim(x)‘ with length equal to the length of ‘alpha‘. If ‘rasterOut=TRUE‘ and ‘x‘ is a SpatRaster, then the output is a list of SpatRaster objects.

References

Rao, C. R. (1982). Diversity and dissimilarity coefficients: A unified approach. Theoretical Population Biology, 21(1), 24-43.

Examples

```
## Not run:
# loading data
data(volcano)
r <- terra::rast(volcano)

# we want to compute Rao's index on this data using a 3x3 window
res <- paRao(x = r, window = 3, alpha = 2, method = "classic")
terra::plot(res[[1]][[1]])

## End(Not run)
```

---

paRaoP — *Parallelized Parametric Rao's index of quadratic entropy (Q)*

Description

This function computes the parametric Rao's index of quadratic entropy (Q), a measure of biodiversity that considers the evolutionary distances between species, utilizing parallel computing for enhanced performance. The computation is applied over a moving window across the input data.

Usage

```
paRaoP(
  x,
  alpha,
  window,
  dist_m,
  na.tolerance,
  diag,
  debugging,
  isfloat,
  mfactor,
  np,
  progBar
)
```

Arguments

`x`	Matrix or data frame; the input data over which the index calculation is performed.
`alpha`	Numeric; specifies the alpha value for the order of diversity in Hill's Index.
`window`	Numeric; half of the side length of the square moving window used in the calculation.
`dist_m`	Character; specifies the type of distance metric used in calculations.

`na.tolerance`	Numeric; the threshold proportion of NA values allowed in the moving window. If exceeded, the calculation for that window is skipped. Values range from 0.0 (no tolerance) to 1.0.
`diag`	Logical; indicates whether the diagonal of the distance matrix should be included in the computation. Typically set to FALSE.
`debugging`	Logical; set to FALSE by default. If TRUE, additional console messages will be displayed for debugging purposes.
`isfloat`	Logical; indicates whether the input data values are floating-point numbers.
`mfactor`	Integer; multiplication factor in case of input data as float numbers.
`np`	Number of processes for parallel computation.
`progBar`	logical. If TRUE a progress bar is shown.

Value

A list of matrices corresponding to the computed Rao's index values. Each matrix in the list represents the calculations performed over the moving window, with dimensions equal to `dim(x)`.

Author(s)

Duccio Rocchini <duccio.rocchini@unibo.it>, Matteo Marcantonio <marcantoniomatteo@gmail.com>

See Also

`paRao` for the related non-parallelized function.

---

paRaoS *Sequential Parametric Rao's index of quadratic entropy (Q)*

Description

Computes the sequential version of the parametric Rao's index of quadratic entropy (Q), a measure used in environmental and ecological studies to assess biodiversity by considering the evolutionary distance between species. The function performs calculations in a sequential manner over a moving window across the input data.

Usage

```
paRaoS(
  x,
  alpha,
  window,
  dist_m,
  na.tolerance,
  diag,
  debugging,
  isfloat,
  mfactor,
  progBar
)
```

Arguments

`x`	Matrix or data frame; the input data over which the index calculation is performed.
`alpha`	Numeric; specifies the alpha value for the order of diversity in Hill's Index.
`window`	Numeric; half of the side length of the square moving window used in the calculation.
`dist_m`	Character; specifies the type of distance metric used in calculations.
`na.tolerance`	Numeric; the threshold proportion of NA values allowed in the moving window. If exceeded, the calculation for that window is skipped. Values range from 0.0 (no tolerance) to 1.0.
`diag`	Logical; indicates whether the diagonal of the distance matrix should be included in the computation. Typically set to FALSE.
`debugging`	Logical; set to FALSE by default. If TRUE, additional console messages will be displayed for debugging purposes.
`isfloat`	Logical; indicates whether the input data values are floating-point numbers.
`mfactor`	Integer; indicates the decimal position to round.
`progBar`	logical. If TRUE a progress bar is shown.

Value

A list of matrices corresponding to the computed Rao's index values. Each matrix in the list represents the calculations performed over the moving window, with dimensions equal to `dim(x)`.

