Package ‘spatgraphs’

December 14, 2017

Type Package
Title Graph Edge Computations for Spatial Point Patterns
Version 3.2-1
Date 2017-12-14
Author Tuomas Rajala
Maintainer Tuomas Rajala <tuomas.rajala@iki.fi>
Description Graphs (or networks) and graph component calculations for spatial locations in 1D, 2D, 3D etc.
License GPL (>= 2)
LazyData TRUE
Imports Rcpp (>= 0.11.6), Matrix, methods
Suggests rgl
LinkingTo Rcpp
RoxygenNote 6.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2017-12-14 11:02:33 UTC

R topics documented:

adj2sg ......................................................... 2
as.sg ......................................................... 3
as.sgadj ...................................................... 3
as.sgc ......................................................... 4
cut.sg ......................................................... 4
edgeLengths .................................................. 5
is_sg ......................................................... 5
plot.sg ....................................................... 6
plot.sgadj ................................................... 6
plot.sgc ....................................................... 7
plot.sgspectral ............................................. 7
Description

sgadj to sg

Usage

adj2sg(x)

Arguments

x  sgadj object
as.sg  

Class creator

Description

Class creator

Usage

as.sg(edges = list(), type = "?", pars = NULL, note = NULL)

Arguments

edges  list of neighbourhoods
type   type
pars   parameters
note   notes

as.sgadj  

Creator for sgadj-class

Description

Creator for sgadj-class

Usage

as.sgadj(edges = NULL, type = "?", pars = NULL, other = "")

Arguments

edges  edge list-of-lists
type   of the graph
pars   parameters for the graph
other  other comments
as.sgc       Creator for sgc

Description
Creator for sgc

Usage
as.sgc(clusters, type = "?", pars = NULL, note = NULL)

Arguments
clusters list of clusters as point indices
type type
pars parameters
note notes

cut.sg       cut edges

Description
cut edges

Usage
## S3 method for class 'sg'
cut(x, data, R, ...)

Arguments
x sg graph object
data point pattern used for computing g
R cutting length
... ignored
Removes edges with length > R.
edgeLengths

Description
Edge lengths

Usage
edgeLengths(g, x, ...)

Arguments
\( g \) \hspace{1cm} \text{sg-object}
\( x \) \hspace{1cm} \text{point pattern}
\( ... \) \hspace{1cm} \text{ignored}

is_sg

Description
verify class sg

Usage
is_sg(x)

Arguments
\( x \) \hspace{1cm} \text{object to check}
plot.sg

Plot a spatial graph

Description

Rudimentary plotting.

Usage

## S3 method for class 'sg'
plot(x, data, which = NULL, add = FALSE, addPoints = FALSE,
     points.pch = 1, points.col = 1, points.cex = 1, max.edges = 10000,
     ...)  

Arguments

x an 'sg' graph object
data The point pattern object, same as for computing the 'g'
which Indices of which out-edges to plot. Default: all
add Add to existing plot? (default: FALSE)
addPoints Add points? Will be added if add=FALSE
points.pch point styling
points.col point styling
points.cex point styling
max.edges limit of edges to try to plot, gets very slow at high count. default 1e4
... passed to 'lines' function

plot.sgadj

plot sgadj

Description

plot sgadj

Usage

## S3 method for class 'sgadj'
plot(x, ...)  

Arguments

x sgadj object
... passed to plot.sg
converts to sg and plots that.
plot.sgc

Description

plot clusters

Usage

## S3 method for class 'sgc'
plot(x, data, atleast = 2, add = FALSE, col, ...)

Arguments

x spatcluster-cluster object
data point pattern object used for computing the graph
atleast plot only cluster with 'atleast' points in them
add add or plot new
col colors for clusters, chosen randomly if missing.
... passed to points

plot.sgspectral

Description

plot spectral clustering results

Usage

## S3 method for class 'sgspectral'
plot(x, data, ...)

Arguments

x spectral_sg result
data point pattern
... ignored
**plot3_sg**

*Plot 3d graph*

**Description**

Plot 3d graph

**Usage**

```r
plot3_sg(x, data, which, ...)
```

**Arguments**

- **x**: `sg` object
- **data**: coordinates
- **which**: points of which out-edges will be plotted
- **...**: passed to `rgl.lines`

**print.sg**

*print method for sg*

**Description**

print method for `sg`

**Usage**

```r
## S3 method for class 'sg'
print(x, ...)
```

**Arguments**

- **x**: `sg` object
- **...**: ignored
print.sgadj  

Description

print method for sgadj

Usage

## S3 method for class 'sgadj'
print(x, ...)

Arguments

x  sgadj object
...

print.sgc  

Description

sgc print method

Usage

## S3 method for class 'sgc'
print(x, ...)

Arguments

x  sgc object
...

