# Package ‘divraster’

May 8, 2026

**Title** Diversity Metrics Calculations for Rasterized Data

**Version** 1.2.3

**Maintainer** Flávio M. M. Mota <`flaviomoc@gmail.com`>

**Description** Alpha and beta diversity for taxonomic (TD), functional (FD), and phylogenetic (PD) dimensions based on rasters. Spatial and temporal beta diversity can be partitioned into replacement and richness difference components. It also calculates standardized effect size for FD and PD alpha diversity and the average individual traits across multilayer rasters. The layers of the raster represent species, while the cells represent communities. Methods details can be found at Cardoso et al. 2022 <`https://CRAN.R-project.org/package=BAT`> and Heming et al. 2023 <`https://CRAN.R-project.org/package=SESraster`>.

**License** GPL (>= 3)

**URL** `https://github.com/flaviomoc/divraster`, `https://flaviomoc.github.io/divraster/`

**BugReports** `https://github.com/flaviomoc/divraster/issues`

**Depends** R (>= 4.1.0)

**Imports** BAT, SESraster, terra, utils, dplyr, sf

**Suggests** ape, knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Flávio M. M. Mota [aut, cre, cph] (ORCID: <`https://orcid.org/0000-0002-0308-7151`>), Neander Marcel Heming [aut] (ORCID: <`https://orcid.org/0000-0003-2461-5045`>), Gabriela Alves-Ferreira [aut] (ORCID: <`https://orcid.org/0000-0001-5661-3381`>)

**Repository** CRAN

**Date/Publication** 2026-01-11 18:20:09 UTC

# Contents

---

## `area.calc` *Calculate SpatRaster Layer Areas and Overlap Areas*

### Description

Calculates the total area for each layer (e.g., species) within a `SpatRaster` object. Optionally, it can also compute the overlapping areas between the primary `SpatRaster` (`x`) and one or two additional single-layer `SpatRaster` objects (`y` and `z`). Results are returned as a `data.frame` and can optionally be saved to a CSV file.

### Usage

```
area.calc(x, y = NULL, z = NULL, filename = "", unit = "km", cellsize = NULL)
```

### Arguments

| | |
|---|---|
| `x` | A `SpatRaster` object for which the area of each layer will be calculated. This `SpatRaster` can have one or multiple layers. |
| `y` | An optional `SpatRaster` object with a **single layer**. If provided, the overlapping area between each layer in `x` and this `y` raster will be calculated. It should have the same extent and resolution as `x`. |

| | |
|---|---|
| `z` | An optional `SpatRaster` object with a **single layer**. If provided, the overlapping area between each layer in `x` and this `z` raster, as well as the three-way overlap (`x`, `y`, and `z`), will be calculated. Requires `y` to also be provided. It should have the same extent and resolution as `x`. |
| `filename` | Character string. If provided (e.g., "results.csv"), the resulting data frame will be saved to a CSV file with this name. If not provided, results are returned only to the R session. |
| `unit` | Character string specifying the unit of measurement for area calculations. Defaults to "km" (kilometers). Other options include "ha" (hectares), "m" (meters), etc. |
| `cellsize` | Numeric. An optional value specifying the cell size (area of a single cell) to be used for calculations. If `NULL` (default), the function will automatically determine the cell size from the input raster `x`. |

## Value

A `data.frame` with the following columns:

- **Layer**: Name of each layer from the input `SpatRaster x`.
- **Area**: The calculated area for each layer in `x` (e.g., total species range area).
- **Overlap_Area_Y** (optional): If `y` is provided, the area where the `x` layer and `y` raster both have a value of 1 (overlap).
- **Overlap_Area_Z** (optional): If `z` is provided, the area where the `x` layer and `z` raster both have a value of 1 (overlap).
- **Overlap_Area_All** (optional): If both `y` and `z` are provided, the area where the `x` layer, `y` raster, and `z` raster all have a value of 1 (triple overlap).

Areas are reported in the specified `unit`.

## Examples

```
library(terra)

# Load example rasters for demonstration
# Ensure these files are present in your package's inst/extdata folder
bin_rast <- terra::rast(system.file("extdata", "ref.tif", package = "divraster"))

# Example 1: Calculate area for 'bin_rast' only
area_only <- area.calc(bin_rast)
area_only
```

---

## `area.calc.flex` *Flexible Area Calculation for Raster*

### Description

This function calculates the area of integer categories in a primary raster (r1). It can optionally compute an overlay area with a second layer (r2) and/or perform calculations within distinct zones defined by a polygon SpatVector.

