Package ‘epm’

September 13, 2024

Type Package

Title EcoPhyloMapper

Version 1.1.3

Depends R (>= 4.0)

Imports terra (>= 1.5-21), sf, ape, viridisLite, pbapply, methods,
Rcpp (>= 0.12.9), RcppProgress

Suggests tmap, data.table, spdep, exactextractr

Description Facilitates the aggregation of species' geographic ranges from vector or raster spatial data, and that enables the calculation of various morphological and phylogenetic community metrics across geography. Citation: Title, PO, DL Swiderski and ML Zelditch (2022) <doi:10.1111/2041-210X.13914>.

License GPL (>= 3)

URL https://github.com/ptitle/epm

BugReports https://github.com/ptitle/epm/issues

NeedsCompilation yes

LinkingTo Rcpp, RcppProgress

LazyData true

RoxygenNote 7.3.2

ByteCompile true

Encoding UTF-8

Author Pascal Title [aut, cre] (<https://orcid.org/0000-0002-6316-0736>),
Donald Swiderski [aut],
Miriam Zelditch [aut]

Maintainer Pascal Title <pascal.title@stonybrook.edu>

Repository CRAN

Date/Publication 2024-09-13 17:30:02 UTC

Contents

addLegend *addLegend*

Description

Adds a legend to an existing plot, with some additional manual controls.

Usage

```
addLegend(
  r,
  params = NULL,
  direction,
  side,
  location = "right",
  nTicks = 3,
  adj = NULL,
  shortFrac = 0.02,
  longFrac = 0.3,
  axisOffset = 0,
  border = TRUE,
  ramp,
  isInteger = "auto",
  ncolors = 64,
  breaks = NULL,
  minmax = NULL,
  locs = NULL,
  label = "",
  cex.axis = 0.8,
  tcl = NA,
  labelDist = 0.7,
  minDigits = 2
)
```

Arguments

`r`	the epmGrid, rasterLayer, SpatRaster or sf object that has been plotted
`params`	If an epmGrid plot was saved to a variable, provide that here. Contents will override other arguments.
`direction`	direction of color ramp. If omitted, then direction is automatically inferred, otherwise can be specified as `horizontal` or `vertical`.
`side`	side for tick marks, see `axis` documentation. Automatically inferred if omitted.
`location`	either a location name (see `Details`), or coordinates for the corners of the bar legend `c(xmin, xmax, ymin, ymax)`.
`nTicks`	number of tick marks, besides min and max.

adj	if location is top, left, bottom or right, use this argument to adjust the location of the legend, defined in percent of the figure width. See Details for additional information.
shortFrac	Percent of the plot width range that will be used as the short dimension of the legend. Only applies to preset location options.
longFrac	Percent of the plot width range that will be used as the long dimension of the legend. Only applies to preset location options.
axisOffset	distance from color bar for labels, as a percent of the plot width.
border	logical, should the color legend have a black border
ramp	either a vector of color names that will be interpolated, or a color ramp function that takes an integer (see for example colorRampPalette). If omitted, defaults to default epm color palette.
isInteger	If auto, automatically determines if r is made up of integer values, otherwise TRUE or FALSE
ncolors	grain size of color ramp
breaks	If a custom set of color breaks were used in plotting r, pass those color breaks here. This overrides the minmax option.
minmax	min and max values from which the color ramp will be derived. If left as NULL, the min and max of r will be used.
locs	locations of tick marks, if NULL automatically placed. If this is supplied as a character vector, then the labels will be plotted verbatim.
label	text to plot alongside the legend
cex.axis	size of axis labels
tcl	length of tick marks (see help for tcl in ?par)
labelDist	distance from axis to axis labels (passed to mgp)
minDigits	minimum number of significant digits for labels

Details

A number of predefined locations exist in this function to make it easy to add a legend to a plot.

Preset locations are: topleft, topright, bottomleft, bottomright, left, right, top and bottom.
If more fine-tuned control is desired, then a numeric vector of length 4 can be supplied to location, specifying the min x, max x, min y and max y values for the legend.

Additionally, the adj argument can be used to more intuitively adjust where the legend is placed. adj is defined as a percentage of the figure width or height, left to right, or bottom to top, respectively. For example, if the legend is at the bottom, adj = 0.8 will place the legend 80% of the distance from the top of the figure, horizontally centered.

If an epmGrid object was plotted with plot.epmGrid, and if use_tmap = FALSE was specified, and if that plot was assigned to a variable, then you can supply that variable here to the params argument, and a number of options will be automatically handed over to this function.

See examples.

Value

Invisibly returns a list with the following components.

- coords: 2-column matrix of xy coordinates for each color bin in the legend.
- width: Coordinates for the short dimension of the legend.
- pal: the color ramp
- tickLocs: the tick mark locations in plotting units

Author(s)

Pascal Title

Examples

```
# create square-cell epmGrid object
tamiasEPM2 <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'square', method = 'centroid')

# need to disable tmap if we want to anything to a plot
plot(tamiasEPM2, use_tmap = FALSE, legend = FALSE)
addLegend(tamiasEPM2, location = 'right', label = 'richness')
addLegend(tamiasEPM2, location = 'top', label = 'richness')

# fine-tune placement
addLegend(tamiasEPM2, location=c(113281, 1265200, -1500000, -1401898), side = 1)

# Using the params option
xx <- plot(tamiasEPM2, use_tmap = FALSE, legend = FALSE,
col = viridisLite::magma)
addLegend(tamiasEPM2, params = xx, location = 'top')

# works with hex grids as well
xx <- plot(tamiasEPM, use_tmap = FALSE, legend = FALSE,
col = viridisLite::magma)
addLegend(tamiasEPM, params = xx, location = 'top')
```

---

addPhylo *addPhylo*

Description

Add a phylogeny to epmGrid object.

Usage

```
addPhylo(x, tree, replace = FALSE, verbose = FALSE)
```

Arguments

`x`	object of class `epmGrid`
`tree`	a phylogeny of class `phylo`, or a set of trees of class `multiPhylo`
`replace`	boolean; if a tree is already a part of `x`, should it be replaced?
`verbose`	if TRUE, list out all species that are dropped/excluded, rather than counts.

Details

If any species in the phylogeny are not found in the epmGrid geographical data, then those species will be dropped from the phylogeny, and a warning will be issued.

If providing a set of trees as a multiPhylo object, it is expected that all trees have the same tips.

Value

object of class `epmGrid`, with a `phylo` object as the list element named `phylo`.

Author(s)

Pascal Title

Examples

```
tamiasEPM
tamiasTree

addPhylo(tamiasEPM, tamiasTree)
```

`addTraits` *addTraits*

Description

Add univariate or multivariate trait data to an epmGrid object.

Usage

```
addTraits(x, data, replace = FALSE, verbose = FALSE)
```

Arguments

`x`	object of class `epmGrid`
`data`	named numeric vector, matrix or dataframe with rownames corresponding to species in `x` or pairwise matrix with row and column names corresponding to species in `x`. If pairwise matrix, the upper triangle of the matrix will be used for calculations.
`replace`	boolean; if data is already a part of `x`, should it be replaced?
`verbose`	if TRUE, list out all species that are dropped/excluded, rather than counts.

Details

If any species in data are not found in the epmGrid geographical data, then those species will be dropped from data, and a warning will be issued.

Value

object of class epmGrid, with trait data as the list element named data.

Author(s)

Pascal Title

Examples

```
tamiasEPM
tamiasTraits

addTraits(tamiasEPM, tamiasTraits)
```

---

betadiv_disparity *Map change in morphological disparity*

Description

Change in morphological disparity is calculating across a moving window of neighboring grid cells. To implement a custom function, see customBetaDiv.

Usage

```
betadiv_disparity(x, radius, slow = FALSE, nThreads = 1)
```

Arguments

x	object of class epmGrid.
radius	Radius of the moving window in map units.
slow	if TRUE, use an alternate implementation that has a smaller memory footprint but that is likely to be much slower. Most useful for high spatial resolution.
nThreads	number of threads for parallelization

Details

For each gridcell neighborhood (defined by the radius), we calculate the proportion of the full disparity contained in those grid cells, and then take the standard deviation of those proportions across the gridcell neighborhood. This way, the returned values reflect how much disparity (relative to the overall total disparity) changes across a moving window.

If the R package spdep is installed, this function should run more quickly.

Value

Returns a sf polygons object (if hex grid) or a SpatRaster object (if square grid).

Author(s)

Pascal Title

References

Foote M. 1993. Contributions of individual taxa to overall morphological disparity. Paleobiology. 19:403–419.

Examples

```
tamiasEPM

tamiasEPM <- addTraits(tamiasEPM, tamiasTraits)

z <- betadiv_disparity(tamiasEPM, radius = 150000)

plot(z)

# using square grid epmGrid
tamiasEPM2 <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'square', method = 'centroid')
tamiasEPM2 <- addTraits(tamiasEPM2, tamiasTraits)
z2 <- betadiv_disparity(tamiasEPM2, radius = 150000)

terra::plot(z2, col = sf::sf.colors(100))
```

---

betadiv_phylogenetic *Map phylogenetic turnover in species communities*

Description

Multisite phylogenetic community dissimilarity is calculated for each cell within a circular moving window of neighboring cells. To implement a custom function, see customBetaDiv.

