Package ‘prior3D’

July 23, 2025

Type Package

Title 3D Prioritization Algorithm

Version 0.1.5

Maintainer Christos Adam <`econp266@econ.soc.uoc.gr`>

Description Three-dimensional systematic conservation planning, conducting nested prioritization analyses across multiple depth levels and ensuring efficient resource allocation throughout the water column. It provides a structured workflow designed to address biodiversity conservation and management challenges in the 3 dimensions, while facilitating users’ choices and parameterization (Doxa et al. 2025 <doi:10.1016/j.ecolmodel.2024.110919>).

License GPL-3

Encoding UTF-8

URL `https://github.com/cadam00/prior3D`,
`https://cadam00.github.io/prior3D/`

BugReports `https://github.com/cadam00/prior3D/issues`

LazyData true

Imports prioritizr (>= 8.0.4), terra, maps (>= 3.4.2), highs, viridis (>= 0.6.5), readxl (>= 1.4.3), rasterdiv (>= 0.3.4), geodiv (>= 1.1.0), methods, stats, utils, graphics, grDevices

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr, rmarkdown

Depends R (>= 4.1.0)

NeedsCompilation no

Author Aggeliki Doxa [aut] (ORCID: <`https://orcid.org/0000-0003-4279-1499`>),
Christos Adam [aut, cre] (ORCID:
<`https://orcid.org/0009-0003-3244-7034`>),
Nikolaos Nagkoulis [aut] (ORCID:

<https://orcid.org/0000-0002-1900-2634>),
Antonios D. Mazaris [aut] (ORCID:
<https://orcid.org/0000-0002-4961-5490>),
Stelios Katsanevakis [aut] (ORCID:
<https://orcid.org/0000-0002-5137-7540>)

Repository CRAN

Date/Publication 2025-01-25 08:10:01 UTC

Contents

`biodiv_df` *Example biodiversity data.frame*

Description

Example biodiversity data.frame

Usage

```
data(biodiv_df)
```

Details

Example of input `biodiv_df` used for functions.

Value

data.frame object with information about features.

Examples

```
data(biodiv_df)
head(biodiv_df)
```

`coherence` *Coherence metrics*

Description

Coherence metrics

Usage

```
coherence(x, w, metric = "sa", normalize = TRUE, plot = TRUE, addlines = TRUE, ...)
```

Arguments

`x`	Output of **Compare_2D_3D** function.
`w`	Integer indicating square window dimensions.
`metric`	Cohension metric to use. It must be one of `"sa"`, `"sku"` and `"rao"`
`normalize`	If `TRUE`, then sum of solutions is normalized at a $[0, 1]$ scale.
`plot`	If `TRUE`, then coherence maps are ploted.
`addlines`	If `TRUE`, then border lines from `maps::map` are ploted.
`...`	Further arguments passed in function, based on metric choice. See more in Details.

Details

`"sa"` and `"sku"` are derived from `geodiv::focal_metrics` and in ellipsis (`...`) further arguments of `geodiv::focal_metrics` are passed.

`metric = "rao"` is derived from `rasterdiv::paRao` and in ellipsis (`...`) further arguments of `rasterdiv::paRao` are passed.

Value

`numeric` vector with 2D and 3D metrics. The result maps are optionally ploted.

References

Rocchini, Duccio, Matteo Marcantonio, Daniele Da Re, Giovanni Bacaro, Enrico Feoli, Giles Foody, Reinhard Furrer, et al. 2021. "From zero to infinity: Minimum to maximum diversity of the planet by spatio-parametric Rao's quadratic entropy." Global Ecology and Biogeography 30 (5): 2315. doi:10.1111/geb.13270.

Rocchini, Duccio, Elisa Thouverai, Matteo Marcantonio, Martina Iannacito, Daniele Da Re, Michele Torresani, Giovanni Bacaro, et al. 2021. "rasterdiv - An Information Theory tailored R package for measuring ecosystem heterogeneity from space: To the origin and back." Methods in Ecology and Evolution 12 (6): 2195. doi:10.1111/2041210X.13583.

Smith, Annie C., Phoebe Zarnetske, Kyla Dahlin, Adam Wilson, and Andrew Latimer. 2023. Geodiv: Methods for Calculating Gradient Surface Metrics. https://CRAN.R-project.org/package=geodiv.

