Package ‘multilandr’

May 10, 2024

Title multilandr: Landscape Analysis at Multiple Spatial Scales

Version 1.0.0

Description Provides a tidy workflow for general landscape-scale analysis. 'multilandr' provides the tools to generate landscapes at multiple spatial scales and calculate landscape metrics, mainly relying on R package 'landscapemetrics'. The package provides utility functions to plot and analyze multi-scale landscapes, calculate and visualize correlations between metrics, filter landscapes that fulfill certain conditions, generate gradients of landscapes for a given metric and prepare datasets for further statistical tasks.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports terra (>= 1.7-71),
GGally (>= 2.2.1),
ggplot2 (>= 3.5.1),
tidyterra (>= 0.6.0),
gridExtra (>= 2.3),
landscapemetrics (>= 2.1.1),
sf (>= 1.0-16),
methods (>= 4.4.0)

Depends R (>= 4.4.0)

LazyData true

Suggests parallel (>= 4.1.0),
testthat (>= 3.0.0)

Collate 'check_raster.R'
'classes.R'
'data.R'
'func_checks.R'
'generate_points.R'
'generate_points_checks.R'
'globals.R'
'methods.R'
'metrics_bind.R'

'metrics_corr.R'
'metrics_corr_checks.R'
'metrics_filter.R'
'metrics_filter_checks.R'
'metrics_gradient.R'
'metrics_gradient_checks.R'
'metrics_list.R'
'metrics_plots.R'
'mland.R'
'mland_checks.R'
'mland_export_gis.R'
'mland_load.R'
'mland_metrics.R'
'mland_metrics_checks.R'
'mland_overlap.R'
'mland_overlap_checks.R'
'mland_plot.R'
'mland_plot_checks.R'
'mland_save.R'
'utils.R'

Config/testthat/edition 3

Contents

`check_raster` *Check input raster*

Description

Checks the validity of raster layers to be inputted in `mland()`, intended to represent land cover. The function directly calls `landscapemetrics::check_landscape()`.

Usage

```
check_raster(raster, verbose = T)
```

Arguments

`raster`	An object of class 'RasterLayer', 'RasterStack', 'RasterBrick', 'SpatRaster', or a list of raster objects (one of 'RasterLayer' or 'SpatRaster').
`verbose`	Print warning messages.

Details

Extracts basic information about the inputted raster: coordinate reference system (crs) - either "geographic", "projected", or NA, units of the coordinate reference system, class for the values of the inputted raster and the number of classes found in the raster.

`conditions` *Define metric conditions*

Description

Helper function to define patch conditions within `generate_points()` or metric conditions within `metrics_filter()`.

Usage

```
conditions(...)
```

Arguments

`...`	Patch or metric conditions in the form of lists. See Details.

Details

Conditions must be defined as lists (one or more).

For patch conditions, within the environment of `generate_points()`, each element within the list defines the condition that the patch must meet in relation to the value of certain patch-level metric, as follows:

```
list(class, metric, minimum value, maximum value)
```

- class: the class (raster value) of the patch that must meet the defined conditions. More than one class can be specified.
- metric: the patch-level metric whose values must meet the defined conditions. Only one metric per condition can be defined. Available patch-level metrics can be found in `metrics_list()` and in documentation of the package `landscapemetrics()`.
- minimum value: the minimum value that the metric must have for the retained patches. If equal to -Inf, and a maximum value is defined, patches whose values in the defined metric are equal or lower to the maximum value will be retained.
- maximum value: the maximum value that the metric must have in the retained patches. If equal to Inf, and a minimum value is defined, patches whose values in the defined metric are equal or higher to the minimum value will be retained.

For metric conditions, within the environment of `metrics_filter()`, each element within the list defines the required metric conditions, as follows:

```
list(rasterlayers, class, radii, metric, minimum value, maximum value)
```

- rasterlayers: the rasterlayers to be considered. If NA, all rasterlayers will be considered. If an extra rasterlayer must be specified, the string "ext" must precede the rasterlayer number (e.g. "ext1", "ext2").
- class: the classes to be considered, as numbers or strings with the names of the classes. If NA, all classes of required rasterlayers will be considered. If NULL, the function will assume that the metric to be considered is a landscape-level metric. Take into account that metrics from extra calculations are considered as landscape-level metrics.
- radii: the radii to be considered. If NA, all radii will be considered.
- metrics: the name of the metric to be considered (as defined with its abbreviation by column "metric" in `metrics_list()`). Only one metric per condition can be defined. Metrics as extra calculations for extra rasterlayers must be provided as "fun_" + the name of the function (e.g. "fun_mean").
- minimum value: the minimum value that the metric must have in the filtered landscapes. If equal to -Inf, and a maximum value is defined, landscapes whose values in the defined metric are equal or lower to the maximum value will be retained.
- maximum value: the maximum value that the metric must have in the filtered landscapes. If equal to Inf, and a minimum value is defined, landscapes whose values in the defined metric are equal or higher to the minimum value will be retained.

See the example sections of functions `generate_points()` and `metrics_filter()` for more details.

Value

A list to be inputted within the argument `patch_conditions` in `generate_points()` or the argument `conditions` in `metrics_filter()`.

`ed_metrics` *'MultiLandMetrics' object*

Description

An object of class 'MultiLandMetrics' generated with `mland_metrics()`, for the purposes of package examples of the following functions: `metrics_filter()`, `metrics_gradient()`, `metrics_corr()`, `metrics_plots()` and `metrics_bind()`. See 'MultiLand-class' for general information about these objects.

Usage

```
ed_metrics
```

Format

An object of class `MultiLandMetrics` of length 1.

Details

The main internal object is a data.frame (accesible through `ed_metrics@data`) with information about the values of two landscape metrics: "pland" (percentage of landscape) and "np" (number of patches).

The object was created from the MultiLand object named "ernesdesign", which received two raster layers from a small portion of the ecoregion "El Chaco" as main inputs. The main rasterlayer was provided by the project "MapBiomas Chaco" for the year 2000. The extra rasterlayer contained the NDVI values of cells within the same extent of the main rasterlayer, and was provided by Landsat.

References

Project MapBiomas Chaco – Collection 4.0 of annual land cover and land use maps, accessed during July 2022 through the following link: MapBiomas Chaco

Landsat-5 image courtesy of the U.S. Geological Survey

See Also

See the examples sections of `mland_metrics()` and `mland()` for more context.

generate_points *Generates point coordinates*

Description

Generates point coordinates over a rasterlayer extent.

Usage

```
generate_points(
  raster,
  approach = "grid",
  n = NULL,
  try = NULL,
  values = NULL,
  patch_conditions = NULL,
  trim = TRUE,
  attempts = 10,
  distance = NULL,
  offset = FALSE,
  closest_cell = FALSE,
  parallel = FALSE,
  cores = 1,
  progress = TRUE
)
```

Arguments

`raster`	An object of class 'SpatRaster', 'RasterLayer', 'RasterStack' or 'RasterBrick'.
`approach`	One of the following: "grid" to generate points through a grid, "random" to generate points at random locations, or "patch" to generate points inside patches that meet pre-defined conditions. See Details.
`n`	Number of point to generate.
`try`	Number of points to be generated in each turn. Only applies if `approach = "random"`. See Details.
`values`	The values of the rasterlayer where points should be placed. Only applies if `approach = "random"`.
`patch_conditions`	The conditions that patches must meet to be included as the patches from which points will be generated. Only applies if `approach = "patch"`. See Details.
`trim`	Logical. If TRUE (default) the number of final points will be trimmed to the value defined in `n`.
`attempts`	Number of attempts to generate new random points given the total required points (`n`) and the minimum distance required in `distance`. Only applies if `approach = "random"`. See Details.

`distance`	Distance between points of the grid (if `approach = "grid"`) or minimum distance between generated points (if `approach = "random"`).
`offset`	Logical. If TRUE, each point coordinates will be randomly displaced around the area occupied by the raster cell size. If FALSE (default), each point will be located at the center of a given raster cell. Only applies if `approach = "random"`.
`closest_cell`	Logical. If `approach = "patch"`, whether to return the coordinates of each patch centroid even if they fall outside the patch (FALSE, default) or to move the point to the closest cell of the patch if this happens (TRUE).
`parallel`	Logical. If TRUE, part of the processes will be parallelized. See Details.
`cores`	Number of cores to use if `parallel = TRUE`.
`progress`	Logical. If TRUE (default), progress of the analysis will be printed.

Details

If `approach = "random"`, the user can restrict the locations of new generated points inside raster cells with certain value or values, by defining them in `values`. Also a minimum distance between the generated points can be defined in `distance` (also applies for the resolution of the grid if `approach = "grid"`).

If `approach = "random"` and a minimum distance was defined, the function will generate new "random" points in sequential passes. In each pass, the function will try to generate new points taking into account the minimum distance between points, by randomly generating a number of points as defined in `try`. The function will perform this task until the new generated points is equal or higher than `n`. If `try = NULL` (default), `try` will equals `n`. If in each turn no new points were added (i.e. new points were located at less than the minimum distance to other previously generated points), the function will record this event. If this event happens more than the number of times defined in `attempts` before the total generated points equals `n`, the function will terminate, and return the points that were successfully generated given the required parameters. The user may try different values for `n`, `try` and `attempts` to get a desirable result.