Usage

```
betadiv_phylogenetic(
  x,
  radius,
  component = "full",
  focalCoord = NULL,
  slow = FALSE,
  nThreads = 1
)
```

Arguments

`x`	object of class `epmGrid`.
`radius`	Radius of the moving window in map units.
`component`	which component of beta diversity to use, can be `"turnover"`, `"nestedness"` or `"full"`
`focalCoord`	vector of x and y coordinate, see details
`slow`	if TRUE, use an alternate implementation that has a smaller memory footprint but that is likely to be much slower. Most useful for high spatial resolution.
`nThreads`	number of threads for parallelization

Details

For each cell, multisite dissimilarity is calculated for the focal cell and its neighbors. If `focalCoord` is specified, then instead of multisite dissimilarity within a moving window of gridcells, pairwise dissimilarity is calculated from the cell at the focal coordinates, to all other cells.

All metrics are based on Sorensen dissimilarity and range from 0 to 1: For each metric, the following components can be specified. These components are additive, such that the full metric = turnover + nestedness.

- turnover: species turnover without the influence of richness differences
- nestedness: species turnover due to differences in richness
- full: the combined turnover due to both differences in richness and pure turnover

If the R package spdep is installed, this function should run more quickly.

Value

Returns a sf polygons object (if hex grid) or a SpatRaster object (if square grid) with multisite community dissimilarity for each grid cell.

Author(s)

Pascal Title

References

Baselga, A. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21 (2012): 1223–1232.

Leprieur, F, Albouy, C, De Bortoli, J, Cowman, PF, Bellwood, DR & Mouillot, D. Quantifying Phylogenetic Beta Diversity: Distinguishing between "True" Turnover of Lineages and Phylogenetic Diversity Gradients. PLoS ONE 7 (2012): e42760–12.

Examples

```
tamiasEPM

tamiasEPM <- addPhylo(tamiasEPM, tamiasTree)

# phylogenetic turnover
beta_phylo_turnover <- betadiv_phylogenetic(tamiasEPM, radius = 70000,
component = 'turnover')
beta_phylo_nestedness <- betadiv_phylogenetic(tamiasEPM, radius = 70000,
component = 'nestedness')
beta_phylo_full <- betadiv_phylogenetic(tamiasEPM, radius = 70000,
component = 'full')

oldpar <- par(mfrow=c(1,3))
plot(beta_phylo_turnover, reset = FALSE, key.pos = NULL)
plot(beta_phylo_nestedness, reset = FALSE, key.pos = NULL)
plot(beta_phylo_full, reset = FALSE, key.pos = NULL)

# using square grid epmGrid
tamiasEPM2 <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'square', method = 'centroid')
tamiasEPM2 <- addPhylo(tamiasEPM2, tamiasTree)

beta_phylo_full <- betadiv_phylogenetic(tamiasEPM2, radius = 70000,
component = 'full')
beta_phylo_full_slow <- betadiv_phylogenetic(tamiasEPM2, radius = 70000,
component = 'full', slow = TRUE)

par(mfrow = c(1,2))
terra::plot(beta_phylo_full, col = sf::sf.colors(100))
terra::plot(beta_phylo_full_slow, col = sf::sf.colors(100))

# dissimilarity from a focal cell
focalBeta <- betadiv_phylogenetic(tamiasEPM, radius = 70000,
component = 'full', focalCoord = c(-1413764, 573610.8))
plot(focalBeta, reset = FALSE)
points(-1413764, 573610.8, pch = 3, col = 'white')

par(oldpar)
```

`betadiv_taxonomic` *Map turnover in species communities*

Description

Multisite taxonomic community dissimilarity is calculated for each cell within a circular moving window of neighboring cells. To implement a custom function, see `customBetaDiv`.

Usage

```
betadiv_taxonomic(
  x,
  radius,
  component = "full",
  focalCoord = NULL,
  slow = FALSE,
  nThreads = 1
)
```

Arguments

`x`	object of class `epmGrid`.
`radius`	Radius of the moving window in map units.
`component`	which component of beta diversity to use, can be `"turnover"`, `"nestedness"` or `"full"`
`focalCoord`	vector of x and y coordinate, see details
`slow`	if TRUE, use an alternate implementation that has a smaller memory footprint but that is likely to be much slower. Most useful for high spatial resolution.
`nThreads`	number of threads for parallelization

Details

For each cell, multisite dissimilarity is calculated from the focal cell and its neighbors. If `focalCoord` is specified, then instead of multisite dissimilarity within a moving window of gridcells, pairwise dissimilarity is calculated from the cell at the focal coordinates, to all other cells.

All metrics are based on Sorensen dissimilarity and range from 0 to 1.
For each metric, the following components can be specified. These components are additive, such that the full metric = turnover + nestedness.

- turnover: species turnover without the influence of richness differences
- nestedness: species turnover due to differences in richness richness and pure turnover

If the R package spdep is installed, this function should run more quickly.

Value

Returns a grid with multi-site community dissimilarity for each cell.

Author(s)

Pascal Title

References

Baselga, A. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21 (2012): 1223–1232.

Examples

```
tamiasEPM

tamiasEPM <- addPhylo(tamiasEPM, tamiasTree)
tamiasEPM <- addTraits(tamiasEPM, tamiasTraits)

# taxonomic turnover
beta_taxonomic_turnover <- betadiv_taxonomic(tamiasEPM, radius = 70000,
component = 'turnover')
beta_taxonomic_nestedness <- betadiv_taxonomic(tamiasEPM, radius = 70000,
component = 'nestedness')
beta_taxonomic_full <- betadiv_taxonomic(tamiasEPM, radius = 70000,
component = 'full')



oldpar <- par(mfrow = c(1, 3))
plot(beta_taxonomic_turnover, reset = FALSE, key.pos = NULL)
plot(beta_taxonomic_nestedness, reset = FALSE, key.pos = NULL)
plot(beta_taxonomic_full, reset = FALSE, key.pos = NULL)



# using square grid epmGrid
tamiasEPM2 <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'square', method = 'centroid')

beta_taxonomic_full <- betadiv_taxonomic(tamiasEPM2, radius = 70000,
component = 'full')
beta_taxonomic_full_slow <- betadiv_taxonomic(tamiasEPM2, radius = 70000,
component = 'full', slow = TRUE)

par(mfrow=c(1,2))
terra::plot(beta_taxonomic_full, col = sf::sf.colors(100))
terra::plot(beta_taxonomic_full_slow, col = sf::sf.colors(100))

# dissimilarity from a focal cell
focalBeta <- betadiv_taxonomic(tamiasEPM, radius = 70000,
component = 'full', focalCoord = c(-1413764, 573610.8))
plot(focalBeta, reset = FALSE)
points(-1413764, 573610.8, pch = 3, col = 'white')

par(oldpar)
```

`calcMeanShape` *Calculate mean shape per grid cell*

Description

For an epmGrid object that contains geometric morphometric shape coordinates, calculate the per-grid-cell mean shape.

Usage

```
calcMeanShape(x)
```

Arguments

x	object of class `epmGrid`

Details

This function will ignore cells that are empty.

Value

a list with 2 elements: (1) matrix where nrow = number of grid cells and ncol = the number of data columns. Each row is a vector of mean shape coordinates. (2) a matrix of xy coordinates corresponding to those grid cells.

Author(s)

Pascal Title

Examples

```
tamiasEPM
tamiasEPM <- addTraits(tamiasEPM, tamiasTraits)
meanshape <- calcMeanShape(tamiasEPM)

head(meanshape[[1]])
head(meanshape[[2]])
```

`coordsFromEpmGrid` *Retrieve coordinates from epmGrid*

Description

Return the centroid coordinates for a specified set of grid cells.

Usage

```
coordsFromEpmGrid(x, sites)
```

Arguments

`x`	object of class `epmGrid`
`sites`	locations of sites, see details.

Details

Sites can be cell indices as a numeric vector, or you can specify `sites = 'all'` to get all grid cells. If the epmGrid object is hexagon-based, then all grid cells that are occupied are returned. If the epmGrid is square-based, then all grid cells, occupied or empty, are returned.

Value

matrix with x and y coordinates.

Author(s)

Pascal Title

Examples

```
tamiasEPM

# from cell indices
cells <- c(2703, 90, 3112, 179)
coordsFromEpmGrid(tamiasEPM, cells)

# for all grid cells
dim(coordsFromEpmGrid(tamiasEPM, 'all'))
```

`createEPMgrid` *Create epmGrid object*

Description

Creates an epmGrid object from a range of species-specific inputs.

