Package 'MCODER'

March 3, 2017

Type Package

Title MCODER, an R Implementation Of MCODE Network Clustering Algorithm

Version 0.1.0

Author Sungjin Kwon [aut, cre], Hyunseok Kim [aut]

Maintainer Sungjin Kwon <ginokwon@yuhs.ac>

Depends R (>= 3.2.2)

Imports sna, igraph

URL https://sourceforge.net/projects/mcoder, https://github.com/ginokwon-yuhs/MCODER

Description Molecular Complex Detection (MCODE) network analysis and visualization based on Bader et al. 2003, "An automated method for finding molecular complexes in large protein interaction networks". The algorithm detects densely interconnected clusters (complex, subgraph). MCODER is an R-implementations of MCODE with additional functions, comprising: a) node annotation, b) visualize of multiple edges and c) visualize of directed graph

License GPL (>= 2)

LazyData TRUE

Date 2017-03-03

RoxygenNote 6.0.1

R topics documented:

`add.annotation` *Adding annotation information to vertices*

Description

Adding annotation information to vertices

Usage

```
add.annotation(x, vertices.par, edges.par, annotation.table)
```

Arguments

`x`	A List of adjacency matrix
`vertices.par`	A matrix of graphical parameters for vertices that is created by the function `graph.par.vertex`
`edges.par`	A matrix of graphical parameters for edges that is created by the function `graph.par.edge`
`annotation.table`	A table for annotation. It must be consist of two columns including vertex label and description

Details

A table for annotation that must include vertex names in its first column and description of vertex in its second column. Return adjacency matrixes of subgraph, tables of graphical parameter for vertices and nodes including annotation information.

Value

A list consists of three matrixes: adjacency matrix for subgraph cluster, matrix of graphical parameters for vertexes, and matrix of graphical parameters for edges

See Also

`cluster.adj.matrix`, `graph.par.vertex`, `graph.par.edge`

Examples

```
# vertex value table (label,K-core,node density, node score)
vertex_table <- calc.vertex.value(x=adj_mat,loop=F)
# finding clusters
clusters <- find.cluster(vertex_table,adj_mat,node.score.cutoff=0.2)
# make table
int_tb <- as.data.frame(c('Gene2','Gene210','Gene6'),stringsAsFactors=F )
colnames(int_tb) <- 'label'
int_tb$desc <- c('Drug_A','Drug_D','Drug_E')
int_tb <- as.matrix(int_tb)
# add annotation
rst <- add.annotation(x=clusters_edges,vertices.par=graph_par_vertex,edges.par=graph_par_edge,annotation.ta
clusters_edges <- rst[[1]]
graph_par_vertex <- rst[[2]]
graph_par_edge <- rst[[3]]
```

`adjacency.matrix` *Transforming a table of connection information into an adjacency matrix*

Description

To transform a table of connections (two vertices connected by edge, i.e. PPI) consist of at least two columns into adjacency matrix

Usage

```
adjacency.matrix(input_tb, v1 = NULL, v2 = NULL, sampling = NULL,
  remove.loop = T, directed = F, multiple.edge = F)
```

Arguments

`input_tb`	A matrix contains connections between two vertices
`v1`	For directed graph, the column name of out-degree vertex in the table x (table of connections). If NULL, it takes first column.
`v2`	v1 For directed graph, the column name of in-degree vertex in the table x (table of connections). If NULL, it takes second column.
`sampling`	A vector of characters for extracting specific nodes. By default NULL, it ignores the sampling step and transform the all connections.
`remove.loop`	If FALSE, adjacency matrix will include loop edges. By default, it removes loop edges.
`directed`	If TRUE, adjacency matrix will be built from v1 (out-degree) to v2 (in-degree) otherwise, it returns a symmetric.
`x`	A list of adjacency matrix that is created by the function `cluster.adj.matrix`
`multitple.edges`	If TRUE, element of adjacency matrix will be bigger than 1 where it has multiple connections between nodes. Otherwise, default value is the result matrix that contains either 1 or 0.

Details

An adjacency matrix of an undirected graph is a symmetric matrix and it contains boolean values. A case for directed network, adjacency matrix contains the directions from vertex (i.e. A) to vertex (i.e. B) such as vertices on the row of adjacency matrix (i.e. A) to vertices of the column of adjacency matrix (i.e. B). For instance, the directional connection from A to B can be defined by 'matrix [A, B]'. However, the MCODE algorithm is an undirected graph algorithm. Therefore, the following result of network clustering in this r package is based on undirected graph.