### Usage

```
area.calc.flex(
  r1,
  r2_raster = NULL,
  r2_vector = NULL,
  threshold = NULL,
  zonal_polys = NULL,
  id_col = NULL,
  omit_zero = TRUE,
  unit = "km"
)
```

### Arguments

| | |
|---|---|
| `r1` | The primary SpatRaster with integer categories. |
| `r2_raster` | An optional SpatRaster for overlay analysis. |
| `r2_vector` | An optional SpatVector for overlay analysis. |
| `threshold` | A numeric value required to binarize 'r2_raster' if it's continuous. |
| `zonal_polys` | An optional SpatVector for zonal analysis. |
| `id_col` | A string specifying the column in 'zonal_polys'. |
| `omit_zero` | A logical value. If TRUE (default), results for category = 0 are removed. |
| `unit` | A string specifying the area unit ("km", "m", or "ha"). |

### Value

A data frame with the area for each category.

### Examples

```
library(terra)

# 1) Primary raster (integer categories)
land_cover <- rast(ncol = 30, nrow = 30,
                   xmin = -50, xmax = -49,
                   ymin = -15, ymax = -14)
values(land_cover) <- sample(1:3, ncell(land_cover), replace = TRUE)
```

```
crs(land_cover) <- "+proj=longlat +datum=WGS84 +no_defs"

# Basic: total area by category
area.calc.flex(land_cover, unit = "km")

# 2) Zonal polygons (two regions)
region1 <- vect("POLYGON ((-50 -15, -49.5 -15, -49.5 -14, -50 -14, -50 -15))")
region2 <- vect("POLYGON ((-49.5 -15, -49 -15, -49 -14, -49.5 -14, -49.5 -15))")
regions <- rbind(region1, region2)
crs(regions) <- crs(land_cover)
regions$region_id <- c("A", "B")

area.calc.flex(
  land_cover,
  zonal_polys = regions,
  id_col = "region_id",
  unit = "km"
)

# 3) Overlay raster (binary mask)
protected <- rast(land_cover)
values(protected) <- sample(0:1, ncell(protected), replace = TRUE)

area.calc.flex(
  land_cover,
  r2_raster = protected,
  unit = "km"
)
```

---

## `area.interval` *Calculate Area by Interval Classes for SpatRasters*

### Description

This function takes a SpatRaster or list of SpatRaster objects, classifies them into intervals based on user-defined or automatically calculated min/max values, and calculates the area for each class across all rasters.

### Usage

```
area.interval(
  raster_list,
  min_value = NULL,
  max_value = NULL,
  interval,
  round = TRUE,
  include_lowest = TRUE,
  right = TRUE,
```

```
    filename = NULL,
    ...
)
```

## Arguments

| | |
|---|---|
| `raster_list` | A SpatRaster object or a list of SpatRaster objects to analyze |
| `min_value` | Numeric. Minimum value for the interval sequence. If NULL (default), automatically calculated from all input rasters |
| `max_value` | Numeric. Maximum value for the interval sequence. If NULL (default), automatically calculated from all input rasters |
| `interval` | Numeric. Interval size for the sequence (e.g., 0.1 for breaks every 0.1 units) |
| `round` | Logical. If TRUE, rounds min_value down and max_value up to the nearest interval. For example, with interval=0.1: min 0.12 becomes 0.1, max 0.98 becomes 1.0. Default TRUE |
| `include_lowest` | Logical. Should the lowest value be included in the classification? Default TRUE |
| `right` | Logical. Should intervals be closed on the right (and open on the left)? Default TRUE |
| `filename` | Character. Optional filename to save the output dataframe as CSV. If NULL (default), the dataframe is not saved |
| `...` | Additional arguments passed to the classify function |

## Value

A data.frame containing area calculations for each interval class and scenario

## Examples

```
library(terra)

r1 <- rast(ncol=10, nrow=10, vals=runif(100, 0.12, 0.98))
r2 <- rast(ncol=10, nrow=10, vals=runif(100, 0, 1))
raster_list <- list(scenario1 = r1, scenario2 = r2)
result <- area.interval(
  raster_list = raster_list,
  interval = 0.1,
  round = TRUE)
```

---

## `bin2crop` *Crop a continuous raster by a binary (0/1) raster footprint (value == 1)*

---

### Description

Crop a continuous raster by a binary (0/1) raster footprint (value == 1)

### Usage

```
bin2crop(
  r_bin,
  r_cont,
  clip = NULL,
  resample_method = "bilinear",
  dissolve = TRUE,
  filename = NULL,
  overwrite = FALSE
)
```

### Arguments

| | |
|---|---|
| `r_bin` | SpatRaster. Binary raster (0/1). Cells with value 1 define the footprint. |
| `r_cont` | SpatRaster. Continuous raster to crop/mask. |
| `clip` | Optional SpatVector. Additional polygon to crop/mask the result. |
| `resample_method` | Character. Method for resampling r_cont to r_bin grid if needed. |
| `dissolve` | Logical. Dissolve contiguous 1-cells when polygonizing. |
| `filename` | Optional character. If provided, writes result to disk. |
| `overwrite` | Logical. Passed to writeRaster if filename is provided. |

### Value

SpatRaster (cropped/masked continuous raster).

