# Package 'shp2graph'

August 22, 2017

**Version** 0-3

**Date** 2017-08-21

**Title** Convert a SpatialLinesDataFrame Object to an 'igraph'-Class Object

**Author** Binbin Lu

**Maintainer** Binbin Lu <lubinbin220@gmail.com>

**Depends** R (>= 3.0.0),igraph

**Imports** methods, maptools,sp

**Description** Functions for converting network data from a SpatialLinesDataFrame object to an 'igraph'-Class object.

**License** GPL (>= 2)

**Repository** CRAN

**NeedsCompilation** yes

**Date/Publication** 2017-08-22 03:08:26 UTC

## R topics documented:

---

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

---

## Description

A function that returns degrees of nodes from provided "nodelist" and "edgelist"; while 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 if edges are directed; FALSE, otherwise |

## Value

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

| | |
|---|---|
| `DegreeL` | An integer vector of degrees for each node in the given "nodelist" |

For directed type:

| | |
|---|---|
| `InDegreeL` | An integer vector of In-degrees for each node in the given "nodelist" |
| `OutDegreeL` | An integer vector of Out-degrees for each node in the given "nodelist" |

## Note

This function returns differently for undirected and directed networks, where both In-degrees and Out-degrees are calculated for a directed network, and only degrees are returned for an undirected networks.

## Author(s)

Binbin Lu <lubinbin220@gmail.com>

Directed *Orientate all the edges in a given edgelist*

## Description

A function 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 |
| direction.v | A vector (of the length equalling to the number of edges in the given "edgelist") with values of 1 (TRUE) or 0 (FALSE), 1 (TRUE) indicates a directed (one-way) edge, while 0 (FALSE) means an undirected (double-way) edge |
| eadf | A data frame of attributes corresponding to all the edges; |

## Details

Within a road network, some road segments might be one-way while the rest are double-way. This suitation makes it complex to define directed or undirected edges in a graph. This function is to orientate each edge according to the given vector, "direction.v": 1 (TRUE) indicates one-way, while 0 (FALSE) represents double-way. All the double-way (undirected) edges in the given "edgelist" are redefined as two directed edges (e.g. a double-way (undirected) edge(nid1, nid2) is redefined as two one-way (directed) edges, (nid1, nid2) and (nid2, nid1)). The one-way (directed) edges are left in their directed forms. In other words, all the edges returned by this function will be directed, and used to be construct a directed graph.

## Value

A list consisted of:

| | |
|---|---|
| newEdgelist | An "edgelist" object with directed edges |
| newEadf | A data frame of attributes for the new "edgelist" |

## Author(s)

Binbin Lu <lubinbin220@gmail.com>

## edgelist *A designed structure to denote edges from a spatial network*

### Description

"edgelist" is an interchange structure of edges from a spatial network into an "igraph" object. It is a three-column matrix, of which each row is designed as [EdgeID,NodeID(from),NodeID(to)].

### Details

Both "nodelist" and "edgelist" are interchange structures of nodes and edges extracted from a spatial network in a "SpatialLines" or "SpatialLinesDataFrame" object. They are always concerned together, and returned by the function readshpnw as initial results for the following steps.

### 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

A function to simplfy multiple-edge into one representative edge, where a multiple-edge structure 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 | A data frame of attributes corresponding to all the edges; |
| ea.prop | A vector (of the length equalling to the number of edge attributes in "eadf") with values of 1, 2, 3 or 4, and each specific value represent a function to assign the attributes of the edited edges, see also Redef.functions:<br>1->sum(v)<br>2->min(v)<br>3->max(v)<br>4->mean(v) |
| Directed | TRUE if edges are directed, FALSE otherwise; |
| DegreeL | DegreeLAn integer vector of degrees for each node in the given "nodelist", and it could be ignored if edges are directed |
| InDegreeL | An integer vector of In-degrees for each node in the given "nodelist", and it could be ignored if edges are undirected |
| OutDegreeL | An integer vector of Out-degrees for each node in the given "nodelist", and it could be ignored if edges are undirected |
| Nexception | A vector of node IDs concerned as exceptions, and all the edges with these nodes included won't be processed; |
| Eexception | A vector of edge IDs concerned as exceptions, and all these edges won't be processed; |

