Package 'shp2graph'

July 2, 2014

Version 0-2

Date 2014-05-14

Title Convert a SpatialLinesDataFrame object to a ``igraph-class'' object

Author Binbin Lu

Maintainer Binbin Lu <lubinbin220@gmail.com>

Depends R (>= 3.0.0),maptools,igraph

Description Functions for converting network data from a SpatialLinesDataFrame object to a ``igraph-class'' object.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-05-14 10:52:28

R topics documented:

`Degree.list` *Degree (In-degree and Out-degree)of nodes*

Description

The function returns degrees of nodes from provided nodelist and edgelist; in-degrees and out-degrees are returned if edges are directed).

Usage

```
Degree.list(nodelist, edgelist, Directed=F)
```

Arguments

`nodelist`	A "nodelist" object
`edgelist`	An "edgelist" object
`Directed`	TRUE value if "edgelist" are directed; otherwise, FALSE

Value

Lists of different contents are returned for undirected and directed edges respectively: For undirected type:

`DegreeL`	An integer vector of degrees cooresponding to each node in "nodelist"

For directed type:

`InDegreeL`	An integer vector of In-degrees cooresponding to each node in "nodelist"
`OutDegreeL`	An integer vector of Out-degrees cooresponding to each node in "nodelist"

Note

The outputs of this function are different between undirected and directed networks, actually DegreeL can also be computed by DegreeL=InDegreeL+OutDegreeL.

Author(s)

Binbin Lu <lubinbin220@gmail.com>

Directed *Orientate each edge in a given edgelist*

Description

This function is to orientate each edge according to the given vector.

Usage

```
Directed(edgelist, direction.v=rep(0,length(edgelist[,1])), eadf=NULL)
```

Arguments

edgelist	An edgelist object, see "edgelist"
direction.v	A vector (of length equal to number of edges in edgelist) with values of 1 or 0, see details below;
eadf	Attribute data frame for all the edges;

Details

In a road network data set, some road segments might be one-way while the rest are double-way. This suitation makes it complex for defining edges in the graph. This property could be specified in "direction.v": 1 means one-way and 0 represents double-way. All the double-way edges in edgelist are clarified as two directed edges (i.e. edge(nid1, nid2) and edge(nid1, nid2) and edge(nid2, nid1)), and duplicate the corresponding attributes. The one-way edges could be regarded as directed.

Value

A list consisted of:

newEdgelist	A new "edgelist" with directed edges
newEadf	A new attribute data frame for the new "edgelist"

Author(s)

Binbin Lu <lubinbin220@gmail.com>

edgelist *A structure for edge information of a network*

Description

This is an intergradation of edges from spatial data to graph data, and each edge cooresponds to one row in the object, of which the row structure is designed as [EdgeID,NodeID(from),NodeID(to)].

Details

This object is the immediate result of establishing edges between nodes from a "SpatialLines" or "SpatialLinesDataFrame" object, see readshpnw. As a transition object, it is a necessary input for many functions in this package.

Note

If the parameter "Detailed" specified in readshpnw is TRUE, all the endpoints of polyines will be extracted as nodes, then the converted graph will have the same spatial details with the transformed "SpatialLines" or "SpatialLinesDataFrame" object. To retrieve the original attributes in the "SpatialLinesDataFrame" object, the original edge ID is also kept and the row structure will be [EdgeID,eid,NodeID(from),NodeID(to)], in which EdgeID refers to the new edge id while eid represents the original edge ID.

Author(s)

Binbin Lu <lubinbin220@gmail.com>

ME.simplification *Simplify multiple edges in a network*

Description

This function simplfies multiple-edge into one representative edge, where multiple-edge refers to a set of edges sharing with the same pair of nodes.

Usage

```
ME.simplification(nodelist, edgelist, eadf=NULL, ea.prop=NULL, Directed=F,
                  DegreeL=NULL, InDegreeL=NULL, OutDegreeL=NULL,Nexception=NULL,
                   Eexception=NULL)
```

Arguments

nodelist	A "nodelist" object
edgelist	An "edgelist" object
eadf	Attribute data frame for all the edges, of which the number of rows equals to the number of edges;
ea.prop	a vector (of lenght equal to the number of columns in "eadf") consisted of 1, 2, 3 or 4, and one value cooresponds to a kind of redefination function, see Redef.functions;
Directed	A logical parameter to specify whether edges are directed or not;
DegreeL	A vector (of length equal to the number of nodes) of degrees for the nodes, and it should be NULL if "Directed" is TRUE;
InDegreeL	A vector (of length equal to the number of nodes) of in-degrees for the nodes;
OutDegreeL	A vector of out-degrees for the nodes;
Nexception	A vector of node IDs considered as exceptions, and all the nodes included won't be processed;
Eexception	A vector of edge IDs considered as exceptions, and all the edges included won't be processed;