### Examples

```
library(terra)

# Create continuous raster (e.g., suitability values 0-1)
r_continuous <- rast(ncol = 50, nrow = 50, xmin = 0, xmax = 10,
                     ymin = 0, ymax = 10)
values(r_continuous) <- runif(ncell(r_continuous), 0, 1)
names(r_continuous) <- "suitability"

# Create binary raster (circular study area)
```

```
r_binary <- rast(r_continuous)
xy <- xyFromCell(r_binary, 1:ncell(r_binary))
center_dist <- sqrt((xy[,1] - 5)^2 + (xy[,2] - 5)^2)
values(r_binary) <- ifelse(center_dist <= 3, 1, 0)
names(r_binary) <- "study_area"

# Crop continuous raster to binary footprint
result <- bin2crop(r_bin = r_binary, r_cont = r_continuous)

# Plot comparison
par(mfrow = c(1, 3))
plot(r_binary, main = "Binary Footprint (Study Area)")
plot(r_continuous, main = "Original Continuous")
plot(result, main = "Cropped Result")
```

---

## `combine.rasters` *Combine GeoTIFF rasters into a multilayer SpatRaster*

### Description

Reads GeoTIFF files from a directory OR combines already-loaded SpatRasters. Computes the union of their extents, resamples them to a common grid, and returns a single multilayer SpatRaster.

### Usage

```
combine.rasters(
  raster_list = NULL,
  dir_path = NULL,
  pattern = NULL,
  method = "bilinear"
)
```

### Arguments

| | |
|---|---|
| `raster_list` | Optional list of SpatRaster objects already loaded in R. If provided, dir_path and pattern are ignored. |
| `dir_path` | Character. Directory containing input GeoTIFF files. Only used if raster_list is NULL. |
| `pattern` | Character. Pattern that file names must contain. Only used if raster_list is NULL. |
| `method` | Character. Resampling method passed to terra::resample(), e.g. "bilinear" (default) or "near" for categorical data. |

### Details

The first raster (file or list element) defines the target resolution, origin and CRS; the union of all extents defines the spatial coverage. Areas where a raster has no data are filled with NA.

## Value

A single multilayer SpatRaster with one layer per input. Layers are named from list names or file basenames without extension.

## Examples

```
library(terra)

# Create 3 separate rasters with different extents
r1 <- rast(ncol = 30, nrow = 30, xmin = 0, xmax = 10,
           ymin = 0, ymax = 10)
values(r1) <- runif(ncell(r1), 0, 100)
crs(r1) <- "+proj=longlat +datum=WGS84 +no_defs"

r2 <- rast(ncol = 30, nrow = 30, xmin = 1, xmax = 11,
           ymin = 1, ymax = 11)
values(r2) <- runif(ncell(r2), 0, 100)
crs(r2) <- crs(r1)

r3 <- rast(ncol = 30, nrow = 30, xmin = -1, xmax = 9,
           ymin = -1, ymax = 9)
values(r3) <- runif(ncell(r3), 0, 100)
crs(r3) <- crs(r1)

# Combine into single multilayer SpatRaster
raster_list <- list(baseline = r1, future_A = r2, future_B = r3)
combined <- combine.rasters(raster_list = raster_list)
combined
```

---

# `differ.rast` *Calculate Absolute or Percentage Difference Between SpatRaster Objects*

## Description

Computes the difference between two `SpatRaster` objects, either as an absolute value or as a percentage of change relative to the first raster (`r1`). This function is commonly used to assess changes in spatial patterns, such as shifts in species richness or environmental variables over time or between scenarios.

## Usage

```
differ.rast(r1, r2, perc = TRUE, filename = "")
```

## Arguments

| | |
|---|---|
| `r1` | A `SpatRaster` object representing the baseline or initial values. Can have one or multiple layers. |
| `r2` | A `SpatRaster` object representing the future or comparison values. Must have the same dimensions, resolution, CRS, and number of layers as `r1`. |
| `perc` | Logical (default is `TRUE`). If `TRUE`, the percentage of change relative to `r1` is calculated: `((r2 - r1) / r1) * 100`. If `FALSE`, the absolute difference (`r2 - r1`) is returned. |
| `filename` | Character string. Optional path and filename to save the resulting `SpatRaster`. Supported formats are those recognized by `terra::writeRaster` (e.g., ".tif", ".grd"). If provided, the `SpatRaster` will be saved to this file. |

## Details

This function performs a cell-wise subtraction (`r2 - r1`).