Becker OScbRA, Minka ARWRvbRBEbTP, Deckmyn. A (2023). maps: Draw Geographical Maps. R package version 3.4.2, https://CRAN.R-project.org/package=maps

See Also

`Compare_2D_3D`

Examples

```
## Not run:
## This example requires commercial solver from 'gurobi' package for
## portfolio = "gap". Else replace it with e.g. portfolio = "shuffle" for using
## a free solver like the one from 'highs' package.

biodiv_raster <- get_biodiv_raster()
depth_raster <- get_depth_raster()
data(biodiv_df)

out_2D_3D <- Compare_2D_3D(biodiv_raster = biodiv_raster,
                           depth_raster = depth_raster,
                           breaks = c(0, -40, -200, -2000, -Inf),
                           biodiv_df = biodiv_df,
                           budget_percents = seq(0, 1, 0.1),
                           budget_weights = "richness",
                           threads = parallel::detectCores(),
                           portfolio = "gap",
                           portfolio_opts = list(number_solutions = 10))

coherence(out_2D_3D, w = 3, metric = "sa")
coherence(out_2D_3D, w = 3, metric = "sku")
coherence(out_2D_3D, w = 3, metric = "rao")

## End(Not run)
```

`Compare_2D_3D` *Compare 2D vs 3D prioritization algorithms*

Description

Compare 2D vs 3D prioritization algorithms

Usage

```
Compare_2D_3D(biodiv_raster, depth_raster, breaks, biodiv_df, val_depth_range = TRUE,
priority_weights = NULL, budget_percents = seq(0,1,0.1), budget_weights = "equal",
penalty = 0, edge_factor = 0.5, gap = 0.1, threads = 1L, sep_priority_weights = ",",
portfolio = "gap", portfolio_opts = list(number_solutions = 10, pool_gap = 0.1),
sep_biodiv_df = ",", locked_in_raster = NULL, locked_out_raster = NULL, verbose = FALSE)
```

Arguments

`biodiv_raster`	SpatRaster object or folder path with 2D feature distributions as layers.
`depth_raster`	SpatRaster object or file path with elevation/bathymetric map.
`breaks`	Numeric vector defining the range of depth layers to use.
`biodiv_df`	`data.frame` or a file path (CSV, TXT, XLS, or XLSX) containing additional information about biodiversity features.
`val_depth_range`	No correction of the splitted 3D distributions based on depth range of the biodiversity features (`"min_z"` and `"max_z"` from `biodiv_df`) is needed.
`priority_weights`	`data.frame` object or file path (CSV, TXT, XLS, or XLSX) containing group names of biodiversity features in the first column and corresponding group weights in the second column. This data.frame attributes distinct prioritization weights to different biodiversity features or groups of features.
`budget_percents`	Numeric value $[0, 1]$ or vector containing budget percentages to use. The default is `seq(0,1,0.1)`.
`budget_weights`	Numeric weight vector for budget_percents allocation among depth levels. Otherwise it can be a string with one of the choices `"equal"`, `"area"` or `"richness"`. Alternatively, it can be a numerical vector with custom weights corresponding to each depth layer, where the first value corresponds to the surface and last one corresponds to the bottom of the sea. The weights are normalized if their sum exceeds 1. If not specified, an equal distribution of budget among depth levels is used, as the default.
`penalty`	Numeric penalty applied to each depth zone, as defined in the `prioritizr::add_boundary_penalties`.
`edge_factor`	Numeric edge factor applied to each depth zone, as defined in the `prioritizr::add_boundary_penalties`.
`gap`	The optimality gap for the solver, as defined in the **prioritizr** package. The default gap is 0.1.
`threads`	The number of solver threads to be used. The default is 1.
`sep_priority_weights`	Separator used in priority_weights file, if priority_weights is in path format.
`portfolio`	The portfolio to be used, choosing between `"extra"`, `"gap"`, `"cuts"` and `"shuffle"` portfolios. The default is `"gap"`. `portfolio=""` indicates that no portfolio is used. For more about `portfolios` see **prioritizr**.
`portfolio_opts`	The **prioritizr** portfolio options to be used.

`sep_biodiv_df`	Separator used in biodiv_df file, if biodiv_df is in path format.
`locked_in_raster`	An optional `locked_in_raster` SpatRaster to be used. Note that these areas are considered as zero-cost.
`locked_out_raster`	An optional `locked_out_raster` SpatRaster to be used. Note that these areas are excluded from the solution.
`verbose`	If `verbose = TRUE`, then solver messages are printed as well. The default is `FALSE`.

Details

To facilitate comparisons between 3D and 2D approaches, the `compare_2D_3D()` function is provided in the package. This function enables users to conduct all steps of the analysis (data generation, setting and solving the optimization problem and producing outputs), by executing both 2D and 3D approaches, with similar settings, that facilitate comparisons. The function generates corresponding maps and graphs for both approaches.

The `split_rast` function is used to convert 2D distributions of biodiversity features (rasters) into a 3D format.

Here the `biodiv_df` can have the following column names (independently of their order and any other names are ignored):

- `"species_name"`: **Mandatory** column with the feature names, which must be the same with biodiv_raster.
- `"pelagic"`: **Mandatory** column about the features' behaviour. `TRUE` means that this feature is pelagic and `FALSE` means that this feature is benthic.
- `"min_z"`: **Optional** column about the minimum vertical range of features. `NA` values are translated as unlimited upward feature movement.
- `"max_z"`: **Optional** column about the maximum vertical range of features. `NA` values are translated as unlimited downward feature movement.
- `"group"`: **Optional** column with the group weights names.