Value

Two types of list for "undirected" and "directed" edges respectively:

For "undirected" network:

newNodelist	The new nodelist with multiple-edges simplified;
newEdgelist	The new edgelist with multiple-edges simplified;
newEadf	The new attribute data frame for the returned edgelist;
DegreeL	The new degree vector cooresponding to the returned nodelist;

For "directed" network:

newNodelist	The new nodelist with multiple-edges simplified;
newEdgelist	The new edgelist with multiple-edges simplified;
newEadf	The new attribute data frame for the returned edgelist;
InDegreeL	The new in-degree vector cooresponding to the returned "nodelist";
OutDegreeL	The new out-degree vector cooresponding to the returned "nodelist";

Note

If input edges are directed, the judgement of multiple-edge will be based on its in-degree and out-degree equalling to 1, then InDegreeL and OutDegreeL should be specified; Otherwise, the criterion will be the degree equalling to 2 and then DegreeL is the significant.

Author(s)

Binbin Lu <lubinbin220@gmail.com>

See Also

SL.extraction, PN.amalgamation

---

nel2igraph *Produce a "igraph" object*

Description

This function is to produce an "igraph" object using the extracted nodelist and edgelist from a spatial network.

Usage

```
nel2igraph(nodelist, edgelist, weight = NULL, eadf = NULL, Directed = FALSE)
```

Arguments

nodelist	A "nodelist" object
edgelist	An "edgelist" object
weight	A numberic vector for weighting all the edges in the edgelist, of which the length equals to the number of edges;
eadf	Attribute data frame for all the edges, of which the number of rows equals to the number of edges;
Directed	Logical scalar, whether to create a directed graph.

Details

1. The vector "weight" will be the default weights of the edges for any relative computation carried out on the produced "igraph" object.

2. The coordinates of vertices are attached as attributes "X" and "Y", which could be retrived with function "get.vertex.attribute" from package **igraph**.

Value

gr	An object of "igraph"

Author(s)

Binbin Lu <lubinbin220@gmail.com>

Examples

```
data(ORN)
rtNEL<-readshpnw(rn, ELComputed=TRUE)
#Add the edge length as the weight for graph
igr<-nel2igraph(rtNEL[[2]],rtNEL[[3]],weight=rtNEL[[4]])
plot(igr, vertex.label=NA, vertex.size=2,vertex.size2=2)
plot(rn)
```

nodelist *An object defined for extracted nodes from a network*

Description

This is a structure for nodes extracted from the spatial network, and each node cooresponds to one row in the object; for each row, the structure is defined as [NodeID,coordinate(X,Y)].

Details

This list is the immediate result of extracting nodes from a "SpatialLines" or "SpatialLinesDataFrame" object, see readshpnw. As a transition object, it is a necessary input for many functions in this package, like Degree.list, ME.simplification, Nodes.coordinates.

Author(s)

Binbin Lu <lubinbin220@gmail.com>

Nodes.coordinates *Return the coordinates of the given "nodelist"*

Description

The function returns the coordinates of the given "nodelist" in two columns X and Y.

Usage

```
Nodes.coordinates(nodelist)
```

Arguments

nodelist	An "nodelist" object

Value

Nodesxy	A matrix with two columns X and Y

Author(s)

Binbin Lu <lubinbin220@gmail.com>

nt.connect *Check the connectivity of given network*

Description

This function is to check the connectivity of a given network. Its principle is to traverse all the nodes and do the classifation: for any two different node they fall into a same category if one can be visited from the other one, i.e. there is a path between them. Different categories (connected parts) are represented in different colors in the output plot and the majority connected part is returned as a "SpatialLinesDataFrame" object.

Usage

```
nt.connect(nt)
```

Arguments

nt	A "SpatialLines" or "SpatialLinesDataFrame" object.

Author(s)

Binbin Lu <lubinbin220@gmail.com>

ORN *Ontario Road Network data set (SpatialLinesDataFrame)*

Description

This data set is a part of Ontario Road Network data set downloaded from Ontario Ministry of Natural Resources.

Usage

```
data(ORN)
```

Format

A "SpatialLinesDataFrame" object.