Usage

```
createEPMgrid(
  spDat,
  resolution = 50000,
  method = "centroid",
  cellType = "hexagon",
  percentThreshold = 0.25,
  retainSmallRanges = TRUE,
  extent = "auto",
```

```
  percentWithin = 0,
  dropEmptyCells = TRUE,
  checkValidity = FALSE,
  crs = NULL,
  nThreads = 1,
  template = NULL,
  verbose = FALSE,
  use.data.table = "auto"
)
```

Arguments

`spDat`	a number of possible input formats are possible. See details below.
`resolution`	vertical and horizontal spacing of grid cells, in units of the polygons' or points' projection.
`method`	approach used for gridding. Either `centroid` or `percentOverlap`. See details below.
`cellType`	either `hexagon` or `square`. See details below.
`percentThreshold`	the percent that a species range must cover a grid cell to be considered present. Specified as a proportion.
`retainSmallRanges`	boolean; should small ranged species be dropped or preserved. See details.
`extent`	if 'auto', then the maximal extent of the polygons will be used. If not 'auto', can be a SpatialPolygon, sf object, or raster, in which case the resulting epmGrid will be cropped and masked with respect to the polygon; or a spatial coordinates object, from which an extent object will be generated; or a numeric vector of length 4 with minLong, maxLong, minLat, maxLat. If 'global', a global extent will be specified. See `interactiveExtent` to draw your own extent.
`percentWithin`	The percentage of a species range that must be within the defined extent in order for that species to be included. This filter can be used to exclude species whose range barely enters the area of interest. The default value of 0 will disable this filter. If `extent == 'auto'`, then this filter will also have no effect, as the extent is defined by the species' ranges.
`dropEmptyCells`	only relevant for hexagonal grids, should empty cells be excluded from the resulting grid. Default is TRUE. Reasons to set this to FALSE may be if you want to retain a grid of a certain extent, regardless of which cells contain species.
`checkValidity`	if `TRUE`, then check polygon validity and repair if needed, using sf::st_make_valid.
`crs`	if supplying occurrence records in a non-spatial format, then you must specify the crs. For unprojected long/lat data, you can simply provide `crs = 4326`.
`nThreads`	if > 1, then employ parallel computing. This won't necessarily improve runtime.
`template`	a grid (SpatRaster, RasterLayer or sf) that will be directly used as the reference grid, bypassing any inference from the input data.
`verbose`	if TRUE, list out all species that are dropped/excluded, rather than counts.
`use.data.table`	if `'auto'`, this is determined by the size of the dataset. Primarily intended for debugging.

Details

Types of accepted inputs for argument `spDat`:

1. a list of polygon objects (sf or sp), named with taxon names.
2. a list of SpatRaster or RasterLayer grids, named with taxon names.
3. a multi-layer RasterStack or multi-layer SpatRaster.
4. a set of occurrence records, multiple accepted formats, see below.
5. a site-by-taxon presence/absence matrix.

If input data consist of **occurrence records** rather than polygons, then a couple of formats are possible:

1. You can provide a list of species-specific spatial point objects.
2. You can provide a single spatial object, where points have a taxon attribute.
3. You can provide a list of non-spatial species-specific dataframes.
4. You can provide a single non-spatial dataframe.

For options (1) and (3), the taxon names must be provided as the list names. For options (2) and (4), the columns must be 'taxon', 'x' and 'y' (or 'long', 'lat'). For options (3) and (4), as these are non-spatial, you must provide a crs object to the `crs` argument, so that the function knows what projection to use.

It is also possible to supply a **matrix with sites as rows and taxa as columns**. The contents of this matrix must be either 0 or 1. If this is the case, then a raster grid must be supplied under the template argument. This will be the grid system used for converting this presence/absence matrix to an epmGrid object. It is expected that the index order of the grid is the same as the row order of the matrix.

If input is a set of **species-specific grids**, then it is expected that all grids belong to the same overall grid system, i.e. that the cells align and that all grids have the same resolution. Grids do not need to have the same extent.

Any SpatialPolygon or SpatialPoints objects are converted to objects of class `sf`.

If `cellType = 'hexagon'`, then the grid is made of polygons via the sf package. If `cellType = 'square'`, then the grid is a raster generated via the terra package. Hexagonal cells have several advantages, including being able to be of different sizes (if the grid is in unprojected long/lat), and may be able to more naturally follow coastlines and non-linear features. However, the raster-based square cells will be much less memory intensive for high resolution datasets. Choice of grid type matters more for spatial resolution (total number of cells), than for number of species.

In the polygon-to-grid conversion process, two approaches are implemented. For `method = 'centroid'`, a range polygon registers in a cell if the polygon overlaps with the cell centroid. For `method = 'percentOverlap'`, a range polygon registers in a cell if it covers that cell by at least `percentThreshold` fraction of the cell.

If `retainSmallRanges = FALSE`, then species whose ranges are so small that no cell registers as present will be dropped. If `retainSmallRanges = TRUE`, then the cell that contains the majority of the the small polygon will be considered as present, even if it's a small percent of the cell.

If `retainSmallRanges = TRUE`, and an extent is provided, then species may still be dropped if they fall outside of that extent.

You may see the message `Failed to compute min/max, no valid pixels found in sampling. (GDAL error 1)`. This just means that a species did not register in any grid cells. If you specified `retainSmallRanges = TRUE`, then those species will be included in a subsequent step. Therefore, this message can be ignored.

For very large datasets, this function will make a determination as to whether or not there is sufficient memory. If there is not, an alternative approach that uses the data.table package will be employed. Please install this R package to take advantage of this feature.

This function is also enhanced by the installation of the exactextractr R package.

Value

an object of class `epmGrid`.

Author(s)

Pascal Title

Examples

```
library(sf)
# example dataset: a list of 24 chipmunk distributions as polygons
head(tamiasPolyList)

# hexagonal grid

tamiasEPM <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'hexagon', method = 'centroid')
tamiasEPM

# square grid
tamiasEPM2 <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'square', method = 'centroid')
tamiasEPM2

# use of a grid from one analysis for another analysis

tamiasEPM <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'hexagon', method = 'centroid')

tamiasEPM <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'hexagon', method = 'centroid', template = tamiasEPM[[1]])

#######

# demonstration of site-by-species matrix as input.
tamiasEPM2 <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'square', method = 'centroid')

## first we will use the function generateOccurrenceMatrix() to get
## a presence/absence matrix
pamat <- generateOccurrenceMatrix(tamiasEPM2, sites = 'all')
```

```
# here, our grid template will be tamiasEPM2[[1]]
tamiasEPM2[[1]]
xx <- createEPMgrid(pamat, template = tamiasEPM2[[1]])


#######
# demonstration with grids as inputs
## We will first generate grids from the range polygons
## (you normally would not do this -- you would have grids from some other source)

# define the extent that contains all range polygons
fullExtent <- terra::ext(terra::vect(tamiasPolyList[[1]]))
for (i in 2:length(tamiasPolyList)) {
fullExtent <- terra::union(fullExtent, terra::ext(terra::vect(tamiasPolyList[[i]])))
}

# create raster template
fullGrid <- terra::rast(fullExtent, res = 50000, crs = terra::crs(terra::vect(tamiasPolyList[[1]])))

# now we can convert polygons to a common grid system
spGrids <- list()
for (i in 1:length(tamiasPolyList)) {
spGrids[[i]] <- terra::trim(terra::rasterize(terra::vect(tamiasPolyList[[i]]), fullGrid))
}
names(spGrids) <- names(tamiasPolyList)

createEPMgrid(spGrids)


#######
# With point occurrences
## demonstrating all possible input formats

# list of sf spatial objects
spOccList <- lapply(tamiasPolyList, function(x) st_sample(x, size = 10, type= 'random'))
tamiasEPM <- createEPMgrid(spOccList, resolution = 100000, cellType = 'hexagon')

# list of coordinate tables
spOccList2 <- lapply(spOccList, function(x) st_coordinates(x))
tamiasEPM <- createEPMgrid(spOccList2, resolution = 100000, cellType = 'square',
crs = st_crs(tamiasPolyList[[1]]))

# single table of coordinates
spOccList3 <- spOccList2
for (i in 1:length(spOccList3)) {
spOccList3[[i]] <- cbind.data.frame(taxon = names(spOccList3)[i], spOccList3[[i]])
colnames(spOccList3[[i]]) <- c('taxon', 'X', 'Y')
}
spOccList3 <- do.call(rbind, spOccList3)
rownames(spOccList3) <- NULL
spOccList3[, "taxon"] <- as.character(spOccList3[, "taxon"])
tamiasEPM <- createEPMgrid(spOccList3, resolution = 100000, cellType = 'square',
```

```
crs = st_crs(tamiasPolyList[[1]]))

# a single labeled spatial object
spOccList4 <- st_as_sf(spOccList3[, c("taxon", "X", "Y")], coords = c("X","Y"),
crs = st_crs(spOccList[[1]]))
tamiasEPM <- createEPMgrid(spOccList4, resolution = 100000, cellType = 'square')
```

---

customBetaDiv *Custom beta diversity metrics*

Description

Define your own function for summarizing information across a moving window of grid cells.

Usage

```
customBetaDiv(
  x,
  fun,
  radius,
  minTaxCount = 1,
  focalCoord = NULL,
  metricName = "custom_metric"
)
```

Arguments

`x`	object of class `epmGrid`
`fun`	a function to apply to grid cell neighborhoods (see details)
`radius`	Radius of the moving window in map units.
`minTaxCount`	the minimum number of taxa needed to apply the function. For instance, should the function be applied to gridcells with just 1 taxon?
`focalCoord`	vector of x and y coordinate, see details
`metricName`	the name you would like to attach to the output

Details

This function will identify the neighbors of every cell and will apply the specified function to those sets of cell neighborhoods.

The custom function should have just one input: a list of taxon names, where the list will represent a set of grid cells (focal cell + neighboring cells).

However, if a set of focal coordinates is provided, then rather than apply the function to each neighborhood of cells, the function should have two inputs: the focal cell and another cell, and that function will be applied to every pair defined by the focal cell and another cell. See examples.

Within the function call, the trait data already attached to the epmGrid object must be referred to as dat, and the phylogenetic tree already attached to the epmGrid must be referred to as phylo.

If the input epmGrid object contains a set of trees, then this function will be applied, using each tree in turn, and will return a list of results. This list can then be passed to `summarizeEpmGridList` to be summarized.

See examples below.