- For **percentage difference**, the formula used is `((r2 - r1) / r1) * 100`. Cells where `r1` is `NA` or `0` will result in `NA` in the output `SpatRaster` for percentage calculations, to avoid division by zero or meaningless percentages.
- It is crucial that `r1` and `r2` are aligned spatially (same extent, resolution, and Coordinate Reference System - CRS) and have the same number of layers, with corresponding layers representing the same variable or species.

## Value

A `SpatRaster` object containing the calculated differences.

- If `perc = TRUE`, the layer name will be "Percentage_Difference".
- If `perc = FALSE`, the layer name will be "Absolute_Difference".

The output `SpatRaster` will have the same dimensions, resolution, and CRS as the input rasters.

## Examples

```
library(terra)

# Load rasters
rich1 <- terra::rast(system.file("extdata", "rich_ref.tif",
package = "divraster"))
rich2 <- terra::rast(system.file("extdata", "rich_fut.tif",
package = "divraster"))

# Calculate absolute difference in richness
abs_diff_rast <- differ.rast(rich1, rich2, perc = FALSE)
abs_diff_rast
plot(abs_diff_rast, main = "Absolute Difference in Richness")

# Calculate percentage difference in richness
perc_diff_rast <- differ.rast(rich1, rich2, perc = TRUE)
```

```
perc_diff_rast
plot(perc_diff_rast, main = "Percentage Difference in Richness")
```

---

## `inputs_chk` *Check if objects are valid*

### Description

Check if objects are valid

### Usage

```
inputs_chk(bin1, bin2, tree)
```

### Arguments

| | |
|---|---|
| `bin1` | A SpatRaster with presence-absence data (0 or 1) for a set of species. |
| `bin2` | A SpatRaster with presence-absence data (0 or 1) for a set of species. Species names in 'bin2' and 'bin1' must match! |
| `tree` | It can be a 'data.frame' with species traits or a 'phylo' with a rooted phylogenetic tree. Species names in 'tree', 'bin1', and 'bin2' must match! |

### Value

Either a success message or an error.

---

## `load.data` *Load data adapted from Mota et al. (2025), Şekercioğlu et al. (2025), Mota et al. (2022), Tobias et al. (2022), and Jetz et al. (2014)*

### Description

Load data adapted from Mota et al. (2025), Şekercioğlu et al. (2025), Mota et al. (2022), Tobias et al. (2022), and Jetz et al. (2014)

### Usage

```
load.data()
```

### Value

A list with binary maps of species for the reference and future climate scenarios, species traits, a rooted phylogenetic tree for the species. The species names across these objects must match! It also includes a polygon of the CCAF, and the protected areas of the CCAF.

## References

Mota, F. M. M. et al. 2025. Impact of Climate Change on the Multiple Facets of Forest Bird Diversity in a Biodiversity Hotspot Within the Atlantic Forest - Diversity and Distributions 31: e70129.

Şekercioğlu, Ç. H. et al. 2025. BIRDBASE: A Global Dataset of Avian Biogeography, Conservation, Ecology and Life History Traits. - Scientific Data 12: 1558.

Mota, F. M. M. et al. 2022. Climate change is expected to restructure forest frugivorous bird communities in a biodiversity hot-point within the Atlantic Forest. - Diversity and Distributions 28: 2886–2897.

Tobias, J. A. et al. 2022. AVONET: morphological, ecological and geographical data for all birds. - Ecology Letters 25: 581–597.

Jetz, W. et al. 2014. Global Distribution and Conservation of Evolutionary Distinctness in Birds. - Current Biology 24: 919–930.

## Examples

```
data <- load.data()
data
```

---

# `occ.avg.dist` *Average pairwise distance between occurrences by species*

## Description

Compute the mean of all unique pairwise great-circle distances between occurrence records for each species, using longitude/latitude coordinates (EPSG:4326). Distances are computed with sf::st_distance().

## Usage

```
occ.avg.dist(df, species_col = "species", lon_col = "lon", lat_col = "lat")
```

## Arguments

| | |
|---|---|
| `df` | A data.frame (or tibble) containing species names and coordinates. |
| `species_col` | Character. Column name containing species names. Default "species". |
| `lon_col` | Character. Column name containing longitudes (decimal degrees). Default "lon". |
| `lat_col` | Character. Column name containing latitudes (decimal degrees). Default "lat". |

### Details

For each species, the function:

- Filters out rows with missing species or coordinates
- Converts lon/lat to an sf point object with CRS = 4326
- Builds a full distance matrix within the species
- Extracts the upper triangle (unique pairs) and averages distances

Species with fewer than 2 valid records return NA.