Except from `biodiv_df`, an additional `data.frame` object can also be used for defining group weights, named `priority_weights`. If used, this `data.frame` object must have two columns:

- `"group"`: **Mandatory** column with the group weights names.
- `"weight"`: **Mandatory** column with the group weights.

In case that no feature weights are desired, then `priority_weights` can be kept to `NULL`.

`breaks` must be in correspondence to depth_raster file. For example, if depth_raster has range $[10, -3000]$, then a breaks vector of `c(0,-40,-200,-2000,-Inf)` will create depth levels $[0, -40]$, $(-40, -200]$, $(-200, -2000]$, $(-2000, -\infty)$ and set to NA cells with values greater than 0.

If `val_depth_range = TRUE` (default), then no correction is done and the depth range of the biodiversity features is derived from the corresponding feature distribution raster and so `"min_z"` and `"max_z"` are ignored. If `val_depth_range = FALSE`, then the function uses the minimum and maximum depth information provided in the biodiv_df, so as to remove feature occurrences outside their expected range.

`budget_percents`: Budget reflects the desired level of protection to be modeled. It ranges from 0 to 1, with 0 indicating no resources available for protection, while 1 signifies resources sufficient to protect the entire study area. Typically, setting a budget of 0.3 corresponds to the 30% conservation target (i.e. 30% of the total area set aside for conservation). Users also have the flexibility to define multiple budget levels using a vector, allowing for the exploration of various protection scenarios. For instance, a vector like `c(0.1, 0.3, 0.5)` represents three scenarios where 10%, 30%, and 50% of the study area are designated for protection.

`budget_weights`: The **Compare_2D_3D** function allows users to specify how the budget is distributed among depth levels. Three allocation methods are available:

1. Equal Distribution: Allocates an equal share of the budget to each depth level (`budget_weights = "equal"`).
2. Proportional to Area: Allocates budget based on the spatial extent of each depth level (`budget_weights = "area"`).
3. Proportional to Species Richness: Prioritizes budget allocation to depth levels with higher species diversity (number of species). (`budget_weights = "richness"`)

Otherwise, it can be a numeric vector with length equal to the number of depth levels, where each number indicates the budget share per depth level.

The solver used for solving the prioritization problems is the best available on the computer, following the solver hierarchy of **prioritizr**.

Value

A list containing the following objects (non-referenced are identical to the input ones):

- split_features: output of `split_rast`
- solution3D: list with 3D solution per budget percentage
- absolute_held3D: `absolute_held` for 3D solutions (see `evaluate_3D`)
- overall_available3D: `overall_available` for 3D solutions (see `evaluate_3D`)
- overall_held3D: `overall_held` for 3D solutions (see `evaluate_3D`)
- relative_helds3D: `relative_held` for 3D solutions (see `evaluate_3D`)
- mean_overall_helds3D: `base::mean` of `overall_held` for 3D solution (see `evaluate_3D`) per budget
- sd_overall_helds3D: `stats::sd` of `overall_held` for 3D solution (see `evaluate_3D`) per budget
- depth_overall_available3D: `depth_overall_available` for 3D solutions (see `evaluate_3D`)
- solution2D: list with 2D solution per budget percentage
- absolute_held2D: `absolute_held` for 2D solutions (see `evaluate_3D`)
- overall_available2D: `overall_available` for 2D solutions (see `evaluate_3D`)
- overall_held2D: `overall_held` for 2D solutions (see `evaluate_3D`)

- relative_helds2D: `relative_held` for 2D solutions (see `evaluate_3D`)
- mean_overall_helds2D: `base::mean` of `overall_held` for 2D solution (see `evaluate_3D`) per budget
- sd_overall_helds2D: `stats::sd` of `overall_held` for 2D solution (see `evaluate_3D`) per budget
- depth_overall_available2D: `depth_overall_available` for 2D solutions (see `evaluate_3D`)
- names_features: names of features used
- total_amount: `total_amount` of features used (see `evaluate_3D`)
- overall_total_amount: `overal_total_amount` of names of features used (see `evaluate_3D`)
- jaccard_coef: `terra_jaccard` per pair of 2D and 3D solutions, given each budget
- depth_levels_names: Depth levels names
- biodiv_raster: `biodiv_raster` used, after cleaning
- biodiv_df: `biodiv_df` used after cleaning

References

Hanson, Jeffrey O, Richard Schuster, Nina Morrell, Matthew Strimas-Mackey, Brandon P M Edwards, Matthew E Watts, Peter Arcese, Joseph Bennett, and Hugh P Possingham. 2024. prioritizr: Systematic Conservation Prioritization in R. https://prioritizr.net.

Lehtomäki, Joona (2016). Comparing prioritization methods, 21 June.
Available at: https://rpubs.com/jlehtoma/priocomp (Accessed 1 June 2024).

