Package ‘landscapetools’

October 13, 2022

Type Package

Title Landscape Utility Toolbox

Version 0.5.0

Description Provides utility functions for some of the less-glamorous tasks involved in landscape analysis. It includes functions to coerce raster data to the common tibble format and vice versa, it helps with flexible reclassification tasks of raster data and it provides a function to merge multiple raster. Furthermore, 'landscapetools' helps landscape scientists to visualize their data by providing optional themes and utility functions to plot single landscapes, rasterstacks, -bricks and lists of raster.

License GPL-3

Encoding UTF-8

LazyData true

ByteCompile true

Depends R (>= 3.1.0)

URL https://ropensci.github.io/landscapetools/

BugReports https://github.com/ropensci/landscapetools/issues

RoxygenNote 6.1.1

Imports ggplot2, raster, tibble, Rcpp

Suggests testthat, covr, knitr, rmarkdown

VignetteBuilder knitr

LinkingTo Rcpp

NeedsCompilation yes

Author Marco Sciaini [aut, cre] (<https://orcid.org/0000-0002-3042-5435>),
Matthias Fritsch [aut],
Maximillian H.K. Hesselbarth [aut]
(<https://orcid.org/0000-0003-1125-9918>),
Jakub Nowosad [aut] (<https://orcid.org/0000-0002-1057-3721>),
Laura Graham [rev] (Laura reviewed the package for rOpenSci, see
https://github.com/ropensci/onboarding/issues/188),

Jeffrey Hollister [rev] (Jeffrey reviewed the package for rOpenSci, see https://github.com/ropensci/onboarding/issues/188)

Maintainer Marco Sciaini <sciaini.marco@gmail.com>

Repository CRAN

Date/Publication 2019-02-25 22:40:03 UTC

R topics documented:

---

landscapetools-package *landscapetools*

Description

landscapetools provides utility functions to work with landscape data (raster* Objects).

Author(s)

Maintainer: Marco Sciaini <sciaini.marco@gmail.com> (0000-0002-3042-5435)

Authors:

- Matthias Fritsch <matthias.fritsch@forst.uni-goettingen.de>
- Maximillian H.K. Hesselbarth <maximilian.hesselbarth@uni-goettingen.de> (0000-0003-1125-9918)
- Jakub Nowosad <nowosad.jakub@gmail.com> (0000-0002-1057-3721)

Other contributors:

- Laura Graham (Laura reviewed the package for rOpenSci, see https://github.com/ropensci/onboarding/issues/188) [reviewer]
- Jeffrey Hollister (Jeffrey reviewed the package for rOpenSci, see https://github.com/ropensci/onboarding/issues/188) [reviewer]

See Also

Useful links:

- `https://ropensci.github.io/landscapetools/`
- Report bugs at `https://github.com/ropensci/landscapetools/issues`

`classified_landscape` *Example map (factor).*

Description

An example map to show landscapetools functionality generated with the nlm_random() algorithm with factorial values.

Usage

```
classified_landscape
```

Format

A raster layer object.

Source

Simulated neutral landscape models with R. `https://github.com/ropensci/NLMR/`

`fractal_landscape` *Example map (fractional brownian motion).*

Description

An example map to show landscapetools functionality generated with the nlm_fbm() algorithm.

Usage

```
fractal_landscape
```

Format

A raster layer object.

Source

Simulated neutral landscape models with R. https://github.com/ropensci/NLMR/

`gradient_landscape` *Example map (planar gradient).*

Description

An example map to show landscapetools functionality generated with the nlm_planargradient() algorithm.

Usage

```
gradient_landscape
```

Format

A raster layer object.

Source

Simulated neutral landscape models with R. https://github.com/ropensci/NLMR/

`random_landscape` *Example map (random).*

Description

An example map to show landscapetools functionality generated with the nlm_random() algorithm.

Usage

```
random_landscape
```

Format

A raster layer object.

Source

Simulated neutral landscape models with R. https://github.com/ropensci/NLMR/

show_landscape *show_landscape*

Description

Plot a Raster* object with the NLMR default theme (as ggplot).