## Value

Two types of list returned for undirected and directed edges, respectively:

For "undirected" edges:

| | |
|---|---|
| newNodelist | New node list with multiple-edges simplified; |
| newEdgelist | New edge list with multiple-edges simplified; |
| newEadf | New attribute data frame for the returned edgelist; |
| DegreeL | New degree vector cooresponding to the newly returned node list; |

For "directed" network:

| | |
|---|---|
| newNodelist | New node list with multiple-edges simplified; |
| newEdgelist | New edgelist with multiple-edges simplified; |
| newEadf | Data frame of attributes for the newly returned edge list; |
| InDegreeL | New In-degree vector cooresponding to the newly returned node list; |
| OutDegreeL | New Out-degree vector cooresponding to the newly returned node list; |

## Note

Edges are recognised as a structure of multiple-edge when they share the same pair of nodes. Note that the order of the nodes should be also the same when edges are directed.

### Author(s)

Binbin Lu <lubinbin220@gmail.com>

### See Also

SL.extraction, PN.amalgamation

---

nel2igraph *Produce an "igraph" object*

---

### Description

A function to produce an "igraph" object with the "nodelist" and "edgelist", which could be returned by the function readshpnw.

### Usage

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

### Arguments

| | |
|---|---|
| nodelist | A "nodelist" object |
| edgelist | An "edgelist" object |
| weight | A numberic vector to weight all the edges in the "edgelist", of which the length equals to the number of edges; |
| eadf | A data frame of attributes corresponding to all the edges; |
| Directed | TRUE if edges are directed, FALSE otherwise; |

### Details

1. The weighting vector, "weight", will be used as default for any weigted calculations with edges in the "igraph" object.

2. The coordinate of each node is attached as attributes "X" and "Y", which could be retrived via the function "get.vertex.attribute" from the package **igraph**.

### Value

| | |
|---|---|
| gr | An "igraph" object |

### 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` *A designed structure to denote nodes from a spatial network*

---

## Description

"nodelist" is an interchange structure of nodes from a spatial network into an "igraph" object. It is a two-column data frame, of which each row is designed as [NodeID,coordinate(X,Y)].

## Details

Both "nodelist" and "edgelist" are interchange structures of nodes and edges extracted from a spatial network in a "SpatialLines" or "SpatialLinesDataFrame" object. They are always concerned together, and returned by the function readshpnw as initial results for the following steps.

## Author(s)

Binbin Lu <lubinbin220@gmail.com>

---

`Nodes.coordinates` *Return the coordinates of nodes in a "nodelist"*

---

## Description

A function that returns the coordinates of nodes in a two-column matrix (X,Y) from a "nodelist"

## Usage

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

## Arguments

| | |
|---|---|
| `nodelist` | A "nodelist" object |

## Value

| | |
|---|---|
| `Nodesxy` | A two-column matrix (X, Y) |

## Author(s)

Binbin Lu <lubinbin220@gmail.com>

## nt.connect *Check the connectivity of a given network*

### Description

A function to check the connectivity of a given network.

### Usage

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

### Arguments

| | |
|---|---|
| nt | A "SpatialLines" or "SpatialLinesDataFrame" object. |

### Details

In this function, all the nodes are traversed, and classified dynamically via a rule, that any pair of different nodes fall into a same category if they could be reached from each other. A map will be plotted spontaneously with differet categories (i.e. self-connected parts) in distinct colors. Finally, the self-connected part with the most nodes will be returned as a new "SpatialLinesDataFrame" object.

### Author(s)

Binbin Lu <lubinbin220@gmail.com>

## ORN *Ontario road network data (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)

---

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

---

## Description

A function to amalgamate edges connected by a pseudo-node.

