Package 'phylospatial'

April 16, 2026

Title Spatial Phylogenetic Analysis

Version 1.4.0

Description Analyze spatial phylogenetic diversity patterns.
Use your data on an evolutionary tree and geographic distributions of the terminal taxa to compute diversity and endemism metrics, test significance with null model randomization, analyze community turnover and biotic regionalization, and perform spatial conservation prioritizations. All functions support quantitative community data in addition to binary data.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports ape, sf, stats, terra, utils, vegan

Suggests canaper, furrr, future, testthat (>= 3.0.0), betapart, knitr, rmarkdown, tmap, magrittr, hillR, picante, phytools, nullcat (>= 0.2.0), parallelDist

URL https://matthewkling.github.io/phylospatial/, https://github.com/matthewkling/phylospatial

Config/testthat/edition 3

Depends R (>= 3.5)

VignetteBuilder knitr

BugReports https://github.com/matthewkling/phylospatial/issues

NeedsCompilation no

Author Matthew Kling [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-9073-4240>)

Maintainer Matthew Kling <mattkling@berkeley.edu>

Repository CRAN

Date/Publication 2026-04-16 07:00:11 UTC

Contents

`benefit` *Calculate taxon conservation benefit*

Description

Nonlinear function that converts proportion of range conserved into conservation "benefit."

Usage

```
benefit(x, lambda = 1)
```

Arguments

`x`	Fraction of taxon range protected (value between 0 and 1).
`lambda`	Shape parameter.

Value

Value between 0 and 1.

`clade_dist` *Pairwise distances among clades or nodes*

Description

This function runs `ape::dist.nodes()` with some additional filtering and sorting. By default, it returns distances between every pair of non-nested clades, i.e. every pair of collateral (non-lineal) nodes including terminals and internal nodes. Package `phytools` is required for this function.

Usage

```
clade_dist(tree, lineal = FALSE, edges = TRUE)
```

Arguments

`tree`	A phylogeny of class `"phylo"`.
`lineal`	Logical indicating whether to retain distances for pairs of nodes that are lineal ancestors/descendants. If `FALSE` (the default), these are set to `NA`, retaining values only for node pairs that are collateral kin.
`edges`	Logical indicating whether to return a distance matrix with a row for every edge in `tree`. If `TRUE` (the default), rows/columns of the result correspond to `tree$edge`. If `FALSE`, rows/columns correspond to nodes as in `ape::dist.nodes()`.

Value

A matrix of pairwise distances between nodes.

Examples

```
if(requireNamespace("phytools", quietly = TRUE)){
  clade_dist(ape::rtree(10))
}
```

moss *Load California moss spatial phylogenetic data*

Description

Get example phylospatial data set based on a phylogeny and modeled distributions of 443 moss species across California. This data set is a coarser version of data from Kling et al. (2024). It contains occurrence probabilities, and is available in raster or polygon spatial formats.

Usage

```
moss(format = "raster")
```

Arguments

format	Either "raster" (default) or "polygon"

Value

a phylospatial object

Source

Kling, Gonzalez-Ramirez, Carter, Borokini, and Mishler (2024) bioRxiv, https://doi.org/10.1101/2024.12.16.628580.

Examples

```
moss()
```

phylospatial *Create a spatial phylogenetic object*

Description

This function creates a phylospatial object. This is the core data type in the phylospatial library, and is a required input to most other functions in the package. The two essential components of a spatial phylogenetic object are a phylogenetic tree and an community data set.

Usage

```
phylospatial(
  comm,
  tree = NULL,
  spatial = NULL,
  data_type = c("auto", "probability", "binary", "abundance", "other"),
  clade_fun = NULL,
  build = TRUE,
  check = TRUE,
  area_tol = 0.01
)
```

Arguments

comm	Community data representing the distribution of terminal taxa across sites. Can be a matrix with a column per terminal and a row per site, a SpatRaster with one layer per terminal, or a sf data with a column per terminal. Taxa whose names do not match between column/layer names in comm and tip labels in tree will be dropped with a warning (unless build = FALSE).
tree	Phylogeny of class phylo. Terminals whose names do not match comm will be dropped with a warning (unless build = FALSE). If this argument is not provided, terminals are assumed to follow a "star" tree with uniform branch lengths, which will lead to non-phylogenetic versions of any analyses done with the resulting phylospatial object. Must be a rooted tree.
spatial	An optional SpatRaster layer or sf object indicating site locations. The number of cells or rows must match comm. Ignored if comm is a SpatRaster or sf object.
data_type	Character giving the data type of comm. Must be "binary", "probability", "abundance", "auto" (the default), or "other". This determines how community values for clades are calculated from the values for terminal taxa. If "binary" (presence-absence), a clade is considered present in a site if any terminal in the clade is present. If "probability," clade probabilities are calculated as the probability that at least one terminal is present in a site. If "abundance," clade abundances are calculated as the sum of abundances for terminals in the clade in a site. If "auto," an attempt is made to guess which of these three data types was provided. This argument is ignored if clade_fun is provided, or if build = FALSE. If "other", a custom clade_fun must be supplied.
clade_fun	Function to calculate the local community weight for a clade based on community weights for tips found in a given location. Must be either NULL (the default, in which case the default function for the selected data_type is used) or a summary function that takes a numeric vector and returns a single numeric output. Ignored if comm already includes clade ranges.
build	Logical indicating whether comm already includes clade ranges that should be used instead of building new ones. Default is TRUE. If FALSE, clade_fun is ignored, no checks are performed to harmonize the tip labels and the community data, and the columns of comm must exactly match the order of tree edges in-

cluding tips and larger clades. If clade ranges are included in `comm` but `build = TRUE`, they will be dropped and new clade ranges will be built.

`check` Logical indicating whether community data should be validated. Default is TRUE.

`area_tol` Numeric value giving tolerance for variation in the area of sites. Default is `0.01`. If the coefficient of variation in the area or length of spatial units (e.g. grid cells) exceeds this value, an error will result. This check is performed because various other functions in the library assume that sites are equal area. This argument is ignored if `check = FALSE` or if no spatial data is provided.

Details

This function formats the input data as a `phylospatial` object. Beyond validating, cleaning, and restructing the data, the main operation it performs is to compute community occurrence data for every internal clade on the tree.

Unoccupied sites (rows where no taxon occurs) are automatically removed from the community matrix during construction to improve performance. The original site indices of occupied rows are stored in `ps$occupied`, and the total number of sites (including unoccupied) in `ps$n_sites`, enabling reconstruction of full-extent spatial outputs. Functions that return spatial results automatically expand occupied-only data back to the full spatial extent.

If your data are in the form of occurrence point localities (e.g. from GBIF or BIEN) rather than a gridded community data set, use `ps_grid()` to rasterize the points onto a spatial grid before passing the result to this function.

Value

A `phylospatial` object, which is a list containing the following elements:

"data_type": Character indicating the community data type

"tree": Phylogeny of class `phylo`

"comm": Community matrix containing only occupied sites, including a column for every terminal taxon and every larger clade. Column order corresponds to tree edge order.

"spatial": A `SpatRaster` or `sf` providing spatial coordinates for all sites (including unoccupied). May be missing if no spatial data was supplied.

"occupied": Integer vector of row indices identifying which sites in the original data are occupied.

