Package 'osc'

December 19, 2019

Type Package

Title Orthodromic Spatial Clustering

Version 1.0.5

Date 2019-12-19

Depends methods, raster, R (>= 2.14)

Suggests testthat, maps

Author Steffen Kriewald, Till Fluschnik, Dominik Reusser, Diego Rybski

Maintainer Steffen Kriewald <kriewald@pik-potsdam.de>

Description

Allows distance based spatial clustering of georeferenced data by implementing the City Clustering Algorithm - CCA. Multiple versions allow clustering for a matrix, raster and single coordinates on a plain (Euclidean distance) or on a sphere (great-circle or orthodromic distance).

License GPL

URL https://www.pik-potsdam.de/~kriewald/osc/

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-12-19 11:10:02 UTC

R topics documented:

assign.data *Assign data to clusters*

Description

After clustering assign additional data from a data frame with columns indicated as latitude and longitude.

Usage

```
assign.data(cluster, points, dist=1000)
```

Arguments

cluster	The from cca() generated data frame with cluster-information.
points	Data frame with additional data, containing at least a "lat" and "long" column with point coordinates which will be assigned.
dist	The assignment distance given in meters. Are the given point coordinates within this distance from an identified cluster, this point will be assigned to the cluster.

Details

Multiple points can be assigned to the same cluster. If no cluster is within the given distance, the cluster_id will be 0.

Value

Returns the data frame given as points with an additional column "cluster_id" referring to the identified cluster. A cluster_id 0 indicates that no cluster was within the given distance.

Examples

```
data(landcover)

# clustering urban areas
urban <- cca(landcover, cell.class=1,s=2000, unit="m")
str(urban)

# plot the result
result <- landcover*NA
result[cellFromXY(result,urban$cluster[,c("long","lat")])]<-urban$cluster[,"cluster_id"]*(-1)
plot(result, col=rainbow(9))

# data.frame with additional information (name)
data.points <- data.frame(
      long=c(13.26,13.28),
      lat=c(52.34,52.20),
      name=c("Pappelhausen","New Garden")
```

```
    )

points(data.points$long, data.points$lat, pch="X")

assign.data(cluster=urban$cluster, points=data.points, dist=3000)
```

---

cca *City Clustering Algorithm (CCA)*

Description

CCA is initialized by selecting an arbitrary populated cell which is burnt. Then, the populated neighbors are also burnt. The algorithm keeps growing the cluster by iteratively burning neighbors of the burnt cells until there are no further populated neighboring cells. Next, another unburned populated cell is picked and the procedure is repeated until all populated cells are assigned to a cluster.

The City Clustering Algorithm (CCA) is based on the burning algorithm [1] and was first introduced in the context of cities [2]. Among other things, it was also used for a global urban percolation study [3].

Usage

```
cca(data, s=1, mode=3, count.cells=FALSE,
    count.max=ncol(data)*3,
    res.x=NULL, res.y=NULL, cell.class=1,
    unit="", compare="")
cca.single(data, s, x,y, mode = 3)
```

Arguments

data	data to be clustered. This can be either a raster, a matrix or a data.frame. See details.
s	The radius/shell size of the burning procedure (i.e. how tolerant to small gaps the algorithm is). The unit is 'number of cells' if data is a numeric matrix and 'meters' if data is a raster or data.frame.
x	The starting position in x-direction. Only used if data is a numeric matrix.
y	The starting position in y-direction. Only used if data is a numeric matrix.
mode	The algorithm for a non-georeferenced matrix comes in three versions which affect which close cells are included to the considered cluster: (mode=1) nearest neighbors (mode=2) cells within a shell (i.e. squares of certain size) (mode=3) cells within a radius Whereas (mode=1) is equivalent to (mode=3) with r=1 and (mode=2) with r=1 is equivalent to (mode=3) with r=2. Only used if data is a numeric matrix.
count.cells	Set this option TRUE, if you want to know the number of cells which belongs to each cluster. Only used if data is a numeric matrix.

count.max	This defines the maximum number of clusters, It is set per default to ncol*3. Only used if data is a numeric matrix.
res.y	The resolution of the data-set expressed as the distance between two cell centers in degrees (geographical coordinate system). Only needed if data is a data.frame and unit="m".
res.x	X-resolution. Only needed if data is a data.frame and unit="m".
cell.class	Only required if data is a raster. Specify which cell class (eg. land use type) will be clustered. Can be an integer or a vector and can be combined with the compare option.
unit	If unit = "m" (meter) the cluster algorithm will be done for a cluster distance in 'meter'. Otherwise, the clustering is done in the degrees. If you want to do a pixel-wise clustering, then choose the resolution as cluster distance.
compare	If compare = "g" then cells greater than the given cell.class will be chosen. If compare = "s" cells smaller then the cell.class will be chosen.

Details

cca is implemented in two versions, depending on the format of the data. For numerical matrices, a matrix based version is called. For raster and data.frame based data, a list based version is used, which is faster for sparse matrices and large cluster distances. (See vignette, section: Comparison - matrix vs list)

For matrix:

The matrix is a simple numerical matrix. A value equal 0 or smaller is treated as an unimportant cell, a value above 0 is treated as a cell of interest. Clusters of connected cells are identified.

