

# Package ‘spatgraphs’

February 10, 2023

**Type** Package

**Title** Graph Edge Computations for Spatial Point Patterns

**Version** 3.4

**Date** 2023-02-10

**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)

**Imports** Rcpp (>= 0.11.6), Matrix, methods

**Suggests** rgl

**LinkingTo** Rcpp

**RoxygenNote** 7.2.3

**Encoding** UTF-8

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2023-02-10 12:00:05 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 . . . . .	7
plot.sgc . . . . .	7
plot.sgspectral . . . . .	8

plot3_sg . . . . .	8
print.sg . . . . .	9
print.sgadj . . . . .	9
print.sgc . . . . .	10
prune_sg . . . . .	10
remove_nodes . . . . .	11
sg2adj . . . . .	12
sg2dxf . . . . .	12
sg2igraph . . . . .	13
sg2sparse . . . . .	13
sg2sym . . . . .	14
sg2wadj . . . . .	14
sg_parse_coordinates . . . . .	14
sg_verify_parameters . . . . .	15
shortestPath . . . . .	15
sparse2sg . . . . .	16
spatcluster . . . . .	16
spatgraph . . . . .	17
spectral_sg . . . . .	18
summary.sg . . . . .	19
summary.sgc . . . . .	19
t.sg . . . . .	20
t.sgadj . . . . .	20
weight_sg . . . . .	21

<b>Index</b>	<b>22</b>
--------------	-----------

---

adj2sg

*sgadj to sg*

---

### 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	<i>Edge lengths</i>
-------------	---------------------

---

**Description**

Edge lengths

**Usage**

```
edgeLengths(g, x, ...)
```

**Arguments**

g	sg-object
x	point pattern
...	ignored

---

is_sg	<i>verify class sg</i>
-------	------------------------

---

**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 = 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	<i>plot sgadj</i>
------------	-------------------

---

**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	<i>plot clusters</i>
----------	----------------------

---

**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	<i>plot spectral clustering results</i>
-----------------	---

---

**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	<i>Plot 3d graph</i>
----------	----------------------

---

**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 segments3d

---

print.sg	<i>Print method for sg</i>
----------	----------------------------

---

**Description**

Print sg class.

**Usage**

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

**Arguments**

x	sg object
...	ignored

**Details**

Print basic info.

---

print.sgadj	<i>print method for sgadj</i>
-------------	-------------------------------

---

**Description**

print method for sgadj

**Usage**

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

**Arguments**

x	sgadj object
...	ignored

---

print.sgc	<i>sgc print method</i>
-----------	-------------------------

---

**Description**

sgc print method

**Usage**

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

**Arguments**

x	sgc object
...	ignored

---

prune_sg	<i>Prune a graph</i>
----------	----------------------

---

**Description**

Prune a graph

**Usage**

```
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**

```
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	<i>Remove edges connected to certain nodes</i>
--------------	--

---

### Description

Remove the existence of particular nodes from the graph.

### Usage

```
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?

### 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

```
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	<i>sg to sgadj</i>
--------	--------------------

---

**Description**

sg to sgadj

**Usage**

sg2adj(x)

**Arguments**

x	sg object
---	-----------

---

sg2dxf	<i>sg to dxf format</i>
--------	-------------------------

---

**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**

Obsolete. Use `igraph::graph_from_adj_list` on the graph edges element.

**Usage**

```
sg2igraph(g, x, ...)
```

**Arguments**

<code>g</code>	sg object
<code>x</code>	possibly the location pattern
<code>...</code>	not used

Not implemented. You can use the `'graph_from_adj_list'`-function in `'igraph'`-package on the edges-element of the graph.

**Examples**

```
## Not run:  
ix <- igraph::graph_from_adj_list(x$edges)  
  
## End(Not run)
```

---

`sg2sparse`*Make a sparse adjacency matrix from sg-object*

---

**Description**

Make a sparse adjacency matrix from sg-object

**Usage**

```
sg2sparse(x)
```

**Arguments**

<code>x</code>	sg-object
----------------	-----------

---

sg2sym	<i>Symmetrisation of sg adjacency matrix wrapper for 1way and 2way symmetrisation</i>
--------	---

---

**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	<i>weighted sg to weighted adjacency matrix</i>
---------	---

---

**Description**

weighted sg to weighted adjacency matrix

**Usage**

```
sg2wadj(x)
```

**Arguments**

x	weighted sg object
---	--------------------

---

sg_parse_coordinates	<i>Parse input for coordinates</i>
----------------------	------------------------------------

---

**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

---

<code>sparse2sg</code>	<i>Make an sg-object from adjacency matrix</i>
------------------------	--

---

**Description**

Make an sg-object from adjacency matrix

**Usage**

```
sparse2sg(x)
```

**Arguments**

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

---

<code>spatcluster</code>	<i>Compute the connected components of a graph</i>
--------------------------	--

---

**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

---

 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

```
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)

# bigger example
xb <- matrix(runif(5000*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

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

---

summary.sg	<i>sg summary</i>
------------	-------------------

---

**Description**

sg summary

**Usage**

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

**Arguments**

object	sg object
...	ignored

---

summary.sgc	<i>sgc summary</i>
-------------	--------------------

---

**Description**

sgc summary

**Usage**

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

**Arguments**

object	sgc object
...	ignored

---

t.sg	<i>Transpose sg object</i>
------	----------------------------

---

**Description**

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

**Usage**

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

**Arguments**

x                   sg-object.

---

t.sgadj	<i>Transpose sgadj object</i>
---------	-------------------------------

---

**Description**

This will transpose the adjacency matrix underlying the graph.

**Usage**

```
## S3 method for class 'sgadj'  
t(x)
```

**Arguments**

x                   sgadj object

---

weight_sg	<i>Set weights to edges of sg</i>
-----------	-----------------------------------

---

**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**

<code>g</code>	sg object
<code>x</code>	point pattern used in <code>g</code>
<code>f</code>	function for the weight
<code>scale</code>	additional scale parameter for the default <code>f</code>
<code>...</code>	ignored

**Details**

Default  $f(x) = \exp(-x^2/\text{scale})$

# Index

adj2sg, [2](#)  
as.sg, [3](#)  
as.sgadj, [3](#)  
as.sgc, [4](#)  
  
cut.sg, [4](#)  
  
edgeLengths, [5](#)  
  
is\_sg, [5](#)  
  
plot.sg, [6](#)  
plot.sgadj, [7](#)  
plot.sgc, [7](#)  
plot.sgspectral, [8](#)  
plot3\_sg, [8](#)  
print.sg, [9](#)  
print.sgadj, [9](#)  
print.sgc, [10](#)  
prune\_sg, [10](#)  
  
relevel, [18](#)  
remove\_nodes, [11](#)  
  
sg2adj, [12](#)  
sg2dxf, [12](#)  
sg2igraph, [13](#)  
sg2sparse, [13](#)  
sg2sym, [14](#)  
sg2wadj, [14](#)  
sg\_parse\_coordinates, [14](#)  
sg\_verify\_parameters, [15](#)  
shortestPath, [15](#)  
sparse2sg, [16](#)  
spatcluster, [16](#)  
spatgraph, [17](#)  
spectral\_sg, [18](#)  
summary.sg, [19](#)  
summary.sgc, [19](#)  
  
t.sg, [20](#)  
  
t.sgadj, [20](#)  
  
weight\_sg, [21](#)