Author(s)

Duccio Rocchini <`duccio.rocchini@unibo.it`>, Matteo Marcantonio <`marcantoniomatteo@gmail.com`>

See Also

`paRao` for the related non-sequential function.

`Pielou` *Pielou's Evenness Index*

Description

Calculates Pielou's Evenness Index for a given raster object over a specified window size. The function can operate in either sequential or parallel mode.

Usage

```
Pielou(
  x,
  window = 3,
  rasterOut = TRUE,
  np = 1,
  na.tolerance = 1,
  cluster.type = "SOCK",
  debugging = FALSE
)
```

Arguments

`x`	A raster object (matrix, SpatRaster, SpatialGridDataFrame, or a list containing one of these).
`window`	The size of the moving window to be used for the calculation. Must be an odd integer.
`rasterOut`	Logical, if TRUE the output will be a raster object; if FALSE a matrix.
`np`	The number of processes to use in parallel mode. If np > 1, parallel computation is enabled.
`na.tolerance`	The tolerance level for NA values within the moving window, expressed as a proportion (0 to 1).
`cluster.type`	The type of cluster to use for parallel computation (e.g., "SOCK", "FORK").
`debugging`	Logical, if TRUE debugging information will be printed.

Value

Returns a raster object or matrix containing the Pielou's Evenness Index values.

`PielouP` *Parallelised Pielou's diversity index*

Description

This function calculates Pielou's diversity index in a parallelized manner, allowing for improved performance on suitable hardware. The diversity index is computed using a moving window approach over the input data.

Usage

```
PielouP(x, window = 1, na.tolerance = 1, debugging = FALSE, np)
```

Arguments

`x`	Input raster data, representing the environmental variable(s) over which the diversity index should be calculated.
`window`	The size of the half-side of the square moving window used in the calculation. This determines the scale at which diversity is assessed.
`na.tolerance`	A numeric value (between 0.0 and 1.0) indicating the proportion of NA values that are acceptable in each moving window over the raster data. If the proportion of NA values in a window exceeds this threshold, the resulting value for that window is set as NA. The default is 0.0, indicating no tolerance for NA values.
`debugging`	Boolean flag indicating whether additional console output should be generated for debugging purposes. Defaults to FALSE.
`np`	The number of processes (cores) which will be spawned. Default value is 2.

Value

A matrix or list of matrices, depending on the input, containing the calculated Pielou diversity index values. Each cell in the output matrix represents the diversity index calculated from the corresponding moving window of the input data.

Author(s)

Marcantonio Matteo <`marcantoniomatteo@gmail.com`>, Martina Iannacito <`martina.iannacito@inria.fr`>, Duccio Rocchini <`duccio.rocchini@unibo.it`>

See Also

`Pielou` for the non-parallelized version of the Pielou's diversity index calculation.

Examples

```
## Not run:
# Demonstration of function with hypothetical data
# Ensure you replace this with actual raster data
demo_raster <- #... (your raster data here)
result <- PielouP(x = demo_raster, win = 3, na.tolerance = 0.1, debugging = FALSE)
# proceed with analyzing 'result'

## End(Not run)
```

`PielouS` *Sequential Pielou's diversity index*

Description

Computes Pielou's diversity index using a sequential method, particularly useful for handling large datasets that might not be efficiently processed in a standard, non-sequential manner.

Usage

```
PielouS(x, window = 1, na.tolerance = 1, debugging = FALSE)
```

Arguments

`x`	Input raster data, representing the environmental variable(s) over which the diversity index should be calculated.
`window`	The size of the half-side of the square moving window used in the calculation. This determines the scale at which diversity is assessed.
`na.tolerance`	A numeric value (between 0.0 and 1.0) indicating the proportion of NA values that are acceptable in each moving window over the raster data. If the proportion of NA values in a window exceeds this threshold, the resulting value for that window is set as NA. The default is 0.0, indicating no tolerance for NA values.
`debugging`	Boolean flag indicating whether additional console output should be generated for debugging purposes. Defaults to FALSE.

