Package ‘raceland’

January 15, 2026

Title Pattern-Based Zoneless Method for Analysis and Visualization of
Racial Topography

Version 1.2.2

Description Implements a computational framework for a pattern-based,
zoneless analysis, and visualization of (ethno)racial topography
(Dmowska, Stepinski, and Nowosad (2020) <doi:10.1016/j.apgeog.2020.102239>).
It is a reimagined
approach for analyzing residential segregation and racial diversity based on
the concept of 'landscape’ used in the domain of landscape ecology.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 3.5)

LinkingTo Rcpp, comat (>= 0.9.0), RcppArmadillo

Imports methods, plotwidgets, terra, sf, Rcpp, comat

Suggests dplyr, pbapply, testthat (>= 2.1.0), knitr, rmarkdown, covr,
raster

VignetteBuilder knitr

URL `https://jakubnowosad.com/raceland/`

BugReports `https://github.com/Nowosad/raceland/issues`

NeedsCompilation yes

Author Jakub Nowosad [aut, cre] (ORCID:
<`https://orcid.org/0000-0002-1057-3721`>),
Anna Dmowska [aut],
Tomasz Stepinski [aut]

Maintainer Jakub Nowosad <`nowosad.jakub@gmail.com`>

Repository CRAN

Date/Publication 2026-01-14 23:50:10 UTC

Contents

`calculate_metrics` *Calculate Metrics*

Description

Calculates exposure matrix and quantifies it by calculating four IT-derived metrics: entropy (ent), joint entropy (joinent), conditional entropy (condent) and mutual information (mutinf). Entropy is associated with measuring racial diversity and mutual information is associated with measuring racial segregation.

Usage

```
calculate_metrics(
  x,
  w,
  neighbourhood = 4,
  fun,
  size = NULL,
  shift = NULL,
  na_action = "replace",
  base = "log2",
  ordered = TRUE,
  threshold = 0.5
)
```

Arguments

`x`	SpatRaster with realizations
`w`	SpatRaster with local densities
`neighbourhood`	The number of directions in which cell adjacencies are considered as neighbours: `4` (rook's case), `8` (queen's case). The default is `4`.

`fun`	Function to calculate values from adjacent cells to contribute to exposure matrix, `"mean"` - calculate average values of local population densities from adjacent cells, `"geometric_mean"` - calculate geometric mean values of local population densities from adjacent cells, or `"focal"` assign value from the focal cell
`size`	Expressed in the numbers of cells, is a length of the side of a square-shaped block of cells. It defines the extent of a local pattern. If `size=NULL` calculations are performed for a whole area
`shift`	Defines the shift between adjacent squares of cells along with the N-S and W-E directions. It describes the density (resolution) of the output grid. The resolution of the output map will be reduced to the original resolution multiplied by the shift. If shift=size the input map will be divided into a grid of non-overlapping square windows. Each square window defines the extent of a local pattern. If shift < size - results in the grid of overlapping square windows.
`na_action`	Decides on how to behave in the presence of missing values in `w`. Possible options are `"replace"`, `"omit"`, `"keep"`. The default, `"replace"`, replaces missing values with 0, `"omit"` does not use cells with missing values, and `"keep"` keeps missing values.
`base`	The unit in which entropy is measured. The default is "log2", which compute entropy in "bits". `"log"` and `"log10"` can be also used
`ordered`	The type of pairs considered. Either ordered (`TRUE`) or unordered (`FALSE`). The default is `TRUE`
`threshold`	The share of NA cells to allow metrics calculation in a square-shaped window

Value

a data.frame

Examples

```
library(terra)
race_raster = rast(system.file("extdata/race_raster.tif", package = "raceland"))
x = create_realizations(race_raster, n = 5)
w = create_densities(x, race_raster, window_size = 10)

#1
df = calculate_metrics(x, w, neighbourhood = 4, fun = "mean")

#2
df2 = calculate_metrics(x, w, neighbourhood = 4, fun = "mean", size = 10, threshold = 0.5)
my_grid = create_grid(x, size = 10)

df3 = dplyr::filter(df2, realization == 2)
result = dplyr::left_join(my_grid, df2, by = c("row", "col"))
plot(result)
```

