Package 'tilemaps'

October 14, 2022

Title Generate Tile Maps

Version 0.2.0

Description Implements an algorithm for generating maps, known as tile maps, in which each region is represented by a single tile of the same shape and size. The algorithm was first proposed in ``Generating Tile Maps'' by Graham McNeill and Scott Hale (2017) <doi:10.1111/cgf.13200>. Functions allow users to generate, plot, and compare square or hexagon tile maps.

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/kaerosen/tilemaps

BugReports https://github.com/kaerosen/tilemaps/issues

Imports clue, ggplot2, igraph, lwgeom, sf, smoothr

Suggests dplyr, knitr, rmarkdown, spData

VignetteBuilder knitr

RoxygenNote 7.1.0

Depends R (>= 2.10)

NeedsCompilation no

Author Kaelyn Rosenberg [aut, cre]

Maintainer Kaelyn Rosenberg <kaerosenberg@gmail.com>

Repository CRAN

Date/Publication 2020-07-10 04:20:02 UTC

R topics documented:

`create_island` *Create a Tile for an Island*

Description

Create a tile for an island that can be added to an existing tile map layout.

Usage

```
create_island(tile_map, position)
```

Arguments

`tile_map`	An `sfc_POLYGON` object representing the layout of a tile map.
`position`	Either a numeric vector of length 2 giving the coordinates for the centroid of the new tile, or a string equal to "upper left", "lower left", "upper right", or "lower right" indicating in which corner of the original tile map the new tile should be located.

Details

Creates a single tile of the same shape and size as the tiles in the given tile map. This tile can be added to the layout of the given tile map to represent an island or region that is not connected to the region represented by the given tile map. The location of the new tile is determined by the `position` argument. Setting the `position` argument equal to "upper left", "lower left", "upper right", or "lower right" will generate a tile which is located in the specified corner of the given tile map. Setting the `position` argument to a numeric vector of length 2 will generate a tile whose centroid is located at the coordinates given in the vector.

Value

Returns an object of class `sfc_POLYGON` representing a single tile of the same shape and size as the tiles in the original tile map.

Examples

```
library(sf)
northeast <- governors[c(6,7,17,18,19,27,28,30,36,37,43),]
tile_map <- generate_map(northeast$geometry, square = FALSE)
tile_map <- append(tile_map, create_island(tile_map, "lower right"))
```

generate_map *Generate a Single Tile Map*

Description

Generate a single square or hexagon tile map.

Usage

```
generate_map(
  data,
  square = TRUE,
  flat_topped = FALSE,
  prop = 0,
  interpolate = 1,
  smoothness = 0,
  shift = c(0, 0)
)
```

Arguments

data	An object of class sfc_MULTIPOLYGON or sfc_POLYGON, which contains the regions that make up the original map.
square	logical. If TRUE, generates a square tile map. If FALSE, generates a hexagon tile map.
flat_topped	logical. If TRUE, hexagons are flat-topped. If FALSE, hexagons are pointy-topped.
prop	A proportion used in specifying the standard deviation of the Gaussian noise added to original region centroids. The standard deviation of the Gaussian noise is calculated as the mean distance between a region centroid and its neighboring regions' centroids multiplied by the value provided for the prop argument.
interpolate	A number between 0 and 1 controlling the linear interpolation between the noisy region centroids and fully-transformed region centroids. If 0, noisy region centroids are used. If 1, fully-transformed centroids are used.
smoothness	numeric. Controls the bandwidth of the Gaussian kernel used for smoothing the transformed boundary polygon. The bandwidth is calculated as the mean distance between adjacent boundary points multiplied by the value provided for the smoothness argument.
shift	A numeric vector of length two specifying the number of grid steps to shift the candidate tile map in the x and y directions before counting the number of tile centroids that lie within the transformed boundary.

Details

Implements an algorithm for generating tile maps proposed in *"Generating Tile Maps"* (McNeill and Hale 2017). The regions of the map must be contiguous. Coordinates cannot be in terms of latitude and longitude. Instead the coordinate reference system must be an appropriate planar projection.

Value

Returns an object of class `sfc_POLYGON`, containing the tiles of the tile map in the same order as the original regions given to the function.

References

McNeill, Graham, and Scott A Hale. 2017. "Generating Tile Maps." In *Computer Graphics Forum*, 36:435–45. 3. Wiley Online Library.

Examples

```
library(sf)
northeast <- governors[c(6,7,17,18,19,27,28,30,36,37,43),]
northeast$tile_map <- generate_map(northeast$geometry, square = FALSE,
                                   flat_topped = TRUE)
```

`governors` *Party Affiliation of US Governors*

Description

A dataset containing the political party affiliation of the governors of the contiguous United States (as of May 2020), as well as an `sfc` object representing the states.