Value

A matrix or list of matrices, depending on the input, containing the calculated Pielou diversity index values. Each cell in the output matrix represents the diversity index calculated from the corresponding moving window of the input data.

Author(s)

Marcantonio Matteo <`marcantoniomatteo@gmail.com`>, Martina Iannacito <`martina.iannacito@inria.fr`>, Duccio Rocchini <`duccio.rocchini@unibo.it`>

See Also

`Pielou` for the standard computation of Pielou's diversity index.

Examples

```
## Not run:
# Demonstration of function with hypothetical data
# Ensure you replace this with actual raster data
demo_raster <- #... (your raster data here)
result <- PielouS(x = demo_raster, win = 3, na.tolerance = 0.1, debugging = FALSE)
# proceed with analyzing 'result'

## End(Not run)
```

`process_raster_result` *Process Raster Results*

Description

This function processes the results of a list of calculations, packaging them into SpatRaster objects and naming them appropriately.

Usage

```
process_raster_result(out, x, alpha, window)
```

Arguments

`out`	A list containing the results of calculations that need to be transformed into SpatRaster objects. Each element in the list corresponds to a different calculation result.
`x`	A list of SpatRaster objects or a similar object used as a template for creating new SpatRaster objects. Specifically, ‘x[[1]]‘ is used as the template.
`alpha`	Numeric or numeric vector indicating the weight(s) used in the distance calculations that generated ’out’. It is used to label the results appropriately.
`window`	Numeric or numeric vector indicating the size(s) of the moving window(s) used in the calculations that generated ’out’. It is used to label the results.

Details

The function is designed to post-process the results of spatial calculations performed on raster data. The typical use case is to process results from a function that performs calculations on different ’windows’ of the data, using varying ’alpha’ parameters, and returns the results as a list. This function takes that list, converts each element to a SpatRaster (using the first SpatRaster in ’x’ as a template), and assigns appropriate names to each based on the ’alpha’ and ’window’ parameters.

Value

A list of SpatRaster objects corresponding to the different processed results. Each SpatRaster is named based on the ’alpha’ and ’window’ parameters used in the calculation. The naming convention is ’alpha.<alpha value>’ for the inner lists and ’window.<window size>’ for the outer list.

Examples

```
## Not run:

# Assume 'result_list' is obtained from a previous calculation, containing
# multiple results to be converted to SpatRaster objects.
# 'raster_template' is a list of SpatRaster objects used as templates.

processed_results <- process_raster_result(out = result_list,
                                           x = raster_template,
                                           alpha = c(1, 2),
                                           window = c(3, 5))

## End(Not run)
```

Rao *Rao's index*

Description

An alias for ‘paRao‘ with ‘alpha‘ fixed at 2.

Usage

```
Rao(x, ...)
```

Arguments

`x`	Input data may be a matrix, a Spatial Grid Data Frame, a SpatRaster, or a list of these objects.
`...`	Other parameters passed to ‘paRao‘.

Value

A return value description.

Examples

```
## Not run:
data(volcano)
r <- terra::rast(volcano)
res <- Rao(x = r, window = 3)
terra::plot(res[[1]][[1]])

## End(Not run)
```

RaoAUC *Accumulation function for parametric Rao's index of quadratic entropy (Q)*

Description

RaoAUC computes the accumulation function (integral or area under the curve) of the parametric version of Rao's index of quadratic entropy (Q) on different classes of numeric matrices using a moving window algorithm.