"n_sites": Total number of sites in the original data, including unoccupied.

"dissim": A community dissimilarity matrix of class `dist` indicating pairwise phylogenetic dissimilarity between occupied sites. Missing unless `ps_add_dissim()` is called.

See Also

`ps_grid()` to convert occurrence point data into a binary or abundance raster that can be used with phylospatial.

Examples

```
# load species distribution data and phylogeny
comm <- terra::rast(system.file("extdata", "moss_comm.tif", package = "phylospatial"))
tree <- ape::read.tree(system.file("extdata", "moss_tree.nex", package = "phylospatial"))

# construct `phylospatial` object
ps <- phylospatial(comm, tree)
ps
```

`plot.phylospatial` *Plot a* `phylospatial` *object*

Description

Plot a `phylospatial` object

Usage

```
## S3 method for class 'phylospatial'
plot(x, y = c("tree", "comm"), max_taxa = 12, ...)
```

Arguments

`x`	`phylospatial` object
`y`	Either `"tree"` or `"comm"`, indicating which component to plot.
`max_taxa`	Integer giving the maximum number of taxon ranges to plot if `y = "tree"`.
`...`	Additional arguments passed to plotting methods, depending on `y` and the class of `x$spatial`. For `y = "tree"`, see plot.phylo; for `y = "comm"`, see plot or plot.sf.

Value

A plot of the tree or community data.

Examples

```
ps <- ps_simulate(20, 20, 20)
plot(ps, "tree")
plot(ps, "comm")
```

plot_lambda *Plot alternative lambda values*

Description

Show a plot illustrating alternative values for the `lambda` parameter in ps_prioritize. Lambda determines the shape of the "benefit" function that determines the conservation value of protecting a given proportion of the geographic range of a species or clade. Positive values place a higher priority on protecting additional populations of largely unprotected taxa, whereas negative values place a higher priority on protecting additional populations of relatively well-protected taxa. The default value used by ps_prioritize is 1.

Usage

```
plot_lambda(lambda = c(-1, -0.5, 0, 0.5, 2, 1))
```

Arguments

`lambda`	A vector of lambda values to plot

Value

Plots a figure

Examples

```
plot_lambda()
plot_lambda(seq(0, 3, .1))
```

ps_add_dissim *Add community dissimilarity data to a* `phylospatial` *object*

Description

This function calculates pairwise phylogenetic dissimilarity between communities and returns the `phylospatial` object with the dissimilarity data added as an element called `dissim`. See ps_dissim for details.

Usage

```
ps_add_dissim(ps, method = "sorensen", ...)
```

Arguments

ps	phylospatial data set.
method	Dissimilarity metric; see ps_dissim for details.
...	Additional arguments passed to ps_dissim, such as fun, endemism, or normalize.

Value

ps with a new dissim element added.

Examples

```
ps <- ps_simulate(data_type = "prob")
ps_add_dissim(ps)
ps_add_dissim(ps, fun = "vegdist", method = "jaccard", endemism = TRUE)
```

---

ps_canape *Categorical Analysis of Neo- and Paleo-Endemism (CANAPE)*

Description

This function classifies sites into areas of significant endemism according to the scheme of Mishler et al. (2014). Categorization is based on randomization quantile values for PE, RPE, and CE (which Mishler et al. call "PE on the comparison tree").

Usage

```
ps_canape(rand, alpha = 0.05)
```

Arguments

rand	An object returned by running ps_rand. It must include the metrics PE, RPE, and CE, and must have been computed with summary = "quantile".
alpha	Numeric value between 0 and 1 giving the one-tailed p-value threshold to use when determining significance.

Details

Endemism significance categories are defined as follows:

- Endemism not significant: neither PE nor CE are significantly high at alpha.
- Significant neoendemism: PE or CE are significantly high at alpha; RPE significantly low at alpha / 2 (two-tailed test).
- Significant paleoendemism: PE or CE are significantly high at alpha; RPE significantly high at alpha / 2 (two-tailed test)..
- Significant mixed-endemism: PE or CE are significantly high at alpha; RPE not significant.
- Significant super-endemism: PE or CE are significantly high at alpha / 5; RPE not significant.

Value

An object of the same class as `rand` containing a variable called `"canape"`, with values 0-4 corresponding to not-significant, mixed-, super-, neo-, and paleo-endemism, respectively.

References

Mishler, B. D., Knerr, N., González-Orozco, C. E., Thornhill, A. H., Laffan, S. W., & Miller, J. T. (2014). Phylogenetic measures of biodiversity and neo-and paleo-endemism in Australian Acacia. Nature Communications, 5(1), 4473.

Examples

```
# classic CANAPE using binary data and the curveball algorithm
# (note that a real analysis would require a much higher `n_rand`)
set.seed(123456)
ps <- ps_simulate(data_type = "binary")
rand <- ps_rand(ps, metric = c("PE", "RPE", "CE"),
                fun = "nullmodel", method = "curveball",
                n_rand = 25, burnin = 10000, progress = FALSE)
canape <- ps_canape(rand)
terra::plot(canape)
```

---

ps_canaper *Binary randomization tests including CANAPE*

Description

This function is a wrapper around `canaper::cpr_rand_test()`. It only works with binary community data. It is largely redundant with `ps_rand()` and `ps_canape()`, which are more flexible in supporting data sets with non-binary community data. However, this function runs faster, and supports custom null models via make.commsim.

Usage

```
ps_canaper(ps, null_model = "curveball", spatial = TRUE, ...)
```

Arguments

`ps`	phylospatial object
`null_model`	see `?canaper::cpr_rand_test()`
`spatial`	Logical: should the function return a spatial object (TRUE, default) or a vector (FALSE).
`...`	further arguments passed to `canaper::cpr_rand_test()`

Details

This function runs `canaper::cpr_rand_test()`; see the help for that function for details.

It also runs `canaper::cpr_classify_endem()` on the result, and includes the resulting classification as an additional variable, 'endem_type', in the output. 'endem_type' values 0-4 correspond to not-significant, neo, paleo, mixed, and super endemisim, respectively.

Value

A `matrix` or `SpatRaster`, or `sf` with a column or layer for each metric.

References

Mishler, B. D., Knerr, N., González-Orozco, C. E., Thornhill, A. H., Laffan, S. W., & Miller, J. T. (2014). Phylogenetic measures of biodiversity and neo-and paleo-endemism in Australian Acacia. Nature Communications, 5(1), 4473.

Nitta, J. H., Laffan, S. W., Mishler, B. D., & Iwasaki, W. (2023). canaper: categorical analysis of neo-and paleo-endemism in R. Ecography, 2023(9), e06638.

See Also

`ps_canape()`, `ps_rand()`

Examples

```
if(requireNamespace("canaper")){
      ps <- ps_simulate(data_type = "binary")
      terra::plot(ps_canaper(ps)$pd_obs_p_upper)
}
```

---

ps_dissim *Quantitative phylogenetic dissimilarity*

Description

This function calculates pairwise phylogenetic dissimilarity between communities. It works with both binary and quantitative community data sets. A wide range of phylogentic community dissimilarity metrics are supported, including phylogenetic Sorensen's and Jaccard's distances, turnover and nestedness components of Sorensen's distance (Baselga & Orme, 2012), and phylogenetic versions of all community distance indices provided through the vegan library. The function also includes options to scale the community matrix in order to focus the analysis on endemism and/or on proportional differences in community composition. The results from this function can be visualized using ps_rgb or ps_regions, or used in a variety of statistical analyses.