### Value

A data.frame with columns:

- species: unique species names
- avg_distance_m: mean pairwise distance in meters

### Examples

```
library(dplyr)
library(sf)

# Create example occurrence data for 3 species
occurrences <- tibble(
  species = c(
    rep("Species_A", 4),
    rep("Species_B", 5),
    rep("Species_C", 2)
  ),
  lon = c(
    # Species A: widespread across Brazil
    -43.2, -47.9, -38.5, -51.2,
    # Species B: clustered in southeast
    -43.9, -44.1, -43.7, -44.3, -43.8,
    # Species C: only 2 points
    -45.0, -46.0
  ),
  lat = c(
    # Species A
    -22.9, -15.8, -12.9, -30.0,
    # Species B
    -19.9, -20.1, -19.8, -20.3, -20.0,
    # Species C
    -23.5, -24.0
  )
)

# Calculate average pairwise distances
result <- occ.avg.dist(occurrences)
result
```

---

## `rast.by.polys` *Summarise raster values by polygons*

---

### Description

Extracts values from a `SpatRaster` for each polygon in a `SpatVector` and returns a data frame that combines polygon identifiers with user-defined summary statistics of the raster values.

### Usage

```
rast.by.polys(
  x,
  polygons,
  id_col = NULL,
  fun = function(v, ...) mean(v, na.rm = TRUE),
  na.rm = TRUE
)
```

### Arguments

| | |
|---|---|
| `x` | A `SpatRaster` object containing one or more layers. |
| `polygons` | A `SpatVector` with polygon geometries used to summarise raster values. |
| `id_col` | Optional character string giving the name of a column in `polygons` to be used as an identifier (for example, `"PA_ID"`). If `NULL` (default), all non-geometry attributes from `polygons` are joined to the summary table. |
| `fun` | A function applied to the vector of raster values extracted for each polygon. The function must return a named vector. It should accept `...` so that arguments such as `na.rm = TRUE` can be passed through. The default is `function(v, ...) mean(v, na.rm = TRUE)`. |
| `na.rm` | Logical; if `TRUE`, missing values are removed before applying `fun`. Passed to `fun` via `...`. |

### Details

This function is a convenience wrapper around `terra::extract()`, combining extraction, summarisation and binding of polygon attributes into a single step. It supports multilayer rasters; in that case the summary statistics are returned for each layer.

### Value

A `data.frame` with one row per polygon. If `id_col` is not `NULL`, the first column is the specified identifier; otherwise, all attribute columns from `polygons` are included. Additional columns contain the summary statistics returned by `fun` for each raster layer.

### Examples

```
library(terra)

# Example SpatRaster and SpatVector
r <- rast(system.file("ex/elev.tif", package = "terra"))
v <- as.polygons(r > 500, dissolve = TRUE)
v$PA_ID <- paste0("PA_", seq_len(nrow(v)))

# Mean elevation per polygon
pa_stats <- rast.by.polys(
  x        = r,
  polygons = v,
  id_col   = "PA_ID"
)

# Multiple statistics per polygon
pa_stats_multi <- rast.by.polys(
  x        = r,
  polygons = v,
  id_col   = "PA_ID",
  fun      = function(v, ...) c(
    mean = mean(v, ...),
    min  = min(v, ...),
    max  = max(v, ...)
  ),
  na.rm    = TRUE
)
```

---

## `spat.alpha` *Alpha calculation for raster*

---

### Description

Calculates alpha diversity for taxonomic (TD), functional (FD), and phylogenetic (PD) dimensions. Adapted from `alpha`

### Usage

```
spat.alpha(bin, tree, cores = 1, filename = "", ...)
```

### Arguments

| | |
|---|---|
| `bin` | A SpatRaster with presence-absence data (0 or 1) for a set of species. |
| `tree` | It can be a 'data.frame' with species traits or a 'phylo' with a rooted phylogenetic tree. Species names in 'tree' and 'bin' must match! |
| `cores` | A positive integer. If cores > 1, a 'parallel' package cluster with that many cores is created and used. |

| | |
|---|---|
| `filename` | Character. Save results if a name is provided. |
| `...` | Additional arguments to be passed passed down from a calling function. |

## Details

Alpha calculations use a tree-based approach for TD, FD, and PD (Cardoso et al. 2014). In the FD calculation, a species traits matrix is transformed into a distance matrix and clustered to create a regional dendrogram (i.e. a dendrogram with all species in the raster stack), from which the total branch length is calculated. When computing FD for each community (i.e. raster cell), the regional dendrogram is subsetted to create a local dendrogram that includes only the species present in the local community. The branch lengths connecting these species are then summed to represent the functional relationships of the locally present species (Petchey and Gaston, 2002, 2006). Similarly, in PD, the cumulative branch lengths connecting species within a community indicate their shared phylogenetic relationships (Faith, 1992). Alpha TD can also be visualized using a tree diagram, where each species is directly connected to the root by an edge of unit length, reflecting the number of different taxa in the community (i.e. species richness) since all taxa are at the same level (Cardoso et al. 2014).