Value

object of class `epmGrid`, or list of `epmGrid` objects

Author(s)

Pascal Title

Examples

```
tamiasEPM <- addPhylo(tamiasEPM, tamiasTree)
tamiasEPM <- addTraits(tamiasEPM, tamiasTraits)

# An example using a multivariate dataset
## For each focal cell + neighbors, calculate the morphological standard deviation
## per grid cell and return the standard deviation.
f <- function(cellList) {
vec <- numeric(length(cellList))
for (i in 1:length(cellList)) {
vec[[i]] <- max(dist(dat[cellList[[i]], ]))
}
return(sd(vec, na.rm = TRUE))
}

xx <- customBetaDiv(tamiasEPM, fun = f, radius = 70000, minTaxCount = 2, metricName = 'maxdist')


# An example using only the phylogeny.
## Calculate standard deviation of phylogenetic diversity across cell neighborhood.
f <- function(cellList) {
vec <- numeric(length(cellList))
for (i in 1:length(cellList)) {
vec[[i]] <- faithPD(phylo, cellList[[i]])
}
return(sd(vec, na.rm = TRUE))
}

xx <- customBetaDiv(tamiasEPM, fun = f, radius = 70000, minTaxCount = 1, metricName = 'faithPD')



# an example that involves both morphological and phylogenetic data
## nonsensical, but for illustrative purposes:
```

```
## ratio of Faith's phylogenetic diversity to morphological range
## the standard deviation of this measure across grid cells
## in a neighborhood.
f <- function(cellList) {
vec <- numeric(length(cellList))
for (i in 1:length(cellList)) {
vec[[i]] <- faithPD(phylo, cellList[[i]]) /
max(dist(dat[cellList[[i]], ]))
}
return(sd(vec, na.rm = TRUE))
}

xx <- customBetaDiv(tamiasEPM, fun = f, radius = 70000, minTaxCount = 2,
  metricName = 'ratio_PD_maxdist')




# from a focal coordinate to all other sites
## Here, the function has 2 inputs.
## Example: calculate the per grid cell mean and take the distance.
f <- function(focalCell, otherCell) {
x1 <- colMeans(dat[focalCell, ])
x2 <- colMeans(dat[otherCell, ])
return(as.matrix(dist(rbind(x1, x2)))[1,2])
}

xx <- customBetaDiv(tamiasEPM, fun = f, radius = 70000, minTaxCount = 1,
 focalCoord = c(-1413764, 573610.8), metricName = 'meandist')



# Example involving a set of trees
tamiasEPM <- addPhylo(tamiasEPM, tamiasTreeSet, replace = TRUE)

## Calculate standard deviation of phylogenetic diversity across cell
## neighborhood.
f <- function(cellList) {
vec <- numeric(length(cellList))
for (i in 1:length(cellList)) {
vec[[i]] <- faithPD(phylo, cellList[[i]])
}
return(sd(vec, na.rm = TRUE))
}

# This time, a list of sf objects will be returned, one for each input tree.
xx <- customBetaDiv(tamiasEPM, fun = f, radius = 70000, minTaxCount = 1,
 metricName = 'faithPD')



# also works with square grid cells
tamiasEPM2 <- createEPMgrid(tamiasPolyList, resolution = 50000,
```

```
cellType = 'square', method = 'centroid')
tamiasEPM2 <- addPhylo(tamiasEPM2, tamiasTree)
tamiasEPM2 <- addTraits(tamiasEPM2, tamiasTraits)


f <- function(cellList) {
vec <- numeric(length(cellList))
for (i in 1:length(cellList)) {
vec[[i]] <- faithPD(phylo, cellList[[i]]) /
max(dist(dat[cellList[[i]], ]))
}
return(sd(vec, na.rm = TRUE))
}

xx <- customBetaDiv(tamiasEPM2, fun = f, radius = 70000, minTaxCount = 2,
 metricName = 'ratio_PD_maxdist')
```

customGridMetric *Custom grid metrics*

Description

Define your own function for summarizing information across grid cells.

Usage

```
customGridMetric(
  x,
  fun,
  column = NULL,
  minTaxCount = 1,
  metricName = "custom_metric"
)
```

Arguments

`x`	object of class `epmGrid`
`fun`	a function to apply to all grid cells (see details)
`column`	If a univariate morphological metric is specified, and the data in `x` are multivariate, which trait should be used? This can also specify which subset of columns a multivariate metric should be applied to.
`minTaxCount`	the minimum number of taxa needed to apply the function. For instance, should the function be applied to gridcells with just 1 taxon?
`metricName`	the name you would like to attach to the output

Details

This function allows you to not be limited to the diversity metrics available via the `gridMetrics` function.

The custom function should have just one input: a vector of taxon names that will then be used to subset the trait or phylogenetic data. Within the function call, the trait data already attached to the epmGrid object must be referred to as dat, and the phylogenetic tree already attached to the epmGrid must be referred to as phylo.

If the input epmGrid object contains a set of trees, then this function will be applied, using each tree in turn, and will return a list of results. This list can then be passed to `summarizeEpmGridList` to be summarized.

See examples below.

Value

object of class `epmGrid`, or list of `epmGrid` objects

Author(s)

Pascal Title

Examples

```
tamiasEPM
tamiasEPM <- addPhylo(tamiasEPM, tamiasTree)
tamiasEPM <- addTraits(tamiasEPM, tamiasTraits)

# In the following examples, notice that any mention of the trait data or
## phylogeny that are already attached to the epmGrid object are referred
## to as dat and phylo.

# example: calculate morphological disparity
## (already implemented in gridMetrics)
f <- function(cells) {
sum(diag(cov(dat[cells,])))
}

# to calculate disparity, we need at least 2 taxa

xx <- customGridMetric(tamiasEPM, fun = f, minTaxCount = 2,
metricName = 'disparity')

# In the example above, gridcells with 1 species are left as NA.
## But if we wanted those gridcells to have a value of 0 rather than NA,
## we could do the following:
f <- function(sp) {
if (length(sp) == 1) {
0
} else {
```

```
sum(diag(cov(dat[sp,])))
}
}

# and change minTaxCount to 1
xx <- customGridMetric(tamiasEPM, fun = f, minTaxCount = 1,
metricName = 'disparity')


# phylogenetic example: mean patristic distance
## this example doesn't actually involve the phylogeny internally,
## we can just supply what is needed to the function
patdist <- cophenetic(tamiasEPM[['phylo']])
patdist[upper.tri(patdist, diag = TRUE)] <- NA
f <- function(cells) {
mean(patdist[cells, cells], na.rm = TRUE)
}

xx <- customGridMetric(tamiasEPM, fun = f, minTaxCount = 1,
metricName = 'mean patristic')

# an example that involves both morphological and phylogenetic data
## nonsensical, but for illustrative purposes:
## ratio of Faith's phylogenetic diversity to morphological range
f <- function(cells) {
faithPD(phylo, cells) / max(dist(dat[cells, ]))
}

xx <- customGridMetric(tamiasEPM, fun = f, minTaxCount = 2,
metricName = 'PD_range_ratio')


# Example involving a set of trees
tamiasEPM <- addPhylo(tamiasEPM, tamiasTreeSet, replace = TRUE)

# get crown clade age of clade containing taxa present in grid cell
f <- function(sp) {
ape::branching.times(phylo)[as.character(ape::getMRCA(phylo, sp))]
}

xx <- customGridMetric(tamiasEPM, fun = f, minTaxCount = 2, metric = 'nodeAge')
```

`dropSpecies` *Drop species from epmGrid*

Description

Removes particular species from a epmGrid object.

Usage

```
dropSpecies(x, sp)
```

Arguments

x	object of class epmGrid
sp	a character vector of species names to be dropped.

Details

If species in sp are not in x, they will be ignored.

Value

new epmGrid object.

Author(s)

Pascal Title

Examples

```
tamiasEPM

new <- dropSpecies(tamiasEPM, sp = c('Tamias_alpinus', 'Tamias_bulleri'))

setdiff(tamiasEPM[['geogSpecies']], new[['geogSpecies']])
```

DRstat *Calculate the DR statistic*

Description

Calculates the tip-specific DR statistic for speciation rates

Usage

```
DRstat(tree)
```

Arguments

tree	phylogeny of class phylo

Value

named numeric vector of speciation rates

Author(s)

Pascal Title

References

Jetz, W., Thomas, G. H., Joy, J. B., Hartmann, K., & Mooers, A. O. (2012). The global diversity of birds in space and time. Nature, 491, 444–448.

Examples

```
tamiasTree
DRstat(tamiasTree)
```

---

epm *EcoPhyloMapper (epm)*

Description

An R package that facilitates the aggregation of species' geographic ranges from vector or raster spatial data, and that enables the calculation of various morphological and phylogenetic metacommunity metrics across geography.

A detailed wiki for the R package can be found on the epm github page: https://github.com/ptitle/epm/wiki#table-of-contents

To cite the epm package in publications, please use:
Pascal O. Title, Donald L. Swiderski and Miriam L. Zelditch. 2022. EcoPhyloMapper: an R package for integrating geographic ranges, phylogeny, and morphology. Methods in Ecology and Evolution. doi:10.1111/2041210X.13914

Details

Creating and enhancing an epmGrid object
Use `createEPMgrid` to create an epmGrid object from species spatial data.
Optionally, you can draw the spatial extent that you would like to use with `interactiveExtent`.
Add in species attributes with `addTraits`, and/or a phylogeny with `addPhylo`.
Use the function `reduceToCommonTaxa` to reduce the epmGrid object to species that are present for all data types.

Calculating diversity metrics
Calculate various diversity metrics with `gridMetrics`, or define your own, using `customGridMetric`.
Calculate moving window turnover metrics with `betadiv_taxonomic`, `betadiv_phylogenetic`, `betadiv_disparity`. You can also define your own beta diversity metric with `customBetaDiv`.
If you have a posterior set of trees, summarize phylogenetic uncertainty with `summarizeEpmGridList`.