Usage

```
ps_dissim(
  ps,
  method = "sorensen",
  fun = c("vegdist", "designdist", "chaodist"),
  endemism = FALSE,
  normalize = FALSE,
  tips_only = FALSE,
  n_cores = NULL,
  ...
)
```

Arguments

ps	phylospatial object.
method	Character indicating the dissimilarity index to use: • "sorensen": Sorensen's dissimilarity, a.k.a. Bray-Curtis distance (the default) • "sorensen_turnover": The turnover component of Sorensen's dissimilarity, a.k.a. Simpson's. • "sorensen_nestedness": The nestedness component of Sorensen's dissimilarity. • Any other valid `method` passed to `fun`. For options, see the documentation for those functions.
fun	Character indicating which general distance function from the `vegan` library to use: "vegdist" (the default), "designdist", or "chaodist". (While these functions are not explicitly designed to calculate phylogenetic beta diversity, their use here incorporates the phylogenetic components.) This argument is ignored if one of the three "sorensen" methods is selected.
endemism	Logical indicating whether community values should be divided by column totals (taxon range sizes) to derive endemism before computing distances.
normalize	Logical indicating whether community values should be divided by row totals (community sums) before computing distances. If `TRUE`, dissimilarity is based on proportional community composition. Normalization is applied after endemism.
tips_only	Logical indicating whether to compute dissimilarity using only terminal taxa (`TRUE`) rather than the full phylogenetic community matrix (`FALSE`, the default). When `TRUE`, branch length weighting is skipped and the result is a standard (non-phylogenetic) community dissimilarity. Endemism and normalization options still apply.
n_cores	Integer controlling the computation backend. The default `NULL` uses `parallelDist` with all available cores if installed, falling back to `vegan` otherwise. Setting `n_cores = 0` forces the `vegan` backend. Setting `n_cores` to a positive integer uses `parallelDist` with that many threads (requires `parallelDist` package). The `parallelDist` backend is faster than `vegan` even single-threaded for

supported methods; unsupported methods (e.g. custom `designdist` formulas, turnover decomposition) always fall back to `vegan` with a message.

`...` Additional arguments passed to `fun`.

Value

A pairwise phylogenetic dissimilarity matrix of class `dist`.

References

Graham, C. H., & Fine, P. V. (2008). Phylogenetic beta diversity: linking ecological and evolutionary processes across space in time. Ecology Letters, 11(12), 1265-1277.

Baselga, A., & Orme, C. D. L. (2012). betapart: an R package for the study of beta diversity. Methods in Ecology and Evolution, 3(5), 808-812.

Pavoine, S. (2016). A guide through a family of phylogenetic dissimilarity measures among sites. Oikos, 125(12), 1719-1732.

See Also

`ps_add_dissim()`

Examples

```
# example data set:
ps <- ps_simulate(n_tips = 50)

# The default arguments give Sorensen's quantitative dissimilarity index
# (a.k.a. Bray-Curtis distance):
d <- ps_dissim(ps)

# Specifying a custom formula explicitly via `designdist`;
# (this is the Bray-Curtis formula, so it's equivalent to the prior example)
d <- ps_dissim(ps, method = "(b+c)/(2*a+b+c)",
      fun = "designdist", terms = "minimum", abcd = TRUE)

# Alternative arguments can specify a wide range of dissimilarity measures;
# here's endemism-weighted Jaccard's dissimilarity:
d <- ps_dissim(ps, method = "jaccard", endemism = TRUE)
```

`ps_diversity` *Calculate spatial phylogenetic diversity and endemism metrics*

Description

This function calculates a variety of alpha phylogenetic diversity metrics, including measures of richness, regularity, and divergence. If continuous community data (probabilities or abundances) are provided, they are used in calculations, giving quantitative versions of the classic binary metrics.

Usage

```
ps_diversity(ps, metric = c("PD", "PE", "CE", "RPE"), spatial = TRUE)
```

Arguments

ps	phylospatial object (created by `phylospatial()` or `ps_simulate()`).
metric	Character vector containing the abbreviation for one or more diversity metrics listed in the details below. Can also specify `"all"` to calculate all available metrics. A small subset of common measures are selected by default.
spatial	Logical: should the function return a spatial object (TRUE, default) or a matrix (FALSE)?

Details

The function calculates the following metrics. Endemism-weighted versions of most metrics are available. All metrics are weighted by occurrence probability or abundance, if applicable.

Richness measures:

- **TD**—Terminal Diversity, i.e. richness of terminal taxa (in many cases these are species): $\sum_t p_t$
- **TE**—Terminal Endemism, i.e. total endemism-weighted diversity of terminal taxa, a.k.a. "weighted endemism": $\sum_t p_t r_t^{-1}$
- **CD**—Clade Diversity, i.e. richness of taxa at all levels (equivalent to PD on a cladogram): $\sum_b p_b$
- **CE**—Clade Endemism, i.e. total endemism-weighted diversity of taxa at all levels (equivalent to PE on a cladrogram): $\sum_b p_b r_b^{-1}$
- **PD**—Phylogenetic Diversity, i.e. total branch length occurring in a site: $\sum_b L_b p_b$
- **PE**—Phylogenetic Endemism, i.e. endemism-weighted PD: $\sum_b L_b p_b r_b^{-1}$
- **ShPD**—Shannon Phylogenetic Diversity, a.k.a. "phylogenetic entropy" (this version is the log of the "effective diversity" version based on Hill numbers): $-\sum_b L_b n_b log(n_b)$
- **ShPE**—Shannon phylogenetic Endemism, an endemism-weighted version of ShPD: $-\sum_b L_b n_b log(e_b) r_b^{-1}$
- **SiPD**—Simpson Phylogenetic Diversity: $1/\sum_b L_b n_b^2$
- **SiPE**—Simpson Phylogenetic Endemism, an endemism-weighted version of SiPD: $1/\sum_b L_b r_b^{-1} e_b^2$

Divergence measures:

- **RPD**—Relative Phylogenetic Diversity, i.e. mean branch segment length (equivalent to PD / CR): $\sum_b L_b p_b / \sum_b p_b$
- **RPE**—Relative Phylogenetic Endemism, i.e. mean endemism-weighted branch segment length (equivalent to PE / CE): $\sum_b L_b p_b r_b^{-1} / \sum_b p_b r_b^{-1}$
- **MPDT**—Mean Pairwise Distance between Terminals, i.e. the classic MPD metric. This is the average of cophenetic distances, weighted by p_t.
- **MPDN**—Mean Pairwise Distance between Nodes, an experimental version of MPD that considers distances between every pair of non-nested clades, putting more weight on deeper branches than does MPDT. This is the mean of distances between all collateral (non-lineal) node pairs including terminal and internal nodes, weighted by p_b.

- Note that divergence can also be assessed by using `ps_rand()` to run null model analyses of richness measures like PD.

Regularity measures:

- **VPDT**—Variance in Pairwise Distances between Terminals, i.e. the classic VPD metric, weighted by p_t.
- **VPDN**—Variance in Pairwise Distances between Nodes, i.e. MPDN but variance.