Usage

```
RaoAUC(
  alphas = 1:5,
  x,
  dist_m = "euclidean",
  window = 9,
  method = "classic",
  rasterAUC = TRUE,
  lambda = 0,
  na.tolerance = 1,
  rescale = FALSE,
  diag = TRUE,
  simplify = 0,
  np = 1,
  cluster.type = "SOCK",
  debugging = FALSE
)
```

Arguments

`alphas`	A continuous vector of alphas in the form start:end over which integrated the parametric Rao's index. Default value is 1:5.
`x`	Input data may be a matrix, a Spatial Grid Data Frame, a SpatRaster, or a list of these objects. In the latter case, if `method="classic"` only the first element of the list will be considered.
`dist_m`	Define the type of distance to be calculated between numerical categories. `dist_m` can be a character string which defines the name of the distance to derive such as "euclidean". The distance names allowed are the same as for `proxy::dist`. Alternatively, `dist_m` can be a function which calculates a user-defined distance, (i.e., `function(x,y) {return(cos(y-x)-sin(y-x))}`) or a matrix of distances. If `method="multidimension"` then only "euclidean", "manhattan", "canberra", "minkowski" and "mahalanobis" can be used. Default value is "euclidean". If `proxy::dist` is a matrix then the function will assume that this is the distance matrix, and therefore no distance will be derived.

`window`	The side of the square moving window, it must be an odd numeric value greater than 1 to ensure that the target pixel is in the centre of the moving window. Default value is 3.
`method`	Currently, there are two ways to calculate the parametric version of Rao's index. If `method="classic"`, then the normal parametric Rao's index will be calculated on a single matrix. If `method="multidimension"` (experimental!) a list of matrices must be provided as input. In the latter case, the overall distance matrix will be calculated in a multi- or hyper-dimensional system by using the distance measure defined through the function argument `dist_m`. Each pairwise distance is then multiplied by the inverse of the squared number of pixels in the considered moving window, and the Rao's Q is finally derived by applying a summation. Default value is *"classic"*.
`rasterAUC`	Boolean, if TRUE the output will be a SpatRaster object with *x* as a raster template.
`lambda`	The value of the lambda of Minkowski's distance. Considered only if `dist_m = "minkowski"` and `method="multidimension"`. Default value is 0.
`na.tolerance`	Numeric value $(0.0 - 1.0)$ which indicates the proportion of NA values that will be tolerated to calculate parametric Rao's index in each moving window over *x*. If the relative proportion of NA's in a moving window is bigger than na.tolerance, then the value of the window will be set as NA, otherwise Rao's index will be calculated considering the non-NA values. Default values are 1.0 (i.e., no tolerance for NA's). Default value is 1.0.
`rescale`	Boolean. Considered only if `method="multidimension"`. If TRUE, each element of `x` is rescaled and centred.
`diag`	Boolean. If TRUE then the diagonal of the distance matrix is filled with 0's, otherwise with NA's. If `diag=TRUE` and `alpha=0`, the output matrix will inexorably be composed of 0's.
`simplify`	Number of decimal places to be retained to calculate distances in Rao's index. Only if *x* is floats.
`np`	The number of processes (cores) which will be spawned. Default value is 2.
`cluster.type`	The type of cluster which will be created. The options are *"MPI"* (which calls "makeMPIcluster"), *"FORK"* (which calls "makeForkCluster"), and *"SOCK"* (which calls "makeCluster"). Default type is *"SOCK"*.
`debugging`	A boolean variable set to FALSE by default. If TRUE, additional messages will be printed. For debugging only.

Details

The accumulation function for the parametric Rao's Index (Q) is calculated integrating numerically over a range of alphas. *RaoAUC* is therefore equal to $(\int_a^b \frac{1}{N^4} \cdot d^{\alpha}_{i,j})^{\frac{1}{\alpha}} dx$. Where *N* is the number of pixels in a moving window, and *alpha* is a weight assigned to distances.

Value

A matrix of dimension `dim(x)`. If `rasterAUC=TRUE`, then the output is a SpatRaster with *x* as a template.

Author(s)

Matteo Marcantonio <marcantoniomatteo@gmail.com>

References

Rocchini, D., M. Marcantonio, and C. Ricotta (2017). Measuring Rao's Q diversity index from remote sensing: An open source solution. Ecological Indicators. 72: 234–238.