Plotting epmGrid objects
Plot epmGrid object with `plot.epmGrid`.

You get finer control over the legend with `addLegend`.
The function `getMultiMapRamp` will be helpful if you are trying to plot multiple epmGrid objects on the same color scale.
Use `plotDispersionField` to plot the assemblage dispersion field for a given site.

Getting derived data from epmGrid objects
Use `calcMeanShape` to get mean morphological shape per grid cell.
Use `coordsFromEpmGrid` to get the spatial coordinates of specific grid cells.
Use `extractFromEpmGrid` to get the species that are found at certain coordinates or within a defined polygon.
Use `generateOccurrenceMatrix` to produce a species-by-site presence/absence matrix.
Use `tableFromEpmGrid` to pull data from epmGrids and rasters from a set of random points for statistical analysis.

Writing to disk
You can save an epmGrid with `write.epmGrid`, and read it back in with `read.epmGrid`.
You can also write an epmGrid object to a spatial file format for use in GIS software with `writeEpmSpatial`.

Author(s)

Pascal O. Title, Donald L. Swiderski, Miriam L. Zelditch

See Also

Useful links:

- `https://github.com/ptitle/epm`
- Report bugs at `https://github.com/ptitle/epm/issues`

`epm-example` *Eco Phylo Mapper datasets*

Description

Included datasets in `epm`

Usage

```
tamiasEPM

tamiasPolyList

tamiasTraits

tamiasTree

tamiasTreeSet
```

Details

Included north american chipmunk dataset:
tamiasTree is a phylogeny for chipmunks from Zelditch et al. 2017
tamiasTreeSet is a distribution of 10 phylogenies for chipmunks, extracted from the mammal tree from Upham et al. 2019 tamiasTraits is a geometric morphometrics dataset of mean values for chipmunks from Zelditch et al. 2017
tamiasPolyList is a set of geographic ranges for chimpunks from IUCN 2021.
tamiasEPM is an `epmGrid` object created with `createEPMgrid` using these datasets.

References

Zelditch, M. L., Ye, J., Mitchell, J. S., & Swiderski, D. L. (2017). Rare ecomorphological convergence on a complex adaptive landscape: Body size and diet mediate evolution of jaw shape in squirrels (Sciuridae). Evolution, 1–17. https://doi.org/10.1111/evo.13168

Upham, N. S., Esselstyn, J. A., & Jetz, W. (2019). Inferring the mammal tree: Species-level sets of phylogenies for questions in ecology, evolution, and conservation. PLoS Biology, 17(12), e3000494. https://doi.org/10.1371/journal.pbio.3000494

IUCN 2021. The IUCN Red List of Threatened Species. 2021-3. https://www.iucnredlist.org. Downloaded on 17 March 2021.

`epmToPhyloComm` *Convert epmGrid to community matrix*

Description

Given specific sites, convert epmGrid to phylocomm matrix, with sites as rows, and species as columns

Usage

```
epmToPhyloComm(x, sites)
```

Arguments

`x`	object of class `epmGrid`
`sites`	locations of sites, see details.

Details

If sites are site coordinates, then dataframe or matrix with two columns; if sites are cell indices, then numeric vector; if `sites = 'all'`, then all cells will be returned as sites.

Value

community matrix, with sites as rows and species as columns

Author(s)

Pascal Title

Examples

```
tamiasEPM

# from cell indices
cells <- c(2703, 90, 3112, 179)
epmToPhyloComm(tamiasEPM, cells)

# from coordinates
library(sf)
# get the projection of the epmGrid object
proj <- summary(tamiasEPM)$crs
# define some points
pts <- rbind.data.frame(
c(-120.5, 38.82),
c(-84.02, 42.75),
c(-117.95, 55.53))
colnames(pts) <- c('x', 'y')
ptsSF <- st_as_sf(pts, coords = 1:2, crs = "epsg:4326")
pts <- st_coordinates(st_transform(ptsSF, crs = proj))

epmToPhyloComm(tamiasEPM, pts)
```

`expandSpeciesCellList` *Expand species list*

Description

The epmGrid object contains an accounting of species per cell in a condensed format. This function returns a complete list of species per cell.

Usage

```
expandSpeciesCellList(x)
```

Arguments

`x`	object of class `epmGrid`

Details

Function to expand condensed species list to full set of cells

Value

list of species for each cell.

Author(s)

Pascal Title

Examples

```
tamiasEPM
head(expandSpeciesCellList(tamiasEPM))
```

---

extractFromEpmGrid *Extract from epmGrid*

Description

Return species from intersection between spatial points or polygons and a epmGrid object.

Usage

```
extractFromEpmGrid(x, spatial, returnCells = FALSE, collapse = TRUE)
```

Arguments

`x`	object of class `epmGrid`
`spatial`	coordinates as either a spatial points object (sp or sf), a matrix/dataframe with two columns, a numeric vector of c(long, lat), or as a spatial polygon object (sp or sf).
`returnCells`	boolean, if `TRUE`, cell indices are returned rather than taxa
`collapse`	boolean; if `TRUE`, then a vector of unique species is returned, pooled from all cells, if `FALSE`, then list is returned with species from every cell as intersected by `spatial`.

Details

If `spatial` is a spatial object, it will be transformed to the same projection as `x` if needed. If `spatial` is not a spatial object, it is assumed to be in the same projection as `x`.

Value

A vector of species if `collapse = TRUE`, or a list of species by cell if `collapse = FALSE`. If `returnCells = TRUE`, a vector of cell indices that correspond to the rows in the epmGrid sf object.

Author(s)

Pascal Title

Examples

```
library(sf)
# get the projection of the epmGrid object
proj <- summary(tamiasEPM)$crs
# define some points
pts <- rbind.data.frame(
c(-120.5, 38.82),
c(-84.02, 42.75),
c(-117.95, 55.53))
colnames(pts) <- c('x', 'y')
ptsSF <- st_as_sf(pts, coords = 1:2, crs = "epsg:4326")
pts <- st_coordinates(st_transform(ptsSF, crs = proj))

# extract with table of coordinates
extractFromEpmGrid(tamiasEPM, pts)

# extract with spatial points object
extractFromEpmGrid(tamiasEPM, ptsSF)

# extract with spatial polygon
hull <- st_convex_hull(st_union(ptsSF))
extractFromEpmGrid(tamiasEPM, hull)


# returns each cell's contents
extractFromEpmGrid(tamiasEPM, hull, collapse=FALSE)

# collapses results to unique set of species
extractFromEpmGrid(tamiasEPM, hull, collapse=TRUE)
```

`faithPD` *Calculate Faith's Phylogenetic Diversity*

Description

Calculates Faith's PD for a specific set of tips

Usage

```
faithPD(phy, tips)
```

Arguments

`phy`	phylogeny of class `phylo`
`tips`	tip names to be included

Details

Returns the sum of total branch lengths that unite a set of species. The root is always included in these calculations. If tip is just one species, then the root-to-tip distance is returned.

Value

numeric value of summed phylogenetic diversity

Author(s)

Pascal Title

References

Faith D.P. (1992) Conservation evaluation and phylogenetic diversity. Biological Conservation, 61, 1-10.

Examples

```
tamiasTree
faithPD(tamiasTree, c('Tamias_minimus', 'Tamias_speciosus'))
```

---

generateOccurrenceMatrix

Convert epmGrid to community matrix

Description

Given specific sites (or all sites), convert epmGrid to a species occurrence matrix, with sites as rows, and species as columns.

Usage

```
generateOccurrenceMatrix(x, sites)
```

Arguments

x	object of class epmGrid
sites	locations of sites, see details.

Details

If sites are site coordinates, then this should be a dataframe or matrix with two columns; if sites are cell indices, then a numeric vector; if `sites = 'all'`, then all cells will be returned as sites.

To get the associated site coordinates, see `coordsFromEpmGrid`.

Value

a presence/absence matrix, with sites as rows and species as columns.

Author(s)

Pascal Title

Examples

```
tamiasEPM

# from cell indices
cells <- c(2703, 90, 3112, 179)
generateOccurrenceMatrix(tamiasEPM, cells)

# get the associated site coordinates
coordsFromEpmGrid(tamiasEPM, cells)

# from coordinates
library(sf)
# get the projection of the epmGrid object
proj <- summary(tamiasEPM)$crs
# define some points
pts <- rbind.data.frame(
c(-120.5, 38.82),
c(-84.02, 42.75),
c(-117.95, 55.53))
colnames(pts) <- c('x', 'y')
ptsSF <- st_as_sf(pts, coords = 1:2, crs = "epsg:4326")
pts <- st_coordinates(st_transform(ptsSF, crs = proj))

generateOccurrenceMatrix(tamiasEPM, pts)
```

`getExtentOfList` *Get extent of list*

Description

Given a list of SpatialPolygons, return an extent object that encompasses all items.

Usage

```
getExtentOfList(shapes)
```

Arguments

shapes	a list of SpatialPolygons or simple features

Value

An object of class bbox.

Author(s)

Pascal Title

Examples

```
getExtentOfList(tamiasPolyList)
```

---

getMultiMapRamp *Extract min and max for multiple epmGrids*

Description

Extracts the range of values across a list of input objects for use in plotting

Usage

```
getMultiMapRamp(...)
```

Arguments

...	objects of class epmGrid, RasterLayer SpatRaster or sf objects.