`create_densities` *Create Densities*

Description

Calculate local densities of subpopulations (race-specific local densities)

Usage

```
create_densities(x, y, window_size)
```

Arguments

`x`	SpatRaster with realizations
`y`	SpatRaster with shares of subpopulations
`window_size`	Size, expressed in the number of cells, of a square-shaped local window for which local densities will be calculated; it is recommended to use the small window_size, i.e., 10

Value

a SpatRaster containing n local densities. Local density layer is calculated for each realization

Examples

```
library(terra)
race_raster = rast(system.file("extdata/race_raster.tif", package = "raceland"))
real_rasters = create_realizations(race_raster, n = 5)
plot(real_rasters)
dens_raster = create_densities(real_rasters, race_raster, window_size = 10)
dens_raster
plot(dens_raster)
```

`create_grid` *Create a grid of square-shaped windows*

Description

Create a grid of square-shaped windows

Usage

```
create_grid(x, size, shift = NULL)
```

Arguments

`x`	A SpatRaster
`size`	Expressed in the numbers of cells, is a length of the side of a square-shaped block of cells. It defines the extent of a local pattern. If `size=NULL` calculations are perfomed for the whole area
`shift`	Defines the shift between adjacent squares of cells along with the N-S and W-E directions. It describes the density (resolution) of the output grid. The resolution of the output map will be reduced to the original resolution multiplied by the shift. If shift=size the input map will be divided into a grid of non-overlapping square windows. Each square window defines the extent of a local pattern. If shift < size - results in the grid of overlapping square windows.

Value

An sf polygon object

Examples

```
library(terra)
race_raster = rast(system.file("extdata/race_raster.tif", package = "raceland"))
x = create_realizations(race_raster, 1)
y = create_grid(x, size = 10)
y
```

`create_realizations` *Create Realizations*

Description

It constructs a high-resolution grid (a racial landscape) in which each cell contains only inhabitants of a single race. Realization is constructed based on race-specific grids. Racial composition at each cell is translated to probabilities of drawing a person of a specific race from a cell. Thus, the race label of a cell is a random variable. To obtain a stochastic realization of racial landscape, we use the cell's race probabilities and a random number generator to randomly assign specific race label to each cell (Monte Carlo procedure).

Usage

```
create_realizations(x, n)
```

Arguments

`x`	SpatRaster with race-specific population densities assign to each cell
`n`	A number of realizations

Value

A SpatRaster object containing n realizations. Single race label in a racial landscape is assigned based on the order of race-specific grids in SpatRaster with input data (For example, the `race_raster` object has five layers named: asian, black, hispanic, other, white. The race labels in racial landscape raster will be 1 - asian, 2- black, 3 - hispanic, 4 - other, 5 - white).

Examples

```
library(terra)
race_raster = rast(system.file("extdata/race_raster.tif", package = "raceland"))
real = create_realizations(race_raster, 10)
plot(real)
```

`plot_realization` *Plot a Realization*

Description

Displays realization taking into account also subpopulation density.

Usage

```
plot_realization(x, y, hex, ...)
```

Arguments

`x`	A SpatRaster with one layer. Each value should correspond to a layer in `y`.
`y`	A SpatRaster with race-specific population densities
`hex`	A character vector with colors specified in hexadecimal format. Each color should correspond to a layer in `y` and value in `x`.
`...`	Additional arguments as for `terra::plotRGB()`

Examples

```
library(terra)
race_raster = rast(system.file("extdata/race_raster.tif", package = "raceland"))
hex_colors = c("#F16667", "#6EBE44", "#7E69AF", "#C77213","#F8DF1D")
realization = create_realizations(race_raster, 1)
plot(race_raster)
plot(realization)

plot_realization(realization, race_raster, hex = hex_colors)
```

