Package ‘simplifyNet’

November 15, 2022

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Network Sparsification</td>
</tr>
<tr>
<td>Version</td>
<td>0.0.1</td>
</tr>
<tr>
<td>Language</td>
<td>en-US</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Alexander Mercier <a href="mailto:amercier@g.harvard.edu">amercier@g.harvard.edu</a></td>
</tr>
</tbody>
</table>

Description


Depends

R (>= 3.4),

Imports

igraph (>= 1.3.1), sanic (>= 0.0.1), Matrix (>= 1.2-18), methods (>= 4.0.2), stats (>= 4.0.2), dplyr (>= 1.0.9)

License

GPL (>= 3)

Encoding

UTF-8

RoxygenNote

7.2.1

NeedsCompilation

no

Author

Andrew Kramer [aut], Alexander Mercier [aut, cre, trl], Shubhankar Tripathi [ctb], Tomlin Pulliam [ctb], John Drake [ctb] (<https://orcid.org/0000-0003-4646-1235>)

Repository

CRAN

Date/Publication

2022-11-15 16:30:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>bestpath</th>
<th>EffR</th>
<th>EffRSparse</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
**bestpath**

---

Sparsification via Best Path

**Description**
Calculates network sparsifier from best path.

**Usage**

```
bestpath(network, directed = FALSE, associative = TRUE)
```

**Arguments**

- `network`: Weighted adjacency matrix, weighted igraph network, or edge list formatted \( \text{l} \ \text{n1} \ \text{n2} \ \text{weight} \) with colnames \( \text{c(“n1”, “n2”, “weight”)}. \)
- `directed`: If TRUE, specifies that the inputted network is directed. Default is FALSE.
- `associative`: Designates if the network is associative where edge weight determines "similarity" or "strength" or dissociative where edge weight denotes "dissimilarity" or "distance".
  - If the network is associative, then the shortest path would be found by looking at \( w^{-1}_e \) where weaker association between nodes suggests a larger distance between nodes for shortest paths.
  - If the network is dissociative, then the shortest path would be between \( w_e \).

**Value**

Edge list of sparsified network via best path.

**Author(s)**

Alexander Mercier
Andrew Kramer

**References**

Examples

```r
# Generate random ER graph with uniformly random edge weights
g = igraph::erdos.renyi.game(50, 0.1)
igraph::E(g)$weight <- runif(length(igraph::E(g)))

# Sparsify g via bestpath
S = simplifyNet::bestpath(g, directed = FALSE, associative = TRUE)

# Show edge list conversion
sg = simplifyNet::net.as(S, net.to = "igraph", directed = FALSE)
igraph::ecount(sg)/igraph::ecount(g)  # fraction of edges in the sparsifier
```

---

**EffR**

**Effective resistances calculator**

**Description**

Calculate or approximate the effective resistances of an inputted, undirected graph. There are three methods.

1. 'ext' which exactly calculates the effective resistances (WARNING! Not ideal for large graphs).
2. 'spl' which approximates the effective resistances of the inputted graph using the original Spielman-Srivastava algorithm.
3. 'kts' which approximates the effective resistances of the inputted graph using the implementation by Koutis et al. (ideal for large graphs where memory usage is a concern).

The relative fidelity of the approximation methods is governed by the variable epsilon.

**Usage**

```r
EffR(network, epsilon = 0.1, type = "kts", tol = 1e-10)
```

**Arguments**

- **network**: Weighted adjacency matrix, weighted igraph network, or edge list formatted `| n1 | n2 | weight |` with colnames `c("n1", "n2", "weight")`.
- **epsilon**: Variable epsilon governs the relative fidelity of the approximation methods 'spl' and 'kts'. The smaller the value the greater the fidelity of the approximation. Default value is 0.1.
- **type**: There are three methods.
  1. 'ext' which exactly calculates the effective resistances (WARNING! Not ideal for large graphs).
  2. 'spl' which approximates the effective resistances of the inputted graph using the original Spielman-Srivastava algorithm.
  3. 'kts' which approximates the effective resistances of the inputted graph using the implementation by Koutis et al. (ideal for large graphs where memory usage is a concern).
- **tol**: Tolerance for the linear algebra (conjugate gradient) solver to find the effective resistances. Default value is 1e-10.
Details

The fidelity of the effective resistance approximation decreases with a decrease in epsilon. However, it is more important for sparsification by effective resistances that the approximations be roughly equivalent relative to one another, as they will be normalized in a probability distribution where exact values are not needed.

The number of edges contained in the sparsifier will be less than or equal to the number of samples taken, q.