In the above equations, b indexes all taxa including terminals and larger clades; t indexes terminals only; p_i is the occurrence value (binary, probability, or abundance) of clade/terminal i in a given community; L_b is the length of the phylogenetic branch segment unique to clade b; and r_i is the sum of p_i across all sites. For Shannon and Simpson indices, only nonzero elements of p_b are used, $n_b = p_b / \sum_b p_b L_b$, and $e_b = p_b / \sum_b p_b L_b r_b^{-1}$.

Value

A matrix, `sf` data frame, or `SpatRaster` with a column or layer for each requested diversity metric.

References

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

Laffan, S. W., & Crisp, M. D. (2003). Assessing endemism at multiple spatial scales, with an example from the Australian vascular flora. Journal of Biogeography, 30(4), 511-520.

Rosauer, D. A. N., 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.

Allen, B., Kon, M., & Bar-Yam, Y. (2009). A new phylogenetic diversity measure generalizing the Shannon index and its application to phyllostomid bats. The American Naturalist, 174(2), 236-243.

Chao, A., Chiu, C. H., & Jost, L. (2010). Phylogenetic diversity measures based on Hill numbers. Philosophical Transactions of the Royal Society B: Biological Sciences, 365(1558), 3599-3609.

Mishler, B. D., Knerr, N., González-Orozco, C. E., Thornhill, A. H., Laffan, S. W., & Miller, J. T. (2014). Phylogenetic measures of biodiversity and neo-and paleo-endemism in Australian Acacia. Nature Communications, 5(1), 4473.

Tucker, C. M., Cadotte, M. W., Davies, T. J., et al. (2016) A guide to phylogenetic metrics for conservation, community ecology and macroecology. Biological Reviews, 92(2), 698-715.

Kling, M. M., Mishler, B. D., Thornhill, A. H., Baldwin, B. G., & Ackerly, D. D. (2019). Facets of phylodiversity: evolutionary diversification, divergence and survival as conservation targets. Philosophical Transactions of the Royal Society B, 374(1763), 20170397.

Examples

```
ps <- ps_simulate()
div <- ps_diversity(ps)
terra::plot(div)
```

ps_expand *Expand occupied-only results to full spatial extent*

Description

Takes a matrix or vector computed on occupied sites only and expands it back to the full site grid, inserting NA for unoccupied sites. This is useful when performing custom analyses on ps$comm (which contains only occupied sites) and mapping the results back to the full raster or spatial object.

Usage

```
ps_expand(ps, x, spatial = FALSE)
```

Arguments

ps	A phylospatial object.
x	A matrix (or vector) with nrow(ps$comm) rows (i.e. one row per occupied site).
spatial	Logical: if TRUE, convert the expanded result to a spatial object using ps$spatial. Default is FALSE.

Value

If spatial = FALSE, a matrix with ps$n_sites rows and NA for unoccupied sites. If spatial = TRUE, a SpatRaster or sf object.

Examples

```
ps <- ps_simulate()

# custom analysis on the occupied-only community matrix
site_totals <- matrix(rowSums(ps$comm), ncol = 1)
colnames(site_totals) <- "total"

# expand to full extent as a matrix
ps_expand(ps, site_totals)

# expand and convert to spatial
ps_expand(ps, site_totals, spatial = TRUE)
```

`ps_geodist` *Geographic distance between sites*

Description

Calculate pairwise geographic distances between occupied sites in a `phylospatial` object. The result is a `dist` object with the same dimensions and site ordering as `ps_dissim()`, making it straightforward to compare phylogenetic dissimilarity with geographic distance (e.g., for distance-decay analyses or Mantel tests).

Usage

```
ps_geodist(ps)
```

Arguments

`ps`	A `phylospatial` object with spatial data.

Details

For raster data, distances are computed between cell centroids. For polygon or line `sf` data, distances are computed between geometry centroids. Great-circle distances are used automatically when the data has a geographic (lon/lat) CRS; Euclidean distances are used for projected data or data without a CRS. Units are meters when a CRS is defined, or unitless coordinate distances otherwise.

Value

A pairwise geographic distance matrix of class `dist`, with one entry per pair of occupied sites. Units are meters when a CRS is defined, or raw coordinate units otherwise.

Examples

```
ps <- moss()
geo <- ps_geodist(ps)
phy <- ps_dissim(ps)

# distance-decay plot
plot(as.vector(geo), as.vector(phy),
     xlab = "Geographic distance (m)",
     ylab = "Phylogenetic dissimilarity",
     pch = ".", col = "#00000020")
```

ps_get_comm *Get* phylospatial *community data*

Description

Get phylospatial community data

Usage

```
ps_get_comm(ps, tips_only = TRUE, spatial = TRUE)
```

Arguments

ps	phylospatial object.
tips_only	Logical indicating whether only the terminal taxa (TRUE, the default) or all taxa (FALSE) should be returned.
spatial	Logical indicating whether a spatial (SpatRaster or sf) object should be returned. Default is TRUE; if FALSE, a matrix is returned.

Value

If spatial = TRUE, a SpatRaster or sf object with a layer/column for every taxon, with NA for unoccupied sites. If spatial = FALSE, a matrix containing only occupied sites (i.e., with nrow equal to the number of occupied sites, not the total number of grid cells). Use ps_expand() to expand an occupied-only matrix back to the full spatial extent if needed.

Examples

```
ps <- ps_simulate()

# the defaults return a spatial object of terminal taxa distributions:
ps_get_comm(ps)

# get distributions for all taxa, as a matrix
pcomm <- ps_get_comm(ps, tips_only = FALSE, spatial = FALSE)
```

ps_grid *Convert occurrence point data to a gridded community data set*

Description

This function takes a set of occurrence point localities (e.g. from GBIF or BIEN) and rasterizes them onto a spatial grid to produce a community data set suitable for passing to phylospatial(). Each species' point records are aggregated within grid cells to produce either binary presence-absence or count data.

Usage

```
ps_grid(
  x,
  grid = NULL,
  res = NULL,
  crs = NULL,
  cols = c(1, 2, 3),
  data_type = c("binary", "count")
)
```

Arguments

x	A data frame or sf points object containing occurrence records. If a data frame, it must contain columns for species identity and geographic coordinates (see cols). Coordinates in a data frame are assumed to be WGS84 longitude and latitude; an error is thrown if values fall outside valid ranges. If an sf object, it must contain a column for species identity and have point geometry; only the first element of cols is used.
grid	An optional SpatRaster to use as the target grid. If provided, res and crs are ignored. Point records falling outside the grid extent are silently dropped.
res	Numeric giving the grid cell resolution when generating a new grid. Units are meters if crs is provided or auto-generated, or in the units of crs otherwise. Ignored if grid is provided. If NULL (the default), a resolution is automatically chosen to produce approximately 1000 grid cells across the extent of the data. See Details.
crs	Optional CRS specification (any format accepted by terra::crs()) for the output grid. Ignored if grid is provided. If NULL (the default), an Albers Equal Area projection centered on the data is automatically generated. See Details.
cols	A vector of length 1 (for sf input) or 3 (for data frame input) identifying the columns in x for species, longitude, and latitude, in that order. Can be character column names or integer column indices but not a mix. Default is c(1, 2, 3). For sf input, only the first element (species column) is used; the remaining elements are ignored.
data_type	Character indicating the type of community data to produce. Either "binary" (default), in which cells receive a 1 if one or more records are present and 0 otherwise, or "count", in which cells receive the number of records.