Usage

```
show_landscape(x, xlab, ylab, discrete, unique_scales, n_col, n_row, ...)

## S3 method for class 'RasterLayer'
show_landscape(x, xlab = "Easting",
  ylab = "Northing", discrete = FALSE, ...)

## S3 method for class 'list'
show_landscape(x, xlab = "Easting", ylab = "Northing",
  discrete = FALSE, unique_scales = FALSE, n_col = NULL,
  n_row = NULL, ...)

## S3 method for class 'RasterStack'
show_landscape(x, xlab = "Easting",
  ylab = "Northing", discrete = FALSE, unique_scales = FALSE,
  n_col = NULL, n_row = NULL, ...)

## S3 method for class 'RasterBrick'
show_landscape(x, xlab = "Easting",
  ylab = "Northing", discrete = FALSE, unique_scales = FALSE,
  n_col = NULL, n_row = NULL, ...)
```

Arguments

x	Raster* object
xlab	x axis label, default "Easting"
ylab	y axis label, default "Northing"
discrete	If TRUE, the function plots a raster with a discrete legend.
unique_scales	If TRUE and multiple raster are to be visualized, each facet can have a unique color scale for its fill
n_col	If multiple rasters are to be visualized, n_col controls the number of columns for the facet
n_row	If multiple rasters are to be visualized, n_row controls the number of rows for the facet
...	Arguments for theme_nlm

Value

ggplot2 Object

Examples

```
## Not run:
x <- gradient_landscape

# classify
y <- util_classify(gradient_landscape,
                   n = 3,
                   level_names = c("Land Use 1", "Land Use 2", "Land Use 3"))

show_landscape(x)
show_landscape(y, discrete = TRUE)

show_landscape(list(gradient_landscape, random_landscape))
show_landscape(raster::stack(gradient_landscape, random_landscape))

show_landscape(list(gradient_landscape, y), unique_scales = TRUE)


## End(Not run)
```

---

theme_nlm *theme_nlm*

Description

Opinionated ggplot2 theme to visualize NLM raster.