Usage

```
governors
```

Format

`sf` data frame with 48 observations and 3 variables:

geometry `sfc_MULTIPOLYGON` object representing states

abbreviation state abbreviations

party political party affiliation of state governor

Source

`spData::us_states` (https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html)

https://www.nga.org/governors/

many_maps *Generate Many Tile Maps*

Description

Generate, plot, and compare many tile maps.

Usage

```
many_maps(
  data,
  labels,
  square = TRUE,
  flat_topped = FALSE,
  prop = c(0, 0.05),
  interpolate = c(0.5, 1),
  smoothness = c(0, 5),
  shift = list(c(0, 0), c(0.5, 0), c(0, 0.5)),
  weights = c(1, 1, 1, 1),
  plot = TRUE,
  size = 2
)
```

Arguments

data	An object of class sfc_MULTIPOLYGON or sfc_POLYGON, which contains the regions that make up the original map.
labels	A character vector with the labels of the regions. Labels must be in the same order as regions given for data argument.
square	logical. If TRUE, generates a square tile map. If FALSE, generates a hexagon tile map.
flat_topped	logical. If TRUE, hexagons are flat-topped. If FALSE, hexagons are pointy-topped.
prop	A numeric vector of proportions used in specifying the standard deviation of the Gaussian noise added to original region centroids. The standard deviation of the Gaussian noise is calculated as the mean distance between a region centroid and its neighboring regions' centroids multiplied by the value provided for the prop argument. A different set of noisy region centroids is created for each given value.
interpolate	A numeric vector of values between 0 and 1 controlling the linear interpolation between the noisy region centroids and fully-transformed region centroids. If 0, noisy region centroids are used. If 1, fully-transformed centroids are used. A different set of interpolated centroids is created for each given value.

smoothness	numeric vector. Controls the bandwidth of the Gaussian kernel used for smoothing the transformed boundary polygon. The bandwidth is calculated as the mean distance between adjacent boundary points multiplied by the value provided for the smoothness argument. A different transformed boundary is created for each given value.
shift	A list of numeric vectors of length two specifying the number of grid steps to shift the candidate tile map in the x and y directions before counting the number of tile centroids that lie within the transformed boundary. A different final tile map is created for each given value.
weights	A numeric vector of length 4 specifying the weights used for calculating the total cost. The first, second, third, and fourth weights are applied to the location, adjacency, angle, and roughness costs, respectively.
plot	logical. If TRUE, prints plot of generated tile maps.
size	numeric. Controls size of labels in plot.

Details

Generates many candidate tile maps using an algorithm proposed in *"Generating Tile Maps"* (McNeill and Hale 2017). The regions of the map must be contiguous. Coordinates cannot be in terms of latitude and longitude. Instead the coordinate reference system must be an appropriate planar projection. The number of maps generated is equal to the product of the lengths of the prop, interpolate, smoothness, and shift arguments.

Value

Returns a data.frame in which each row corresponds to one map and the columns contain the generated maps, the parameters used for creating the maps, and the costs associated with each map. The data.frame is ordered by the total cost.

References

McNeill, Graham, and Scott A Hale. 2017. "Generating Tile Maps." In *Computer Graphics Forum*, 36:435–45. 3. Wiley Online Library.

Examples

```
library(sf)
northeast <- governors[c(6,7,17,18,19,27,28,30,36,37,43),]
ne_maps <- many_maps(northeast$geometry, northeast$abbreviation,
                     prop = 0, interpolate = 1, smoothness = c(0,20),
                     shift = list(c(0,0), c(0,0.5)))
```

plot_many_maps *Plot Many Maps*

Description

Plot many maps of a single area.

Usage

```
plot_many_maps(map_list, labels, size = 2)
```

Arguments

map_list	A list of sfc_POLYGON objects, each containing regions of a map to be plotted.
labels	A character vector containing the labels for the regions of the sfc_POLYGON objects.
size	numeric. Controls size of labels in plot.

Details

Each element of the map_list argument must have the same number of features, with the first feature of each element corresponding to the same region, the second feature of each element corresponding to the same region, etc. Region labels must be in the same order as the regions of each sfc_POLYGON object.

Value

Prints a plot with labels of the maps in the map_list argument.

Examples

```
library(sf)
northeast <- governors[c(6,7,17,18,19,27,28,30,36,37,43),]
ne_maps <- many_maps(northeast$geometry, northeast$abbreviation,
                     prop = 0, interpolate = 1, smoothness = c(0,20),
                     shift = list(c(0,0), c(0,0.5)), plot = FALSE)
plot_many_maps(ne_maps$map, northeast$abbreviation)
```

Index