Value

Return either exact or approximate effective resistances for each edge in the same order as "weight" in the edge list.

Author(s)

Alexander Mercier

References


Examples

```r
e_list = matrix(c(1,1,2,2,3,3,1,1,1), 3, 3) #Triangle graph, \( K_3 \), with edge weights equal to 1
effr = simplifyNet::EffR(e_list, epsilon = 0.1, type = 'kts', tol = 1e-10)
```

---

EffRSparse

### Description

Sparsify an undirected network by sampling edges proportional to \( w_e \times R_e \) where \( w_e \) is the weight of edge \( e \) and \( R_e \) is the effective resistance of edge \( e \).

Approximately preserves the graph Laplacian, \( L \), with increasing fidelity as the number of samples taken increases.

### Usage

```r
EffRSparse(network, q, effr, seed, n)
```
Arguments

- **network**: Weighted adjacency matrix, weighted `igraph` network, or edge list formatted as `n1 n2 | weight` with colnames `c("n1", "n2", "weight")`.
- **q**: The numbers of samples taken. The fidelity to the original network increases as the number of samples increases, but decreases the sparseness.
- **effR**: Effective resistances corresponding to each edge. Should be in the same order as "weight".
- **seed**: Set the seed to reproduce results of random sampling.
- **n**: The number of nodes in the network. Default is the max node index of the edge list.

Details

The performance of this method is dependent on the size of the network and fidelity of the effective resistance approximation. The network should be "sufficiently large."
For more details, see: [https://epubs.siam.org/doi/epdf/10.1137/080734029](https://epubs.siam.org/doi/epdf/10.1137/080734029)

Value

A sparsified network, `H`, edge list where the number of edges is dependent on the number of samples taken, `q`.

Author(s)

Daniel A. Spielman,
Alexander Mercier

References


Examples

```r
#Generate random ER graph with uniformly random edge weights
g = igraph::erdos.renyi.game(100, 0.1)
igraph::E(g)$weight <- runif(length(igraph::E(g)))
#Approximate effective resistances
effR = simplifyNet::EffR(g)
#Use effective resistances to create spectral sparsifier by edge sampling
S = simplifyNet::EffRSparse(g, q = 200, effR = effR, seed = 150)
sg = simplifyNet::net.as(S, net.to="igraph", directed=FALSE)
igraph::ecount(sg)/igraph::ecount(g)#fraction of edges in the sparsifier
```
**EList_Mtrx**  
*Edge list to adjacency matrix*

**Description**  
Convert an edge list to an adjacency matrix.

**Usage**  
`EList_Mtrx(E_List, directed = FALSE, n)`

**Arguments**

- **E_List**  
  Edge list formatted | n1 | n2 | weight |.
- **directed**  
  Specifies if the network is directed or undirected. Default is set to undirected.
- **n**  
  Specify number of nodes. Default is `max(E_List[,1:2])`.

**Value**  
Adjacency matrix constructed from edge list, `E_List`, of the class dgCMatrix.

**Author(s)**

Alexander Mercier

---

**gns**  
*Global Network Sparsification*

**Description**  
Remove all edges under certain edge weight threshold.

**Usage**  
`gns(network, remove.prop, cutoff, directed = FALSE)`

**Arguments**

- **network**  
  Weighted adjacency matrix, weighted igraph network, or edge list formatted | n1 | n2 | weight | with colnames c("n1", "n2", "weight").
- **remove.prop**  
  The proportion of highest weighted edges to retain. A value between 0 and 1.
- **cutoff**  
  Threshold value for edge weight thresholding.
- **directed**  
  If TRUE, specifies that the inputted network is directed. Default is FALSE.
irefit

Value

Edge list of sparsified network

Author(s)

Andrew Kramer
Alexander Mercier

Examples

#Generate random ER graph with uniformly random edge weights
g = igraph::erdos.renyi.game(100, 0.1)
igraph::E(g)$weight <- runif(length(igraph::E(g))
#Sparsify g via GNS
S = gns(g, remove.prop = 0.5)
sg = simplifyNet::net.as(S, net.to="igraph", directed=FALSE)
igraph::ecount(sg)/igraph::ecount(g)#fraction of edges in the sparsifier

irefit

Iterative refitting

Description

Iterative sparsification based refitting.