Usage

```
theme_nlm(base_family = NA, base_size = 11.5,
  plot_title_family = base_family, plot_title_size = 18,
  plot_title_face = "bold", plot_title_margin = 10,
  subtitle_family = NA, subtitle_size = 13, subtitle_face = "plain",
  subtitle_margin = 15, strip_text_family = base_family,
  strip_text_size = 12, strip_text_face = "plain",
  strip.background = "grey80", caption_family = NA, caption_size = 9,
  caption_face = "plain", caption_margin = 10,
  axis_text_size = base_size, axis_title_family = base_family,
  axis_title_size = 9, axis_title_face = "plain",
  axis_title_just = "rt", plot_margin = ggplot2::unit(c(0, 0, 0, 0),
  "lines"), grid_col = "#cccccc", grid = TRUE, axis_col = "#cccccc",
  axis = FALSE, ticks = FALSE, legend_title = "Z",
  legend_labels = NULL, legend_text_size = 8, legend_title_size = 10,
  ratio = 1, viridis_scale = "D", ...)

theme_nlm_discrete(base_family = NA, base_size = 11.5,
  plot_title_family = base_family, plot_title_size = 18,
  plot_title_face = "bold", plot_title_margin = 10,
  subtitle_family = NA, subtitle_size = 13, subtitle_face = "plain",
  subtitle_margin = 15, strip_text_family = base_family,
  strip_text_size = 12, strip_text_face = "plain",
  strip.background = "grey80", caption_family = NA, caption_size = 9,
  caption_face = "plain", caption_margin = 10,
  axis_text_size = base_size, axis_title_family = base_family,
  axis_title_size = 9, axis_title_face = "plain",
  axis_title_just = "rt", plot_margin = ggplot2::unit(c(0, 0, 0, 0),
  "lines"), grid_col = "#cccccc", grid = TRUE, axis_col = "#cccccc",
  axis = FALSE, ticks = FALSE, legend_title = "Z",
  legend_labels = NULL, legend_text_size = 8, legend_title_size = 10,
  ratio = 1, viridis_scale = "D", ...)

theme_nlm_grey(base_family = NA, base_size = 11.5,
  plot_title_family = base_family, plot_title_size = 18,
  plot_title_face = "bold", plot_title_margin = 10,
  subtitle_family = NA, subtitle_size = 13, subtitle_face = "plain",
  subtitle_margin = 15, strip_text_family = base_family,
  strip_text_size = 12, strip_text_face = "plain",
  strip.background = "grey80", caption_family = NA, caption_size = 9,
  caption_face = "plain", caption_margin = 10,
  axis_text_size = base_size, axis_title_family = base_family,
  axis_title_size = 9, axis_title_face = "plain",
  axis_title_just = "rt", plot_margin = ggplot2::unit(c(0, 0, 0, 0),
  "lines"), grid_col = "#cccccc", grid = TRUE, axis_col = "#cccccc",
  axis = FALSE, ticks = FALSE, legend_title = "Z",
  legend_labels = NULL, legend_text_size = 8, legend_title_size = 10,
  ratio = 1, ...)

theme_nlm_grey_discrete(base_family = NA, base_size = 11.5,
  plot_title_family = base_family, plot_title_size = 18,
  plot_title_face = "bold", plot_title_margin = 10,
  subtitle_family = NA, subtitle_size = 13, subtitle_face = "plain",
  subtitle_margin = 15, strip_text_family = base_family,
  strip_text_size = 12, strip_text_face = "plain",
  strip.background = "grey80", caption_family = NA, caption_size = 9,
  caption_face = "plain", caption_margin = 10,
  axis_text_size = base_size, axis_title_family = base_family,
  axis_title_size = 9, axis_title_face = "plain",
  axis_title_just = "rt", plot_margin = ggplot2::unit(c(0, 0, 0, 0),
  "lines"), grid_col = "#cccccc", grid = TRUE, axis_col = "#cccccc",
  axis = FALSE, ticks = FALSE, legend_title = "Z",
  legend_labels = NULL, legend_text_size = 8, legend_title_size = 10,
  ratio = 1, ...)

theme_facetplot(base_family = NA, base_size = 11.5,
  plot_title_family = base_family, plot_title_size = 18,
  plot_title_face = "bold", plot_title_margin = 10,
  subtitle_family = NA, subtitle_size = 13, subtitle_face = "plain",
  subtitle_margin = 15, strip.background = "grey80",
  caption_family = NA, caption_size = 9, caption_face = "plain",
  caption_margin = 10, ratio = 1, viridis_scale = "D", ...)

theme_facetplot_discrete(base_family = NA, base_size = 11.5,
  plot_title_family = base_family, plot_title_size = 18,
  plot_title_face = "bold", plot_title_margin = 10,
  subtitle_family = NA, subtitle_size = 13, subtitle_face = "plain",
  subtitle_margin = 15, strip.background = "grey80",
  caption_family = NA, caption_size = 9, caption_face = "plain",
  caption_margin = 10, ratio = 1, viridis_scale = "D", ...)
```

Arguments

base_family	base font family size
base_size	base font size
plot_title_family	plot title family
plot_title_size	plot title size
plot_title_face	plot title face
plot_title_margin	plot title ggplot2::margin
subtitle_family	plot subtitle family
subtitle_size	plot subtitle size
subtitle_face	plot subtitle face
subtitle_margin	plot subtitle ggplot2::margin bottom (single numeric value)
strip_text_family	facet facet label font family
strip_text_size	facet label font family, face and size
strip_text_face	facet facet label font face
strip.background	strip background
caption_family	plot caption family

`caption_size` plot caption size

`caption_face` plot caption face

`caption_margin` plot caption ggplot2::margin

`axis_text_size` axis text size

`axis_title_family` axis title family

`axis_title_size` axis title size

`axis_title_face` axis title face

`axis_title_just` axis title justification

`plot_margin` plot ggplot2::margin (specify with ‘ggplot2::margin“)

`grid_col` grid color

`grid` grid TRUE/FALSE

`axis_col` axis color

`axis` axis TRUE/FALSE

`ticks` ticks TRUE/FALSE

`legend_title` Title of the legend (default `"Z"`)

`legend_labels` Labels for the legend ticks, if used with `show_landscape` they are automatically derived.