See Also

paRao

Examples

```
# Minimal example; RaoAUC with alphas ranging from 1 to 10
a <- matrix(c(10,10,10,20,20,20,20,30,30), ncol=3, nrow=3)
out <- RaoAUC(alphas=1:10, x=a, window=3, dist_m="euclidean", na.tolerance=1, rasterAUC=TRUE)
```

Renyi *Renyi Diversity Index Calculation*

Description

Computes Renyi diversity index for a given raster object. This function allows specifying window size, alpha values, and various other parameters for the calculation of the Renyi index.

Usage

```
Renyi(
  x,
  window = 3,
  alpha = 1,
  base = exp(1),
  rasterOut = TRUE,
  np = 1,
  na.tolerance = 1,
  cluster.type = "SOCK",
  debugging = FALSE
)
```

Arguments

x	A raster object which can be a matrix, SpatialGridDataFrame, SpatRaster, list, or RasterStack.
window	The size of the moving window; must be an odd integer.
alpha	A numeric vector of alpha values for the Renyi index.

base	The logarithm base for the calculation, default is natural logarithm.
rasterOut	Logical; if TRUE, returns a SpatRaster object, otherwise returns a list.
np	Number of processes for parallel computation.
na.tolerance	Tolerance level for NA values, must be within [0-1].
cluster.type	Type of cluster for parallel computation, either "SOCK" or "MPI".
debugging	Logical; if TRUE, provides additional console output for debugging.

Value

A SpatRaster object or a list of calculated Renyi indices.

Examples

```
## Not run:
result <- Renyi(ndvi.8bit, window = 3, alpha = c(0, 1, 2))

## End(Not run)
```

RenyiP *Parallel Computation of Renyi's Diversity Index*

Description

This function computes Renyi's diversity index for each cell of a matrix, using a parallelized approach and considering a specified moving window.

Usage

```
RenyiP(
  x,
  window = 1,
  alpha = 1,
  base = exp(1),
  na.tolerance = 1,
  debugging = FALSE,
  np = 1
)
```

Arguments

x	A numeric matrix representing the data on which the index is to be calculated.
window	The width of the moving window to consider for each cell. The actual window size will be ‘(2 * window + 1) x (2 * window + 1)‘. Default is 1.
alpha	The alpha parameter for Renyi's index, influencing sensitivity to species abundance. Default is 1.

`base`	The base of the logarithm used in Renyi's formula. Default is 'exp(1)' (natural logarithm).
`na.tolerance`	The tolerance level for missing data within the moving window. A window will be processed only if the proportion of non-missing data is above this threshold. Value should be between 0 and 1. Default is 1.
`debugging`	Boolean flag to enable or disable debugging messages. Default is FALSE.
`np`	Number of processes for parallel computation.#'

Value

A matrix of the same dimensions as 'x', where each cell contains the Renyi's diversity index calculated for the window around the cell.

Examples

```
data <- matrix(runif(100), nrow = 10)
renyi_index <- RenyiP(data, window = 1, np = 1)
```

`RenyiS` *Sequential Renyi's diversity index*

Description

This function calculates the Renyi's diversity index index for each cell in a matrix, considering a specified moving window around each cell.

Usage

```
RenyiS(
  x,
  window = 1,
  alpha = 1,
  base = exp(1),
  na.tolerance = 1,
  debugging = FALSE
)
```

Arguments

`x`	A numeric matrix representing the data on which the index is to be calculated.
`window`	The width of the moving window to consider for each cell. The actual window size will be '(2 * window + 1) x (2 * window + 1)'. Default is 1.
`alpha`	(Not used in this function, included for compatibility) The alpha parameter for diversity indices, default is 1.

`base`	The base of the logarithm used in the Shannon formula, default is ‘exp(1)‘ (natural logarithm).
`na.tolerance`	The tolerance level for missing data within the moving window. A window will be processed only if the proportion of non-missing data is above this threshold. Value should be between 0 and 1. Default is 1.
`debugging`	Boolean flag to enable or disable debugging messages. Default is FALSE.

Details

Berger-Parker’s diversity index calculated sequentially over a raster matrix.