Source

http://www.geographynetwork.ca/website/orn/viewer.htm

References

Peterborough, ON: Ontario Ministry of Natural Resources, Ontario Road Network. Available: http://www.geographynetwork.ca/website/orn/viewer.htm, 2006, (Accessed Dec. 4, 2009)

Examples

```
data(ORN)
plot(rn)
```

`PN.amalgamation` *Amalgamate edges connected by a pseudo-node*

Description

This function is to amalgamate edges connected by a pseudo-node, i.e. to get rid of pseudo-nodes in a network.

Usage

```
PN.amalgamation(nodelist, edgelist, eadf=NULL, ea.prop=NULL, Directed=F,
                DegreeL=NULL, InDegreeL=NULL, OutDegreeL=NULL,Nexception=NULL,
                Eexception=NULL)
```

Arguments

`nodelist`	A "nodelist" object
`edgelist`	An "edgelist" object
`eadf`	Attribute data frame for all the edges, of which the number of rows equals to the number of edges;
`ea.prop`	a vector (of lenght equal to the number of columns in "eadf") consisted of 1, 2, 3 or 4, and one value cooresponds to a kind of redefination function, see Redef.functions;
`Directed`	A logical parameter to specify whether edges are directed or not;
`DegreeL`	A vector (of length equal to the number of nodes) of degrees for the nodes, and it should be NULL if "Directed" is TRUE;
`InDegreeL`	A vector (of length equal to the number of nodes) of in-degrees for the nodes;
`OutDegreeL`	A vector of out-degrees for the nodes;
`Nexception`	A vector of node IDs considered as exceptions, and all the nodes included won't be processed;
`Eexception`	A vector of edge IDs considered as exceptions, and all the edges included won't be processed;

Value

Two types of list for "undirected" and "directed" edges respectively: For undirected type:

`newNodelist`	The new nodelist with pseudo-nodes removed;
`newEdgelist`	The new edgelist with edges connected by a pseudo-node amalgamated;
`newEadf`	The new attribute data frame for the returned "edgelist";

DegreeL	The new degree vector cooresponding to the returned "nodelist";

For directed type:

newNodelist	The new "nodelist" with pseudo-node removed;
newEdgelist	The new "edgelist" with edges connected by a pseudo-node amalgamated;
newEadf	The new attribute data frame for the returned "edgelist";
InDegreeL	The new in-degree vector cooresponding to the returned "nodelist";
OutDegreeL	The new out-degree vector cooresponding to the returned "nodelist";

Note

If input edges are directed, a node is recognised as a pseudo-node with in-degree and out-degree equalling to 1, then InDegreeL and OutDegreeL are required; Otherwise, a node is recognised as a pseudo-node with degree equalling to 2.

Author(s)

Binbin Lu <lubinbin220@gmail.com>

See Also

SL.extraction, ME.simplification

Examples

```
data(ORN)
rtNEL<-readshpnw(rn, ELComputed=TRUE)
res.sl<-SL.extraction(rtNEL[[2]],rtNEL[[3]])
res.me<-ME.simplification(res.sl[[1]],res.sl[[2]],DegreeL=res.sl[[4]])
res.pn<-PN.amalgamation(res.me[[1]],res.me[[2]],DegreeL=res.me[[4]])
ptcoords<-Nodes.coordinates(res.pn[[1]])
plot(rn)
points(ptcoords, col="green")
plot(rn)
points(Nodes.coordinates(rtNEL[[2]]), col="red")
```

---

points2network *Integrate a point data set with a network*

Description

In order to associate a point data set with a given network, this function finds a corresponding node for each point in the data set under a certain rule(see details).

Usage

```
points2network(ntdata,pointsxy,mapping.method=1,ELComputed=FALSE,longlat=F,
               Detailed=F, ea.prop=NULL)
```

Arguments

ntdata	a "SpatialLinesDataFrame" object
pointsxy	A two column matrix contains X-Y coordinates of the points.
mapping.method	Mapping methods between the given points and network will be used, see the details below
ELComputed	if TRUE, the edge length will be computed
longlat	if TRUE, use distances on an ellipse with WGS84 parameters
Detailed	if TRUE, all the points of polylines will be regarded as nodes in the network; if FALSE, only endpoints of polylines are treated as nodes
ea.prop	A vector contains 1 or 0 values (the length equals to the number of arributes of the "ntdata"): 1: the corresponding attribute will be kept and re-orgnised for the new network; 0: the corresponding attribute will not be kept.