If `approach = "patch"`, the function will return as many points as patches that meet certain conditions in relation to pre-defined metric values. Conditions can be defined in argument `patch_conditions`, for which the helper function `conditions()` is available:

```
conditions(list(class, metric, minimum value, maximum value),
           list(class, metric, minimum value, maximum value), ...)
```

- class: the class (raster value) of the patch that must meet the defined conditions. More than one class can be specified.
- metric: the patch-level metric whose values must meet the defined conditions. Only one metric per condition can be defined. Available patch-level metrics can be found in `metrics_list()` and in documentation of the package `landscapemetrics()`.
- minimum value: the minimum value that the metric must have for the retained patches. If equal to -Inf, and a maximum value is defined, patches whose values in the defined metric are equal or lower to the maximum value will be retained.
- maximum value: the maximum value that the metric must have in the retained patches. If equal to Inf, and a minimum value is defined, patches whose values in the defined metric are equal or higher to the minimum value will be retained.

Retained patches will be those patches that meet all patch conditions at the same time. Returned point's coordinates will equal the centroid of each patch. If `closest_cell = TRUE`, the point's coordinates of the centroids that did not fall inside the patch will be moved to the closest cell belonging to that patch.

If `parallel = TRUE` the function will parallelize part of the processes. Parallelization is done to obtain the coordinates of the patches if `approach = "patch"`. The number of cores must be declared in `cores` (parallelization requires at least two cores). To use this functionality, package `parallel` must be installed. So far, parallelization will run in LINUX and MAC, but not in Windows.

Value

An object of class 'SpatVector' containing the coordinates of the generated points.

See Also

`mland()`

Examples

```
# Loads raster
elchaco <- terra::rast(system.file("extdata", "elchaco.tif", package = "multilandr"))

# Returns points at "random" locations, but inside cells of value equals to 1.
chaco_coords <- generate_points(elchaco, approach = "random", values = 1, n = 500)

# The same but points must be separated by at least 300 m between each other. Also, each point
# is randomly displaced inside the raster cell.
chaco_coords2 <- generate_points(elchaco, approach = "random", values = 1, n = 500,
                              try = 100, distance = 300, offset = TRUE)

## Not run:
# Returns as many points as patches that meet the defined condition. This is
# all patches of value equal to 1 of area between 9 and 11 hectares.
patch_sites <- generate_points(elchaco, approach = "patch",
                            patch_conditions = conditions(list(1, "area", 8, 12)))

## End(Not run)
```

---

metrics_bind *Metric's data preparation*

Description

Merge data.frame with metric's values with a data.frame with other data.

Usage

```
metrics_bind(
  x,
  data,
  raster = NULL,
  classes = NULL,
  radii = NULL,
  c_level = NULL,
  l_level = NULL,
  ext_raster = NULL,
  classnames = FALSE
)
```

Arguments

`x`	An object of class 'MultiLandMetrics' generated with `mland_metrics()`.
`data`	A data.frame with data from each sampling point/site. See Details.
`raster`, `ext_raster`, `classes`, `radii`, `l_level`, `c_level`	Parameters to subset data.frame containing the metrics values. See Details.
`classnames`	Logical. If TRUE, classes names will be returned as the names of the classes previously provided (if so) when `x` was generated. Default FALSE.

Details

Merges data.frame with metrics values, contained in an object of class 'MultiLandMetrics' (returned by `mland_metrics()`) with a data.frame with other data for each site. In this way, the returned data.frame will be prepared for later statistical or visual analyses. The data.frame provided in `data` must have a column named "site" or "point_id", containing unique identifiers for each sampling site, which must match with the identifiers present in the data.frame contained in `x` (i.e. data.frame with metrics values for each site). If "site", the function will assume that the site names are provided as identifiers. If "point_id", the function will assume that point ids are being provided. In any case, these identifiers must match the site identifiers in `x`.

Argument `raster`, `ext_raster`, `classes`, `radii`, `l_level` and `c_level` can be defined to subset the data.frame contained in `x`. In each one of these, an all-positive or an all-negative vector can be passed, whether to include (all-postive) or exclude (all-negative) the elements to be taken into account for the subsetting:

- rasterlayers: a numeric vector with the number of the rasterlayers to be included/excluded. For example: `c(1, 2, 4)` to include rasterlayers 1, 2 and 4; `c(-2, -3)` to exclude rasterlayers 2 and 3.
- classes: must be a list with as many elements as defined rasterlayers in argument `rasterlayers`. Each element of the list must be a numeric vector (classes identities) with the classes to be included/excluded. If provided a character vector, `metrics_bind()` assumes that classes names are provided. For example, for the case with 2 rasterlayers: `list(c(3, 20, 35), c("Forest", "Crops"))` would include classes 3, 20 and 35 from rasterlayer 1 and classes "Forest" and "Crops" for rasterlayer 2. For the case of a unique rasterlayer, there is no need to input a list. For example, for the case of a unique rasterlayer and the exclusion of some classes:

c(-5, -10, -15) to exclude classes 5, 10 and 15 of the unique rasterlayer; c("-Forest", "-Grassland") to exclude classes "Forest" and "Grassland". Note the "-" before each class name to indicate the exclusion of the classes.

- radii: a numeric vector to include/exclude particular radii. For example: c(1000, 2000) to include only radii of 1000 and 2000 m; c(-500, -1500) to exclude radii of 500 and 1500 m.
- c_level: character vector with the class-level metrics to be included/excluded from the analysis. For example: c("np", "pland") will include only the metrics "number of patches" ("np") and "percentage of the landscape" ("pland") in the analysis, whereas c("-np", "-pland") will exclude them. Note the "-" before each metric name to indicate the exclusion of the metrics.
- l_level: character vector with the landscape-level metrics to be included/excluded from the analysis. Extra calculations for extra rasterlayers are considered as landscape-level metrics, and must be provided as "fun_" + the name of the function.

Value

A data.frame equal to sampling data provided in data but with additional columns containing the values of the metrics for each sampling site.

See Also

mland_metrics()

Examples

```
# Get sites names from ed_metrics and creates ad-hoc data.frame with random values of
# "richness" (the response variable). Only for the purpose of this example
sites <- ed_metrics@points$name
sampling_data <- data.frame(site = rep(sites, each = 10),
                            richness = sample(1:500, 150))

# With no filters, all columns with all metrics at all spatial scales are added to
# the sampling data
new_data <- metrics_bind(ed_metrics, sampling_data)

# Subset for metrics of class "Forest", radius 5000 and metric "pland"
new_data <- metrics_bind(ed_metrics, sampling_data, classnames = TRUE,
                         classes = "Forest", radii = 3000, c_level = "pland")

# In this format, the data.frame can be passed to a fitting model
fit <- lm(richness ~ r1_Forest_pland_3000, data = new_data)
```

---

metrics_corr *Pairwise metric correlations*

Description

Calculates pairwise correlations between landscape metrics.

Usage

```
metrics_corr(
  x,
  method = "pearson",
  fun = NULL,
  raster = NULL,
  classes = NULL,
  radii = NULL,
  c_level = NULL,
  l_level = NULL,
  ext_raster = NULL,
  classnames = FALSE,
  display = "radii",
  ...
)
```

Arguments

`x`	An object of class 'MultiLandMetrics' generated with `mland_metrics()`.
`method`	The method to be used to calculate pair correlations: "pearson" (default), "spearman" or "kendall".
`fun`	A user-defined function to calculate correlations. See Details.
`raster`, `ext_raster`, `classes`, `radii`, `l_level`, `c_level`	Parameters to subset calculations of correlations. See Details.
`classnames`	Logical. If TRUE, row and column of returned matrices will be identified with the names of the classes, if available in `x`. Default FALSE.
`display`	Defines how correlations are presented: "radii" (default), "rl" or "both". See Details.
`...`	Other arguments passed to function `cor()` or to the user-defined function provided in `fun`.

Details

Correlations are calculated, by default, through the function `cor()`, by specifying the method through the argument `method`. Alternatively, a user-defined function can be provided in the argument `fun`. If not NULL, the function will assume that a user-defined function have been provided. This must be a function already loaded in the environment, and must take at least two arguments. These initial pair of arguments should be capable of receiving two numeric vectors (one in each argument), process them in some way, and return a numeric value (i.e. the supposed correlation).

Arguments `raster`, `ext_raster`, `classes`, `radii`, `c_level` and `l_level` can be defined to subset the calculations of pair correlations. In each one of these, an all-positive or an all-negative vector can be passed, whether to include (all-postive) or exclude (all-negative) the elements to be taken into account for the subsetting:

- raster: a numeric vector with the number of the rasterlayers to be included/excluded. For example: `c(1, 2, 4)` to include rasterlayers 1, 2 and 4; `c(-2, -3)` to exclude rasterlayers 2 and 3.

- ext_raster: a numeric vector with the number of the extra rasterlayers to be included/excluded, as in the raster slot.
- classes: must be a list with as many elements as defined rasterlayers in argument `raster`. Each element of the list must be a numeric vector (classes identities) with the classes to be included/excluded. If provided a character vector, `metrics_corr()` assumes that classes names are provided. For example, for the case with 2 rasterlayers: `list(c(3, 20, 35), c("Forest", "Crops"))` would include classes 3, 20 and 35 from rasterlayer 1 and classes "Forest" and "Crops" for rasterlayer 2. For the case of a unique rasterlayer, there is no need to input a list. For example, for the case of a unique rasterlayer and the exclusion of some classes: `c(-5, -10, -15)` to exclude classes 5, 10 and 15 of the unique rasterlayer; `c("-Forest", "-Grassland")` to exclude classes "Forest" and "Grassland". Note the "-" before each class name to indicate the exclusion of the classes.
- radii: a numeric vector to include/exclude particular radii. For example: `c(1000, 2000)` to include only radii of 1000 and 2000 m; `c(-500, -1500)` to exclude radii of 500 and 1500 m.
- c_level: character vector with the class-level metrics to be included/excluded from the analysis. For example: `c("np", "pland")` will include only the metrics "number of patches" ("np") and "percentage of the landscape" ("pland") in the analysis, whereas `c("-np", "-pland")` will exclude them. Note the "-" before each metric name to indicate the exclusion of the metrics.
- l_level: character vector with the landscape-level metrics to be included/excluded from the analysis. Extra calculations for extra rasterlayers are considered as landscape-level metrics, and must be provided as "fun_" + the name of the function (e.g. "fun_mean").