Examples

```
#' # ideal undirected graph for MCODE
adj_mat <- adjacency.matrix(raw_table1,v1='X',v2='Y')
# directed graph with multple edges
adj_mat_directed <- adjacency.matrix(raw_table2,v1='outdegree_node',v2='indegree_node',direct=T,multiple.ed
```

calc.local.density *Calculating density value of a node*

Description

To calculate vertex density and used for function fluff

Usage

```
calc.local.density(x, vertex)
```

Arguments

x	A vector of character that is the label of vertex
vertex	An adjacency matrix

Value

A numeric vector of the vertex density

See Also

post.fluff

calc.node.scores *Calculating density, score and k-core of the single vertex*

Description

Calculate density, score and k-core of the single vertex. The function for calculating k-core of vertex is imported from package 'sna' kcores

Usage

```
calc.node.scores(x, adj.mat, minimum.edges = 2, loop = F)
```

Arguments

x	A vector of character that is the name of vertex
adj.mat	An adjacency matrix and it is created by function adjacency.matrix
minimum.edges	The minimum numbers of connected edges to a vertex (degree). By default, vertex should have more than two edges
loop	By default, loop edges would remove loops. The method of the maximum number of edges depends on loop inclusion

Details

The k-core value is calculated by the function that is imported from the package, sna. k-core measures cliquishness of a network. The principle of MCODE algorithm is to detect densely interconnected clusters. This idea goes from local density within neighborhoods. Node density is calculated by the number of edges, D = |E|/|E|max in Graph = (V,E). |E| = number of edges to a vertex. |E|max = maximum number of possible connection to a vertex. . If graph includes loops, |E|max = |V| (|V| + 1) / 2 and graph excludes loops, |E|max = |V| (|V| - 1) / 2. |V| is the number of connected vertexes to seed vertex.

Value

A list includes the density, score and k-core of vetex

References

Butts CT, Social Network Analysis with sna, 2008, Journal of Statistical Software, 24 (6), Bader GD et al. BMC Bioinformatics 2003, 4, 2. PMID 12525261

See Also

calc.vertex.value, adjacency.matrix

---

calc.vertex.value *Calculating the density, score and k-core of the all vertices in a network*

Description

Calculating the density, score and k-core of the all vertices in a network

Usage

```
calc.vertex.value(x, loop = F, minimum.edges = 2)
```

Arguments

x	An adjacency matrix
loop	By default, loop edges would remove loops. The method of the maximum number of edges depends on loop inclusion
minimum.edges	The minimum numbers of connected edges to a vertex (degree). By default, vertex should have more than two edges

Value

A table including k-core, density and scores of the all vertices in a network

Examples

```
adj_mat <- adjacency.matrix(raw_table1,v1='X',v2='Y')
#vertex value table (label,K-core,node density, node score)
vertex_table <- calc.vertex.value(x=adj_mat,loop=F)
```

cluster.adj.matrix *Extracting adjacency matrixes of subgraphs*

Description

Extracting adjacency matrixes of subgraphs

Usage

```
cluster.adj.matrix(x, adj.mat)
```

Arguments

x	List of clusters is created by the function find.cluster
adj.mat	An adjacency matrix

Value

A list consists of adjacency matrixes that are separated by individual sub clusters

See Also

find.cluster

Examples

```
adj_mat <- adjacency.matrix(raw_table1,v1='X',v2='Y')
vertex_table <- calc.vertex.value(x=adj_mat,loop=F)
# finding clusters
clusters <- find.cluster(vertex_table,adj_mat,node.score.cutoff=0.2)
# edge matrix order by cluster number
clusters_edges <- cluster.adj.matrix(x=clusters,adj_mat)
```

find.cluster *Detecting subgraphs (clusters) in a network*

Description

Detecting subgraphs (clusters) in a network. The derived score is used in clustering

Usage

```
find.cluster(vertex.score, adj.mat, node.score.cutoff = 0.2, min.k.core = 2,
  max.depth = 100)
```

Arguments

vertex.score	A table of score is created by the function calc.vertex.value
adj.mat	An adjacency matrix
node.score.cutoff	The cutoff for derived score. Otherwise, default value is 0.2
min.k.core	The minimum K-core for clustering. Default value is 2
max.depth	The maximum depth is the number of path along the longest path from seed node to destination. By default, maximum depth is 100.