## 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` | A data frame of attributes corresponding to all the edges; |
| `ea.prop` | A vector (of the length equalling to the number of edge attributes in "eadf") with values of 1, 2, 3 or 4, and each specific value represent a function to assign the attributes of the edited edges, see also Redef.functions:<br>1->sum(v)<br>2->min(v)<br>3->max(v)<br>4->mean(v) |
| `Directed` | TRUE if edges are directed, FALSE otherwise; |
| `DegreeL` | An integer vector of degrees for each node in the given "nodelist", and it could be ignored if edges are directed |
| `InDegreeL` | An integer vector of In-degrees for each node in the given "nodelist", and it could be ignored if edges are undirected |
| `OutDegreeL` | An integer vector of Out-degrees for each node in the given "nodelist", and it could be ignored if edges are undirected |
| `Nexception` | A vector of node IDs concerned as exceptions, and all the edges with these nodes included won't be processed; |
| `Eexception` | A vector of edge IDs concerned as exceptions, and all these edges won't be processed; |

## Value

Two types of list returned for undirected and directed edges, respectively:

For "undirected" edges:

| | |
|---|---|
| newNodelist | New node list with pseudo-nodes removed; |
| newEdgelist | New edge list with pseudo-nodes removed; |
| newEadf | New attribute data frame for the returned edgelist; |
| DegreeL | New degree vector cooresponding to the newly returned node list; |

For "directed" network:

| | |
|---|---|
| newNodelist | New node list with pseudo-nodes removed; |
| newEdgelist | New edgelist with pseudo-nodes removed; |
| newEadf | Data frame of attributes for the newly returned edge list; |
| InDegreeL | New In-degree vector cooresponding to the newly returned node list; |
| OutDegreeL | New Out-degree vector cooresponding to the newly returned node list; |

## Note

Node in a directed network is recognised a a pseudo-node with in-degree and out-degree equalling to 1; while it could be a pseudo-node with degree equalling to 2 in a undirected network.

## 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 into a network*

## Description

A function to integrate an individual point data set into a network under different rules (see details below).

## Usage

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

## Arguments

| | |
|---|---|
| `ntdata` | A "SpatialLinesDataFrame" or "SpatialLines" object |
| `pointsxy` | A two-column matrix of point coordinates (X, Y) |
| `approach` | specified by an integer ranging from 1 to 4 to define the approach for integration:<br>1: Mapping each point to the nearest node in the network/graph<br>2: Mapping each point to the nearest point (add them as nodes if they are not) on the network<br>3: Add a new edge(virtual edge) between each point and the nearest node<br>4: Add a new edge(virtual edge) between each point and the nearest point |
| `ELComputed` | If TRUE, the length of each edge will be calculated and returned |
| `longlat` | If TRUE, distances on an ellipse with WGS84 parameters will be returned |
| `Detailed` | If TRUE, all the vertices within the polylines of a spatial network will be recognised as nodes; if FALSE, only two endpoints of each polyline are treated as nodes |
| `ea.prop` | A vector (of the length equalling to the number of edge attributes in in the network "ntdata") with values of 1 or 0:<br>1: the corresponding column in the data frame will be kept for the new network;<br>0: the corresponding column in the data frame will not be kept. |

## Value

A list consisted of:

| | |
|---|---|
| `nodelist` | A "nodelist" object |
| `edgelist` | An "edgelist" object |
| `CoorespondIDs` | A vector of the cooresponding node ID for each point in "pointsxy" |
| `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 attributes of the returned edges, of which the structure is a data frame with [EdgeID,(attributes inherited from the input network data)] |
| VElist | A list of virtual edges if added, i.e. approaches 3 and 4 are used |
| Edgelength | If "ELComputed" is TRUE, lengths of all the edges will be returned as a numeric vector; otherwise 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, approach=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, approach=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, approach=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, approach=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 integration between a point data set and spatila network*

---

## Description

A specific function to visualizethe integration between a point data set and spatila network, i.e. results returned by the function points2network

## Usage

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

## Arguments

| | |
|---|---|
| ntdata | A "SpatialLinesDataFrame" or "SpatialLines" object |
| nodelist | A "nodelist" object |
| pointsxy | A two-column matrix of point coordinates (X, Y) |
| CoorespondIDs | A vector of the cooresponding node ID for each point in "pointsxy" |
| VElist | A list of virtual edges if added, i.e. approaches 3 and 4 are used |

## 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

A function to read a spatial network from a "SpatialLines" or "SpatialLinesDataFrame" object, and resolve it into a "nodelist" and "edgelist" for the following conversion.