Names of the available metrics of the 'MultiLandMetrics' object provided in `x` can be accessed with `x@metrics` and `x@ext_calc`.

Note that patch-level metrics, if exists in `x` metric's data.frame, are excluded from calculations, as this function works at a landscape-scale analysis.

Argument `display` defines how correlation values will be presented. If equals to "radii" (default), correlation values are disaggregated by radii. If "rl", correlation values are disaggregated by rasterlayer: correlations between different radii will be presented. If "both", correlation values are firstly disaggregated by rasterlayer, and by radii secondly. Disaggregations by rasterlayers only make sense for 'MultiLandMetrics' objects with more than one rasterlayer.

Value

A list with matrices containing correlation values between pair of metrics. Matrices are disaggregated by radius if `display = "radii"`, by rasterlayer if `display = "rl"` or by rasterlayer and radii if `display = "both"`. Metrics names are presented as row and column names of the matrices, with the following format: "level"*"metric_name"*"radius". For a landscape-level metric, a plausible metric name could be "l_np_1500" indicating a landscape-level metric, which is "np" ("number of patches") at a scale (radius) of 1500 m. For a class-level metric a plausible metric name could be "c4_pland_1000", indicating a class-level metric of class 4 (the value of the raster), which is "pland" ("percentage of landscape") at a scale (radius) of 1000 m. If more that one rasterlayer is being analyzed, the prefix "r1", "r2", "r3", ..., "rn" (referring to rasterlayer 1, 2, 3, ..., n) is added to the metric name.

See Also

mland_metrics(), metrics_plots()

Examples

```
# Calculates pearson correlations between metrics of a MultiLandMetrics object
metrics_corr(ed_metrics)

# Only for radius 5000 m and with classes names rather than classes values
metrics_corr(ed_metrics, radii = 5000, classnames = TRUE)

# Only selecting the metric "pland"
metrics_corr(ed_metrics, radii = 5000, classnames = TRUE, c_level = "pland")

# Excluding the metric "pland"
metrics_corr(ed_metrics, radii = 5000, classnames = TRUE, c_level = "-pland")

# Excluding the metric radii of 4000 and 5000 m
metrics_corr(ed_metrics, radii = c(-4000, -5000), classnames = TRUE)

# Correlations of metric "pland" between classes 1 to 3, and between radii
# 1000 and 5000 m, disaggregating by rasterlayer.
metrics_corr(ed_metrics, radii = c(1000, 5000), classes = 1:3,
             c_level = "pland", display = "rl")
```

---

metrics_filter *Filters metrics*

Description

Selects landscapes that meet certain pre-defined conditions in relation to its metrics from a 'MultiLandMetrics' object.

Usage

```
metrics_filter(
  x,
 conditions = list(rasterlayers = NULL, classes = NULL, radii = NULL, metric = NULL,
   min_value = NULL, max_value = NULL),
  output = "MLM"
)
```

Arguments

x	An object of class 'MultiLandMetrics' generated with mland_metrics().
conditions	List. Conditions to be met by the landscapes. See Details.

`output`	One of the following: "MLM" to return an updated version of the 'MultiLand-Metrics' object provided in x (default), "spatial" to return a 'SpatVector' with the points of the selected landscapes, "data" to return a data.frame with the metric values information or "coords" to return a data.frame with geographical information of the filtered points.

Details

Selects landscapes that meet certain conditions in relation to the values of their landscape metrics. The function will retain those points associated with the landscapes that meet all the defined conditions at the same time. Conditions must be provided through a list, for which the helper function `conditions()` is available:

```
conditions(list(rasterlayers, class, radii, metric, minimum value, maximum value),
           list(rasterlayers, class, radii, metric, minimum value, maximum value),
              ...)
```

- rasterlayers: the rasterlayers to be considered. If NA, all rasterlayers will be considered. If an extra rasterlayer must be specified, the string "ext" must precede the rasterlayer number (e.g. "ext1", "ext2").
- class: the classes to be considered, as numbers or strings with the names of the classes. If NA, all classes of required rasterlayers will be considered. If NULL, the function will assume that the metric to be considered is a landscape-level metric. Take into account that metrics from extra calculations are considered as landscape-level metrics.
- radii: the radii to be considered. If NA, all radii will be considered.
- metrics: the name of the metric to be considered (as defined with its abbreviation by column "metric" in `metrics_list()`). Only one metric per condition can be defined. Metrics as extra calculations for extra rasterlayers must be provided as "fun_" + the name of the function (e.g. "fun_mean").
- minimum value: the minimum value that the metric must have in the filtered landscapes. If equal to -Inf, and a maximum value is defined, landscapes whose values in the defined metric are equal or lower to the maximum value will be retained.
- maximum value: the maximum value that the metric must have in the filtered landscapes. If equal to Inf, and a minimum value is defined, landscapes whose values in the defined metric are equal or higher to the minimum value will be retained.

A plausible list of conditions could be the following:

```
conditions(list(1, 2, 1000, "pland", 20, 30),
           list(1, 4, 1000, "np", 1, 15),
           list("ext1", NULL, 1000, "fun_mean", 70, 80))
```

And it would indicate that landscapes of radius equal to 1000 m should present values of "pland" (percentage of the landscape) for class 2 from rasterlayer 1, between 20 and 30%. At the same time, landscapes of radius equal to 1000 m should present values of "np" (number of patches) for class 4 from rasterlayer 1, between 1 and 15 patches. Finally, all selected landscapes of radius equal to 1000 m should present values for "fun_mean" (applied to extra rasterlayer "ext1") between 70 and 80. Note that the slot for "class" is NULL, as extra rasterlayers do not hold classes.

Value

A 'MultiLandMetrics' if `output = "MLM"`, a 'SpatVector' if `output = "spatial"`, a data.frame if `output = "data"` or a data.frame with geographical information of the points if `output = "coords"`.

See Also

`metrics_gradient()`

Examples

```
# Filter landscapes that have between 20 and 30% of forest at a radius of 2000 m
# and output the data.frame with metrics values
conds <- conditions(list(NA, "Forest", 2000, "pland", 20, 30))
otf_subset <- metrics_filter(otf_metrics,
                             conditions = conds,
                             output = "data")

# The same but returning a data.frame with information of the retained points
conds <- conditions(list(NA, "Forest", 2000, "pland", 20, 30))
otf_subset_points <- metrics_filter(otf_metrics,
                                    conditions = conds,
                                    output = "coords")

# Filter landscapes that have between 20 and 30% of forest at a radius of 2000 m
# and a maximum of 60% of Crops.
conds <- conditions(list(NA, "Forest", 2000, "pland", 20, 30),
                    list(NA, "Crops", 2000, "pland", -Inf, 60))
otf_subset2 <- metrics_filter(otf_metrics,
                              conditions = conds,
                              output = "data")
```

`metrics_gradient` *Generates optimized metrics gradient*

Description

Selects a set of points whose associated landscapes comprise an optimized gradient for a given landscape metric.

Usage

```
metrics_gradient(
  x,
  rasterlayer = NULL,
  class = NULL,
  radius = NULL,
  metric = NULL,
  n,
  cutpoints = NULL,
  breaks = NULL,
  output = "MLM"
)
```

Arguments

`x`	An object of class 'MultiLandMetrics' generated with `mland_metrics()`.
`rasterlayer`	The rasterlayer to be considered. If an extra rasterlayer must be specified, the string "ext" must precede the rasterlayer number (e.g. "ext1", "ext2")
`class`	The class to be considered, as a number or as a string with the name of the class.
`radius`	The radius to be considered.
`metric`	The metric to be considered. Metrics as extra calculations for extra rasterlayers must be provided as "fun_" + the name of the function.
`n`	The number of points that will comprise the gradient. See Details.
`cutpoints`	A sequence of numbers that will serve as numeric approximations to select the points that will comprise the gradient. See Details.
`breaks`	A unique number with the number of breaks that will generate the cutpoints for the specified metric values. Default is 10. See Details.
`output`	One of the following: "MLM" to return an updated version of the 'MultiLandMetrics' object provided in `x` (default), "spatial" to return a 'SpatVector' with the points of the selected landscapes, "data" to return a data.frame with the metric values information or "coords" to return a data.frame with geographical information of the selected points.

Details

Selects a subset of landscapes that overall will generate an optimized gradient of values for a given landscape metric of a specified rasterlayer, class and radius. One can define a gradient as optimized if its values fulfill to cover a good range of values between a minimum and a maximum value. The final gradient will comprise the number of points specified in argument `n`. Note that only one landscape metric can be specified at a time.