Details

If the user would like to plot multiple epmGrid objects with a standardized color ramp, then the returned values from this function can be supplied to plot.epmGrid. Also works with RasterLayer and sf objects. For sf object, only one attribute can be specified.

Value

a numeric vector of length 2: overall min and max value.

Author(s)

Pascal Title

Examples

```
library(terra)
tamiasEPM

# create a dummy raster for demonstration purposes.
ras <- rast()
values(ras) <- runif(ncell(ras), min = 0, max = 40)

getMultiMapRamp(tamiasEPM, ras)
```

`getSpPartialDisparities` *Partial Disparity*

Description

Calculate species-specific partial disparity, relative to some group mean.

Usage

```
getSpPartialDisparities(dat, groupMean = NULL)
```

Arguments

`dat`	matrix of multivariate morphological data
`groupMean`	if `NULL`, calculated from `dat`, otherwise can be provided as a vector of mean values

Details

Calculates partial disparities, as in Foote 1993. By default, the group mean is calculated from the full input data.

Value

numeric vector

Author(s)

Pascal Title

Examples

```
tamiasTraits[1:5, 1:5]
getSpPartialDisparities(tamiasTraits)
```

gridMetrics *Grid Metrics*

Description

Calculate various morphological and phylogenetic community metrics for every cell in a `epmGrid` object. To implement other metrics not available here, see `customGridMetric`.

Usage

```
gridMetrics(
  x,
  metric,
  column = NULL,
  verbose = TRUE,
  dataType = c("auto", "univariate", "multivariate", "pairwise")
)
```

Arguments

`x`	object of class `epmGrid`
`metric`	name of metric to use, see Details.
`column`	If a univariate morphological metric is specified, and the data in `x` are multivariate, which trait should be used? This can also specify which subset of columns a multivariate metric should be applied to.
`verbose`	Print various messages to the console. Default is TRUE.
`dataType`	Specify the type of input data that the metric will be calculated from. Defaults to `'auto'` in which case this is determined based on the data structure.

Details

Univariate trait metrics

- mean
- median
- range
- variance
- mean_NN_dist: mean nearest neighbor distance
- min_NN_dist: minimum nearest neighbor distance
- evenness: variance of nearest neighbor distances, larger values imply decreasing evenness
- arithmeticWeightedMean (see below)
- geometricWeightedMean (see below)

Multivariate trait metrics

- mean: mean of pairwise distance matrix derived from multivariate data
- median: median of pairwise distance matrix derived from multivariate data
- disparity
- partialDisparity: contribution of species in each gridcell to overall disparity, returned as the ratio of summed partial disparities to total disparity.
- range
- mean_NN_dist: mean nearest neighbor distance
- min_NN_dist: minimum nearest neighbor distance
- evenness: variance of nearest neighbor distances, larger values imply decreasing evenness.

Phylogenetic metrics

- pd: Faith's phylogenetic diversity, including the root
- meanPatristic
- meanPatristicNN: mean nearest neighbor in patristic distance
- minPatristicNN: minimum nearest neighbor in patristic distance
- phyloEvenness: variance of nearest neighbor patristic distances, larger values imply decreasing evenness
- phyloDisparity: sum of squared deviations in patristic distance
- PSV: Phylogenetic Species Variability
- PSR: Phylogenetic Species Richness
- DR: non-parametric estimate of speciation rates

Range-weighted metrics

- weightedEndemism: Species richness inversely weighted by range size.
- correctedWeightedEndemism: Weighted endemism standardized by species richness
- phyloWeightedEndemism: Phylogenetic diversity inversely weighted by range size associated with each phylogenetic branch.

If data slot contains a pairwise matrix, `column` is ignored. Weighted mean options are available where, for each cell, a weighting scheme (inverse of species range sizes) is applied such that small-ranged species are up-weighted, and broadly distributed species are down-weighted. This can be a useful way to lessen the influence of broadly distributed species in the geographic mapping of trait data.

It may be desirable to have metrics calculated for a dataset where only taxa shared across geography, traits and phylogeny are included. The function `reduceToCommonTaxa` does exactly that.

If a set of trees are associated with the input epmGrid object `x`, then the metric is calculated for each tree, and a list of epmGrid objects is returned. This resulting list can be summarized with the function `summarizeEpmGridList`. For instance the mean and variance can be calculated, to show the central tendency of the metric across grid cells, and to quantify where across geography variability in phylogenetic topography manifests itself.

To implement other metrics not available here, see `customGridMetric`.

Value

object of class `epmGrid` where the grid represents calculations of the metric at every cell. The species identities per grid cell are those that had data for the calculation of the metric. If taxa were dropped from the initial epmGrid object, then they have been removed from this epmGrid. If a set of trees was involved, then returns a list of `epmGrid` objects.

References

partial disparity
Foote, M. (1993). Contributions of individual taxa to overall morphological disparity. Paleobiology, 19(4), 403–419. https://doi.org/10.1017/s0094837300014056

PSV, RSV
Helmus, M. R., Bland, T. J., Williams, C. K., & Ives, A. R. (2007). Phylogenetic Measures of Biodiversity. The American Naturalist, 169(3), E68–E83. https://doi.org/10.1086/511334

DR
Jetz, W., Thomas, G. H., Joy, J. B., Hartmann, K., & Mooers, A. O. (2012). The global diversity of birds in space and time. Nature, 491(7424), 444–448. https://doi.org/10.1038/nature11631

weighted endemism
Crisp, M. D., Laffan, S., Linder, H. P., & Monro, A. (2001). Endemism in the Australian flora. Journal of Biogeography, 28(2), 183–198. https://doi.org/10.1046/j.1365-2699.2001.00524.x

phylo weighted endemism
Rosauer, D., Laffan, S. W., Crisp, M. D., Donnellan, S. C., & Cook, L. G. (2009). Phylogenetic endemism: a new approach for identifying geographical concentrations of evolutionary history. Molecular Ecology, 18(19), 4061–4072. https://doi.org/10.1111/j.1365-294x.2009.04311.x

Examples

```
tamiasEPM <- addPhylo(tamiasEPM, tamiasTree)
tamiasEPM <- addTraits(tamiasEPM, tamiasTraits)

# univariate morphological example
x <- gridMetrics(tamiasEPM, metric='mean', column='V2')
plot(x, use_tmap = FALSE)

# multivariate morphological
x <- gridMetrics(tamiasEPM, metric='disparity')
plot(x, use_tmap = FALSE)

# phylogenetic metrics
x <- gridMetrics(tamiasEPM, metric='meanPatristic')
plot(x, use_tmap = FALSE)
```

identify.epmGrid *Interactively identify species in epmGrid*

Description

Plots a epmGrid object and allows you to click on the plot to return the species found in the cell you clicked on.

Usage

```
## S3 method for class 'epmGrid'
identify(x, returnCell = FALSE, ...)
```

Arguments

`x`	object of class `epmGrid` or `sf`
`returnCell`	boolean; if FALSE, then species names are returned, if TRUE, then cell indices are returned.
`...`	additional arguments passed to sf::plot

Details

This is a wrapper function for the `identify` function in base graphics. This is primarily intended as a useful function for data exploration and spot-checking.

Value

A vector of species names or cell indices.

Author(s)

Pascal Title

interactiveExtent *Interactively choose extent*

Description

Given a list of polygons or point occurrences, sets up an interactive plot to allow the user to draw the desired extent. This can be used to define the extent in `createEPMgrid`.

Usage

```
interactiveExtent(polyList, cellType = "square", bb = NULL)
```

Arguments

polyList	a list of Simple Feature polygons or points.
cellType	either hexagon or square.
bb	c(xmin, xmax, ymin, ymax) to limit the extent for the interactive plot.

Details

This function returns both a sf polygon and the same polygon as a WKT string. Either can be supplied to createEPMgrid as the extent. A recommended strategy is to use this function to find your extent, and to copy/paste the WKT string into your R script so that you can retain it for future use, and maintain reproducibility. See example.

What is chosen for cellType has no effect on what you might choose in createEPMgrid. Square cells will probably be fastest. If hexagons are selected, grid cell points are plotted instead of polygons to speed up plotting.

You may see the message Failed to compute min/max, no valid pixels found in sampling. (GDAL error 1) . This just means that a species did not register in any grid cells. This can be ignored.

The basemap is from https://www.naturalearthdata.com/.

Value

A list with a polygon, and its WKT string

Author(s)

Pascal Title

Examples

```
if (interactive()) {
ex <- interactiveExtent(tamiasPolyList)

# You can use this as the extent in createEPMgrid
grid <- createEPMgrid(tamiasPolyList, resolution = 50000, extent = ex$wkt)

# One way to make your code reproducible would be to copy/paste the wkt
# in your code for future use:
ex <- interactiveExtent(tamiasPolyList)
ex$wkt
customExtent <- "POLYGON ((-2238201 3532133, -2675450 1722657, -2470677 -317634,
-1863632 -1854074, -521614.8 -2170280, -349356.8 799040.9, -2238201 3532133))"

grid <- createEPMgrid(tamiasPolyList, resolution = 50000, extent = customExtent)

}
```

plot.epmGrid *Plot epmGrid*

Description

Plot a epmGrid object. This function uses the tmap package for plotting by default.