Details

MCODE algorithm finds the densely connected vertices that are started from a seed vertex and use several parametric variables such as cutoff score, the maximum depth, and k-core. First, connected vertices its scores have to be bigger than a derived score, vertex score = vertex density * k-core, derived score = seed score * score cutoff. Second, destination node should be in a range of maximum depth from a seed node. Third, k-core value should be bigger than or equal to the minimum k-core value. The cutoff must be in a range between from 0 to 1. The minimum K-core value is bigger than or equal to 2 and it represents complexity.

Value

A list of clusters

Examples

```
adj_mat <- adjacency.matrix(raw_table1,v1='X',v2='Y') # import raw data
vertex_table <- calc.vertex.value(x=adj_mat,loop=F) # adjacency matrix transfoming
# finding clusters
clusters <- find.cluster(vertex_table,adj_mat,node.score.cutoff=0.2)
```

---

graph.par.edge *Making matrix of graphical parameters for edges*

Description

Making matrix of graphical parameters for edges

Usage

```
graph.par.edge(x, directed = F, multiple.edge = F)
```

Arguments

x	A list of adjacency matrixes for each cluster (subgraph)
directed	If TRUE, x (input adjacency matrix) must be directed graph. By defualt, result table generates connections on an upper triangular matrix.
multiple.edge	If TRUE, x must involve multiple edges. By defualt, duplicated connections should not be on result table.

Details

Default colour of edge is 'blue', width of edge is 1

Value

A matrix contains graphical variables for edges including vertex x (from), vertex y (to), number (cluster number), colour (edge colour), and width (edge width)

---

graph.par.vertex *Making matrix of graphical parameters for vertices*

Description

Making matrix of graphical parameters for vertices

Usage

```
graph.par.vertex(x)
```

Arguments

x	A list of clusters (subgraph) created by the function `find.cluster`

Details

The only possible shape of vertex is 'rectangle' and 'circle'. Default vertex frame colour is 'white', vertex size is 27, vertex label size is 1, vertex shape is 'circle' and vertex border colour is 'black'.

Value

A matrix contains graphical variables for vertices including x (vertex name), number (cluster number), colour (vertex frame colour), size (vertex size), font (vertex label size), shape (vertex shape), lab_col (vertex label colour), and border (vertex border colour)

---

network.format *Transforms an adjacency matrix of clusters into the format of network for igraph*

Description

Transforms an adjacency matrix of clusters into the format of network for igraph

Usage

```
network.format(x, vertices.par = NULL, edges.par = NULL, direct = F)
```

Arguments

x	An adjacency matrix
vertices.par	A matrix of graphical parameters for vertices that is created by the function graph.par.vertex
edges.par	A matrix of graphical parameters for edges that is created by the function graph.par.edge
directed	If TRUE, x must be a directed adjacency matrix. By default, it returns an undirected graph.

Value

A list of network format for igraph

References

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006.

Examples

```
adj_mat <- adjacency.matrix(input_tb,v1='X',v2='Y',remove.loop=T,directed=F,multiple.edge=T)
vertex_table <- calc.vertex.value(x=adj_mat,loop=F) # tansforming to matrix
# finding clusters
clusters <- find.cluster(vertex_table,adj_mat,node.score.cutoff=0.2)
# subgraphs adjancecy matrixes
clusters_edges <- cluster.adj.matrix(x=clusters,adj_mat)
#prepare for essential plotting input
graph_par_vertex <- graph.par.vertex(x=clusters)
graph_par_edge <- graph.par.edge(x=clusters_edges) #option here for multiple edge, direction
# adjacency matrix to graph format, includes graphical parameters
IG <- network.format(x=clusters_edges,vertices.par=graph_par_vertex,edges.par=graph_par_edge,direct=F)
```

---

plot.network *Plotting function of networks*

Description

Plotting function of networks

Usage

```
## S3 method for class 'network'
plot(input, network.layout = "layout.fruchterman.reingold",
  arrow.size = 0.5)
```