The algorithm will select those points whose associated landscapes present values for the specified landscape metric that are the most close to the specified `cutpoints`. Alternatively, the user can provide a number of `breaks` from which the sequence of cutpoints will be generated. If both arguments are specified, the function will consider the values inputted in `cutpoints`. If both arguments are NULL, the algorithm will simply select `n` random points.

Value

A 'MultiLandMetrics' if `output = "MLM"`, a 'SpatVector' if `output = "spatial"`, a data.frame if `output = "data"` or a data.frame with geographical information of the points if `output = "coords"`.

See Also

`metrics_filter()`

Examples

```
# Generates an optimized gradient for the landscape metric "pland", for the class "Forest".
pland_gradient <- metrics_gradient(otf_metrics, rasterlayer = 1, class = "Forest",
                                   radius = 2000, metric = "pland", n = 15, breaks = 10)
# Note that, in this case, specifications for the rasterlayer and the radius are
# redundant, and could be simply ignored and left as default, asthe object otf_metrics
# only comprises a unique rasterlayer and radius.

# By default, the output is an updated version of the object otf_metrics. In order to
# inspect the returned values, let's select only the dataframe containing the
# metric's values.
foo <- subset(pland_gradient@data, metric == "pland" & classname == "Forest",
              select = value)

# Next, we output the range of values we have obtained, note there are 15 points, as
# previously specified in the function definition in the argument 'n'
round(sort(foo$value), digits = 2)

# 1.15  1.57  8.17  8.19 15.24 22.32 29.27 36.32 43.17 43.20 49.79 50.25 55.44 57.62 64.53

# Alternatively, we can define specific cutpoints around the landscapes will be selected
# in termsof its numeric closeness.
pland_gradient <- metrics_gradient(otf_metrics, rasterlayer = 1, class = "Forest",
                                   radius = 2000,metric = "pland", n = 15,
                                   cutpoints = seq(1, 60, 5))

# Again, we inspect the dataframe with the metric values to see our results.
foo <- subset(pland_gradient@data, metric == "pland" & classname == "Forest",
              select = value)

round(sort(foo$value), digits = 2)

# 1.15  6.02  6.03 10.99 15.97 20.99 26.01 31.02 35.95 41.14 41.34 45.93 51.41 54.56 55.44

# Both alternatives generated a well-ranged gradient of values for the forest metric "pland"
```

`metrics_list` *Metrics list*

Description

List of available landscape metrics provided by package landscapemetrics to be calculated with `mland_metrics()`. It simply calls `landscapemetrics::list_lsm()`. For more information regarding the definition and equations of metrics, please check the user manual of landscapemetrics.

Usage

```
metrics_list(
  level = NULL,
  metric = NULL,
  name = NULL,
  type = NULL,
  what = NULL
)
```

Arguments

`level`	Character vector. Level of metrics. Either "patch", "class" or "landscape" (or a vector with a combination of these). Default NULL considers all levels.
`metric`	Abbreviation of metrics (e.g. "area").
`name`	Full name of metrics (e.g. "core area").
`type`	Character vector. Type according to FRAGSTATS grouping. One or more of the following: "area and edge", "core area", "shape", "aggregation", "complexity", and or "diversity". Default NULL considers all types.
`what`	Selected level of metrics: either "patch", "class" or "landscape". It is also possible to specify functions as a vector of strings, e.g. what = c("lsm_c_ca", "lsm_l_ta").

Value

A data.frame with the list of available landscape metrics, including information regarding the level, type, metric, name and function name provided by package landscapemetrics.

References

Hesselbarth, M.H.K., Sciaini, M., With, K.A., Wiegand, K., Nowosad, J. 2019. landscapemetrics: an open-source R tool to calculate landscape metrics. - Ecography 42:1648-1657(ver. 0).

McGarigal, K., SA Cushman, and E Ene. 2012. FRAGSTATS v4: Spatial Pattern Analysis Program for Categorical and Continuous Maps. Computer software program produced by the authors at the University of Massachusetts, Amherst.
Available at the following web site: https://www.umass.edu/landeco/

`metrics_plots` *Pairwise metric plots*

Description

Plots pair of metric values in two-dimensional plots.

Usage

```
metrics_plots(
  x,
  raster = NULL,
  classes = NULL,
  radii = NULL,
  c_level = NULL,
  l_level = NULL,
  ext_raster = NULL,
  classnames = FALSE,
  upper = TRUE,
  diag = TRUE,
  smooth = TRUE,
  method = "loess",
  se = FALSE,
 st_points = list(shape = 21, size = 2, col = "black", fill = "white", alpha = 1),
  st_lines = list(lty = 1, lwd = 1, col = "black", alpha = 0.6),
  ...
)
```

Arguments

`x`	An object of class 'Multiland' generated with `mland()`.
`raster`, `ext_raster`, `classes`, `radii`, `l_level`, `c_level`	Parameters to subset plots. See Details.
`classnames`	logical. Whether to show classes with its previously defined names (if defined) when generating the 'MultiLand' object (TRUE), or not (FALSE, default).
`upper`	logical. Whether to plot upper-diagonal plots or not. Default TRUE
`diag`	logical. Whether to plot diagonal density plots or not. Default TRUE.
`smooth`	logical. If TRUE (default) a pattern between the pair of metric values is plotted, with a smoothing method as defined in `method`.
`method`	Smoothing method (function) to use, as in `ggplot2::geom_smooth()`. It accepts "loess" (default), "lm", "gam", among others. See ?ggplot2::geom_smooth() for more details.
`se`	logical. Whether to show (TRUE) or not (FALSE) confidence intervals when `smooth = TRUE`.
`st_points`	List of aesthetic arguments for points plotting: `shape` for points shape, `size` for points size, `col` for points border color, `fill` for points fill color and `alpha` for point transparency.
`st_lines`	List of aesthetic arguments for lines plotting (if `smooth = TRUE`): `lty` for linetype, `lwd` for linewidth, `col` for line color and `alpha` for line transparency.
`...`	Other parameters to be passed to `ggplot2::geom_smooth()`, if `smooth = TRUE`.

Details

`metrics_plots()` mainly relies on `GGally::ggpairs()` to generate pair plots between metrics values. Arguments `upper` and `diag` are specific arguments of `GGally::ggpairs()`, here adapted to the context of continuous values only.

Argument `raster`, `classes`, `radii`, `l_level` and `c_level` can be defined to subset the plots. In each one of these, an all-positive or an all-negative vector can be passed, whether to include (all-postive) or exclude (all-negative) the elements to be taken into account for the subsetting:

- rasterlayers: a numeric vector with the number of the rasterlayers to be included/excluded. For example: `c(1, 2, 4)` to include rasterlayers 1, 2 and 4; `c(-2, -3)` to exclude rasterlayers 2 and 3.
- classes: must be a list with as many elements as defined rasterlayers in argument `raster`. Each element of the list must be a numeric vector (classes identities) with the classes to be included/excluded. If provided a character vector, `metrics_corr()` assumes that classes names are provided. For example, for the case with 2 rasterlayers: `list(c(3, 20, 35), c("Forest", "Crops"))` would include classes 3, 20 and 35 from rasterlayer 1 and classes "Forest" and "Crops" for rasterlayer 2. For the case of a unique rasterlayer, there is no need to input a list. For example, for the case of a unique rasterlayer and the exclusion of some classes: `c(-5, -10, -15)` to exclude classes 5, 10 and 15 of the unique rasterlayer; `c("-Forest", "-Grassland")` to exclude classes "Forest" and "Grassland". Note the "-" before each class name to indicate the exclusion of the classes.
- radii: a numeric vector to include/exclude particular radii. For example: c(1000, 2000) to include only radii of 1000 and 2000 m; c(-500, -1500) to exclude radii of 500 and 1500 m.
- metrics: character vector with the metrics to be included/excluded from the plots For example: `c("np", "pland")` will include only the metrics "number of patches" ("np") and "percentage of the landscape" ("pland") in the analysis, whereas `c("-np", "-pland")` will exclude them. Note the "-" before each metric name to indicate the exclusion of the metrics. Extra calculations for extra rasters must be provided as "fun_" + the name of the function. Names of the available metrics of the 'MultiLandMetrics' object provided in `x` can be accessed with `x@metrics` and `x@ext_calc`.

Value

A panel with several plots returned by `GGally::ggpairs()` relating pair of metrics values. Metrics names are presented at the top and right of the panel (strips), with the following format: "level"*"metric_name"*"radius". For a landscape-level metric, a plausible metric name could be "l_np_1500" indicating a landscape-level metric, which is "np" ("number of patches") at a scale (radius) of 1500 m. For a class-level metric a plausible metric name could be "c4_pland_1000", indicating a class-level metric of class 4 (the value of the raster), which is "pland" ("percentage of landscape") at a scale (radius) of 1000 m. If more that one rasterlayer is being analyzed, the prefix "r1", "r2", "r3", ..., "rn" (referring to rasterlayer 1, 2, 3, ..., n) is added to the metric name.