Details

When grid is NULL, a grid is automatically generated from the point data. If crs is also NULL, an Albers Equal Area (AEA) projection is created based on the geographic extent of the points, with the central meridian and latitude at the midpoint of the coordinate ranges, and standard parallels at 1/6 and 5/6 of the latitude range. This ensures that grid cells are equal-area, which is an assumption of various functions in the phylospatial package. Users working with other spatial layers (e.g., environmental rasters) may want to supply a grid or crs that matches their existing data.

When res is NULL and no grid is supplied, the resolution is automatically chosen so that the grid contains approximately 1000 cells. Specifically, the resolution is set to sqrt(extent_area

/ 1000), where extent_area is the area of the bounding box of the projected points. The auto-generated grid is buffered by half a grid cell on each side to ensure that edge points are not excluded.

The res parameter is interpreted in the units of the output CRS. When the auto-generated AEA projection is used, units are meters. If a geographic (lon/lat) CRS is supplied, res is in degrees and the resulting cells will not be equal-area; a warning is issued in this case.

Coordinate cleaning and taxonomic name matching are outside the scope of this function. Users are encouraged to clean occurrence data beforehand (e.g., using the CoordinateCleaner package) and to ensure that species names in the occurrence data match the tip labels of their phylogeny before proceeding to phylospatial().

Value

A SpatRaster with one layer per species, suitable for passing directly to phylospatial() as the comm argument. Layer names correspond to unique values in the species column.

See Also

phylospatial() for constructing a spatial phylogenetic object from the output.

Examples

```
# simulate some occurrence records
set.seed(42)
occ <- data.frame(
  species = sample(paste0("sp", 1:10), 500, replace = TRUE),
  x = runif(500, -122, -118),
  y = runif(500, 34, 38)
)

# grid the occurrences with auto resolution (~1000 cells)
comm <- ps_grid(occ)
terra::plot(comm[[1:4]])

# specify columns by name
comm <- ps_grid(occ, cols = c("species", "x", "y"))

# grid at a specific resolution (50 km)
comm <- ps_grid(occ, res = 50000)

# use a custom grid
template <- terra::rast(res = 0.5, xmin = -123, xmax = -117, ymin = 33, ymax = 39,
                        crs = "EPSG:4326")
comm2 <- ps_grid(occ, grid = template)

# from sf points
occ_sf <- sf::st_as_sf(occ, coords = c("x", "y"), crs = 4326)
comm3 <- ps_grid(occ_sf, cols = "species")
```

ps_ordinate *Community phylogenetic ordination*

Description

Perform an ordination that reduces a spatial phylogenetic data set into k dimensions, using one of several alternative ordination algorithms.

Usage

```
ps_ordinate(ps, method = c("cmds", "nmds", "pca"), k = 3, spatial = TRUE)
```

Arguments

ps	A phylospatial object. Unless method = "pca", ps must have a non-null dissim component, generated by ps_add_dissim.
method	Ordination method, either "cmds" (the default, classical MDS, implemented via stats::cmdscale(), "nmds" (nonmetric MDS, implemented via vegan::metaMDS(); this is slower but often preferred), or "pca" (principal component analysis, implemented via stats::prcomp()),.
k	Positive integer giving the desired number of output dimensions; default is 3.
spatial	Logical indicating whether a spatial object (inherited from ps) should be returned. Default is TRUE.

Value

A matrix or spatial object with k variables.

See Also

For visualization using ordination onto RGB color space, see ps_rgb().

Examples

```
ps <- ps_add_dissim(ps_simulate(50, 5, 5))
ord <- ps_ordinate(ps, method = "cmds", k = 4)
terra::plot(ord)
```

`ps_prioritize` *Phylogenetic conservation prioritization*

Description

Create a ranking of conservation priorities using optimal or probabilistic forward stepwise selection. Prioritization accounts for the occurrence quantities for all lineages present in the site, including terminal taxa and larger clades; the evolutionary branch lengths of these lineages on the phylogeny, which represent their unique evolutionary heritage; the impact that protecting the site would have on these lineages' range-wide protection levels; the compositional complementarity between the site, other high-priority sites, and existing protected areas; the site's initial protection level; the relative cost of protecting the site; and a free parameter "lambda" determining the shape of the conservation benefit function.

Usage

```
ps_prioritize(
  ps,
  init = NULL,
  cost = NULL,
  lambda = 1,
  protection = 1,
  max_iter = NULL,
  method = c("optimal", "probable"),
  trans = function(x) replace(x, which(rank(-x) > 25), 0),
  n_reps = 100,
  n_cores = 1,
  summarize = TRUE,
  spatial = TRUE,
  progress = interactive()
)
```

Arguments

`ps`	phylospatial object.
`init`	Optional numeric vector or spatial object giving the starting protection status of each site across the study area. Values should be between 0 and 1 and represent the existing level of conservation effectiveness in each site. If this argument is not specified, it is assumed that no existing reserves are present.
`cost`	Optional numeric vector or spatial object giving the relative cost of protecting each site. Values should be positive, with greater values indicating higher cost of conserving a site. If this argument is not specified, cost is assumed to be uniform across sites.
`lambda`	Shape parameter for taxon conservation benefit function. This can be any real number. Positive values, such as the default value `1`, place higher priority on conserving the first part of the range of a given species or clade, while negative

values (which are not typically used) place higher priority on fully protecting the most important taxa (those with small ranges and long branches) rather than partially protecting all taxa. See the function plot_lambda for an illustration of alternative `lambda` values.

`protection` Degree of protection of proposed new reserves (number between 0 and 1, with same meaning as `init`).

`max_iter` Integer giving max number of iterations to perform before stopping, i.e. max number of sites to rank.

`method` Procedure for selecting which site to add to the reserve network at each iteration:

- "optimal": The default, this selects the site with the highest marginal value at each iteration. This is a optimal approach that gives the same result each time.
- "probable": This option selects a site randomly, with selection probabilities calculated as a function of sites' marginal values. This approach gives a different prioritization ranking each time an optimization is performed, so `n_reps` optimizations are performed, and ranks for each site are summarized across repetitions.

`trans` A function that transforms marginal values into relative selection probabilities; only used if `method = "probable"`. The function should take a vector of positive numbers representing marginal values and return an equal-length vector of positive numbers representing a site's relative likelihood of being selected. The default function returns the marginal value if a site is in the top 25 highest-value sites, and zero otherwise.

`n_reps` Number of random repetitions to do; only used if `method = "probable"`. Depending on the data set, a large number of reps (more than the default of 100) may be needed in order to achieve a stable result. This may be a computational barrier for large data sets; multicore processing via `n_cores` can help.

`n_cores` Number of compute cores to use for parallel processing; only used if `method = "probable"`.

`summarize` Logical: should summary statistics across reps (TRUE, default) or the reps themselves (FALSE) be returned? Only relevant if `method = "probable"`.

`spatial` Logical: should the function return a spatial object (TRUE, default) or a matrix (FALSE)?

`progress` Logical: should a progress bar be displayed?