See Also

`evaluate_3D`, `terra_jaccard`, `split_rast`, `plot_Compare_2D_3D`

Examples

```
## Not run:
## This example requires commercial solver from 'gurobi' package for
## portfolio = "gap". Else replace it with e.g. portfolio = "shuffle" for using
## a free solver like the one from 'highs' package.

biodiv_raster <- get_biodiv_raster()
depth_raster <- get_depth_raster()
data(biodiv_df)

out_2D_3D <- Compare_2D_3D(biodiv_raster = biodiv_raster,
                           depth_raster = depth_raster,
                           breaks = c(0, -40, -200, -2000, -Inf),
                           biodiv_df = biodiv_df,
                           budget_percents = seq(0, 1, 0.1),
                           budget_weights = "richness",
                           threads = parallel::detectCores(),
                           portfolio = "gap",
                           portfolio_opts = list(number_solutions = 10))
```

```
plot_Compare_2D_3D(out_2D_3D, to_plot = "all", add_lines=TRUE)

# Arbitrary random weights
priority_weights <- data.frame(c("A", "B", "C"), c(0.001, 1000, 1))
names(priority_weights) <- c("group", "weight")
biodiv_df$group <- rep(c("A", "B", "C"), length.out=20)
out_2D_3D <- Compare_2D_3D(biodiv_raster = biodiv_raster,
                           depth_raster = depth_raster,
                           breaks = c(0, -40, -200, -2000, -Inf),
                           biodiv_df = biodiv_df,
                           priority_weights = priority_weights,
                           budget_percents = seq(0, 1, 0.1),
                           budget_weights = "richness",
                           threads = parallel::detectCores(),
                           portfolio = "gap",
                           portfolio_opts = list(number_solutions = 10))

plot_Compare_2D_3D(out_2D_3D, to_plot = "all", add_lines=TRUE)

## End(Not run)
```

`evaluate_3D` *Evaluate prioritization solution over 3D feature distributions*

Description

Evaluate prioritization solution over 3D feature distributions.

Usage

```
evaluate_3D(solution, split_features)
```

Arguments

`solution`	prioritization solution SpatRaster object.
`split_features`	A list of SpatRaster objects representing depth zones, where each element corresponds to a different depth level, ranging from surface to the bottom of the sea. The function uses the output of the split_rast function, but other multilevel (3D) distribution data that conform to this structure is also acceptable. First list element corresponds to the shallowest distribution and the last list element to the deepest one.

Details

This function evaluates a prioritization solution over 3D feature distributions, by estimating the relative and overall percentages of features under protection, as designated by the prioritization solution.

Value

A list containing the following evaluation metrics:

- relative_held_raw: relative percentage under protection per feature and per depth level
- relative_held: mean percentage under protection of all features per depth level
- overall_held: overall percentage under protection per feature
- overall_available: relative percentage under protection per feature and per depth level, over total amount of each feature
- depth_overall_available: averages of overall_available per depth
- absolute_held: absolute value per feature and per depth under protection
- total_amount: absolute value per feature and depth level

Examples

```
## Not run:
## This example requires commercial solver from 'gurobi' package for
## portfolio = "gap". Else replace it with e.g. portfolio = "shuffle" for using
## a free solver like the one from 'highs' package.

biodiv_raster <- get_biodiv_raster()
depth_raster <- get_depth_raster()
data(biodiv_df)

# You can split features' 2D distributions into 3D ones and then run only 3D analysis
split_features <- split_rast(biodiv_raster,
                             depth_raster,
                             breaks = c(0, -40, -200, -2000, -Inf),
                             biodiv_df)

out_3D <- prioritize_3D(split_features = split_features,
                        depth_raster = depth_raster,
                        breaks = c(0, -40, -200, -2000, -Inf),
                        biodiv_df = biodiv_df,
                        budget_percents = seq(0, 1, 0.1),
                        budget_weights = "richness",
                        threads = parallel::detectCores(),
                        portfolio = "gap",
                        portfolio_opts = list(number_solutions = 10))

evaluate_3D(out_3D$solution3D$budget0.3, split_features)

## End(Not run)
```

`get_biodiv_raster` *Example biodiversity raster*

Description

Example biodiversity raster

Usage

```
get_biodiv_raster()
```

Details

Example of input `biodiv_raster` used for functions.

Value

SpatRaster object with distribution of features.

References

Kaschner, K., Kesner-Reyes, K., Garilao, C., Segschneider, J., Rius-Barile, J., Rees, T., & Froese, R. (2019). AquaMaps: Predicted range maps for aquatic species. https://www.aquamaps.org

Examples

```
biodiv_raster <- get_biodiv_raster()
terra::plot(biodiv_raster[[1:4]])
```

`get_depth_raster` *Example depth raster*

Description

Example depth raster

Usage

```
get_depth_raster()
```

Details

Example of input `depth_df` object for functions.

Value

SpatRaster object with depth levels for Mediterranean.