Usage

```
## S3 method for class 'epmGrid'
plot(
  x,
  log = FALSE,
  legend = TRUE,
  col,
  basemap = "worldmap",
  colorRampRange = NULL,
  minTaxCount = "auto",
  zoom = TRUE,
  ignoredColor = gray(0.9),
  lwd,
  borderCol = "black",
  alpha = 1,
  includeFrame = FALSE,
  use_tmap = TRUE,
  fastPoints = FALSE,
  title = "",
  add = FALSE,
  ...
)
```

Arguments

`x`	object of class `epmGrid`
`log`	boolean; should the cell values be logged?
`legend`	boolean; should legend be included?
`col`	either a vector of color names that will be interpolated, or a color ramp function that takes an integer (see for example `colorRampPalette`).
`basemap`	if `'none'`, then only the grid is plotted. If `'worldmap'`, then vector map is plotted. If `'interactive'`, then the plot is sent to the web browser.
`colorRampRange`	numeric vector of min and max value for scaling the color ramp. Automatically inferred if set to `NULL`. This is relevant if multiple plots are desired on the same scale. See `getMultiMapRamp`.

minTaxCount	an integer, or 'auto'. Should cells containing certain numbers of taxa be grayed out? For example, should single-taxon cells be ignored because the metric only makes sense for multi-taxon cells? This is predetermined for all metrics in gridMetrics if minTaxCount = 'auto'.
zoom	Should plot zoom in on cells with data. Default is TRUE.
ignoredColor	color for ignored cells. See details.
lwd	grid cell border width
borderCol	color for grid cell borders
alpha	opacity of all colors and borders, ranging from 0 (fully transparent) to 1 (fully opaque)
includeFrame	boolean; include frame around plot?
use_tmap	boolean; if FALSE, plotting will be done via sf instead of tmap package
fastPoints	Intended for debugging purposes. For hex grids and use_tmap = F, plot points instead of polygons. Helpful for sorting out plotting details without waiting for slow polygon plotting.
title	text to add to the plot
add	logical, add to existing plot?
...	additional arguments that can be passed to sf::plot or terra::plot if use_tmap = FALSE

Details

If x is a metric as generated with gridMetrics that returns 0 for single-species cells, then those cells (that have a value of 0) will be plotted in gray (or any color as specified with ignoredColor) if minTaxCount = 'auto'. You can specify other values as well. For instance, if you use the function customGridMetric to calculate phylogenetic signal, which is a metric that only makes sense for cells with 3 or more taxa, then you could then specify minTaxCount = 3. Setting minTaxCount = 1 shows all cells with data.

If the tmap package is not installed, then this function will default to plotting with sf::plot.

If you would like more control over the legend, then plot with tmap = FALSE and legend = FALSE, and then call the function addLegend.

Value

Nothing is returned if plotting with tmap (the default). If plotting with use_tmap = FALSE, and if the plot is directed to a variable, then this variable will contain relevant information to be passed on to the function addLegend:

Author(s)

Pascal Title

Examples

```
plot(tamiasEPM, use_tmap = FALSE)

plot(tamiasEPM, legend = FALSE, use_tmap = FALSE, col = viridisLite::inferno)
addLegend(tamiasEPM, location = 'top', ramp = viridisLite::inferno)


# Example for how to plot multiple epmGrids on the same color scale
# for illustration purposes, we will compare weighted endemism to
# phylogenetic weighted endemism
library(tmap)

tamiasEPM <- addPhylo(tamiasEPM, tamiasTree)
epm1 <- gridMetrics(tamiasEPM, metric='weightedEndemism')
epm2 <- gridMetrics(tamiasEPM, metric='phyloWeightedEndemism')
# get global min and max values
minmax <- getMultiMapRamp(epm1, epm2)

map1 <- plot(epm1, colorRampRange = log(minmax), log = TRUE, legend = FALSE)
map2 <- plot(epm2, colorRampRange = log(minmax), log = TRUE, legend = FALSE)
# tmap_arrange(map1, map2)

# view your plot in the web-browser as a dynamic plot.
plot(tamiasEPM, basemap = 'interactive')


# Adding a custom legend, and passing along arguments via params
xx <- plot(tamiasEPM, use_tmap = FALSE, legend = FALSE,
col = viridisLite::magma)
addLegend(tamiasEPM, params = xx, location = 'bottom')
```

`plotDispersionField` *Plot dispersion fields*

Description

For a set of specified coordinates, plot a richness map for the species that are found at those coordinates.

Usage

```
plotDispersionField(
  x,
  coords,
  plotCoords = TRUE,
  legend = TRUE,
  col,
  lwd = 0.5,
```

```
  basemap = "worldmap",
  borderCol = "black",
  alpha = 1,
  includeFrame = FALSE,
  use_tmap = TRUE,
  add = FALSE
)
```

Arguments

x	object of class epmGrid
coords	coordinates as either a spatial points object (sp or sf), a matrix/dataframe with two columns or a numeric vector of c(long, lat).
plotCoords	boolean; should the coordinates be plotted as well?
legend	boolean; should legend be included?
col	either a vector of color names that will be interpolated, or a color ramp function that takes an integer (see for example colorRampPalette).
lwd	grid cell border width
basemap	if 'none', then only the grid is plotted. If 'worldmap', then vector map is plotted. If 'interactive', then the mapview package is used.
borderCol	color for grid cell borders
alpha	opacity of all colors and borders, ranging from 0 (fully transparent) to 1 (fully opaque)
includeFrame	boolean; include frame around plot?
use_tmap	boolean; if FALSE, plotting will be done via sf instead of tmap package
add	logical, add to existing plot?

Details

Assemblage dispersion fields represent an overlapping of geographic ranges for the taxa that occur in the focal grid cells.

Value

Nothing is returned.

Author(s)

Pascal Title

References

Graves, G. R., & Rahbek, C. (2005). Source pool geometry and the assembly of continental avifaunas. Proceedings of the National Academy of Sciences, 102(22), 7871–7876.

Examples

```
# plotDispersionField(tamiasEPM, c(-1944951, 69588.74))
plotDispersionField(tamiasEPM, c(-1944951, 69588.74), use_tmap = FALSE)
```

`plotSpRange` *plot a single species' range*

Description

Plot one species' geographic range, as encoded in the epmGrid object.

Usage

```
plotSpRange(
  x,
  taxon,
  taxonColor = "orange",
  basemap = "worldmap",
  lwd = 0.5,
  alpha = 1,
  use_tmap = TRUE,
  add = FALSE
)
```

Arguments

`x`	object of class `epmGrid`
`taxon`	taxon to plot
`taxonColor`	color for plotting taxon's range
`basemap`	if `'none'`, then only the grid is plotted. If `'worldmap'`, then vector map is plotted. If `'interactive'`, then the plotting is done via your web browser.
`lwd`	grid cell border width
`alpha`	opacity of all colors and borders, ranging from 0 (fully transparent) to 1 (fully opaque)
`use_tmap`	if false, use sf or terra packages for plotting
`add`	logical. If TRUE, adds the gridded taxon range to existing plot.

Value

nothing is returned

Author(s)

Pascal Title

Examples

```
tamiasEPM

plotSpRange(tamiasEPM, 'Tamias_speciosus', use_tmap = FALSE)
```

`rasterToGrid` *Convert raster to sf grid*

Description

Convert a raster to sf polygons object, matching the attributes of the target object.

Usage

```
rasterToGrid(x, target, fun = "mean", crop = TRUE, na.rm = TRUE)
```

Arguments

`x`	rasterLayer or rasterStack or SpatRaster
`target`	epmGrid or sf object
`fun`	function for summarizing raster cells to polygons
`crop`	if TRUE, the raster will be cropped to the bounding box of the target
`na.rm`	determines how `NA` cells are summarized

Details

By default, raster cells that overlap with target grid cell polygons will be averaged. If target is a raster grid, then `terra::resample` is used.

Value

sf polygons object, or a list of such objects if input has multiple layers.

Author(s)

Pascal Title

Examples

```
library(terra)

# We have a terra grid object (for example, climate data read in as a raster)
# Here, we are just generating some random data for demo
env <- rast(vect(tamiasEPM[[1]]), resolution = 100000)
env[] <- sample(1:100, ncell(env), replace = TRUE)
plot(env)
```

```
# Now, if we are interested in doing analyses of environmental data in relation to
# the epmGrid data we have, we want to convert the env data to the same grid structure
# where the cells align and where raster grid values are resampled and averaged.

newgrid <- rasterToGrid(env, target = tamiasEPM, fun = 'mean')
plot(newgrid)


# again but this time the input has multiple layers
env <- rast(vect(tamiasEPM[[1]]), resolution = 100000, nlyr = 3)
values(env[[1]]) <- sample(1:100, ncell(env), replace = TRUE)
values(env[[2]]) <- sample(1:200, ncell(env), replace = TRUE)
values(env[[3]]) <- sample(1:300, ncell(env), replace = TRUE)

newgrid <- rasterToGrid(env, target = tamiasEPM, fun = 'mean')
```

read.epmGrid *Read a epmGrid object*

Description

Load a saved epmGrid object.

Usage

```
read.epmGrid(filename)
```

Arguments

filename	filename, with extension rds

Details

This function will read in epmGrid objects that were saved with write.epmGrid.

Value

object of class epmGrid

Author(s)

Pascal Title

Examples

```
#save
write.epmGrid(tamiasEPM, paste0(tempdir(), '/tamiasEPM'))

# read back in
tamiasEPM <- read.epmGrid(paste0(tempdir(), '/tamiasEPM.rds'))

# delete the file
unlink(paste0(tempdir(), '/tamiasEPM.rds'))
```

---

reduceToCommonTaxa *Subset epmGrid to shared taxa*

Description

An epmGrid object may contain more taxa with morphological data than taxa with phylogenetic information, or vice versa. This function subsets all epmGrid components to the set of taxa shared across geographic, phenotypic and phylogenetic datasets. This might desirable to ensure that all diversity metrics are based on the same set of taxa.