Details

This function uses the forward stepwise selection algorithm of Kling et al. (2019) to generate a ranked conservation prioritization. Prioritization begins with the starting protected lands network identified in `init`, if provided. At each iteration, the marginal conservation value of fully protecting each site is calculated, and a site is selected to be added to the reserve network. Selection can happen either in an "optimal" or "probable" fashion as described under the `method` argument. This process is repeated until all sites are fully protected or until `max_iter` has been reached, with sites selected early in the process considered higher conservation priorities.

The benefit of the probabilistic approach is that it relaxes the potentially unrealistic assumption that protected land will actually be added in the optimal order. Since the algorithm avoids compositional

redundancy between high-priority sites, the optimal approach will never place high priority on a site that has high marginal value but is redundant with a slightly higher-value site, whereas the probabilistic approach will select them at similar frequencies (though never in the same randomized run).

Every time a new site is protected as the algorithm progresses, it changes the marginal conservation value of the other sites. Marginal value is the increase in conservation benefit that would arise from fully protecting a given site, divided by the cost of protecting the site. This is calculated as a function of the site's current protection level, the quantitative presence probability or abundance of all terminal taxa and larger clades present in the site, their evolutionary branch lengths on the phylogeny, the impact that protecting the site would have on their range-wide protection levels, and the free parameter `lambda`. `lambda` determines the relative importance of protecting a small portion of every taxon's range, versus fully protecting the ranges of more valuable taxa (those with longer evolutionary branches and smaller geographic ranges).

Value

Matrix or spatial object containing a ranking of conservation priorities. Lower rank values represent higher conservation priorities. All sites with a lower priority than `max_iter` have a rank value equal to the number of sites in the input data set (i.e. the lowest possible priority).

If `method = "optimal"`**.** the result contains a single variable "priority" containing the ranking.

If `method = "probable"` **and** `summarize = TRUE`**,** the "priority" variable gives the average rank across reps, variables labeled "pctX" give the Xth percentile of the rank distribution for each site, variables labeled "topX" give the proportion of reps in which a site was in the top X highest-priority sites, and variables labeled "treX" give a ratio representing "topX" relative to the null expectation of how often "topX" should occur by chance alone.

If `method = "probable"` **and** `summarize = FALSE`**,** the result contains the full set of `n_rep` solutions, each representing the the ranking, with low values representing higher priorities..

References

Kling, M. M., Mishler, B. D., Thornhill, A. H., Baldwin, B. G., & Ackerly, D. D. (2019). Facets of phylodiversity: evolutionary diversification, divergence and survival as conservation targets. Philosophical Transactions of the Royal Society B, 374(1763), 20170397.

See Also

`benefit()`, `plot_lambda()`

Examples

```
# simulate a toy `phylospatial` data set
set.seed(123)
ps <- ps_simulate()

# basic prioritization
p <- ps_prioritize(ps)

# specifying locations of initial protected areas
# (can be binary, or can be continuous values between 0 and 1)
```

```
# here we'll create an `init` raster with arbitrary values ranging from 0-1,
# using the reference raster layer that's part of our `phylospatial` object
protected <- terra::setValues(ps$spatial, seq(0, 1, length.out = 400))
cost <- terra::setValues(ps$spatial, rep(seq(100, 20, length.out = 20), 20))
p <- ps_prioritize(ps, init = protected, cost = cost)

# using probabilistic prioritization
p <- ps_prioritize(ps, init = protected, cost = cost,
      method = "prob", n_reps = 1000, max_iter = 10)
terra::plot(p$top10)
```

`ps_quantize` *Stratified randomization of a phylospatial object*

Description

Generates a randomized version of a phylospatial object by extracting the tip community matrix, permuting it using `nullcat::quantize()`, and rebuilding the phylospatial object using the permuted tip matrix.

Usage

```
ps_quantize(ps, wt_row = NULL, wt_col = NULL, ...)
```

Arguments

`ps`	Object of class `phylospatial`
`wt_row`, `wt_col`	Optional square numeric weight matrices controlling which pairs of rows (sites) or columns (species) are likely to exchange values during randomization. Enables spatially constrained or functional-group constrained null models; e.g. a geographic distance decay matrix from ps_geodist can be transformed and used as `wt_row`. See nullcat for details. If left unspecified (the default), gives unweighted randomization.
`...`	Additional arguments passed to quantize, such as `method`, `n_strata`, `transform`, `fixed`, `n_iter`, etc.

Details

The nullcat quantize routine involves three steps: converting a quantitative matrix to categorical strata, permuting the resulting categorical matrix using one of several categorical null model algorithms, and mapping the randomized categories back to quantitative values. Supply arguments via `...` to control options for each of these stages.

For repeated randomizations to generate a null distribution, it is more efficient to use `ps_rand(fun = "quantize")`, which is structured to avoid unnecessarily recomputing overhead that is shared across randomizations.

Value

A randomized version of ps

See Also

ps_rand(), ps_geodist()

Examples

```
if (requireNamespace("nullcat", quietly = TRUE)) {
  ps <- ps_simulate(data_type = "prob")
  ps_rand <- ps_quantize(ps, n_strata = 4,
    n_iter = 1000,
    method = "curvecat", fixed = "cell")

  # spatially constrained randomization
  geo <- as.matrix(ps_geodist(ps))
  W <- exp(-geo / median(geo))
  ps_rand <- ps_quantize(ps, n_strata = 4,
    n_iter = 1000,
    method = "curvecat", fixed = "cell",
    wt_row = W)
}
```

---

ps_rand *Null model randomization analysis of alpha diversity metrics*

Description

This function compares phylodiversity metrics calculated in ps_diversity to their null distributions computed by randomizing the community matrix or shuffling the tips of the phylogeny, indicating statistical significance under the assumptions of the null model. Various null model algorithms are available for binary, probability, and count data.

Usage

```
ps_rand(
  ps,
  metric = c("PD", "PE", "RPE", "CE"),
  fun = "tip_shuffle",
  method = NULL,
  n_iter = 1000,
  n_rand = 100,
  summary = "quantile",
  spatial = TRUE,
  n_cores = 1,
  progress = interactive(),
```

```
  wt_row = NULL,
  wt_col = NULL,
  ...
)
```

Arguments

ps	phylospatial object.
metric	Character vector giving one or more diversity metrics to calculate; see ps_diversity for options. Can also specify "all" to calculate all available metrics.
fun	Null model function to use. Must be either "tip_shuffle", "nullmodel", "nullcat", "quantize", or an actual function: • "tip_shuffle" (the default): randomly shuffles the identities of terminal taxa. • "nullmodel": uses nullmodel and simulate.nullmodel, from the vegan package, which offer a wide range of randomization algorithms with different properties. • "nullcat": uses null model algorithms from the nullcat package. Only works with binary community data. This is the recommended path for binary data when mixing diagnostics (via ps_suggest_n_iter()) or spatial weights (via wt_row or wt_col) are desired. • "quantize": uses quantize, which converts a quantitative matrix to discrete strata, applies a categorical variant of the selected null model, and then maps randomized strata back to values. Only works with quantitative (probability or abundance) community data. • Any other function that accepts a community matrix as its first argument and returns a randomized version of the matrix.
method	Method used by the selected function. • For fun = "nullmodel", one of the method options listed under commsim. Be sure to select a method that is appropriate to your community data_type (binary, quantitative, abundance). • For fun = "nullcat" or fun = "quantize", one of the categorical algorithms listed under nullcat_methods (e.g. "curvecat", "swapcat"). • Ignored if fun is "tip_shuffle" or if it is a custom function.
n_iter	Integer giving the number of swap iterations per randomized matrix. Controls how thoroughly each null matrix is mixed before sampling. Default is 1000. Used with fun = "nullcat" (passed as n_iter to nullcat), fun = "quantize" (passed to quantize_prep), and fun = "nullmodel" with sequential methods like curveball (passed as burnin to simulate.nullmodel; ignored for non-sequential vegan methods). Use ps_suggest_n_iter() to estimate an appropriate value for your dataset.
n_rand	Integer giving the number of random communities to generate.
summary	Character indicating which summary statistic to return. If "quantile", the default, the function returns the proportion of randomizations in which the observed diversity metric was greater than the randomized metric. If "zscore", it returns a "standardized effect size" or z-score relating the observed value to the mean and standard deviation of the randomizations.