See Also

`mland_metrics()`, `metrics_corr()`

Examples

```
## Not run:
# Pair plots between metrics "pland" of classes 1 to 4, for radius 3000 m
metrics_plots(ed_metrics, classes = 1:4, radii = 3000, classnames = TRUE,
              c_level = "pland")

# Without smooth pattern
metrics_plots(ed_metrics, classes = 1:4, radii = 3000, classnames = TRUE,
              c_level = "pland", smooth = FALSE)

# Changing aesthetics
metrics_plots(ed_metrics, classes = 1:4, radii = 3000, classnames = TRUE,
              c_level = "pland", smooth = FALSE, size = 1.5, shape = 21,
              fill = "red", alpha = 0.4)

# Assessing two radii values at the same time
metrics_plots(ed_metrics, classes = 1:4, radii = c(1000, 5000),
              classnames = TRUE, c_level = "pland", smooth = FALSE,
              size = 1.5, shape = 21, fill = "red", alpha = 0.4)

# An example with hundreds of points
metrics_plots(otf_metrics, classes = c("Forest", "Crops"))

# Plots can be combined with ggplot2::theme
# library(ggplot2)
metrics_plots(otf_metrics, classes = c("Forest", "Crops")) +
  theme_bw()

## End(Not run)
```

`mland` *Generates object of class 'MultiLand'*

Description

Creates an object of class 'MultiLand', which is the main object to be used by other functions of the package to generate plots, calculate landscape metrics and perform other relevant analyses.

Usage

```
mland(
  points_layer,
  rast_layer = NULL,
  radii,
  classnames = NULL,
  site_ref = NULL,
  bufftype = "round",
  segs = 20,
  ext_rast_layer = NULL,
  rast_names = NULL,
  on_the_fly = FALSE,
  progress = TRUE
)
```

Arguments

`points_layer`	An object of class 'SpatVector', 'SpatialPoints', 'SpatialPointsDataFrame' or 'sf', or a string with the path to a vector file.
`rast_layer`, `ext_rast_layer`	An object of class 'SpatRaster', 'RasterLayer', 'RasterStack', 'RasterBrick', or a list of raster objects (any of 'RasterLayer' or 'SpatRaster').
`radii`	A numeric vector with the radii (in meters) from which buffers will be created.
`classnames`	A list matching each raster value with a class name. See Details.
`site_ref`	A string with the name of the column containing the identity of the sites in points layer data (argument `points_layer`). See Details.
`bufftype`	Type of buffer to be created: "round" for circular buffers (default) or "square".
`segs`	Number of line segments to use to approximate a quarter circle during buffer generation. Only valid when `bufftype = "round"`. Default is 20.
`rast_names`	A character vector with the names of the rasterlayers provided in `rast_layer` and `ext_rast_layer`. See Details.
`on_the_fly`	Logical. If FALSE (default) intersections between buffers and rasterlayers will be calculated. If TRUE, only buffers will be generated. See Details.
`progress`	Logical. If TRUE (default), progress of the analysis will be printed.

Details

`mland()` is the primary function of the package. It creates an object of class 'MultiLand' that holds relevant objects and information about points, buffers and intersections between the latters and rasterlayers.

The function firstly creates buffers with center in the sites defined in `points_layer`, and size defined by the values of `radii`. If each point defined in `points_layer` has an associated name or id for ulterior identification, the user should provide the name of the attribute inside `points_layer` containing this information, by passing this string through the argument `site_ref`.

Argument `rast_layer` must be provided with rasterlayers with discrete values from which different landscape metrics (provided by package landscapemetrics) could be calculated. Extra rasterlayers can be provided in `ext_rast_layer`, from which other metrics can be calculated. For instance, an extra rasterlayer could be one depicting continuous values of slope, from which a mean value per landscape may be calculated. Raster layers should be in a coordinate reference system with meters as the unit, and all raster and vector layers must be in the same coordinate reference system. The user may check the validity of the raster layer with `check_raster()`.

If `on_the_fly = FALSE` (default), intersections between buffers and rasterlayers defined in `rast_layer` and/or `ext_rast_layer` will be generated. Otherwise, if `on_the_fly = TRUE`, only buffers will be generated. The latter approach may be particularly useful for 'MultiLand' objects with numerous points (hundreds or thousands), in order to avoid returning an object excessively heavy for memory.

If this is the case, intersections between buffers and rasterlayers will be generated when required ("on the fly"). For instance, to calculate metrics with `mland_metrics()`.

The names of the provided rasterlayers can be defined in `rast_names`. If so, this must be a character vector with as many names (strings) as provided rasterlayers in arguments `rast_layer` and `ext_rast_layer`, in the mentioned order. If there is no need of a name for a particular rasterlayer, the given element in the vector should be `NA`. Definition of these names could be useful when applying other functions of the package to the object generated here. For instance, to get the name of the rasterlayer of a particular row of the data.frame with landscape metrics exported by `mland_metrics()`.

Classes names can be associated with each value of the rasterlayers defined in `rast_layer`, for a easier future identification. If so, the user must provide a list with as many elements as rasterlayers defined in `rast_layer`. If a 'SpatRaster' with multiple layers, a 'RasterStack' or a 'RasterBrick' is provided, the number of rasterlayers are extracted from these objects. Each element of the list must be a vector built from concatenated pairs of values, with the value of the raster (the class) in the first place and the associated class name in the second place. For example, in the case only one rasterlayer is provided (with four unique values: 1, 2, 3 and 4), a plausible definition for the argument `classnames` could be the following:

```
list(c(1, "Forest", 2, "Crops", 3, "Urban", 4, "Grassland"))
```

If, for instance, two rasterlayers are provided (with four unique values for the first layer, and two unique values for the second one), a plausible definition would be:

```
list(c(1, "Forest", 2, "Crops", 3, "Urban", 4, "Grassland"),
     c(1, "Burnt Areas", 2, "Non-burnt Areas"))
```

Value

An object of class 'MultiLand'. This object can be used to generate useful plots with `mland_plot()`, calculate metrics with `mland_metrics()` and calculate buffer's overlapping with `mland_overlap()`. See ?MultiLand for more details on the content of this object.

See Also

`mland_plot()`, `mland_metrics()`, `mland_overlap()`, `generate_points()`

Examples

```
# Loads main raster with land covers
elchaco <- terra::rast(system.file("extdata", "elchaco.tif", package = "multilandr"))

# Main raster should have discrete values (e.g. land covers). This can be
# checked with the function check_raster():

check_raster(elchaco)

# Loads extra raster with NDVI values
elchaco_ndvi <- terra::rast(system.file("extdata", "elchaco_ndvi.tif",
                                        package = "multilandr"))
```

```
# Classes names
cl_names <- c(1, "Forest",
              2, "Grassland",
              3, "Crops",
              4, "Pastures",
              5, "Water",
              6, "Urban")

# Loads points
elchaco_sites <- terra::vect(system.file("extdata", "elchaco_sites.gpkg",
                                          package = "multilandr"))

# Creates 'MultiLand' object by loading main raster, an extra raster and points.
ernesdesign <- mland(points_layer = elchaco_sites,
                     rast_layer = elchaco,
                     radii = seq(1000, 5000, 1000),
                     classnames = list(cl_names),
                     site_ref = "name",
                     ext_rast_layer = elchaco_ndvi,
                     rast_names = c("landuse", "NDVI"),
                     segs = 20)

# Returns basic information about the object
ernesdesign

# Returns the classes of each rasterlayer and its names, if initially provided
ernesdesign@classes

## Not run:
# Loads another main raster. Same classes as "elchaco", but a different year.
elchaco2 <- terra::rast(system.file("extdata", "elchaco2.tif",
                                    package = "multilandr"))

# Creates 'MultiLand' with two rasterlayers.
ernesdesign2 <- mland(points_layer = elchaco_sites,
                      rast_layer = list(elchaco, elchaco2),
                      radii = seq(1000, 5000, 1000),
                      classnames = list(cl_names, cl_names),
                      site_ref = "name")

# Creates the same object but with "on_the_fly = T". Intersections between
# buffers and rasters will not be generated in this step
ernesdesign3 <- mland(points_layer = elchaco_sites,
                      rast_layer = list(elchaco, elchaco2),
                      radii = seq(1000, 5000, 1000),
                      classnames = list(cl_names, cl_names),
                      site_ref = "name",
                      on_the_fly = T)

# Creates a MultiLand object with hundreds of points. In this case, these
# points were generated with generate_points(), another function from this
# package. Also, "on_the_fly = TRUE" assures that no intersections between buffers
# and the raster are created in this step.

# Loads points
otf_sites <- terra::vect(system.file("extdata", "otf_sites.gpkg",
                                     package = "multilandr"))

# Creates MultiLand object
otf_design <- mland(points_layer = otf_sites,
                    rast_layer = elchaco,
                    radii = 2000,
                    classnames = list(c(1, "Forest",
                                        2, "Grassland",
                                        3, "Crops",
                                        4, "Pastures",
                                        5, "Water",
                                        6, "Urban")),
                    on_the_fly = TRUE)

## End(Not run)
```

`mland_export_gis` *Exports a 'MultiLand' object as GIS data*

Description

Exports points, buffers and intersections between buffers and rasterlayers, as vector and raster files.

Usage

```
mland_export_gis(
  x,
  raster = NULL,
  points = NULL,
  radii = NULL,
  ext_raster = NULL,
  name = NULL,
  gdal = c("COMPRESS=DEFLATE", "PREDICTOR=2", "ZLEVEL=9"),
  ...
)
```

Arguments

`x`	An object of class 'MultiLand' generated with `mland()`.
`raster, ext_raster`	Numeric. The rasterlayers to be exported.
`points`	Numeric or character vector of points to be processed. See Details.
`radii`	Numeric vector of radii to be processed.
`name`	Character. Name of the zip file where files will be exported.

`gdal`	GeoTiff creation options for rasters (GeoTiff file format). `mland_export_gis()` uses the following compression options: c("COMPRESS=DEFLATE", "PREDICTOR=2", "ZLEVEL=9").
`...`	Other arguments passed to terra::writeRaster.