Details

The value of mapping.method can be 1,2,3 or 4, which respectively means:

1: Mapping each point to the nearest node in the network/graph

2: Mapping each point to the nearest point (add them as nodes if they are not) on the network

3: Add a new edge(virtual edge) between each point and the nearest node

4: Add a new edge(virtual edge) between each point and the nearest point

Value

A list consisted of:

nodelist	A "nodelist" object
edgelist	An "edgelist" object
CoorespondIDs	A vector contains cooresponding node IDs for each point
nodexlist	A vector contains X-coordinates of all the nodes
nodeylist	A vector contains Y-coordinates of all the nodes
Eadf	a data frame of edge attributes, dataframe(EdgeID,... (attributes extracted from original file)...)
VElist	A list of virtual edges if existed
Edgelength	If ELComputed is TRUE, Edgelength will be a vector consisted of edge lengths cooresponding to each edge, else it will be NULL

Author(s)

Binbin Lu <lubinbin220@gmail.com>

See Also

ptsinnt.view

Examples

```
## Not run:
data(ORN)
pts<-spsample(rn, 100, type="random")
ptsxy<-coordinates(pts)[,1:2]
ptsxy<-cbind(ptsxy[,1]+0.008,ptsxy[,2]+0.008)
#Mapping each point to the nearest node in the network/graph
res<-points2network(ntdata=rn,pointsxy=ptsxy, mapping.method=1)
ptsinnt.view(ntdata=rn, nodelist=res[[1]], pointsxy=ptsxy,
             CoorespondIDs=res[[3]])
#Mapping each point to the nearest point (add them as nodes if they are not) on
#the network
res<-points2network(ntdata=rn,pointsxy=ptsxy, mapping.method=2,ea.prop=rep(0,37))
ptsinnt.view(ntdata=rn, nodelist=res[[1]], pointsxy=ptsxy, CoorespondIDs=res[[3]])
#Add a new edge(Virtual edge) between each point and the nearest node
res<-points2network(ntdata=rn,pointsxy=ptsxy, mapping.method=3,ea.prop=rep(0,37))
VElist<-res[[7]]
ptsinnt.view(ntdata=rn, nodelist=res[[1]], pointsxy=ptsxy, CoorespondIDs=res[[3]],
             VElist=VElist)
#Add a new edge(Virtual edge) between each point and the nearest point
res<-points2network(ntdata=rn,pointsxy=ptsxy, mapping.method=4,ea.prop=rep(0,37))
VElist<-res[[7]]
ptsinnt.view(ntdata=rn, nodelist=res[[1]], pointsxy=ptsxy, CoorespondIDs=res[[3]],
             VElist=VElist)

## End(Not run)
```

`ptsinnt.view` *Visualize the result of integrating a set of data points with given network*

Description

The function is to visualize the result of integrating a set of data points with a given network. It is an immediate visualization tool of the result from function points2network to give user an impression how the points connected with the given network.

Usage

```
ptsinnt.view(ntdata, nodelist, pointsxy, CoorespondIDs, VElist=NULL)
```

Arguments

`ntdata`	A "SpatialLines" or "SpatialLinesDataFrame" object;
`nodelist`	An nodelist object, see nodelist;
`pointsxy`	A two columns vector of X-Y coordinates of the given points set;
`CoorespondIDs`	A vector(of the length equal to the number of points) of cooresponding node IDs of each point;
`VElist`	A list of virtual edges if existed

Author(s)

Binbin Lu <lubinbin220@gmail.com>

See Also

points2network

Examples

```
data(ORN)
pts<-spsample(rn, 100, type="random")
ptsxy<-coordinates(pts)[,1:2]
ptsxy<-cbind(ptsxy[,1]+0.008,ptsxy[,2]+0.008)
#Mapping each point to the nearest node in the network/graph
res<-points2network(ntdata=rn,pointsxy=ptsxy, mapping.method=1)
#Visualize the results without virtual edges
ptsinnt.view(ntdata=rn, nodelist=res[[1]], pointsxy=ptsxy,
             CoorespondIDs=res[[3]])
#Visualize the results with virtual edges
res<-points2network(ntdata=rn,pointsxy=ptsxy, mapping.method=3,
                    ea.prop=rep(0,37))
VElist<-res[[7]]
ptsinnt.view(ntdata=rn, nodelist=res[[1]], pointsxy=ptsxy,
             CoorespondIDs=res[[3]], VElist=VElist)
```

---

readshpnw *Read a network from a "SpatialLines" or "SpatialLinesDataFrame" object*

Description

This function is to split nodes and edges from a "SpatialLines" or "SpatialLinesDataFrame" object, and return "nodelist" and "edgelist".