## Usage

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

## Arguments

| | |
|---|---|
| ntdata | A "SpatialLinesDataFrame" or "SpatialLines" object |
| ELComputed | If TRUE, the length of each edge will be calculated and returned |
| longlat | If TRUE, distances on an ellipse with WGS84 parameters will be returned |
| Detailed | If TRUE, all the vertices within the polylines of a spatial network will be recognised as nodes; if FALSE, only two endpoints of each polyline are treated as nodes |
| ea.prop | A vector (of the length equalling to the number of edge attributes in in the network "ntdata") with values of 1 or 0:<br>1: the corresponding column in the data frame will be kept for the new network;<br>0: the corresponding column in the data frame will not be kept. |

## Details

This function plays the first step to convert a spatial network (in a "SpatialLines" or "SpatialLinesDataFrame" object) into an "igraph" object by returning a "nodelist" and "edgelist".

## Value

A list consisted of:

| | |
|---|---|
| Detailed | TRUE if the output is under a "Detailed" mode; |
| nodelist | A "nodelist" object |
| edgelist | An "edgelist" object |
| Edgelength | If "ELComputed" is TRUE, lengths of all the edges will be returned as a numeric vector; otherwise it will be NULL. |
| Eadf | A data frame of attributes of the returned edges, of which the structure is a data frame with [EdgeID,(attributes inherited from the input network data)] |
| 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 edited edges*

---

## Description

A function to provide a collection of functions to redefine attributes of edited edges in the functions PN.amalgamation and ME.simplification.

## Usage

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

## Arguments

| | |
|---|---|
| `v` | An input vector for the specified function; |
| `typ` | A value equalling to 1, 2, 3 or 4 to specify a redefinition function:<br>1->sum(v)<br>2->min(v)<br>3->max(v)<br>4->mean(v) |

## Note

It is easy to include more specific methods for attribute redefinition via incorporating the corresponding functions. In other words, you can define more methods by numbering them with 5, 6, etc.

## Author(s)

Binbin Lu <lubinbin220@gmail.com>

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

## Description

A function 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; |
| edgelist | An "edgelist" object |
| eadf | A data frame of attributes corresponding to all the edges; |
| Directed | TRUE if edges are directed, FALSE otherwise; |
| DegreeL | DegreeLAn integer vector of degrees for each node in the given "nodelist", and it could be ignored if edges are directed |
| InDegreeL | An integer vector of In-degrees for each node in the given "nodelist", and it could be ignored if edges are undirected |
| OutDegreeL | An integer vector of Out-degrees for each node in the given "nodelist", and it could be ignored if edges are undirected |
| Nexception | A vector of node IDs concerned as exceptions, and all the edges with these nodes included won't be processed; |
| Eexception | A vector of edge IDs concerned as exceptions, and all these edges won't be processed; |

## Value

Two types of list returned for undirected and directed edges, respectively:

For "undirected" edges:

| | |
|---|---|
| newNodelist | New node list with self-loops removed; |
| newEdgelist | New edge list with self-loops removed; |
| newEadf | New attribute data frame for the returned edgelist; |
| DegreeL | New degree vector cooresponding to the newly returned node list; |

For "directed" network:

| | |
|---|---|
| newNodelist | New node list with self-loops removed; |
| newEdgelist | New edgelist with self-loops removed; |
| newEadf | Data frame of attributes for the newly returned edge list; |
| InDegreeL | New In-degree vector cooresponding to the newly returned node list; |
| OutDegreeL | New Out-degree vector cooresponding to the newly returned node list; |

## Note

An edge is recognised as a structure of self-loop when it starts and ends at the same point (location).

## Author(s)

Binbin Lu <lubinbin220@gmail.com>

## See Also

PN.amalgamation, ME.simplification

# Index