Usage

```
reduceToCommonTaxa(x)
```

Arguments

x	object of class epmGrid

Value

new epmGrid object.

Author(s)

Pascal Title

Examples

```
tamiasEPM
# randomly drop a few species for demonstration
tamiasEPM <- addPhylo(tamiasEPM, ape::drop.tip(tamiasTree, sample(tamiasTree$tip.label, 5)))
tamiasEPM <- addTraits(tamiasEPM, tamiasTraits[-(3:5),])

new <- reduceToCommonTaxa(tamiasEPM)

tamiasEPM
new
```

`singleSpCellIndex` *Identify single-species cells*

Description

Given a epmGrid object, return the grid cell indices of those cells that have just one species.

Usage

```
singleSpCellIndex(x)
```

Arguments

`x`	object of class `epmGrid`

Details

This function can be useful when further analyzing epmGrid objects generated by `gridMetrics`, as it might make sense to exclude these single-species cells in further analyses.

Value

numeric vector of grid cell indices.

Author(s)

Pascal Title

Examples

```
singleSpCellIndex(tamiasEPM)
```

`spCountIndex` *Identify cells that have a certain number of taxa*

Description

Given a epmGrid object, return the grid cell indices of those cells that have the specified number of taxa.

Usage

```
spCountIndex(x, count)
```

Arguments

`x`	object of class `epmGrid`
`count`	number of species to consider (can be a vector of integers)

Details

This function can be useful when further analyzing epmGrid objects generated by `gridMetrics`, as it might make sense to exclude certain grid cells in further analyses.

Value

numeric vector of grid cell indices.

Author(s)

Pascal Title

Examples

```
spCountIndex(tamiasEPM, count = 1)
spCountIndex(tamiasEPM, count = 1:3)
```

`summarizeEpmGridList` *Summarize lists of epmGrid objects*

Description

If a diversity metric was calculated for an epmGrid object that contained a phylogenetic distribution, then a list of resulting epmGrid objects was returned. This function will take that list, and apply a summary statistic, returning a single epmGrid object. If the input list is from `betadiv_phylogenetic`, then that list of sf or SpatRaster objects can also be summarized with this function.

Usage

```
summarizeEpmGridList(x, fun = mean)
```

Arguments

`x`	a list of objects of class `epmGrid` or `sf` or `SpatRaster`.
`fun`	a function to apply to grid cells across the list x.

Details

It is assumed that across the objects in list x, the only difference is the values for the grid cells.

Value

a single object of class `epmGrid` or `sf` or `SpatRaster`.

Author(s)

Pascal Title

Examples

```
tamiasEPM <- addPhylo(tamiasEPM, tamiasTreeSet)
tamiasEPM <- addTraits(tamiasEPM, tamiasTraits)

x <- gridMetrics(tamiasEPM, metric='meanPatristicNN')
z <- summarizeEpmGridList(x, fun = var)

# using a custom function
f <- function(y) sum(y) / length(y)

z <- summarizeEpmGridList(x, fun = f)

# works with square grid epmGrids too
tamiasEPM2 <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'square', method = 'centroid')
tamiasEPM2 <- addPhylo(tamiasEPM2, tamiasTreeSet)
tamiasEPM2 <- addTraits(tamiasEPM2, tamiasTraits)

x <- gridMetrics(tamiasEPM2, metric='meanPatristicNN')

z <- summarizeEpmGridList(x, fun = median)


# With phylogenetic distribution

tamiasEPM <- addPhylo(tamiasEPM, tamiasTreeSet, replace = TRUE)
beta_phylo_turnover <- betadiv_phylogenetic(tamiasEPM, radius = 70000,
component = 'turnover')

z <- summarizeEpmGridList(beta_phylo_turnover)

tamiasEPM2 <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'square', method = 'centroid')
tamiasEPM2 <- addPhylo(tamiasEPM2, tamiasTreeSet)

beta_phylo_turnover <- betadiv_phylogenetic(tamiasEPM2, radius = 70000,
component = 'turnover')

z <- summarizeEpmGridList(beta_phylo_turnover, fun = median)
```

`summary.epmGrid` *epmGrid summary*

Description

Generates a summary of a epmGrid object.

Usage

```
## S3 method for class 'epmGrid'
summary(object, ...)
```

Arguments

`object`	object of class `epmGrid`
`...`	further arguments passed to `summary`

Details

Summary information includes

Value

A list containing the summary information is returned invisibly.

Author(s)

Pascal Title

Examples

```
summary(tamiasEPM)
attr <- summary(tamiasEPM)
attr
```

`tableFromEpmGrid` *Data table from epmGrid*

Description

Given one or several epmGrid objects, sf objects, rasterLayers, SpatRasters, create a table of values and associated coordinate data.

Usage

```
tableFromEpmGrid(..., n = NULL, minTaxCount = 1, coords = NULL, id = FALSE)
```

Arguments

`...`	objects of class `epmGrid`, `sf`, `sp`, `SpatRaster`, `RasterLayer` or `RasterStack`. All should have the same projection.
`n`	number of cells to randomly subsample, no subsampling if `NULL`
`minTaxCount`	integer; cells with at least this many taxa will be included.
`coords`	if NULL, then points are sampled as needed, otherwise, data will be extracted at these specified coordinates.
`id`	boolean, should the grid cell index (of the first item in the inputs) be returned as well?

Details

A set of cells are identified in the input objects. If `n=NULL`, then all cells are used, otherwise cells are randomly subsampled. Values at those cells are then returned. This table construction can be particularly useful for subsequent statistical analyses.

Only cells with data in all inputs are returned. If n is greater than the number of cells with data, then fewer than n cells will be returned.

The first element provided should be a `epmGrid` object, and that will be the one used as a template for the sampled grid system.

If `coords` is provided, then data are extracted at those coordinates, and no subsetting of those points is done.

Value

data.frame with input variables, as well as `"x"` and `"y"`.

Author(s)

Pascal Title

Examples

```
tamiasEPM
tamiasEPM <- addPhylo(tamiasEPM, tamiasTree)
tamiasEPM <- addTraits(tamiasEPM, tamiasTraits)
morphoDisp <- gridMetrics(tamiasEPM, metric='disparity')
meanPat <- gridMetrics(tamiasEPM, metric='meanPatristic')

tableFromEpmGrid(tamiasEPM, morphoDisp, meanPat, n = 100,
minTaxCount = 2)

# this time request grid cell ID's, which would be useful
# for linking this table back to the grid system
tableFromEpmGrid(tamiasEPM, morphoDisp, meanPat, n = 100,
minTaxCount = 2, id = TRUE)

# from predetermined set of coordinates
pts <- sf::st_sample(tamiasEPM[[1]], size = 10)
```

```
tableFromEpmGrid(tamiasEPM, morphoDisp, meanPat, n = 100,
minTaxCount = 1, coords = pts)
```

write.epmGrid *Save epmGrid object*

Description

Write a epmGrid object to disk.

Usage

```
write.epmGrid(x, filename)
```

Arguments

x	object of class epmGrid
filename	filename with no extension

Details

This function writes a .rds file with xz compression. This file can be read back in with read.epmGrid.

Value

Nothing is returned, but object is written to disk.

Author(s)

Pascal Title

Examples

```
#save
write.epmGrid(tamiasEPM, paste0(tempdir(), '/tamiasEPM'))

# read back in
tamiasEPM <- read.epmGrid(paste0(tempdir(), '/tamiasEPM.rds'))

# delete the file
unlink(paste0(tempdir(), '/tamiasEPM.rds'))
```

writeEpmSpatial — *Write epmGrid Spatial Object to Disk*

Description

Writes the grid to disk for use in other GIS applications.

Usage

```
writeEpmSpatial(x, filename, ...)
```

Arguments

x	object of class epmGrid
filename	filename to be written to, with the appropriate file extension
...	additional arguments to be passed to st_write or writeRaster.

Details

For hexagonal grid systems, appending .shp to the filename will result in a shapefile, whereas appending .gpkg results in a geopackage file. See st_write for additional options. For square grid cells, appending .tif will result in a GeoTiff file being written to disk. If no extensions are included with the filename, then this function will default to geopackage files for hexagonal grids and GeoTiffs for square grids.

Value

the object is written to disk, nothing is returned.

Author(s)

Pascal Title

Examples

```
tamiasEPM
tamiasEPM2 <- createEPMgrid(tamiasPolyList, resolution = 50000,
cellType = 'square', method = 'centroid')
writeEpmSpatial(tamiasEPM, filename = paste0(tempdir(), '/tamiasGrid.shp'))
writeEpmSpatial(tamiasEPM, filename = paste0(tempdir(), '/tamiasGrid.gpkg'))
unlink(paste0(tempdir(), '/tamiasGrid.gpkg'))
# will automatically append .gpkg
writeEpmSpatial(tamiasEPM, filename = paste0(tempdir(), '/tamiasGrid'))

writeEpmSpatial(tamiasEPM2, filename = paste0(tempdir(), '/tamiasGrid.tif'))
unlink(paste0(tempdir(), '/tamiasGrid.tif'))
# will automatically append .tif
writeEpmSpatial(tamiasEPM2, filename = paste0(tempdir(), '/tamiasGrid'))
```

```
# remove files generated by example
unlink(paste0(tempdir(), '/tamiasGrid', c('.dbf', '.gpkg', '.prj', '.shp', '.shx', '.tif')))
```

Index