Value

A matrix of the same dimensions as ‘x‘, where each cell contains the Renyi’s diversity index calculated for the window around the cell.

Examples

```
data <- matrix(runif(100), nrow = 10)
renyi_index <- RenyiS(data, window = 1)
```

`Shannon` *Shannon’s Evenness Index*

Description

Calculates Shannon’s Evenness Index for a given raster object over a specified window size. The function can operate in either sequential or parallel mode.

Usage

```
Shannon(
  x,
  window = 3,
  rasterOut = TRUE,
  np = 1,
  na.tolerance = 1,
  cluster.type = "SOCK",
  debugging = FALSE
)
```

Arguments

`x`	A raster object (matrix, SpatRaster, SpatialGridDataFrame, or a list containing one of these).
`window`	The size of the moving window to be used for the calculation. Must be an odd integer.
`rasterOut`	Logical, if TRUE the output will be a raster object; if FALSE a matrix.
`np`	The number of processes to use in parallel mode. If np > 1, parallel computation is enabled.
`na.tolerance`	The tolerance level for NA values within the moving window, expressed as a proportion (0 to 1).
`cluster.type`	The type of cluster to use for parallel computation (e.g., "SOCK", "FORK").
`debugging`	Logical, if TRUE debugging information will be printed.

Value

Returns a raster object or matrix containing the Shannon's Evenness Index values.

`ShannonP` *Calculate Shannon-Wiener Index on a Matrix*

Description

This function computes Shannon-Wiener Index for each cell of a matrix, using a parallelized approach and considering a specified moving window.

Usage

```
ShannonP(x, window = 1, na.tolerance = 1, debugging = FALSE, np = 1)
```

Arguments

`x`	A numeric matrix representing the data on which the index is to abe calculated.
`window`	The width of the moving window to consider for each cell. The actual window size will be '(2 * window + 1) x (2 * window + 1)'. Default is 1.
`na.tolerance`	The tolerance level for missing data within the moving window. A window will be processed only if the proportion of non-missing data is above this threshold. Value should be between 0 and 1. Default is 1.
`debugging`	Boolean flag to enable or disable debugging messages. Default is FALSE.
`np`	Number of processes for parallel computation.

Value

A matrix of the same dimensions as 'x', where each cell contains the Shannon-Wiener Index calculated for the window around the cell.

Examples

```
data <- matrix(runif(100), nrow = 10)
shannon_index <- ShannonP(data, window = 1, np = 1 )
```

`ShannonS` *Calculate Shannon-Wiener Index on a Matrix*

Description

This function calculates the Shannon-Wiener Index for each cell in a matrix, considering a specified moving window around each cell.

Usage

```
ShannonS(x, window = 1, na.tolerance = 1, debugging = FALSE)
```

Arguments

`x`	A numeric matrix representing the data on which the index is to be calculated.
`window`	The width of the moving window to consider for each cell. The actual window size will be '(2 * window + 1) x (2 * window + 1)'. Default is 1.
`na.tolerance`	The tolerance level for missing data within the moving window. A window will be processed only if the proportion of non-missing data is above this threshold. Value should be between 0 and 1. Default is 1.
`debugging`	Boolean flag to enable or disable debugging messages. Default is FALSE.

Value

A matrix of the same dimensions as 'x', where each cell contains the Shannon-Wiener Index calculated for the window around the cell.

Examples

```
data <- matrix(runif(100), nrow = 10)
shannon_index <- ShannonS(data, window = 1)
```

`validateInputs` *Validate Input Parameters for Diversity Index Calculation*

Description

Validates the input parameters for diversity index calculation functions. Checks for valid raster types, window sizes, alpha values, and NA tolerance levels.

Usage

```
validateInputs(x, window, alpha = 1, na.tolerance)
```

Arguments

`x`	Raster object to be validated.
`window`	Size of the moving window for calculations.
`alpha`	Diversity index parameter, default is 1.
`na.tolerance`	Proportion of acceptable NA values within the window (range: 0 to 1).

Value

None. Throws an error if any input is invalid.

Index