References

GEBCO Compilation Group. (2021). GEBCO 2021 Grid. doi:10.5285/c6612cbe50b30cffe053-6c86abc09f8f.

Examples

```
depth_raster <- get_depth_raster()
terra::plot(depth_raster)
```

`get_rast` *Read multiple rast files*

Description

Read multiple rast files contained in a folder path. Raster files must have either .asc or .tif extension.

Usage

```
get_rast(path)
```

Arguments

`path`	Path string of folder containing rast files.

Value

A SpatRaster object.

Examples

```
feature_folder <- system.file("get_rast_example", package="prior3D")
get_rast(feature_folder)
```

`plot_3D` *Plot output of prioritize_3D*

Description

Plot summarized output of prioritize_3D

Usage

```
plot_3D(x, to_plot = "all", add_lines = TRUE)
```

Arguments

`x`	Output of prioritize_3D.
`to_plot`	Any of `"maps"`, `"relative_held"` or `"all"`. The default is `"all"`. See more in Details.
`add_lines`	If `TRUE`, then border lines from **maps::map** are ploted as well.

Details

This function plots the summarized output of prioritize_3D for all the selected budgets. The produced plot can contain information about:

- `"maps"`: produced maps normalized at a $[0, 1]$ scale.
- `"relative_held"`: percentage of protection for all features per depth level.
- `"all"`: both `"maps"` and `"relative_held"`.

Value

A plot.

References

Becker, R.A., Wilks, A.R., Brownrigg, R., & Minka, T.P. (2023). maps: Draw Geographical Maps. R package version 3.4.2, https://CRAN.R-project.org/package=maps

See Also

`prioritize_3D`

Examples

```
## Not run:
## This example requires commercial solver from 'gurobi' package for
## portfolio = "gap". Else replace it with e.g. portfolio = "shuffle" for using
## a free solver like the one from 'highs' package.

biodiv_raster <- get_biodiv_raster()
depth_raster <- get_depth_raster()
data(biodiv_df)

# You can split features' 2D distributions into 3D ones and then run only 3D analysis
split_features <- split_rast(biodiv_raster,
                             depth_raster,
                             breaks = c(0, -40, -200, -2000, -Inf),
                             biodiv_df,
                             val_depth_range=TRUE)

out_3D <- prioritize_3D(split_features = split_features,
                        depth_raster = depth_raster,
                        breaks = c(0, -40, -200, -2000, -Inf),
                        biodiv_df = biodiv_df,
```

```
                           priority_weights = NULL,#priority_weights,
                           budget_percents = seq(0, 1, 0.1),
                           budget_weights = "equal",
                           penalty = 0,
                           edge_factor = 0.5,
                           gap = 0.1,
                           threads = parallel::detectCores(),
                           sep_priority_weights = ",",
                           portfolio = "gap",
                           portfolio_opts = list(number_solutions = 10),
                           sep_biodiv_df = ",",
                           locked_in_raster = NULL,
                           locked_out_raster = NULL)

plot_3D(out_3D, to_plot="all", add_lines=FALSE)
plot_3D(out_3D, to_plot="all", add_lines=TRUE)
plot_3D(out_3D, to_plot="maps", add_lines=TRUE)
plot_3D(out_3D, to_plot="relative_held", add_lines=TRUE)

## End(Not run)
```

`plot_Compare_2D_3D` *Plot output of Compare_2D_3D*

Description

Plot summarized output of Compare_2D_3D

Usage

```
plot_Compare_2D_3D(x, to_plot = "all", add_lines = TRUE)
```

Arguments

`x`	Output of Compare_2D_3D.
`to_plot`	Any of `"maps"`, `"relative_held"` or `"all"`. The default is `"all"`. See more in Details.
`add_lines`	If `TRUE`, then border lines from **maps::map** are ploted as well.

Details

This function plots the summarized output of Compare_2D_3D for all selected budgets. The produced plot can contain information about:

- `"maps"`: produced maps normalized at a $[0, 1]$ scale.
- `"relative_held"`: percentage of protection for all features per depth level.
- `"all"`: both `"maps"` and `"relative_held"`.

Value

A plot.

References

Becker, R. A., Wilks, A. R., Brownrigg, R., & Minka, T. P. (2023). maps: Draw Geographical Maps. R package version 3.4.2, https://CRAN.R-project.org/package=maps