ignored
prune_sg  

**Description**
Prune a graph

**Usage**
```r
prune_sg(g, level = 1, verbose = FALSE)
```

**Arguments**
- `g` : sg object
- `level` : pruning level
- `verbose` : verbosity

**Details**
Remove edges from a graph by their path connectivity.

**Examples**
```r
x <- matrix(runif(50*2), ncol=2)
g <- spatgraph(x, "MST")
gp <- prune_sg(g, level = 2)
plot(g, x, lty=2)
plot(gp, x, add=TRUE, col=2)
```

---

remove_nodes  

**Description**
Remove the existence of particular nodes from the graph.

**Usage**
```r
remove_nodes(g, i, fuse = FALSE, verb = FALSE)
```

**Arguments**
- `g` : sg object
- `i` : indices of nodes for which to remove the edges
- `fuse` : Should the neighbours of removed nodes be connected?
- `verb` : verbose?
sg2adj

Details

Basically, just clear the neighbourhood of selected indices. If fuse=TRUE, connect neighbours together (excluding i’s). Should work over several remove nodes along a path.

Note: g should be symmetric. use sg2sym to force symmetry, it is not checked.

Warning: In development.

Examples

```r
x <- matrix(runif(200), ncol=2)
g <- spatgraph(x, "RST", c(1,0))
g <- sg2sym(g)
i <- sample(100, 50)
k <- setdiff(1:100, i)
gs <- remove_nodes(g, i, fuse=TRUE)
plot(g, x, add=FALSE)
points(x[k,], pch=19, col=4)
plot(gs, x, add=TRUE, lty=2, col=3)
```

sg2adj

sg to sgadj

Description

sg to sgadj

Usage

```r
sg2adj(x)
```

Arguments

- `x`: sg object

sg2dxf

sg to dxf format

Description

sg to dxf format

Usage

```r
sg2dxf(g, x, file)
```
Arguments

\begin{itemize}
\item \texttt{g} \hspace{2em} \textit{sg object}
\item \texttt{x} \hspace{2em} \textit{pattern object used for computing g}
\item \texttt{file} \hspace{2em} \textit{filename for output}
\end{itemize}

\textbf{Description}

\textit{sg to igraph}

\textbf{Usage}

\texttt{sg2igraph(x)}

\textbf{Arguments}

\begin{itemize}
\item \texttt{x} \hspace{2em} \textit{sg object}
\end{itemize}

-----

\textbf{sg2sparse} \hspace{2em} \textit{Make a sparse adjacency matrix from sg-object}

\textbf{Description}

Make a sparse adjacency matrix from sg-object

\textbf{Usage}

\texttt{sg2sparse(x)}

\textbf{Arguments}

\begin{itemize}
\item \texttt{x} \hspace{2em} \textit{sg-object}
\end{itemize}
sg2sym

**Symmetrisation of sg adjacency matrix wrapper for 1way and 2way symmetrisation**

**Description**
Symmetrisation of sg adjacency matrix wrapper for 1way and 2way symmetrisation

**Usage**
sg2sym(x, way = 1)

**Arguments**
- **x**: sg object
- **way**: 1: OR rule, 2: AND rule for keeping edges.

sg2wadj

**weighted sg to weighted adjacency matrix**

**Description**
weighted sg to weighted adjacency matrix

**Usage**
sg2wadj(x)

**Arguments**
- **x**: weighted sg object

sg_parse_coordinates

**Parse input for coordinates**

**Description**
Extract the coordinate locations from the input object.

**Usage**
sg_parse_coordinates(x, verbose = FALSE)

**Arguments**
- **x**: Input object containing the coordinates in some format.
- **verbose**: Print out info of the coordinates.
**sg_verify_parameters**  
*Verify input parameters for the graph*

**Description**
Mainly for internal use.

**Usage**
```
sg_verify_parameters(coord, type, par, maxR, doDists, preGraph)
```

**Arguments**
- `coord`: Coordinates of the locations
- `type`: Type of graph
- `par`: Parameter(s) for the graph
- `maxR`: Maximum range for edges, helps in large patterns.
- `doDists`: Precompute distances? Speeds up some graphs, takes up memory.
- `preGraph`: Precomputed graph, taken as a super-graph

---

**shortestPath**  
*shortest path on the graph*

**Description**
Dijkstra’s algorithm

**Usage**
```
shortestPath(i, j, g, x = NULL, dbg = FALSE)
```

**Arguments**
- `i`: index from
- `j`: index to
- `g`: sg object
- `x`: optional point pattern from which g was computed
- `dbg`: verbose
spare2sg

**Description**

Make an sg-object from adjacency matrix

**Usage**

```r
spare2sg(x)
```

**Arguments**

- `x` square matrix. non-0 elements are taken as edge presence.