## Value

A SpatRaster with alpha result.

## References

Cardoso, P. et al. 2014. Partitioning taxon, phylogenetic and functional beta diversity into replacement and richness difference components. - Journal of Biogeography 41: 749–761.

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

Petchey, O. L. and Gaston, K. J. 2002. Functional diversity (FD), species richness and community composition. - Ecology Letters 5: 402–411.

Rodrigues, A. S. L. and Gaston, K. J. 2002. Maximising phylogenetic diversity in the selection of networks of conservation areas. - Biological Conservation 105: 103–111.

## Examples

```
library(terra)
bin1 <- terra::rast(system.file("extdata", "ref.tif",
package = "divraster"))
traits <- read.csv(system.file("extdata", "traits.csv",
package = "divraster"), row.names = 1)
tree <- ape::read.tree(system.file("extdata", "tree.tre",
package = "divraster"))
spat.alpha(bin1)
spat.alpha(bin1, traits)
spat.alpha(bin1, tree)
```

---

## `spat.alpha.vec` *Alpha calculation for vector*

### Description

Alpha calculation for vector

### Usage

```
spat.alpha.vec(x, tree, resu, ...)
```

### Arguments

| | |
|---|---|
| `x` | A numeric vector with presence-absence data (0 or 1) for a set of species. |
| `tree` | It can be a data frame with species traits or a phylogenetic tree. |
| `resu` | Numeric. A vector to store results. |
| `...` | Additional arguments to be passed passed down from a calling function. |

### Value

A vector with alpha result.

---

## `spat.alpha2` *Alternative Method to Calculate Alpha Taxonomic Diversity*

### Description

Calculates the alpha taxonomic diversity, specifically **species richness**, for each cell in a `SpatRaster` object containing species presence-absence data. This function provides a straightforward method to sum the number of species present in each grid cell.

### Usage

```
spat.alpha2(bin, cores = 1, filename = "")
```

### Arguments

| | |
|---|---|
| `bin` | A `SpatRaster` object with multiple layers, where each layer represents a species and cell values are binary (0 for absence, 1 for presence). Species names should correspond to layer names (e.g., `names(bin)`). |
| `cores` | A positive integer (default is 1). If `cores > 1`, a parallel processing cluster is created using the `parallel` package to speed up calculations across raster cells. |
| `filename` | Character string. Optional path and filename to save the resulting `SpatRaster`. Supported formats are those recognized by `terra::writeRaster` (e.g., ".tif", ".grd"). If provided, the `SpatRaster` will be saved to this file. |

## Details

This function calculates species richness by summing the presence (value 1) of all species across layers for each individual raster cell. It is an alternative to `spat.alpha()` when only Taxonomic Diversity (TD) is required, offering a more direct and potentially faster computation for this specific metric. `NA` values in input cells are ignored during the sum calculation.

## Value

A `SpatRaster` object with a single layer named "Richness". Each cell in this `SpatRaster` contains the calculated species richness (number of species present). The output `SpatRaster` will have the same dimensions, resolution, and CRS as the input `bin`.

## Examples

```
library(terra)

# Load an example SpatRaster with binary presence-absence data
bin_rast <- terra::rast(system.file("extdata", "ref.tif", package = "divraster"))

# Calculate species richness (alpha taxonomic diversity)
richness_map <- spat.alpha2(bin_rast)
richness_map

# Plot the resulting richness map
plot(richness_map, main = "Species Richness Map")
```

---

# `spat.beta` *Spatial beta diversity for raster*

## Description

Calculates spatial beta diversity for taxonomic (TD), functional (FD), and phylogenetic (PD) dimensions. See `raster.beta`.