Details

If argument `points` is a character vector, `mland_export_gis()` will assume that the 'MultiLand' object inputted in argument `x` was created with `site_ref` = `TRUE`. This is, there is an attribute in points layer data with the names for each individual point. Therefore, the inputted values in argument `points` will be taken as these identification names. Otherwise, if a numeric vector is declared, the inputted values will be taken as the automatically generated point ids (created when running `mland()`).

Value

GIS data from a 'MultiLand' object is exported through a zip file.

See Also

`mland()`, `mland_save()`, `mland_load()`

Examples

```
## Not run:
# Loads a 'MultiLand' object
ernesdesign <- system.file("extdata", "ernesdesign.zip", package = "multilandr")
ernesdesign <- mland_load(ernesdesign)

# Exports as GIS data
mland_export_gis(ernesdesign, dir = "ernesdesign")

## End(Not run)
```

`mland_load` *Load 'MultiLand' or 'MultiLandMetrics' object*

Description

Imports a zip file into an object of class 'MultiLand' that was previously saved with `mland_save()`. Alternatively, loads to the environment an RDS object depicting a 'MultiLandMetrics' object.

Usage

```
mland_load(path, ...)
```

Arguments

`path`	A string depicting the path to a zip file, to load objects of class 'MultiLand', or to a RDS file to load objects of class 'MultiLandMetrics'.
`...`	Other parameters passed to `readRDS()` when trying to load an object of class 'MultiLandMetrics'.

Value

A 'MultiLand' or a 'MultiLandMetrics' object.

See Also

`mland_save()`, `mland()`, `mland_metrics()`

Examples

```
# Loads mland object from a zip file, previously created with mland_save()
mland_obj <- system.file("extdata", "ernesdesign.zip", package = "multilandr")
ernesdesign <- mland_load(mland_obj)

# Loads a MultiLandMetrics object previously generated with mland_metrics() and
# exported as a RDS object with mland_save() or saveRDS()

mlm_obj <- system.file("extdata", "ed_metrics.rds", package = "multilandr")
ed_metrics <- mland_load(mlm_obj)
```

`mland_metrics` *Calculates landscape metrics*

Description

Calculates landscape metrics of patch, class and/or landscape level via the package landscapemetrics and user-defined functions from an object of class 'MultiLand'.

Usage

```
mland_metrics(
  x,
  raster = NULL,
  points = NULL,
  radii = NULL,
  classes = NULL,
  level = NULL,
  metric = NULL,
  name = NULL,
  type = NULL,
  what = NULL,
  report_absences = TRUE,
  absence_values = NULL,
  ext_calc = NULL,
  na.exclude = TRUE,
  coords = FALSE,
  update = NULL,
  output = "MLM",
  progress = TRUE,
  ...
)
```

Arguments

`x`	An object of class 'Multiland' generated with `mland()`.
`raster`	Vector depicting the rasterlayers of `x` from which metrics will be calculated. It can be a numeric vector, for rasterlayer numbers, or a character vector, for rasterlayer names (if provided during the generation of `x`). If NULL, all rasterlayers will be considered.
`points`	Numeric or character vector with the points from which metrics will be calculated. If NULL, all points will be considered. See Details.
`radii`	Numeric vector depicting the radii from which metrics will be calculated. If NULL, all radii will be considered.
`classes`	List containing the classes or classes names from which metrics will be calculated. If NULL, all classes will be considered. See Details.
`level`, `metric`, `name`, `type`, `what`	Arguments passed to `landscapemetrics::calculate_lsm()`, which define which metrics will be calculated. See Details.
`report_absences`	Logical. If TRUE (default), intersections with absences of particular classes will be returned in the final data.frame. See Details.
`absence_values`	A list depicting which value for each class-level metric should be printed if `report_absences = TRUE`. See Details.
`ext_calc`	A list containing vectors, each one of length equal to 2 or more: the first element of the vector with the identification number of the extra rasterlayer defined in `x`, and next elements with a string with the name of the function to be applied to the defined raster. See Details.
`na.exclude`	Logical. Whether to exclude (default) or not the NA values when performing extra calculations to extra rasterlayers. Only applies if `ext_calc` is not NULL. See Details.
`coords`	Logical. If TRUE, the coordinates of the points will be returned in the data.frame containing the values of the required metrics. Default FALSE.
`update`	An object of class 'MultiLandMetrics', if it is intended to be updated with new or updated metrics data. See Details.
`output`	Either "MLM" (default) to output an object of class 'MultiLandMetrics' or "data" to output only the data.frame with metric values.

`progress`	Logical. If TRUE (default), progress of calculations will be printed.
`...`	Other arguments passed to `landscapemetrics::calculate_lsm()`. See Details.

Details

Calculates landscape metrics from an object of class `MultiLand` created with `mland()`. The function allows to define which metrics will be calculated in the form defined by the function `calculate_lsm()`] from package landscapemetrics, by specifying one or more of the following arguments:

- level: level of metrics. Either "patch", "class" or "landscape" (or vector with combination).
- metric: abbreviation of metrics (e.g. "area").
- name: full name of metrics (e.g. "core area").
- type: type according to FRAGSTATS grouping (e.g. "aggregation metrics").
- what: selected level of metrics: either "patch", "class" or "landscape". It is also possible to specify functions as a vector of strings, e.g. what = c("lsm_c_ca", "lsm_l_ta").

Available metrics can be seen in `metrics_list()` and in the associated documentation of package `landscapemetrics`.

`mland_metrics()` also allows to define some other parameters that filter how metrics are calculated, by defining the rasterlayers, points, radii and classes to be taken into account.

If `report_absences = TRUE` (default), the function will print values of class-level metrics from classes that are not present in particular landscapes, as a distinct row in the final data.frame. This is particularly useful for certain class-level metrics in which the absence of the class should be acknowledged, for instance, the percentage of landscape ('pland') for a forest class. For this metric, a value of 0 (zero) should be printed for those landscapes where the class forest is not present. By default, if `report_absences = TRUE`, the function will consider `NA` as the value to be declared in the case that the class is absent in the landscape. To declare a different value for a particular class-level metric, this can be declared inside argument `absence_values`. If not NULL, this must be a list with the value that one ore more class-level metric should have in the case of an absence of a class. For example, in the case of "pland", the argument should be defined as follows: `absence_values = list("pland" = 0)`. Note that the metric must be identified with its abbreviation. You can see abbreviations for all available metrics in `metrics_list()`, under the column "metric".

If argument `points` is a character vector, `mland_metrics()` assumes that the 'MultiLand' object inputted in argument `x` was created with `site_ref = TRUE`. This is, there is an column/attribute in the points layer with the names for each distinct point. Therefore, the inputted values in argument `points` will be taken as these identification names. Otherwise, if a numeric vector is inputted, these values will be taken as the automatically generated point ids (created when running `mland()`).

The user may specify which classes will be considered when calculating the metrics, by passing this information in the argument `classes`. Of course, this information only applies for class-level metrics. The argument must be a list with as many elements as rasterlayers to be considered (defined in argument `raster`, in ascending order: 1, 2, 3, ...). Each element must be a numeric vector with the classes values (raster values) to be considered, or a character vector with the names of the classes (if provided when generating `x`).

Other arguments can be passed to function `landscapemetrics::calculate_lsm()` through argument `...`. These include specific arguments relative to the calculation of particular landscape metrics. See the documentation of this function for more information.

Extra calculations can be performed through `ext_calc`. The functions defined here will take the values of the extra rasterlayers defined in `x` as input. For instance, a plausible definition could be `ext_calc = list(1, "mean")`, which will take the values from the extra rasterlayer 1, and calculate its mean for each landscape. If `na.exclude = TRUE` (default), NA values will be excluded from this task.

A previously generated 'MultiLandMetrics' object can be updated with new information regarding other metrics, probably from other points, radii, rasterlayers, etc, that haven´t been calculated in the previous time (or not). In this way, the returned object will be the object provided in this argument, plus the additions of information about new metrics, and changes to previously metric calculations. Note that if a particular metric is calculated for a given rasterlayer, points, radii and or class, that were previously generated in the object provided in `update`, the information of these metrics from the latter will be overwritten. Also note that if in the previous 'MultiLandMetrics' object `report_absences` was `TRUE` for a given set of metrics and other parameters (e.g. points, radii, rasterlayers, etc.), and in the new call `report_absences` is `FALSE` (for the same set of other parameters), the rows depicting landscapes with empty classes from the previous call will be mantained. If the intention is the removal of these rows, the user should create a fresh new 'MultiLandMetrics' from scratch.

Value

If `output = "MLM"`, an object of class 'MultiLandMetrics' will be returned. This object can then be passed to functions `metrics_corr()`, metrics_plots, `metrics_filter()`, `metrics_gradient()` and `metrics_bind()`. See ?MultiLandMetrics for more information regarding the content of this object. Otherwise, if `output = "data"`, only a data.frame with the calculated metrics will be returned.