Usage

```
readshpnw(data=list(), ELComputed=FALSE, longlat=FALSE, Detailed=FALSE, ea.prop=NULL)
```

Arguments

data	A "SpatialLines" or "SpatialLinesDataFrame" object;
ELComputed	if TRUE, the edge length will be computed
longlat	If TRUE, distance will be calculated on an ellipse with WGS84 parameters;
Detailed	Default is FALSE, only two endpoints (starting and ending) of each polyline are recognised as nodes and collected in the "nodelist"; If TRUE, all the endpoints will be recognised as nodes and collected.

`ea.prop` If data is a "SpatialLinesDataFrame" object and "Detailed" is TRUE, the ea.prop shoul be a given as a vector (of lenght equal to the number of columns in data.frame(data)) with values 0 or 1, for defining the rules of re-attributing the new edges: 0 means that equalization is used for the arrtibute inherition from the original data; 1 implies that weighted-mean based on the edge length is adopted.

Details

This function is the first step to convert a newtwork data into an object of "igraph". With a given "SpatialLines" or "SpatialLinesDataFrame", it produces a nodelist, "edgelist", and data frame for edge attributes.

Value

A list consisted of:

`Detailed` TRUE if the output is under a "Detailed" mode, and "edgelist" will have a different structure;

`nodelist` A "nodelist" object

`edgelist` An "edgelist" object

`Edgelength` A vector (of length equal to the number of edges) of edge lengths if "ELComputed" is TRUE;

`Eadf` A data frame of edge attributes, [EdgeID,...(items extracted from the "SpatialLinesDataFrame" object)...]

`nodexlist` A vector contains X-coordinates of all the nodes

`nodeylist` A vector contains Y-coordinates of all the nodes

Author(s)

Binbin Lu <lubinbin220@gmail.com>

Examples

```
data(ORN)
rtNEL<-readshpnw(rn)
nl<-rtNEL[[2]]
el<-rtNEL[[3]]
#Compute edge length
rtNEL<-readshpnw(rn, ELComputed=TRUE)
edgelength<-rtNEL[[4]]
eadf<-rtNEL[[5]]
```

Redef.functions *A collection of functions for redefining attributes of new edges*

Description

The function is used to redifine attributes of new edges when the optimization functions PN.amalgamation and ME.simplification are conducted.

Usage

```
Redef.functions(v, typ=1)
```

Arguments

v	An input vector for the specified function;
typ	a value from "1, 2, 3, 4" to specify the redefinition method: 1->sum(v), 2->min(v),3->max(v),4->mean(v)

Note

With specific cases, it is easy to extend this collection with some other functions.

Author(s)

Binbin Lu <lubinbin220@gmail.com>

SL.extraction *Extract self-loops form given network*

Description

This function is to extract self-loops away in given network.

Usage

```
SL.extraction(nodelist, edgelist, eadf=NULL,Directed=F, DegreeL=NULL,
              InDegreeL=NULL, OutDegreeL=NULL,Nexception=NULL,Eexception=NULL)
```

Arguments

nodelist	A nodelist object, see nodelist;
edgelist	An edgelist object, see edgelist;
eadf	Attribute data frame for all the edges;
Directed	A logical parameter to specify whether edges are directed or not;
DegreeL	A vector (of length equal to the number of nodes) of degrees for the nodes, and it should be null if "Directed" is TRUE;
InDegreeL	A vector (of length equal to the number of nodes) of in-degrees for the nodes;
OutDegreeL	A vector of out-degrees for the nodes;
Nexception	A vector of node IDs considered as exceptions, and all the nodes included won't be processed;
Eexception	A vector of edge IDs considered as exceptions, and all the edges included won't be processed;

Details

If edges in the "edgelist" are directed, the InDegreeL and OutDegreeL will be calculated (if not given) and updated synchronized with self-loop extration; otherwise only DegreeL is returned.

Value

Two types of list for undirected and directed edges respectively: For undirected type:

newNodelist	The new "nodelist" with self-loops extracted;
newEdgelist	The new "edgelist" with self-loops extracted;
newEadf	The new attribute data frame for the amended edgelist;
DegreeL	The new degree vector cooresponding to the returned nodelist;

For directed type:

newNodelist	The new "nodelist" with self-loops extracted;
newEdgelist	The new "edgelist" with self-loops extracted;
newEadf	The new attribute data frame for the returned edgelist;
InDegreeL	The new in-degree vector cooresponding to the returned "nodelist";
OutDegreeL	The new out-degree vector cooresponding to the returned "nodelist";

Author(s)

Binbin Lu <lubinbin220@gmail.com>

See Also

PN.amalgamation, ME.simplification

Index