spatial	Logical: should the function return a spatial object (TRUE, default) or a matrix (FALSE).
n_cores	Integer giving the number of compute cores to use for parallel processing.
progress	Logical: should a progress bar be displayed?
wt_row, wt_col	Optional square numeric weight matrices controlling which pairs of rows (sites) or columns (species) are likely to exchange values during randomization. Only used with fun = "nullcat" or fun = "quantize". Enables spatially or trait-constrained null models; e.g. a geographic distance decay matrix from ps_geodist can be used as wt_row. See nullcat for details.
...	Additional arguments passed to the selected function: quantize, nullcat, simulate.nullmodel, or a custom function. Note that the nsim argument to simulate.nullmodel should not be used here; specify n_rand instead.

Value

A matrix with a row for every row of x, a column for every metric specified in metric, and values for the summary statistic. Or if spatial = TRUE, a sf or SpatRaster object containing these data.

See Also

ps_diversity(), ps_geodist()

Examples

```
# simulate a `phylospatial` data set and run randomization with default settings
ps <- ps_simulate(data_type = "prob")
rand <- ps_rand(ps)

# using the `quantize` function with the `curvecat` algorithm
if(requireNamespace("nullcat")){
    rand <- ps_rand(ps,
      fun = "quantize", method = "curvecat",
      transform = sqrt, n_strata = 4, fixed = "cell")
}

# binary data with nullcat's curvecat algorithm
ps2 <- ps_simulate(data_type = "binary")
if(requireNamespace("nullcat")){
    rand <- ps_rand(ps2, fun = "nullcat", method = "curvecat", n_iter = 1000)
}

# spatially constrained randomization using geographic distance weights
if(requireNamespace("nullcat")){
    geo <- as.matrix(ps_geodist(ps2))
    W <- exp(-geo / median(geo))
    rand <- ps_rand(ps2, fun = "nullcat", method = "curvecat",
                    n_iter = 1000, wt_row = W)
}

# using binary data, with a vegan `nullmodel` algorithm
```

```
rand <- ps_rand(ps2, "PD", "nullmodel", "r2")

# using abundance data
ps3 <- ps_simulate(data_type = "abund")
rand <- ps_rand(ps3, metric = c("ShPD", "SiPD"),
      fun = "nullmodel", method = "abuswap_c")
```

`ps_regions` *Cluster analysis to identify phylogenetic regions*

Description

Perform a clustering analysis that categorizes sites into biogeographic regions based on phylogenetic community compositional similarity.

Usage

```
ps_regions(ps, k = 5, method = "average", endemism = FALSE, normalize = TRUE)
```

Arguments

`ps`	A `phylospatial` object. If `method` is anything other than `"kmeans"`, it must contain a `dissim` component generated by ps_add_dissim.
`k`	Number of spatial clusters to divide the region into (positive integer). See ps_regions_eval to help choose a value of k by comparing the variance explained by different numbers of regions.
`method`	Clustering method. Options include all methods listed under hclust, and `"kmeans"`. If `"kmeans"` is selected, the `dissim` component of `ps` is ignored.
`endemism`	Logical indicating whether community values should be divided by column totals (taxon range sizes) to derive endemism. Only used if `method = "kmeans"`; in other cases this information should instead be supplied to ps_add_dissim.
`normalize`	Logical indicating whether community values should be divided by row totals (community sums). If `TRUE`, dissimilarity is based on proportional community composition. This happens after endemism is derived. Only used if `method = "kmeans"`; in other cases this information should instead be supplied to ps_add_dissim.

Value

A raster or matrix with an integer indicating which of the `k` regions each site belongs to.

References

Daru, B. H., Elliott, T. L., Park, D. S., & Davies, T. J. (2017). Understanding the processes underpinning patterns of phylogenetic regionalization. Trends in Ecology & Evolution, 32(11), 845-860.

Examples

```
ps <- ps_simulate()

# using kmeans clustering algorithm
terra::plot(ps_regions(ps, method = "kmeans"))

# to use a hierarchical clustering method, first we have to `ps_add_dissim()`
terra::plot(ps_regions(ps_add_dissim(ps), k = 7, method = "average"))
```

---

ps_regions_eval *Evaluate region numbers*

Description

This function compares multiple potential values for k, the number of clusters in to use in `ps_regions()`, to help you decide how well different numbers of regions fit your data set. For each value of k, it performs a cluster analysis and calculates the proportion of total variance explained (SSE, the sum of squared pairwise distances explained). It also calculates second-order metrics of the relationship between k and SSE. While many data sets have no optimal value of k and the choice is often highly subjective, these evaluation metrics can help you identify potential points where the variance explained stops increasing quickly as k increases.

Usage

```
ps_regions_eval(ps, k = 1:20, plot = TRUE, ...)
```

Arguments

`ps`	A `phylospatial` object. Must contain a `dissim` component generated by ps_add_dissim.
`k`	Vector of positive integers giving possible values for `k`. Values greater than the number of sites in the data set will be ignored.
`plot`	Logical indicating whether to print a plot of the results (`TRUE`, the default) or return a data frame of the results (`FALSE`).
`...`	Further arguments passed to ps_regions.

Value

The function generates a data frame with the following columns. If `plot = FALSE` the data frame is returned, otherwise the function prints a plot of the latter variables as a function of `k`:

- "k": The number of clusters.
- "sse": The proportion of total variance explained, with variance defined as squared pairwise community phylogenetic dissimilarity between sites.
- "curvature": The local second derivative. Lower (more negative) values indicate more attractive break-point values of k.

- "dist11": The distance from the point to the 1:1 line on a plot of k vs sse in which k values over the interval from 1 to the number of sites are rescaled to the unit interval. Higher values indicate more attractive values for k.

Examples

```
ps <- ps_add_dissim(ps_simulate())
ps_regions_eval(ps, k = 1:15, plot = TRUE)
```

---

ps_rgb *Map phylospatial data onto RGB color bands*

Description

Perform an ordination that reduces a spatial phylogenetic data set into three dimensions that can be plotted as the RGB bands of color space to visualize spatial patterns of community phylogenetic composition. This function is a wrapper around ps_ordinate().