See Also

`metrics_corr()`, `metrics_plots()`, `metrics_filter()`, `metrics_gradient()`, `metrics_bind()`

Examples

```
## Not run:
# Loads a 'MultiLand' object
ernesdesign <- system.file("extdata", "ernesdesign.zip", package = "multilandr")
ernesdesign <- mland_load(ernesdesign)

# Creates a 'MultiLandMetrics' object. It will calculate the "percentage of landscape"
# ("pland") and "number of patches" ("np") for all classes. Note that an absence value
# for each metric is declared, as the absence of a class for these metrics should be
# acknowledged as a 0 (percentage of zero and zero patches).
ed_metrics <- mland_metrics(ernesdesign, level = "class", metric = c("pland", "np"),
                            absence_values = list("pland" = 0, "np" = 0))

# Returns data.frame with the values of all metrics for each landscape
head(ed_metrics@data)

# Shows which metrics were calculated and are contained in the data.frame
ed_metrics@metrics
```

```
# If output = "data", only the data.frame will be returned
data <- mland_metrics(ernesdesign, level = "class", metric = "pland",
                      classes = c("Forest", "Crops"),
                      absence_values = list("pland" = 0),
                      output = "data")

# Calculate landscape metrics plus extra calculations for extra rasterlayer 1,
# the mean value, and a user defined function, which is the mean divided
# standard deviation.

# User-defined function
mean_sd <- function(x){ mean(x)/sd(x) }

ed_metrics2 <- mland_metrics(ernesdesign, level = "class",
                             metric = c("pland", "np"),
                             absence_values = list("pland" = 0, "np" = 0),
                             ext_calc = list(c(1, "mean"), c(1, "mean_sd")))

# We can calculate metrics for extra rasterlayers only
ed_metrics3 <- mland_metrics(ernesdesign, ext_calc = list(c(1, "mean", "mean_sd")))

# If metrics of different levels must be calculated, a better approach is to declare
# them inside the argument 'what', by naming the function associated with the metric.
# Also in this case, only the landscapes with a radius of 5000 m are considered.
# A list of available metrics with its names, abbreviations and function names can
# be seen in metrics_list() and in the documentation of the package landscapemetrics.
ed_metrics4 <- mland_metrics(ernesdesign,
                             what = c("lsm_c_area_mn", "lsm_l_np", "lsm_l_shdi"),
                             radii = 5000)

# Calculates patch-level metrics of a particular landscape
ed_patchs <- mland_metrics(ernesdesign, points = "Algarrobo",
                           level = "patch", class = "Forest",
                           radii = 1000)

## End(Not run)
```

---

mland_overlap *Buffers overlapping*

Description

Returns matrices informing the degree of overlapped area between buffers of a 'MultiLand' object.

Usage

```
mland_overlap(
  x,
  points = NULL,
  radii = NULL,
  digits = 2,
  perc = TRUE,
  title = "id"
)
```

Arguments

x	An object of class 'Multiland' generated with mland().
points	Numeric or character vector depicting the points to be considered. If NULL, all points will be taken into account. See Details.
radii	Numeric vector depicting the radii to be considered. If NULL, all radii will be taken into account.
digits	Numeric. Number of digits for the values of overlapped areas. Default is 2.
perc	Logical. If TRUE (default) the degree of overlapped areas will be presented as percentages. If FALSE, proportions will be outputted.
title	One of the following: "id" to output each point with its id (default), or "sitename" to output each point with its pre-defined point name in x.

Details

If argument points is a character vector, mland_overlap() will assume that the 'MultiLand' object inputted in argument x was created with site_ref = TRUE. This is, there is a column/attribute in the points layer with the names for each distinct point. Therefore, the inputted values in argument points will be taken as these identification names. Otherwise, if a numeric vector is declared, the inputted values will be taken as the automatically generated point ids (created when executing mland()).

Value

A list with as many elements as different radius in x. Each element contains a matrix with the percentages (or proportions if perc = FALSE) of overlapping of buffer areas.

Examples

```
## Not run:
# Loads a 'MultiLand' object
ernesdesign <- system.file("extdata", "ernesdesign.zip", package = "multilandr")
ernesdesign <- mland_load(ernesdesign)

# Returns a matrix with the percentage of overlapping between buffers of each radii
mland_overlap(ernesdesign)

# Selects only one radius and return the site names rather than the ids
mland_overlap(ernesdesign, radii = 5000, title = "sitename")

## End(Not run)
```

mland_plot *Plots landscapes from 'MultiLand' objects*

Description

Returns multiple plots for each landscape generated from each point and buffer, with their radii and classes, defined by the user through a 'MultiLand' object (generated by `mland()`). Aesthetic parameters of plots can be customized.

Usage

```
mland_plot(
  x,
  raster = NULL,
  points = NULL,
  radii = NULL,
  ext_raster = NULL,
  title = "id",
  ncol = NULL,
  nrow = NULL,
  st_points = list(shape = 21, size = 2, col = "black", fill = "white", alpha = 1),
  st_buffers = list(lty = 1, lwd = 1, col = "black", alpha = 0.6),
  st_classes = list(palette = "Spectral", fill = NULL, alpha = NULL, na_value =
    c("white", 1)),
  st_ext = c("chartreuse", "firebrick1")
)
```

Arguments

`x`	An object of class 'MultiLand' generated with `mland()`.
`raster, ext_raster`	Numeric. The rasterlayer to be plotted. Only one rasterlayer can be plotted at the same time, either defined in `raster` or `ext_raster`.
`points`	Numeric or character vector of points to be plotted. See Details.
`radii`	Numeric vector of radii to be plotted.
`title`	One of the following: "id" to plot titles as each point id (default), or "sitename" to plot titles as each pre-defined point name in `x`. See Details.
`ncol, nrow`	Number of columns and rows wherein individual plots will be arranged.
`st_points`	List of aesthetic arguments for points plotting: `shape` for points shape, `size` for points size, `col` for points border color, `fill` for points fill color and `alpha` for point transparency.
`st_buffers`	List of aesthetic arguments for buffers plotting: `lty` for buffers linetype, `lwd` for buffers linewidth, `col` for buffers border color and `alpha` for border transparency.

`st_classes`	List of aesthetic arguments for classes plotting: `palette`, for classes color palette, `fill` a vector of fill colors for classes, `alpha`, a vector of alpha values for classes, and `na_value` for the color of NA values. See Details.
`st_ext`	Character vector of length 2, depicting the color for the minimum and maximum values of the raster defined in `ext_raster`.

Details

If argument `points` is a character vector, `mland_plot()` will assume that the 'MultiLand' object inputted in argument `x` was created with `site_ref = TRUE`. This is, there is a column/attribute in points layer data with the names for each distinct point. Therefore, the inputted values in argument `points` will be taken as these identification names. Otherwise, if a numeric vector is inputted, these values will be taken as the automatically generated point ids (created when running `mland()`).

If `title = "sitename"`, the title of individual plots will be the names of each point. For this, the names of the points in `x` must had been defined when the object was created with `mland()` (i.e. `x@site_ref = TRUE`). Otherwise, the argument will be ignored and the titles will be the ids of the points.

A pre-defined palette can be chosen to differentiate classes inside `palette = "palette_name"`, inside the list defined in `st_classes`. Any palette from `hcl.pals()` can be chosen. Otherwise, the user can define specific colors for each class, inside `fill`. This must be a vector built with concatenated pair of values, the first value being the class (or class name, if defined during `x` generation), and the second value the color (either the name of the color or the hex code of the color). For example, in the case the rasterlayer has four unique values: (1, 2, 3 and 4), a plausible color definition could be the following:

```
list(c(1, "green", 2, "red", 3, "black", 4, "yellow"))
```

Value

Multiple plots (in a unique plotting device) of landscapes around defined points, radii and classes of a MultiLand object.

Examples

```
## Not run:
# Loads a 'MultiLand' object
ernesdesign <- system.file("extdata", "ernesdesign.zip", package = "multilandr")
ernesdesign <- mland_load(ernesdesign)

# Plots all points and radii
mland_plot(ernesdesign)

# Plots points 1 to 3 and only radius 3000 m
mland_plot(ernesdesign, points = 1:3, radii = 3000)

# Plot with pre-defined colors, and specifying other arguments
cols <- c(1, "forestgreen",
          2, "darkolivegreen2",
          3, "firebrick3",
                4, "goldenrod1",
                5, "deepskyblue3",
                6, "black")

mland_plot(ernesdesign, points = 9:11, radii = c(1000, 2000, 3000),
           title = "sitename", nrow = 1,
           st_points = list(shape = 9),
           st_buffers = list(lty = "dashed"),
           st_classes = list(fill = cols))

# Plot a unique landscape by calling it with its name
mland_plot(ernesdesign, points = "Peje", title = "sitename",
           st_points = list(shape = 15, col = "red"),
           st_classes = list(palette = "Hawaii"))

# Plot extra rasterlaer
mland_plot(ernesdesign, radii = 3000, ext_raster = 1, title = "sitename")

# Plot extra rasterlater with customized colors
mland_plot(ernesdesign, radii = 3000, ext_raster = 1, title = "sitename",
           st_ext = c("blue", "red"))

## End(Not run)
```

`mland_save` *Saves a 'MultiLand' or 'MultiLandMetrics' object*

Description

Exports an object of class 'MultiLand' to be read in the future with `mland_load()`, or an object of class 'MultiLandMetrics' as if it was saved with `saveRDS()`.

Usage

```
mland_save(
  x,
  name = NULL,
  gdal = c("COMPRESS=DEFLATE", "PREDICTOR=2", "ZLEVEL=9"),
  ...
)
```

Arguments

`x`	Object of class 'MultiLand' or 'MultiLandMetrics'.
`name`	If `x` is an object of class 'MultiLand', the name of the zip file where files will be saved (without the '.zip'). If `x` is an object of class 'MultiLandMetrics', the name of the R file (.rds). If NULL (default), the name will be 'mland_' or 'mlandmetrics_' + a large random number.

