Package ‘spatgraphs’

October 27, 2015

Type Package
Title Graph Edge Computations for Spatial Point Patterns
Version 3.0
Date 2015-10-26
Author Tuomas Rajala
Maintainer Tuomas Rajala <tuomas.rajala@iki.fi>
Description Graphs (or networks) and graph component calculations for spatial locations in *D.
License GPL (>= 2)
LazyData TRUE
Imports Rcpp (>= 0.11.6), Matrix, rgl, methods
LinkingTo Rcpp
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-10-27 12:03:40

R topics documented:

adj2sg ........................................... 2
as.sg ........................................... 3
as.sgadj ........................................ 3
as.sgc ........................................... 4
cut.sg ........................................... 4
distanceLengths ................................ 5
is_sg ............................................. 5
plot.sg .......................................... 6
plot.sgadj ....................................... 6
plot.sgc ......................................... 7
plot.sgspectral .................................. 7
plot3.sg ........................................ 8
print.sg ........................................ 8
adj2sg

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

**Usage**

```r
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

**Usage**

```r
## 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**  
  sg-object
- **x**  
  point pattern
- **...**  
  ignored

---

**is_sg**  

### Description

verify class sg

### Usage

`is_sg(x)`

### Arguments

- **x**  
  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 = 1e+04,
     ...)

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**

*plot clusters*

### Description

plot clusters

### Usage

```r
## 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**

*plot spectral clustering results*

### Description

plot spectral clustering results

### Usage

```r
## 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

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

## 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
- `...` ignored

---

**print.sgc**

**Description**

sgc print method

**Usage**

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

**Arguments**

- `x` sgc object
- `...` ignored
**prune.sg**  
*Prune a graph*

**Description**

Prune a graph

**Usage**

```
prune.sg(g, level = 1, verbose = FALSE)
```

**Arguments**

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

---

**sg2adj**  
*sg to sgadj*

**Description**

sg to sgadj

**Usage**

```
sg2adj(x)
```

**Arguments**

- `x`: sg object
sg2dxf  
sg to dxf format

Description
sg to dxf format

Usage
sg2dxf(g, x, file)

Arguments
  g  sg object
  x  pattern object used for computing g
  file  filename for output

sg2igraph  
sg to igraph

Description
sg to igraph

Usage
sg2igraph(x)

Arguments
  x  sg object

sg2sparse  
Make a sparse adjacency matrix from sg-object

Description
Make a sparse adjacency matrix from sg-object

Usage
sg2sparse(x)

Arguments
  x  sg-object
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_GRAPH_PARAMETERS  Supported graphs constants

Description
Supported graphs constants

Usage
SG_GRAPH_PARAMETERS

Format
List of 10
$ geometric :List of 1
  ..$ R: chr "numeric>0"  
$ knn :List of 1
  ..$ k: chr "integer>0"
$ mass_geometric:List of 1
  ..$ mass: chr "numeric vector of sizes"
$ markcross :List of 1
  ..$ mass: chr "numeric vector of sizes"
$ gabriel : list()
$ MST : list()
$ SIG : list()
$ RST :List of 1
  ..$ center: chr "coordinates of the center"
$ RNG : list()
$ CCC :List of 1
  ..$ types: chr "factor vector of types"

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
sparse2sg

Make an sg-object from adjacency matrix

Description
Make an sg-object from adjacency matrix

Usage
sparse2sg(x)

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

spatcluster

Compute the connected components of a graph

Description
Compute the connected components of a graph

Usage
spatcluster(x, verbose = TRUE, sym = FALSE)

Arguments
x sg-object
verbose print info
sym force symmetry of edges
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’.
Examples

# basic example
x <- matrix(runif(50*2), ncol=2)
g <- spatgraph(x, "knn", par=3)
plot(g, x)

# big example
xb <- matrix(runif(10000*2), ncol=2)
gb <- spatgraph(xb, "RNG", maxR=0.1)

spectral.sg

spectral clustering

Description

spectral clustering

Usage

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

sg summary

Description

sg summary

Usage

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

Arguments

object          sg object
...            ignored
**Summary**

For class `sgc`.

**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 ‘sgadj’).

**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**

```r
## 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**

```r
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

*Topic *datasets *

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG_GRAPH_PARAMETERS</td>
<td>13</td>
</tr>
<tr>
<td>adj2sg</td>
<td>2</td>
</tr>
<tr>
<td>as.sg</td>
<td>3</td>
</tr>
<tr>
<td>as.sgadj</td>
<td>3</td>
</tr>
<tr>
<td>as.sgc</td>
<td>4</td>
</tr>
<tr>
<td>cut.sg</td>
<td>4</td>
</tr>
<tr>
<td>edgeLengths</td>
<td>5</td>
</tr>
<tr>
<td>is_sg</td>
<td>5</td>
</tr>
<tr>
<td>plot.sg</td>
<td>6</td>
</tr>
<tr>
<td>plot.sgadj</td>
<td>6</td>
</tr>
<tr>
<td>plot.sgc</td>
<td>7</td>
</tr>
<tr>
<td>plot.sgspectral</td>
<td>7</td>
</tr>
<tr>
<td>plot3.sg</td>
<td>8</td>
</tr>
<tr>
<td>print.sg</td>
<td>8</td>
</tr>
<tr>
<td>print.sgadj</td>
<td>9</td>
</tr>
<tr>
<td>print.sgc</td>
<td>9</td>
</tr>
<tr>
<td>prune.sg</td>
<td>10</td>
</tr>
<tr>
<td>relevel</td>
<td>16</td>
</tr>
<tr>
<td>sg2adj</td>
<td>10</td>
</tr>
<tr>
<td>sg2dxf</td>
<td>11</td>
</tr>
<tr>
<td>sg2igraph</td>
<td>11</td>
</tr>
<tr>
<td>sg2sparse</td>
<td>11</td>
</tr>
<tr>
<td>sg2sym</td>
<td>12</td>
</tr>
<tr>
<td>sg2wadj</td>
<td>12</td>
</tr>
<tr>
<td>SG_GRAPH_PARAMETERS</td>
<td>13</td>
</tr>
<tr>
<td>sg_parse_coordinates</td>
<td>13</td>
</tr>
<tr>
<td>sg_verify_parameters</td>
<td>13</td>
</tr>
<tr>
<td>shortestPath</td>
<td>14</td>
</tr>
<tr>
<td>sparse2sg</td>
<td>15</td>
</tr>
<tr>
<td>spatcluster</td>
<td>15</td>
</tr>
<tr>
<td>spatgraph</td>
<td>16</td>
</tr>
<tr>
<td>spectral.sg</td>
<td>17</td>
</tr>
<tr>
<td>summary.sg</td>
<td>17</td>
</tr>
<tr>
<td>summary.sgc</td>
<td>18</td>
</tr>
<tr>
<td>t.sg</td>
<td>18</td>
</tr>
<tr>
<td>t.sgadj</td>
<td>19</td>
</tr>
<tr>
<td>weight.sg</td>
<td>19</td>
</tr>
</tbody>
</table>