For raster:

A sub-function will be called to extract the coordinates of a given cell type (cell.class). Also, a threshold determining which cells can be burnt is possible by using compare = "g" (eg. minimum population to consider a cell as populated) Following steps; see data.frame.

For data.frame:

A data frame with two columns specifying the longitude (first column) and latitude (second column) coordinates. The algorithm identifies all points with a distance to each other smaller than the cluster distance s. If unit="m" the orthodromic distance, otherwise the Euclidean distance will be used.

Value

For matrix:

A matrix that defines for each cell to which cluster it belongs.

For raster / data.frame:

List with two entries - 1. data frame with longitude and latitude coordinates of the cells and the cluster_id. and 2. a vector giving the size of the cluster in units of the primitive cell. The first number is the size of the cluster with cluster_id 1, second the size of the cluster with cluster_id 2, and so on.

Author(s)

Steffen Kriewald, Till Fluschnik, Dominik Reusser, Diego Rybski

References

[1] Stauffer, D., & Aharony, A. (2014). Introduction to percolation theory. Taylor & Francis.

[2] Rozenfeld, H. D., Rybski, D., Andrade, J. S., Batty, M., Stanley, H. E., & Makse, H. A. (2008). Laws of population growth. Proceedings of the National Academy of Sciences, 105(48), 18702-18707.

[3] Fluschnik, T., Kriewald, S., Garcia Cantu Ros, A., Zhou, B., Reusser, D., Kropp, J., & Rybski, D. (2016). The size distribution, scaling properties and spatial organization of urban clusters: a global and regional percolation perspective. ISPRS International Journal of Geo-Information, 5(7), 110.

Examples

```
# for a matrix
data(population)
image(population)

clusters <- cca(population, s=5)
cols <- c("white",rep(rainbow(10), length.out=length(table(clusters))) )
image(clusters, col=cols, xlab="", ylab="")

one.cluster <- cca.single(population, s=1, x=125, y=125)
image(one.cluster, col=cols, xlab="", ylab="")

# for a raster-object
data(landcover)

# clustering urban areas
urban <- cca(landcover, cell.class=1,s=2000, unit="m")
str(urban)

# plot the result
result <- landcover*NA
result[cellFromXY(result,urban$cluster[,c("long","lat")])]<-urban$cluster[,"cluster_id"]*(-1)
plot(result, col=rainbow(9))
```

`coordinate.list` *List of coordinates for clustering*

Description

Extracts coordinates of cells with defined cell class from a raster object.

Usage

```
coordinate.list(raster, cell.class, compare = "")
```

Arguments

raster	raster with values
cell.class	number or vector of cell-values for clustering
compare	character of type "", "g" or "s". If "g" or "s" all coordinates of cells with value greater "g" resp. smaller "s" than cell.class will be extracted

Details

Works also for very large raster, but can take some time.

Value

Returns a data frame with lat-, long-coordinates

Examples

```
data("landcover")

coordinate.list(landcover, 1:10)
```

---

exampledata *Example data for the clustering algorithm.*

Description

This is test data for the package.

Usage

```
data(exampledata)
```

Format

A data frame with 235 observations on the following 4 variables.

`x` a numeric vector of x-coordinates

`y` a numeric vector of y-coordinates

`z` a numeric vector of population data

`cluster` a numeric vector of clusters

Examples

```
data(exampledata)
str(exampledata)
```

landcover *Fictional landcover data to demonstrate the cca for Raster-Data*

Description

A fictional landcover dataset with six different landtypes indicated by number from 0 to 5. Cells with value one a considered as urban in the examples.

Usage

```
data(landcover)
```

Format

The format is: Formal class 'RasterLayer' [package "raster"] with 12 slots

Examples

```
data(landcover)
str(landcover)
```

osc.buffer *Simple Buffer algorithm*

Description

Simple buffer based on euclidean distance are created around all cells equal to one.

Usage

```
osc.buffer(input, width=max(dim(input)), return.width=F, complete=F)
```

Arguments

input	Matrix or Raster containing 1 indicating a cluster, no NA values are allowed
width	Width of the buffer in cells
return.width	Logical value, if TRUE then the distance to the nearest cell with value 1 will be returned for all cells within the buffer.
complete	Logical value, if TRUE the buffer will be increased until the whole raster is covered. Can only be used in combination with return.width=TRUE.

Value

Returns matrix or raster, depending on input, with 1 for cluster and -1 for surrounding buffer. If return.width=TRUE negative numbers indicating the distance (width) to the nearest cell with value 1.

Examples

```
data(landcover)
landcover[landcover[]>1] <- 0

plot(osc.buffer(landcover, 6))

plot(osc.buffer(landcover, 6, return.width = TRUE))

plot(osc.buffer(landcover, return.width = TRUE, complete = TRUE))
```

---

population *Example population data for the city clustering algorithm*

Description

Example population data for the city clustering algorithm

Usage

```
data(population)
```

Format

The format is: num [1:1525, 1:1000] 0 0 0 0 0 0 0 0 0 0 ...

Examples

```
data(population)
str(population)
```

Index