`gdal`	GeoTiff creation options for rasters (GeoTiff file format). `mland_save()` uses the following compression options: c("COMPRESS=DEFLATE", "PREDICTOR=2", "ZLEVEL=9"). Only relevant if x is an object of class 'MultiLand'.
`...`	If x is an object of class 'MultiLand', ... should depict other arguments passed to terra::writeRaster, the function to write rasterlayers (from intersections and plain rasterlayers). Otherwise, if x is an object of class 'MultiLandMetrics', ... should depict other arguments passed to `save()`. See Details.

Details

'MultiLand' objects should be exported with this function rather than exporting as an external representation of R objects with `saveRDS()`. This is because objects of classes 'SpatVector' and 'SpatRaster' (from package terra) contained inside a 'MultiLand' object cannot be exported as regular R objects. The exported object will be a zip file, and can be loaded again into an R session with `mland_load()`.

Relevant arguments can be passed to the function terra::writeRaster, which is used to write rasterlayers from a 'MultiLand' object. Particularly, in the argument `gdal` one can specify relevant options regarding raster compression. This may reduce raster sizes significantly. Definition of some other arguments inside terra::writeRaster may affect exportation of rasterlayer objects, in the context of a 'MultiLand' object.

Objects of class 'MultiLandMetrics', instead, do not contain 'SpatVector' or 'SpatRaster' objects and can be exported as regular R objects with `saveRDS()`. The user may use `saveRDS()` or `mland_save()`, and the outcome will be identical.

Value

If x is an object of class 'MultiLand', a zip file or a directory containing all information regarding the 'MultiLand' object provided in 'x'. Otherwise, if x is an object of class 'MultiLandMetrics', the function will export the R object as if it was exported as a regular R object with `saveRDS()`.

See Also

`mland_load()`, `mland()`, `mland_metrics()`

Examples

```
## Not run:
# Load MultiLand object
mland_obj <- system.file("extdata", "ernesdesign.zip", package = "multilandr")
ernesdesign2 <- mland_load(mland_obj)

# Save it again
mland_save(ernesdesign2)

# Save it again but defining a higher compression for rasterlayers
mland_save(ernesdesign2, gdal = "COMPRESS=DEFLATE")

# Loads a MultiLandMetrics object previously generated with mland_metrics()
mlm_obj <- system.file("extdata", "ed_metrics.rds", package = "multilandr")
ed_metrics2 <- mland_save(mlm_obj)
```

```
# Save it again. In this case, mland_save() is the same as using saveRDS()
mland_save(ed_metrics2)

## End(Not run)
```

MultiLand-class *Class "MultiLand"*

Description

Objects of class 'MultiLand' are created with the function `mland()`, and holds relevant objects and information to be passed to other functions of the package. The slot `@buffers` holds an object of class 'SpatVector' with the buffers for each point contained in the slot `@points` and each radius defined in slot `@radii`. The slot `@buffers@data` holds a data.frame with the identification of each buffer (individualized by a point id and a radius value).

Details

If the slot `@on_the_fly` equals `FALSE`, the slot `@landscapes` holds the intersections (objects of class 'SpatRaster') between buffers and the rasterlayers inputted by the user when running `mland()`. Intersections between buffers and rasterlayers with discrete values (inputted in argument `raster` in `mland()`) are contained inside a list named 'lsm_rasters', whereas intersections between extra rasterlayers (inputted in argument `ext_rast_layer` in `mland()`) and buffers are contained inside a list named 'ext_rasters'. Each list ('lsm_rasters' and 'ext_rasters') contains a list with as many elements as previously inputted rasterlayers. Additionally, each element of this latter list holds an additional internal list, with as many elements as intersections (i.e. rasters generated by the intersections between buffers defined by each point and radius, and the rasterlayer). The name of each element of each internal list reveals the information related to a given intersection, with the following coding: "RasterLayerL-P-R" or "ExtRasterLayerL-P-R", where L is the given rasterlayer, P is the id of the point and R is the radius. For example, a plausible intersection may be named as "RasterLayer1-5-1500", indicating that this element holds a raster layer which is the result of the intersection between RasterLayer1 and the buffer around point 5 and radius 1500 m.

If slot `@on_the_fly` equals `FALSE`, the slot `@landscapes` holds a list containing two named lists as 'lsm_rasters' and 'ext_rasters'. Each one contains a list with as many rasterlayers were initially inputted by the user when running `mland()` in arguments `rast_layer` and `ext_rast_layer`. This means that no intersections were made when creating the 'MultiLand' object. Intersections will be created "on the fly" when other functions of the package requires them.

Slots

`call` The call when function `mland()` was called.

`idkey` A unique identification id for the 'MultiLand' object.

`crs_proj` A string depicting the CRS of points layer.

`points` An object of class 'SpatVector'. Holds the points inputted by the user.

`buffers` An object of class 'SpatVector'. Holds the buffers layers.

`site_ref` String holding the name of the attribute that the user defined as the one that identifies individual points and is contained inside the layer of points.

`radii` Vector of numbers containing the radii that defined the creation of buffers.

`n_layers` Number of rasterlayers (defined in argument `raster` in `mland()`) from which i ntersections between were created (or will be if slot `@on_the_fly = TRUE`).

`n_classes` A numeric vector depicting the number of classes (raster values) per raster layer (defined in argument `raster` in `mland()`).

`classes` A data.frame depicting the classes (and classes names) for each rasterlayer (defined in argument `raster` in `mland()`).

`on_the_fly` A logical value indicating whether intersections between buffers and rasterlayers were created (FALSE) or not (TRUE).

`landscapes` If `on_the_fly` = `FALSE`, this slot holds the intersections between buffers and rasterlayers. Otherwise, if `on_the_fly` = `TRUE`, it holds the raw rasterlayers.

`l_ref` A data.frame relating each point and radius with a "row_id", equal to the position of its buffer in the slot `@buffers` and to the position of the intersection for each point/radius in the slot `@landscapes` (if `on_the_fly` = `TRUE`).

`rast_names` A list containing two data.frame with the names assigned by the user for the main rasterlayers and extra rasterlayers defined in argument `rast_layer` and `ext_rast_layer` in `mland()`.

Examples

```
# Shows information of object 'MultiLand'
## Not run:
ernesdesign
otf_design

## End(Not run)
```

`MultiLandMetrics-class` *Class 'MultiLandMetrics'*

Description

Objects of class `MultiLandMetrics` are returned by `mland_metrics()`. It holds all the information relative to the metrics that were calculated by the parameters inputted by the user. This object class can be passed to functions `metrics_corr()`, `metrics_plots()`, `metrics_filter()`, `metrics_gradient()` and `metrics_bind()` for further analyses.

Slots

`call` The call when function `mland_metrics()` was called.

`idkey` A unique identification id for the 'MultiLandMetrics' object.

`crs_proj` A string depicting the CRS of points layer.

`n_layers` Number of rasterlayers from which metrics were calculated.

`rast_names` A list with dataframes containing the names of the rasterlayers of the ’MultiLand’ object the function `mland_metrics()` worked with.

`classes` A data.frame depicting the rasterlayers, classes and classes names from which metrics were calculated.

`n_classes` Numeric vector depicting the number of distinct classes per rasterlayer from which metrics were calculated.

`points` A data.frame containing points coordinates and other attributes.

`n_points` Number of points from which metrics were calculated.

`radii` Distinct radii from which metrics were calculated.

`metrics` A data.frame depicting the metrics that were calculated, classified by level.

`data` Main data.frame with the values of the metrics that were calculated for each point, radius, rasterlayers and other parameters pre-defined by the user.

`ext_calcs` A data.frame depicting the extra calculations that were made in given extra rasterlayers.

Examples

```
# Shows information of object 'MultiLandMetrics'
## Not run:
ed_metrics
otf_metrics

## End(Not run)
```

`otf_metrics` *’MultiLandMetrics’ object*

Description

An object of class ’MultiLandMetrics’ generated with `mland_metrics()`, for the purposes of package examples of the following functions: `metrics_filter()`, `metrics_gradient()`, `metrics_corr()`, `metrics_plots()` and `metrics_bind()`. See ’MultiLand-class’ for general information about these objects.

Usage

```
otf_metrics
```

Format

An object of class `MultiLandMetrics` of length 1.

Details

The main internal object is a data.frame (accesible through `otf_metrics@data`) with information about the values of two landscape metrics: "pland" (percentage of landscape) and "np" (number of patches).

The object was created from the MultiLand object named "otf_design", which received a raster layer from a small portion of the ecoregion "El Chaco" as main input. The rasterlayer was provided by the project "MapBiomas Chaco" for the year 2000.

See Also

See the examples sections of `mland_metrics()` and `mland()` for more context.

#' @references Project MapBiomas Chaco – Collection 4.0 of annual land cover and land use maps, accessed during July 2022 through the following link: MapBiomas Chaco

---

`show,MultiLand-method` *Show 'MultiLand' object*

Description

Show 'MultiLand' object

Usage

```
## S4 method for signature 'MultiLand'
show(object)
```

Arguments

`object`	Prints relevant information about a 'MultiLand' object.

`show,MultiLandMetrics-method` *Show 'MultiLandMetrics' object*

Description

Show 'MultiLandMetrics' object

Usage

```
## S4 method for signature 'MultiLandMetrics'
show(object)
```

Arguments

`object`	Prints relevant information about a 'MultiLandMetrics' object.

Index