Usage

```
ps_rgb(ps, method = c("nmds", "cmds", "pca"), trans = identity, spatial = TRUE)
```

Arguments

ps	A phylospatial object with a non-null dissim component, generated by ps_add_dissim.
method	Ordination method, either "pca" (principal component analysis implemented via stats::prcomp()), "cmds" (classical MDS, implemented via stats::cmdscale()), or "nmds" (the default, nonmetric MDS, implemented via vegan::metaMDS(); this is slower but often preferred).
trans	A function giving a transformation to apply to each dimension of the ordinated data. The default is the identity function. Specifying rank generates a more uniform color distribution.
spatial	Logical indicating whether a spatial object (inherited from ps) should be returned. Default is TRUE.

Value

A matrix or spatial object with three variables containing RGB color values in the range 0-1.

Examples

```
ps <- ps_add_dissim(ps_simulate(50, 20, 20))
RGB <- ps_rgb(ps, method = "cmds")
terra::plotRGB(RGB * 255, smooth = FALSE)
```

`ps_simulate` *Simulate a toy spatial phylogenetic data set*

Description

This function generates a simple `phylospatial` object that can be used for testing other functions in the package. It is not intended to be realistic.

Usage

```
ps_simulate(
  n_tips = 10,
  n_x = 20,
  n_y = 20,
  data_type = c("probability", "binary", "abundance"),
  spatial_type = c("raster", "none"),
  seed = NULL
)
```

Arguments

`n_tips`	Number of terminals on phylogeny.
`n_x`	Number of raster cells in x dimension of landscape.
`n_y`	Number of raster cells in y dimension of landscape.
`data_type`	Community data type for simulated ranges: either "probability" (default), "binary", or "abundance".
`spatial_type`	Either "raster" or "none".
`seed`	Optional integer to seed random number generator.

Value

`phylospatial` object, comprising a random phylogeny and community matrix in which each terminal has a circular geographic range with a random radius and location. The spatial reference data is a SpatRaster.

Examples

```
# using all the defaults
ps_simulate()

# specifying some arguments
plot(ps_simulate(n_tips = 50, n_x = 30, n_y = 40, data_type = "abundance"), "comm")
```

ps_suggest_n_iter *Suggest number of iterations for null model convergence*

Description

Estimates the number of iterations needed for the null model randomization in `ps_rand()` to reach its stationary distribution, given a dataset and algorithm. This is a convenience wrapper around `nullcat::suggest_n_iter()` that extracts the appropriate community matrix from a `phylospatial` object. Use this before running `ps_rand()` with `fun = "nullcat"` or `fun = "quantize"` to choose an appropriate value for `n_iter`. The function runs multiple independent chains of the randomization algorithm, records a mixing diagnostic at each step, and identifies the point at which chains stabilize.

Usage

```
ps_suggest_n_iter(ps, fun = c("nullcat", "quantize"), plot = TRUE, ...)
```

Arguments

`ps`	A `phylospatial` object.
`fun`	Character: `"nullcat"` or `"quantize"`, matching the intended `fun` argument to `ps_rand()`.
`plot`	Logical: if `TRUE`, plot the mixing traces with the suggested burn-in marked.
`...`	Additional arguments passed to `nullcat::suggest_n_iter()` and through to `nullcat::trace_cat()`, such as `method`, `n_iter`, `n_chains`, `n_strata`, `fixed`, etc.

Value

An integer giving the suggested minimum number of iterations, with additional diagnostic information as attributes. See `nullcat::suggest_n_iter()` for details.

See Also

`ps_trace()`, `ps_rand()`

Examples

```
if (requireNamespace("nullcat", quietly = TRUE)) {
  set.seed(123)

  # binary data with nullcat
  ps_bin <- ps_simulate(data_type = "binary")
  ps_suggest_n_iter(ps_bin, fun = "nullcat", method = "curvecat",
                    n_iter = 5000, n_chains = 3, plot = TRUE)

  # quantitative data with quantize
```

```
  ps <- ps_simulate(data_type = "prob")
  ps_suggest_n_iter(ps, fun = "quantize", method = "curvecat",
                    n_strata = 4, fixed = "cell",
                    n_iter = 2000, n_chains = 3, plot = TRUE)
}
```

`ps_trace` *Trace diagnostics for null model mixing*

Description

Runs multiple independent chains of the null model randomization algorithm and records a mixing diagnostic at each step, producing trace plots to assess convergence. This is a convenience wrapper around `nullcat::trace_cat()` that extracts the appropriate community matrix from a `phylospatial` object.

Usage

```
ps_trace(ps, fun = c("nullcat", "quantize"), plot = TRUE, ...)
```

Arguments

`ps`	A `phylospatial` object.
`fun`	Character: `"nullcat"` or `"quantize"`, matching the intended `fun` argument to `ps_rand()`.
`plot`	Logical: if `TRUE`, plot the traces.
`...`	Additional arguments passed to `nullcat::trace_cat()`, such as `method`, `n_iter`, `thin`, `n_chains`, `n_strata`, `fixed`, `stat`, etc.

Value

An object of class `"cat_trace"`. See `nullcat::trace_cat()` for details.

See Also

`ps_suggest_n_iter()`, `ps_rand()`

Examples

```
if (requireNamespace("nullcat", quietly = TRUE)) {
  ps_bin <- ps_simulate(data_type = "binary")
  tr <- ps_trace(ps_bin, fun = "nullcat", method = "curvecat",
                 n_iter = 2000, n_chains = 5, plot = TRUE)

  ps <- ps_simulate(data_type = "prob")
  tr <- ps_trace(ps, fun = "quantize", method = "curvecat",
                 n_strata = 4, fixed = "cell",
```

```
                  n_iter = 2000, n_chains = 5, plot = TRUE)
}
```

`quantize` *Stratified randomization of community matrix*

Description

This is a simple wrapper around `nullcat::quantize()`, included in `phylospatial` mainly for backward compatibility.

Usage

```
quantize(x = NULL, ...)
```

Arguments

`x`	Community matrix with species in rows, sites in columns, and nonnegative quantities in cells.
`...`	Additional arguments passed to `nullcat::quantize()`.

Details

The nullcat quantize routine involves three steps: converting a quantitative matrix to categorical strata, permuting the resulting categorical matrix using one of several categorical null model algorithms, and mapping the randomized categories back to quantitative values. Supply arguments via `...` to control options for each of these stages.

Value

A randomized version of `x`.

Examples

```
if (requireNamespace("nullcat", quietly = TRUE)) {
      # example quantitative community matrix
      comm <- matrix(runif(2500), 50)

      # examples of different quantize usage
      rand <- quantize(comm)
      rand <- quantize(comm, n_strata = 4, transform = sqrt, fixed = "row")
      rand <- quantize(comm, method = "swapcat", n_iter = 500)
}
```

`to_spatial` *Convert a site-by-variable matrix into a SpatRaster or sf object*

Description

Convert a site-by-variable matrix into a SpatRaster or sf object

Usage

```
to_spatial(m, template)
```

Arguments

`m`	Matrix or vector with the same number of rows as sites in `template`. Note that `ps$comm` contains only occupied sites and must be expanded with `ps_expand()` before passing to this function.
`template`	`SpatRaster` layer with number of cells equal to the number of rows in m, or `sf` data frame with same number of rows as m.

Value

`SpatRaster` with a layer for every column in `m`, or `sf` data frame with a variable for every column in `m`, depending on the data type of `template`.

Examples

```
ps <- moss()
# ps$comm contains only occupied sites, so expand before converting:
to_spatial(ps_expand(ps, ps$comm[, 1:5]), ps$spatial)
```

Index