`legend_text_size` legend text size, default 8

`legend_title_size` legend text size, default 10

`ratio` ratio for tiles (default 1, if your raster is not a square the ratio should be `raster::nrow(x) / raster::ncol(x)`)

`viridis_scale` Five options are available: "viridis - magma" (= "A"), "viridis - inferno" (= "B"), "viridis - plasma" (= "C"), "viridis - viridis" (= "D", the default option), "viridis - cividis" (= "E")

`...` optional arguments to ggplot2::theme

Details

A focused theme to visualize raster data that sets a lot of defaults for the `ggplot2::theme`.

The functions are setup in such a way that you can customize your own one by just wrapping the call and changing the parameters. The theme itself is heavily influenced by hrbrmstr and his package hrbrthemes (`https://github.com/hrbrmstr/hrbrthemes/`).

`util_as_integer` *util_as_integer*

Description

Coerces raster values to integers

Usage

```
util_as_integer(x)

## S3 method for class 'RasterLayer'
util_as_integer(x)
```

Arguments

x	raster

Details

Coerces raster values to integers, which is sometimes needed if you want further methods that rely on integer values.

Value

RasterLayer

Examples

```
# Mode 1
util_as_integer(fractal_landscape)
```

`util_binarize` *Binarize continuous raster values*

Description

Classify continuous raster values into binary map cells based upon given break(s).

Usage

```
util_binarize(x, breaks)

## S3 method for class 'RasterLayer'
util_binarize(x, breaks)
```

Arguments

`x`	Raster* object
`breaks`	Vector with one or more break percentages

Details

Breaks are considered to be habitat percentages (p). If more than one percentage is given multiple layers are written in the same brick.

Value

RasterLayer / RasterBrick

Examples

```
breaks <- c(0.3, 0.5)
binary_maps <- util_binarize(gradient_landscape, breaks)
```

`util_classify` *util_classify*

Description

Classify continuous landscapes into landscapes with discrete classes

Usage

```
util_classify(x, n, weighting, level_names, real_land, mask_val)

## S3 method for class 'RasterLayer'
util_classify(x, n = NULL, weighting = NULL,
  level_names = NULL, real_land = NULL, mask_val = NULL)
```

Arguments

`x`	raster
`n`	Number of classes
`weighting`	Vector of numeric values that are considered to be habitat percentages (see details)
`level_names`	Vector of names for the factor levels.
`real_land`	Raster with real landscape (see details)
`mask_val`	Value to mask (refers to real_land)

Details

Mode 1: Calculate the optimum breakpoints using Jenks natural breaks optimization, the number of classes is determined with n. The Jenks optimization seeks to minimize the variance within categories, while maximizing the variance between categories.

Mode 2: The number of elements in the weighting vector determines the number of classes in the resulting matrix. The classes start with the value 1. If non-numerical levels are required, the user can specify a vector to turn the numerical factors into other data types, for example into character strings (i.e. class labels). If the numerical vector of weightings does not sum up to 1, the sum of the weightings is divided by the number of elements in the weightings vector and this is then used for the classificat#' .

Mode 3: For a given 'real' landscape the number of classes and the weightings are extracted and used to classify the given landscape (any given weighting parameter is overwritten in this case!). If an optional mask value is given the corresponding class from the 'real' landscape is cut from the landscape beforehand.