Usage

irefit(
  network,
  func,
  tol,
  rank = "none",
  connected = FALSE,
  directed = FALSE,
  per = 0.5
)

Arguments

network Weighted adjacency matrix, weighted igraph network, or edge list formatted | n1 | n2 | weight | with colnames c("n1", "n2", "weight").
func Model function whose input is the network and whose output is a single real value or a list of reevaluated weights in the first index and a real value in the second.
A wrapper function may have to be written.
tol Allowed error around the original output of func approximated by the sparsified network within which edges are removed. Specifies if method converges.
**rank**

Ranking of edges. Lower ranked edges are removed first. Must be the same length as `nrow(E_List)`.

**connected**

If TRUE, connectivity of the network is prioritized over scoring by `func`.

**directed**

If TRUE, specifies that the inputted network is directed. Default is FALSE.

**per**

Percentage of edges to add/remove from the sparsifier at each step.

**Value**

Sparsified network, H, which still maintains evaluator function, `func`, plus/minus `tol`.

**Author(s)**

Alexander Mercier

Andrew Kramer

**Examples**

```r
#Set scoring function
mean.weight.degree <- function(graph){
  graph.ob <- igraph::graph_from_edgelist(graph[,1:2])
  igraph::E(graph.ob)$weight <- graph[,3]
  return(mean(igraph::strength(graph.ob)))
}

#Generate random graph
g <- igraph::erdos.renyi.game(100, 0.1)
igraph::E(g)$weight <- rexp(length(igraph::E(g)), rate=10) #random edge weights from exp(10)
E_List <- cbind(igraph::as_edgelist(g), igraph::E(g)$weight)
colnames(E_List) <- c("n1", "n2", "weight")
sparse_dist <- simplifyNet::irefit(E_List, func=mean.weight.degree, tol = 0.1)
```

---

### Local Adaptive Network Sparsification

**Description**

Remove all edges under certain probability of the fractional edge weight, alpha.

**Usage**

```r
lans(network, alpha, output, directed = FALSE)
```
Arguments

network  
Weighted adjacency matrix, weighted igraph network, or edge list formatted | n1 | n2 | weight | with colnames c("n1", "n2", "weight").

alpha  
The alpha value is a predetermined threshold to designate statistically important edges by their fractional edge weight at each node. If the probability of choosing that edge via the CDF is less than or equal to alpha, then the edge is not included.

output  
If the output should be directed or undirected. Default is that the output is the same as the input based on adjacency matrix symmetry. If the default is overridden, set as either "undirected" or "directed".

directed  
If TRUE, specifies that the inputted network is directed. Default is FALSE.

Details

For more information on finding alpha values, see: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0016431#s5

Value

Weighted adjacency matrix of sparsified network.

Author(s)

Andrew Kramer
Alexander Mercier

References


Examples

#Generate random ER graph with uniformly random edge weights  
g = igraph::erdos.renyi.game(100, 0.1)  
igraph::E(g)$weight <- runif(length(igraph::E(g)))  
#Sparsify g via LANS  
S = lans(g, alpha = 0.3, output = "undirected", directed = FALSE)  
#Convert sparsifier to edge list  
S_List = simplifyNet::Mtrx_EList(S, directed = FALSE)  
sg = simplifyNet::net.as(S_List, net.to="igraph", directed=FALSE)  
igraph::ecount(sg)/igraph::ecount(g)#fraction of edges in the sparsifier
Mtrx_EList | Adjacency matrix to edge list

Description
Convert an adjacency matrix to an edge list.

Usage
Mtrx_EList(A, directed = FALSE)

Arguments
A | Weighted adjacency matrix.
directed | Specifies if the network is directed or undirected. Default is set to undirected.

Value
An edge list, E_List, of adjacency matrix, A, of the form | n1 | n2 | weight |.

Author(s)
Alexander Mercier

net.as | Network format converter

Description
Convert a network in weighted adjacency matrix, edge list, or igraph to a weighted adjacency matrix, edge list, or igraph format.

Usage
net.as(network, net.to = "E_List", directed = FALSE)

Arguments
network | Weighted adjacency matrix, weighted igraph network, or edge list formatted | n1 | n2 | weight | with colnames c("n1", "n2", "weight").
net.to | Specifies to what format the imputed network is to be converted:
(1) 'E_List' convert to an edge list of the format | n1 | n2 | weight | with colnames c("n1", "n2", "weight").
(2) 'Adj' convert to a weighted adjacency matrix.
(3) 'igraph' convert to a weighted igraph object.
directed | If TRUE, specifies that the inputted network is directed. Default is FALSE.
net.as

Value

A network of the format specified by net.to.

Author(s)

Alexander Mercier
Index

bestpath, 2

EffR, 3
EffRSparse, 4
EList_Mtrx, 6

gns, 6

irefit, 7

lans, 8

Mtrx_EList, 10

net.as, 10