Arguments

input	A list of network format
network.layout	The drawing algorithm for plotting. Default drawing algorithm is 'Frucherman Reingold'. It is also possible to apply other drawing algorithms, which are supported on R package, 'igraph'. Other layouts are in layout.reingold.tilford
arrow.size	size of arrow. Default size is 0.5

Examples

```
### draw an whole network (directed), Example 1
adj_mat_directed <- adjacency.matrix(raw_table2,v1='outdegree_node',v2='indegree_node',direct=T,multiple.ed
# vertex value table (label,K-core,node density, node score)
clusters <- list()
clusters[[1]] <- rownames(adj_mat_directed)
clusters_edges <- list()
clusters_edges[[1]] <- adj_mat_directed
##prepare for essential plotting input
graph_par_vertex <- graph.par.vertex(x=clusters)
graph_par_edge <- graph.par.edge(x=clusters_edges)
# transform to network
IG <- network.format(x=clusters_edges,vertices.par=graph_par_vertex,edges.par=graph_par_edge,direct=T)
# plotting
plot(plot.network(IG[1])


### draw clusters with multiple edges, example 2
adj_mat_directed <- adjacency.matrix(raw_table2,v1='outdegree_node',v2='indegree_node',direct=F,multiple.ed
vertex_table <- calc.vertex.value(x=adj_mat_directed,loop=F)

# finding clusters
clusters <- find.cluster(vertex_table,adj_mat_directed,node.score.cutoff=0.2)

# Fluff
clusters <- post.fluff(x=clusters,adj_mat_directed,vertex_table)
clusters <- post.fluff.bind(x=clusters)

# haircut
clusters <- post.haircut(clusters,adj_mat_directed)

# edge matrix order by cluster number
clusters_edges <- cluster.adj.matrix(x=clusters,adj_mat_directed)

#prepare for essential plotting input
graph_par_vertex <- graph.par.vertex(x=clusters)
graph_par_edge <- graph.par.edge(x=clusters_edges)

# network format
IG <- network.format(x=clusters_edges,vertices.par=graph_par_vertex,edges.par=graph_par_edge,direct=T)

# plotting
for(i in 1:length(IG)) { plot.network(IG[i]) }
```

`post.fluff` *Performing post processing, Fluff*

Description

Performing post processing, Fluff, after detecting cluster complex

Usage

```
post.fluff(x, adj.mat, vertex.score.table, density.cutoff = 0.6)
```

Arguments

x	A list of clusters
adj.mat	An adjacency matrix
vertex.score.table	A matrix of the score of the vertices that is created by the function calc.vertex.value
density.cutoff	A cutoff value of the node density

Details

The fluff step is the post process after clustering. In this step, cluster expands to neighbourhoods. It is decided by derived density, derived density = seed density * cutoff. Density of candidate vertices has to be bigger than the derived density of the seed node. If a neighbour vertex has satisfied enough density, it belongs to the expanded cluster. If procedure contains either fluff and haircut, fluff has to be performed before haircut.

Value

A list of clusters

Examples

```
# finding clusters
clusters <- find.cluster(vertex_table,adj_mat,node.score.cutoff=0.2)
# Fluff
clusters <- post.fluff(x=clusters,adj_mat,vertex_table)
clusters <- post.fluff.bind(x=clusters)
```

---

post.fluff.bind *Combining overlapping clusters (subgraphs)*

Description

Combining overlapping clusters (subgraphs), after it performed post-process step post.fluff

Usage

```
post.fluff.bind(x)
```

Arguments

x	A list of clusters that is generated after the post processing function post.fluff

Value

A list of clusters

See Also

post.fluff

post.haircut *Performing post processing, Haircut*

Description

Performing post processing, Haircut, after detecting cluster complex

Usage

```
post.haircut(x, adj.mat)
```

Arguments

x	A list of clusters (subgraph) that is created by the function find.cluster
adj.mat	An adjacency matrix

Details

The haircut removes singly connected vertices within a cluster. Post process have to performed after clustering. This haircut process comes after fluff when fluff is needed.

Value

A list of clusters

Examples

```
# finding clusters
clusters <- find.cluster(vertex_table,adj_mat,node.score.cutoff=0.2)
# Haircut, post process
clusters <- post.haircut(clusters,adj_mat)
```

Index