---

spatcluster

**Description**

Compute the connected components of a graph

**Usage**

```r
spatcluster(x, verbose = TRUE, sym = FALSE)
```

**Arguments**

- `x` sg-object
- `verbose` print info
- `sym` force symmetry of edges
spatgraph

Compute the edges of a spatial graph

**Description**

Given a spatial point pattern, we compute the edges of a graph (network) for a specified type of edge relationship.

**Usage**

```r
spatgraph(x, type = "geometric", par = NULL, verbose = FALSE, maxR = 0,
            doDists = FALSE, preGraph = NULL)
```

**Arguments**

- **x**: Input point pattern object
- **type**: Type of the graph
- **par**: Parameter(s) for the graph
- **verbose**: Print details
- **maxR**: Maximum range for edges, helps in large patterns.
- **doDists**: Precompute distances? Speeds up some graphs, takes up memory.
- **preGraph**: Precomputed graph, taken as a super-graph

**Details**

Several edge definitions are supported:

- **geometric**: `par=numeric>0`. Geometric graph, `par = connection radius`.
- **knn**: `par=integer>0`. k-nearest neighbours graph, `par = k`.
- **mass_geometric**: Connect two points if `||x-y||<m(x)`. par=vector giving the `m(x_i)`'s
- **markcross**: Connect two points if `||x-y||<m(x)+m(y)`. par = vector giving the `m(x_i)`'s
- **gabriel**: Gabriel graph. Additional parameter for allowing `par=k` instead of 0 points in the circle.
- **MST**: Minimal spanning tree.
- **SIG**: Spheres of Influence.
- **RST**: Radial spanning tree, `par=origin of radiation, coordinate vector`
- **RNG**: Relative neighbourhood graph
- **CCC**: Class-Cover-Catch, `par=factor vector of point types`. The factor vector is converted to integers according to R’s internal representation of factors, and the points with type 1 will be the target. Use `relevel` to change the target.

The parameter 'maxR' can be given to bring $n^3$ graphs closer to $n^2$. k-nearest neighbours will warn if maxR is too small (<k neighbours for some points), others, like RNG, don’t so be careful. Voronoi diagram aka Delaunay triangulation is not supported as other R-packages can do it, see e.g. package ‘deldir’.
spectral_sg

Examples

# basic example
x <- matrix(runif(50*2), ncol=2)
g <- spatgraph(x, "knn", par=2)
plot(g, x)

# bigger example
xb <- matrix(runif(5000*2), ncol=2)
rb <- spatgraph(xb, "RNG", maxR=0.1)

spectral_sg(g, m = 2, K = 3)

Arguments

g sg object. Should be weighted (with weight_sg-function)
m levels to consider
K number of assumed clusters

summary.sg

Description

sg summary

Usage

## S3 method for class 'sg'
summary(object, ...)

Arguments

object sg object
...
 ignored
### summary.sgc

**sgc summary**

**Description**

sgc summary

**Usage**

```r
## S3 method for class 'sgc'
summary(object, ...)
```

**Arguments**

- `object`: sgc object
- `...`: ignored

### t.sg

**Transpose sg object**

**Description**

This will transpose the adjacency matrix underlying the graph. Will transform to and from sgadj-object (see `sg2adj`)

**Usage**

```r
## S3 method for class 'sg'
t(x)
```

**Arguments**

- `x`: sg-object.
t.sgadj

Transpose sgadj object

Description
This will transpose the adjacency matrix underlying the graph.

Usage

### S3 method for class 'sgadj'

t(x)

Arguments

- x: sgadj object

weight_sg

Set weights to edges of sg

Description

For each edge e(i,j) between points i,j, set the weight f(||x_i-x_j||)

Usage

weight_sg(g, x, f = function(x) exp(-x^2/scale), scale = 1, ...)

Arguments

- g: sg object
- x: point pattern used in g
- f: function for the weight
- scale: additional scale parameter for the default f
- ...: ignored

Details

Default f(x) = exp(-x^2/scale)
Index

adj2sg, 2  t.sgadj, 19
as.sg, 3     weight_sg, 19
as.sgadj, 3
as.sgc, 4

cut.sg, 4

dgeLengths, 5

is_sg, 5

plot.sg, 6
plot.sgadj, 6
plot.sgc, 7
plot.sgspectral, 7
plot3_sg, 8
print.sg, 8
print.sgadj, 9
print.sgc, 9
prune_sg, 10

relevel, 16
remove_nodes, 10

sg2adj, 11
sg2dxf, 11
sg2igraph, 12
sg2sparse, 12
sg2sym, 13
sg2wadj, 13

sg_parse_coordinates, 13
sg_verify_parameters, 14
shortestPath, 14
sparse2sg, 15
spatcluster, 15
spatgraph, 16
spectral_sg, 17
summary.sg, 17
summary.sgc, 18

t.sg, 18