`pop_vector` *An sf object*

Description

It is an sf POLYGON object with census block-level data. It consists of 7 variables: GISJOIN - block ID, ASIAN, BLACK, HISPANIC, OTHER, WHITE - number of people of given race/ethnicity in each block

Usage

```
pop_vector
```

Format

An sf object

`quantify_raceland` *Quantify a racial landscape*

Description

This function is a wrapper of several steps (functions) implemented in the raceland package: `create_realizations()`, `create_densities()`, `calculate_metrics()`, and `create_grid()`.

Usage

```
quantify_raceland(
  x,
  n,
  window_size,
  neighbourhood = 4,
  fun,
  size = NULL,
  na_action = "replace",
  base = "log2",
  ordered = TRUE,
  threshold = 0.5
)
```

Arguments

`x`	SpatRaster with race-specific population densities assign to each cell
`n`	A number of realizations
`window_size`	Size, expressed in the number of cells, of a square-shaped local window for which local densities will be calculated; it is recommended to use the small window_size, i.e., 10
`neighbourhood`	The number of directions in which cell adjacencies are considered as neighbours: `4` (rook's case), `8` (queen's case). The default is `4`.
`fun`	Function to calculate values from adjacent cells to contribute to exposure matrix, `"mean"` - calculate average values of local population densities from adjacent cells, `"geometric_mean"` - calculate geometric mean values of local population densities from adjacent cells, or `"focal"` assign value from the focal cell
`size`	Expressed in the numbers of cells, is a length of the side of a square-shaped block of cells. It defines the extent of a local pattern. If `size=NULL` calculations are performed for a whole area
`na_action`	Decides on how to behave in the presence of missing values in `w`. Possible options are `"replace"`, `"omit"`, `"keep"`. The default, `"replace"`, replaces missing values with 0, `"omit"` does not use cells with missing values, and `"keep"` keeps missing values.
`base`	The unit in which entropy is measured. The default is "log2", which compute entropy in "bits". `"log"` and `"log10"` can be also used
`ordered`	The type of pairs considered. Either ordered (`TRUE`) or unordered (`FALSE`). The default is `TRUE`
`threshold`	The share of NA cells to allow metrics calculation in a square-shaped window

Value

An sf polygon object with five columns - `row` and `col` allowing for identification of each square polygon, `ent` - entropy measuring racial diversity, `mutinf` - mutual information, which is associated with measuring racial segregation, and `geometry` containing spatial geometries.

Examples

```
library(terra)
race_raster = rast(system.file("extdata/race_raster.tif", package = "raceland"))
rl = quantify_raceland(race_raster, n = 10, window_size = 10,
 neighbourhood = 4, fun = "mean", size = 20)
```

`race_raster.tif` *A raster file*

Description

A raster file covering an area of 60x60 cells. The raster file contains 5 layers - a high resolution (30m) race-specific grids with values of subpopulation densities for Asian, Black, Hispanic, other and Whites. `system.file("extdata/race_raster.tif", package = "raceland")`

Format

A raster file

`zones_to_raster` *Convert zones to rasters*

Description

Convert zones to rasters

Usage

```
zones_to_raster(v, resolution, variables, ...)
```

Arguments

`v`	An sf object (POLYGON or MULTIPOLYGON)
`resolution`	A numeric vector of length 1 or 2 to set the resolution
`variables`	A character vector with columns names from `v`. The values from these columns will be (1) rasterized and (2) recalculated to densities. Each column will be represented as an layer in the output RasterStack
`...`	Additional arguments as for `terra::rasterize()`

Value

a SpatRaster

Examples

```
library(sf)
library(terra)
plot(pop_vector)
popdens_raster = zones_to_raster(pop_vector, resolution = 30,
                           variables = c("ASIAN", "BLACK", "HISPANIC", "OTHER", "WHITE"))
plot(popdens_raster)
```

Index