Value

RasterLayer

Examples

```
# Mode 1
util_classify(fractal_landscape,
              n = 3,
              level_names = c("Land Use 1", "Land Use 2", "Land Use 3"))

# Mode 2
util_classify(fractal_landscape,
              weighting = c(0.5, 0.25, 0.25),
              level_names = c("Land Use 1", "Land Use 2", "Land Use 3"))

# Mode 3
real_land <- util_classify(gradient_landscape,
              n = 3,
              level_names = c("Land Use 1", "Land Use 2", "Land Use 3"))

fractal_landscape_real <- util_classify(fractal_landscape, real_land = real_land)
fractal_landscape_mask <- util_classify(fractal_landscape, real_land = real_land, mask_val = 1)

## Not run:
landscapes <- list(
'1 nlm' = fractal_landscape,
'2 real' = real_land,
'3 result' = fractal_landscape_real,
'4 result with mask' = fractal_landscape_mask
)

show_landscape(landscapes, unique_scales = TRUE, nrow = 1)

## End(Not run)
```

util_merge *util_merge*

Description

Merge a primary raster with other rasters weighted by scaling factors.

Usage

```
util_merge(primary_nlm, secondary_nlm, scalingfactor = 1, rescale)

## S3 method for class 'RasterLayer'
util_merge(primary_nlm, secondary_nlm,
  scalingfactor = 1, rescale = TRUE)
```

Arguments

primary_nlm	Primary Raster* object
secondary_nlm	A list or stack of Raster* objects that are merged with the primary Raster* object
scalingfactor	Weight for the secondary Raster* objects
rescale	If TRUE (default), the values are rescaled between 0-1.

Value

Rectangular matrix with values ranging from 0-1

Examples

```
x <- util_merge(gradient_landscape, random_landscape)
show_landscape(x)
```

util_raster2tibble *Converts raster data into tibble*

Description

Writes spatial raster values into tibble and adds coordinates.

Usage

```
util_raster2tibble(x)

util_raster2tibble(x)
```

Arguments

x	Raster* object

Details

You will loose any resolution, extent or reference system. The output is raw tiles.

Value

a tibble

Examples

```
maptib <- util_raster2tibble(fractal_landscape)
## Not run:
library(ggplot2)
ggplot(maptib, aes(x,y)) +
    coord_fixed() +
    geom_raster(aes(fill = z))

## End(Not run)
```

util_rescale *util_rescale*

Description

Linearly rescale element values in a raster to a range between 0 and 1.

Usage

```
util_rescale(x)

util_rescale(x)
```

Arguments

`x`	Raster* object

Details

Rasters generated by `nlm_` functions are scaled between 0 and 1 as default, this option can be set to `FALSE` if needed.

Value

Raster* object with values ranging from 0-1

Examples

```
unscaled_landscape <- gradient_landscape + fractal_landscape
util_rescale(unscaled_landscape)
```

`util_tibble2raster` *Converts tibble data into a raster*

Description

Writes spatial tibble values into a raster.

Usage

```
util_tibble2raster(x)

util_tibble2raster(x)
```

Arguments

`x`	a tibble

Details

Writes tiles with coordinates from a tibble into a raster. Resolution is set to 1 and the extent will be c(0, max(x), 0, max(y)).

You can directly convert back the result from 'util_raster2tibble()' without problems. If you have altered the coordinates or otherwise played with the data, be careful while using this function.

Value

Raster* object

Examples

```
maptib <- util_raster2tibble(random_landscape)
mapras <- util_tibble2raster(maptib)
all.equal(random_landscape, mapras)
```

`util_writeESRI` *util_writeESRI*

Description

Export raster objects as ESRI ascii files.

Usage

```
util_writeESRI(x, filepath)

## S3 method for class 'RasterLayer'
util_writeESRI(x, filepath)
```

Arguments

`x`	Raster* object
`filepath`	path where to write the raster to file

Details

`raster::writeRaster` or `SDMTools::write.asc` both export files that are recognised by most GIS software, nevertheless they both have UNIX linebreaks. Some proprietary software (like SPIP for example) require an exact 1:1 replica of the output of ESRI's ArcMap, which as a Windows software has no carriage returns at the end of each line. `util_writeESRI` should therefore only be used if you need this, otherwise `raster::writeRaster` is the better fit for exporting raster data in R.

Examples

```
## Not run:
util_writeESRI(gradient_landscape, "gradient_landscape.asc")

## End(Not run)
```

Index