## Usage

```
spat.beta(x, tree, filename = "", func = "jaccard", abund = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | A SpatRaster with presence-absence data (0 or 1) for a set of species. (This maps to `layers` in `BAT::raster.beta`). |
| `tree` | It can be a 'data.frame' with species traits or a 'phylo' with a rooted phylogenetic tree. Species names in 'tree' and 'x' must match! |
| `filename` | Character. Save results if a name is provided. |
| `func` | Character. Distance function for beta diversity calculation. Defaults to "jaccard". Passed to `BAT::beta`. |

| | |
|---|---|
| abund | Logical. Whether to use abundance data (TRUE) or presence-absence (FALSE). Defaults to FALSE. Passed to `BAT::beta`. |
| ... | Additional arguments to be passed to internal functions within `BAT::raster.beta` (e.g., `BAT::beta`). Note: `BAT::raster.beta` does not accept a 'neighbour' argument. |

### Value

A SpatRaster with beta results (total, replacement, richness difference, and ratio).

### Examples

```
library(terra)
bin1 <- terra::rast(system.file("extdata", "fut.tif",
package = "divraster"))
traits <- read.csv(system.file("extdata", "traits.csv",
package = "divraster"), row.names = 1)
tree <- ape::read.tree(system.file("extdata", "tree.tre",
package = "divraster"))
spat.beta(bin1)
spat.beta(bin1, traits)
spat.beta(bin1, tree)
```

---

## `spat.rand` *Standardized Effect Size (SES) for raster*

### Description

Calculates the standardized effect size for functional and phylogenetic alpha diversity. See `bootspat_str` and `bootspat_naive`

### Usage

```
spat.rand(
  x,
  tree,
  aleats,
  random = c("site", "species", "both", "spat"),
  cores = 1,
  filename = "",
  ...
)
```

### Arguments

| | |
|---|---|
| `x` | SpatRaster. A SpatRaster containing presence-absence data (0 or 1) for a set of species. |
| `tree` | It can be a 'data.frame' with species traits or a 'phylo' with a rooted phylogenetic tree. Species names in 'tree' and 'x' must match! |
| `aleats` | positive integer. A positive integer indicating how many times the calculation should be repeated. |
| `random` | character. A character indicating the type of randomization. The currently available randomization methods are "spat", "site", "species" or "both" (site and species). |
| `cores` | positive integer. If cores > 1, a 'parallel' package cluster with that many cores is created and used. |
| `filename` | character. Output filename. |
| `...` | additional arguments to be passed passed down from a calling function. |

### Value

SpatRaster with Mean, SD, Observed, and SES.

### Examples

```
x <- terra::rast(system.file("extdata", "ref.tif",
package = "divraster"))
traits <- read.csv(system.file("extdata", "traits.csv",
package = "divraster"), row.names = 1)
tree <- ape::read.tree(system.file("extdata", "tree.tre",
package = "divraster"))
spat.rand(x, tree, 3, "site")
spat.rand(x, traits, 3, "site")
```

---

## `spat.trait` *Average trait calculation for raster*

### Description

Compute average for each trait.

### Usage

```
spat.trait(x, trait, cores = 1, filename = "", ...)
```

### Arguments

| | |
|---|---|
| `x` | A SpatRaster with presence-absence data (0 or 1) for a set of species. |
| `trait` | A 'data.frame' with species traits. Rownames must have species names that match with 'x'! |
| `cores` | A positive integer. If cores > 1, a 'parallel' package cluster with that many cores is created and used. |
| `filename` | Character. Save results if a name is provided. |
| `...` | Additional arguments to be passed passed down from a calling function. |

### Value

SpatRaster with average traits.

### Examples

```
library(terra)
bin1 <- terra::rast(system.file("extdata", "ref.tif",
package = "divraster"))
traits <- read.csv(system.file("extdata", "traits.csv",
package = "divraster"), row.names = 1)
spat.trait(bin1, traits)
```

---

## `spat.trait.vec` *Average trait calculation for vector*

### Description

Average trait calculation for vector

### Usage

```
spat.trait.vec(x, col_trait, ...)
```

### Arguments

| | |
|---|---|
| `x` | A numeric vector with presence-absence data (0 or 1) for a set of species. |
| `col_trait` | A numeric vector with trait numbers. |
| `...` | Additional arguments to be passed passed down from a calling function. |

### Value

Vector of average trait.

---

`suit.change` *Species Suitability Change Between Climate Scenarios*

---

## Description

Compares two `SpatRaster` objects, each containing species presence-absence data for multiple species under different climate scenarios (e.g., baseline vs. future). It calculates and encodes the change in habitat suitability (gain, loss, unchanged, unsuitable) for each species in each raster cell.

## Usage