See Also

`Compare_2D_3D`

Examples

```
## Not run:
## This example requires commercial solver from 'gurobi' package for
## portfolio = "gap". Else replace it with e.g. portfolio = "shuffle" for using
## a free solver like the one from 'highs' package.

biodiv_raster <- get_biodiv_raster()
depth_raster <- get_depth_raster()
data(biodiv_df)

out_2D_3D <- Compare_2D_3D(biodiv_raster = biodiv_raster,
                           depth_raster = depth_raster,
                           breaks = c(0, -40, -200, -2000, -Inf),
                           biodiv_df = biodiv_df,
                           budget_percents = seq(0, 1, 0.1),
                           budget_weights = "richness",
                           threads = parallel::detectCores(),
                           portfolio = "gap",
                           portfolio_opts = list(number_solutions = 10))

plot_Compare_2D_3D(out_2D_3D, to_plot="all", add_lines=FALSE)
plot_Compare_2D_3D(out_2D_3D, to_plot="all", add_lines=TRUE)
plot_Compare_2D_3D(out_2D_3D, to_plot="maps", add_lines=TRUE)
plot_Compare_2D_3D(out_2D_3D, to_plot="relative_held", add_lines=TRUE)

## End(Not run)
```

`plot_sumrast` *Plot sum list of SpatRaster objects.*

Description

Plot sum list of SpatRaster objects.

Usage

```
plot_sumrast(x, normalize = TRUE, add_lines = TRUE, ...)
```

Arguments

`x`	List of SpatRaster objects.
`normalize`	If `TRUE`, then sum of solutions is normalized at a $[0, 1]$ scale.
`add_lines`	If `TRUE`, then border lines from **maps::map** are ploted as well.
`...`	Further arguments passed to **terra::plot**

Value

A plot.

See Also

`sumrast`

Examples

```
set.seed(42)
x <- terra::rast(matrix(rbinom(100, 1, 0.2), nrow=10))
y <- terra::rast(matrix(rbinom(100, 1, 0.8), nrow=10))
plot_sumrast(list(x, y), add_lines = FALSE)
```

`prioritize_3D` *3D prioritization algorithm*

Description

3D prioritization algorithm

Usage

```
prioritize_3D(split_features, depth_raster, breaks, biodiv_df,
priority_weights = NULL, budget_percents = seq(0,1,0.1), budget_weights = "equal",
penalty = 0, edge_factor = 0.5, gap = 0.1, threads = 1L, sep_priority_weights = ",",
portfolio = "gap", portfolio_opts = list(number_solutions = 10, pool_gap = 0.1),
sep_biodiv_df = ",", locked_in_raster = NULL, locked_out_raster = NULL, verbose = FALSE)
```

Arguments

`split_features`	`list` of SpatRaster objects representing depth zones, where each element corresponds to a different depth level, ranging from surface to the bottom of the sea. The function uses the output of the split_rast function, but other multilevel (3D) distribution data that conform to this structure is also acceptable. First list element corresponds to the shallowest distribution and the last list element to the deepest one.
`depth_raster`	SpatRaster object or file path with elevation/bathymetric map.
`breaks`	Numeric vector defining the range of depth layers to use.

`biodiv_df`	`data.frame` or a file path (CSV, TXT, XLS, or XLSX) containing additional information about biodiversity features.
`priority_weights`	`data.frame` object or file path (CSV, TXT, XLS, or XLSX) containing group names of biodiversity features in the first column and corresponding group weights in the second column. This data.frame attributes distinct prioritization weights to different biodiversity features or groups of features.
`budget_percents`	Numeric value $[0, 1]$ or vector containing budget percentages to use. The default is `seq(0,1,0.1)`.
`budget_weights`	Numeric weight vector for budget_percents allocation among depth levels. Otherwise it can be a string with one of the choices `"equal"`, `"area"` or `"richness"`. Alternatively, it can be a numerical vector with custom weights corresponding to each depth layer, where the first value corresponds to the surface and last one corresponds to the bottom of the sea. The weights are normalized if their sum exceeds 1. If not specified, an equal distribution of budget among depth levels is used, as the default.
`penalty`	A single numeric penalty applied to each depth zone, as defined in the `prioritizr::add_boundary_penalties`.
`edge_factor`	A single numeric edge factor applied to each depth zone, as defined in the `prioritizr::add_boundary_penalties`.
`gap`	The optimality gap for the solver, as defined in the **prioritizr** package. The default gap is 0.1.
`threads`	The number of solver threads to be used. The default is 1.
`sep_priority_weights`	Separator used in priority_weights file, if priority_weights is in path format.
`portfolio`	The portfolio to be used, choosing between `"extra"`, `"gap"`, `"cuts"` and `"shuffle"` portfolios. The default is `"gap"`. `portfolio=""` indicates that no portfolio is used. For more about `portfolios` see **prioritizr**.
`portfolio_opts`	The **prioritizr** portfolio options to be used.
`sep_biodiv_df`	Separator used in biodiv_df file, if biodiv_df is in path format.
`locked_in_raster`	An optional `locked_in_raster` SpatRaster to be used. Note that these areas are considered as zero-cost.
`locked_out_raster`	An optional `locked_out_raster` SpatRaster to be used. Note that these areas are excluded from the solution.
`verbose`	If `verbose = TRUE`, then solver messages are printed as well. The default is `FALSE`.