```
suit.change(r1, r2, filename = "")
```

## Arguments

| | |
|---|---|
| `r1` | A `SpatRaster` with multiple layers. Each layer represents a species' presence-absence data (0 for absence, 1 for presence) for the **baseline climate scenario**. Layer names should correspond to species names. |
| `r2` | A `SpatRaster` with multiple layers. Each layer represents a species' presence-absence data (0 for absence, 1 for presence) for the **future climate scenario**. Layer names should correspond to species names and must match those in `r1`. |
| `filename` | Character string. Optional path and filename to save the resulting `SpatRaster` stack. Supported formats are those recognized by `terra::writeRaster` (e.g., ".tif", ".grd"). If provided, the `SpatRaster` will be saved to this file. |

## Details

This function processes each species layer independently. It's crucial that both input `SpatRasters` (`r1` and `r2`) have the same extent, resolution, and the same number of layers, with corresponding layers representing the same species. The function expects binary (0 or 1) presence-absence data.

## Value

A `SpatRaster` object with multiple layers, where each layer corresponds to a species from the input SpatRasters. Cell values are encoded as follows: 1 = Gain: Species absent in r1 (baseline) becomes present in r2 (future). 2 = Loss: Species present in r1 (baseline) becomes absent in r2 (future). 3 = Unchanged (Presence): Species present in both r1 and r2. 4 = Unsuitable (Both): Species absent in both r1 and r2. The dimensions, resolution, and layer names of the output raster will match those of the input r1 and r2.

## Examples

```
library(terra)

# Load example rasters for baseline and future climate scenarios
r1 <- terra::rast(system.file("extdata", "ref.tif", package = "divraster"))
r2 <- terra::rast(system.file("extdata", "fut.tif", package = "divraster"))
```

```
# Calculate suitability change
change_map <- suit.change(r1, r2)
change_map
```

---

`temp.beta` *Temporal beta diversity calculation for raster*

---

## Description

Calculates temporal beta diversity for taxonomic (TD), functional (FD), and phylogenetic (PD) dimensions. Adapted from `beta`

## Usage

```
temp.beta(bin1, bin2, tree, filename = "", cores = 1, ...)
```

## Arguments

| | |
|---|---|
| `bin1` | A SpatRaster with presence-absence data (0 or 1) for a set of species. |
| `bin2` | A SpatRaster with presence-absence data (0 or 1) for a set of species. Species names in 'bin2' and 'bin1' must match! |
| `tree` | It can be a 'data.frame' with species traits or a 'phylo' with a rooted phylogenetic tree. Species names in 'tree', 'bin1', and 'bin2' must match! |
| `filename` | Character. Save results if a name is provided. |
| `cores` | A positive integer. If cores > 1, a 'parallel' package cluster with that many cores is created and used. |
| `...` | Additional arguments to be passed passed down from a calling function. |

## Details

The TD beta diversity partitioning framework we used was developed by Podani and Schmera (2011) and Carvalho et al. (2012) and expanded to PD and FD by Cardoso et al. (2014).

## Value

A SpatRaster with beta results (total, replacement, richness difference, and ratio).

## References

Cardoso, P. et al. 2014. Partitioning taxon, phylogenetic and functional beta diversity into replacement and richness difference components. - Journal of Biogeography 41: 749–761.

Carvalho, J. C. et al. 2012. Determining the relative roles of species replacement and species richness differences in generating beta-diversity patterns. - Global Ecology and Biogeography 21: 760–771.

Podani, J. and Schmera, D. 2011. A new conceptual and methodological framework for exploring and explaining pattern in presence - absence data. - Oikos 120: 1625–1638.

Hidasi-Neto, J. et al. 2019. Climate change will drive mammal species loss and biotic homogenization in the Cerrado Biodiversity Hotspot. - Perspectives in Ecology and Conservation 17: 57–63.

### Examples

```
library(terra)
bin1 <- terra::rast(system.file("extdata", "ref.tif",
package = "divraster"))
bin2 <- terra::rast(system.file("extdata", "fut.tif",
package = "divraster"))
traits <- read.csv(system.file("extdata", "traits.csv",
package = "divraster"), row.names = 1)
tree <- ape::read.tree(system.file("extdata", "tree.tre",
package = "divraster"))
temp.beta(bin1, bin2)
temp.beta(bin1, bin2, traits)
temp.beta(bin1, bin2, tree)
```

---

## `temp.beta.vec` *Temporal beta diversity calculation for vector*

### Description

Temporal beta diversity calculation for vector

### Usage

```
temp.beta.vec(x, nspp, spp, tree, resu, ...)
```

### Arguments

| | |
|---|---|
| `x` | A numeric vector with presence-absence data (0 or 1) for a set of species. |
| `nspp` | Numeric. Number of species. |
| `spp` | Character. Species name. |
| `tree` | It can be a data frame with species traits or a phylogenetic tree. |
| `resu` | Numeric. A vector to store results. |
| `...` | Additional arguments to be passed passed down from a calling function. |

### Value

A vector with beta results (total, replacement, richness difference, and ratio).

# Index