Details

This function is used to generate prioritization maps. Single budget settings (ex. `total_budget = 0.3`) produce standard maps, as typical Marxan outputs. Multiple budgets, by using a vector (ex. `c(0.1,0.3,0.5)`, result in cumulative (frequency) maps, illustrating areas selected by various budget levels. Although this output follows a different approach, it resembles to typical Zonation output maps.

The main reason for `biodiv_df` here is defining prioritization weights for features. In this package weights are defined per group of features (if needed). `biodiv_df` has `mandatory` column names (and any other names are ignored):

- `"species_name"`: **Mandatory** column with the feature names, which must be the same with `split_rast`.
- `"group"`: **Mandatory** column with the group weights names.

Except from `biodiv_df`, an additional `data.frame` object can also be used, named `priority_weights`. If used, this `data.frame` object must have two columns:

- `"group"`: **Mandatory** column with the group weights names.
- `"weight"`: **Mandatory** column with the group weights.

In case that no feature weights are desired, then `biodiv_df` and `priority_weights` can be both kept to `NULL`.

`breaks` must be in correspondence to depth_raster file. For example, if depth_raster has range $[10, -3000]$, then a breaks vector of `c(0,-40,-200,-2000,-Inf)` will create depth levels $[0, -40]$, $(-40, -200]$, $(-200, -2000]$, $(-2000, -\infty)$ and set to NA cells with values greater than 10.

`budget_percents`: Budget reflects the desired level of protection to be modeled. It ranges from 0 to 1, with 0 indicating no resources available for protection, while 1 signifies resources sufficient to protect the entire study area. Typically, setting a budget of 0.3 corresponds to the 30% conservation target (i.e. 30% of the total area set aside for conservation). Users also have the flexibility to define multiple budget levels using a vector, allowing for the exploration of various protection scenarios. For instance, a vector like `c(0.1, 0.3, 0.5)` represents three scenarios where 10%, 30%, and 50% of the study area are designated for protection.

`budget_weights`: The **prioritize_3D** function allows users to specify how the budget is distributed among depth levels. Three allocation methods are available:

1. Equal Distribution: Allocates an equal share of the budget to each depth level (`budget_weights = "equal"`).
2. Proportional to Area: Allocates budget based on the spatial extent of each depth level (`budget_weights = "area"`).
3. Proportional to Species Richness: Prioritizes budget allocation to depth levels with higher species diversity (number of species) (`budget_weights = "richness"`).

Otherwise, it can be a numeric vector with length equal to the number of depth levels, where each number indicates the budget share per depth level.

The solver used for solving the prioritization problems is the best available on the computer, following the solver hierarchy of **prioritizr**.

Value

A list containing the following objects (non-referenced are identical to the input ones):

- solution3D: list with 3D solution per budget percentage

- absolute_held3D: `absolute_held` for 3D solutions (see `evaluate_3D`)
- overall_available3D: `overall_available` for 3D solutions (see `evaluate_3D`)
- overall_held3D: `overall_held` for 3D solutions (see `evaluate_3D`)
- relative_helds3D: `relative_held` for 3D solutions (see `evaluate_3D`)
- mean_overall_helds3D: `base::mean` of `overall_held` for 3D solution (see `evaluate_3D`) per budget
- sd_overall_helds3D: `base::mean` of `overall_held` for 3D solution (see `evaluate_3D`) per budget
- depth_overall_available3D: `depth_overall_available` for 3D solutions (see `evaluate_3D`)

References

Hanson, Jeffrey O, Richard Schuster, Nina Morrell, Matthew Strimas-Mackey, Brandon P M Edwards, Matthew E Watts, Peter Arcese, Joseph Bennett, and Hugh P Possingham. 2024. prioritizr: Systematic Conservation Prioritization in R. https://prioritizr.net.

Lehtomäki, Joona (2016). Comparing prioritization methods, 21 June.
Available at: https://rpubs.com/jlehtoma/priocomp (Accessed 1 June 2024).

See Also

`evaluate_3D`, `terra_jaccard`, `plot_3D`

Examples

```
## Not run:
## This example requires commercial solver from 'gurobi' package for
## portfolio = "gap". Else replace it with e.g. portfolio = "shuffle" for using
## a free solver like the one from 'highs' package.

biodiv_raster <- get_biodiv_raster()
depth_raster <- get_depth_raster()
data(biodiv_df)

# You can split features' 2D distributions into 3D ones and then run only 3D analysis
split_features <- split_rast(biodiv_raster,
                             depth_raster,
                             breaks = c(0, -40, -200, -2000, -Inf),
                             biodiv_df)

out_3D <- prioritize_3D(split_features = split_features,
                        depth_raster = depth_raster,
                        breaks = c(0, -40, -200, -2000, -Inf),
                        biodiv_df = biodiv_df,
                        budget_percents = seq(0, 1, 0.1),
                        budget_weights = "richness",
                        threads = parallel::detectCores(),
                        portfolio = "gap",
                        portfolio_opts = list(number_solutions = 10))
```

```
plot_3D(out_3D, to_plot="all", add_lines=TRUE)

# Arbitrary random weights
priority_weights <- data.frame(c("A", "B", "C"), c(0.001, 1000, 1))
names(priority_weights) <- c("group", "weight")
biodiv_df$group <- rep(c("A", "B", "C"), length.out=20)
out_3D <- prioritize_3D(split_features = split_features,
                        depth_raster = depth_raster,
                        biodiv_df = biodiv_df,
                        priority_weights = priority_weights,
                        breaks = c(0, -40, -200, -2000, -Inf),
                        budget_percents = seq(0, 1, 0.1),
                        budget_weights = "richness",
                        threads = parallel::detectCores(),
                        portfolio = "gap",
                        portfolio_opts = list(number_solutions = 10))

plot_3D(out_3D, to_plot="all", add_lines=TRUE)

## End(Not run)
```

`split_rast` *Split 2D feature distributions into 3D ones*

Description

Split 2D feature distributions into 3D ones

Usage

```
split_rast(biodiv_raster, depth_raster, breaks, biodiv_df, val_depth_range=TRUE,
sep_biodiv_df=",")
```

Arguments

`biodiv_raster`	SpatRaster object or folder path with 2D feature distributions as layers.
`depth_raster`	SpatRaster object or file path with elevation/bathymetric map.
`breaks`	Numeric vector defining the range of depth layers to use.
`biodiv_df`	`data.frame` or a file path (CSV, TXT, XLS, or XLSX) containing additional information about biodiversity features.
`val_depth_range`	No correction of the splitted 3D distributions based on depth range of the biodiversity features ("`min_z`" and "`max_z`" from `biodiv_df`) is needed.
`sep_biodiv_df`	The separator used in biodiv_df file, if biodiv_df is in path format.

Details

This function is used to convert 2D distributions of biodiversity features (rasters) into a 3D format. Here the `biodiv_df` can have the following column names (independently of their order and any other names are ignored):

- `"species_name"`: **Mandatory** column with the feature names, which must be the same with biodiv_raster.
- `"pelagic"`: **Mandatory** column about the features' behaviour. `TRUE` means that this feature is pelagic and `FALSE` means that this feature is benthic.
- `"min_z"`: **Optional** column about the minimum vertical range of features. `NA` values are translated as unlimited upward feature movement.
- `"max_z"`: **Optional** column about the maximum vertical range of features. `NA` values are translated as unlimited downward feature movement.

`breaks` must be in correspondence to depth_raster file. For example, if depth_raster has range $[10, -3000]$, then a breaks vector of `c(0,-40,-200,-2000,-Inf)` will create depth levels $[0, -40]$, $(-40, 200]$, $(-200, -2000]$, $(-2000, -\infty)$ and set to NA cells with values greater than 0.

If `val_depth_range` = `TRUE` (default), then no correction is done and the depth range of the biodiversity features is derived from the corresponding feature distribution raster and so `"min_z"` and `"max_z"` are ignored. If `val_depth_range` = `FALSE`, then the function uses the minimum and maximum depth information provided in the biodiv_df, so as to remove feature occurrences outside their expected range.

Value

A list containing species distributions for each bathymetric layer, that are necessary for further 3D analysis. List names are indicating the depth levels.

Examples

```
biodiv_raster <- get_biodiv_raster()
depth_raster <- get_depth_raster()
data(biodiv_df)

# You can split features' 2D distributions into 3D ones and then run only 3D analysis
split_features <- split_rast(biodiv_raster,
                             depth_raster,
                             breaks = c(0, -40, -200, -2000, -Inf),
                             biodiv_df)
```

`sumrast` *Sum list of SpatRaster objects.*

Description

Sum list of SpatRaster objects.

Usage

```
sumrast(x, normalize = TRUE)
```

Arguments

`x`	List of SpatRaster objects.
`normalize`	If `TRUE`, then sum of solutions is normalized at a $[0, 1]$ scale.

Value

A SpatRaster object.

See Also

`plot_sumrast`

Examples

```
set.seed(42)
x <- terra::rast(matrix(rbinom(100, 1, 0.2), nrow=10))
y <- terra::rast(matrix(rbinom(100, 1, 0.8), nrow=10))
sumrast(list(x, y))
```

`terra_jaccard` *Jaccard similarity coefficient among two SpatRaster objects*

Description

Jaccard coefficient among two SpatRaster objects

Usage

```
terra_jaccard(x, y)
```

Arguments

`x`	SpatRaster object with binary values.
`y`	SpatRaster object with binary values.

Details

Jaccard similarity coefficient evaluates the percentage number equal to the intersection between two sets, divided by the size of the union of these sets.

Value

A numeric value $[0, 1]$.

Examples

```
set.seed(42)
x <- terra::rast(matrix(rbinom(100, 1, 0.2), nrow=10))
y <- terra::rast(matrix(rbinom(100, 1, 0.8), nrow=10))
terra_jaccard(x, y)
```

Index