Package 'linkcomm'

October 6, 2020

Type Package

Title Tools for Generating, Visualizing, and Analysing Link Communities in Networks

Version 1.0-13

Date 2020-10-03

Author Alex T. Kalinka <alex.t.kalinka@gmail.com>, with contributions from Alain Guenoche

Maintainer Alex T. Kalinka <alex.t.kalinka@gmail.com>

Description Link communities reveal the nested and overlapping structure in networks, and uncover the key nodes that form connections to multiple communities. linkcomm provides a set of tools for generating, visualizing, and analysing link communities in networks of arbitrary size and type. The linkcomm package also includes tools for generating, visualizing, and analysing Overlapping Cluster Generator (OCG) communities. Kalinka and Tomancak (2011) <10.1093/bioinformatics/btr311>.

License GPL (>= 2)

URL https://alextkalinka.github.io/linkcomm/, https://github.com/alextkalinka/linkcomm

Depends igraph, RColorBrewer

Imports dynamicTreeCut, grid, utils

Suggests fastcluster, R.rsp

VignetteBuilder R.rsp

Encoding UTF-8

LazyData yes

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-10-06 05:10:06 UTC

R topics documented:

linkcomm-package *The* linkcomm *package*

Description

linkcomm provides tools for the generation, visualization, and analysis of link communities in networks of arbitrary size and type.

Details

Link communities reveal the nested and overlapping structure in networks, and uncover the key nodes that form connections to multiple communities. linkcomm provides tools for generating, visualizing, and analysing link communities in networks of arbitrary size and type.

For a more detailed overview of how to use the package:

```
vignette(topic = "linkcomm",package = "linkcomm")
```

To run an interactive demonstration of linkcomm within R:

```
demo(topic = "linkcomm",package = "linkcomm")
```

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Ahn, Y.Y., Bagrow, J.P., and Lehmann, S. (2010). Link communities reveal multiscale complexity in networks. *Nature* **466**, 761-764.

Becker, E., Robisson, B., Chapple, C.E., Guenoche, A. and Brun, C. (2012) Multifunctional proteins revealed by overlapping clustering in protein interaction network. *Bioinformatics* **28**, 84-90.

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Spencer, R. (2010). http://scaledinnovation.com/analytics/communities/comlinks.html

See Also

getLinkCommunities, getOCG.clusters, plot.linkcomm, pp_rnapol, lesmiserables, karate, weighted, igraph, RColorBrewer, grid

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)
```

```
## Plot a graph layout of the link communities.
plot(lc, type = "graph")

## Use a Spencer circle layout.
plot(lc, type = "graph", layout = "spencer.circle")

## Calculate a community-based measure of node centrality.
getCommunityCentrality(lc)

## Find nested communities.
getAllNestedComm(lc)

## Uncover the relatedness between communities.
getClusterRelatedness(lc)
```

`corLinkcommCentrality` *Correlation of Community Centrality with Classic Centrality*

Description

This function calculates the correlation between the community centrality and classic centrality measures for a set of nodes in a network, and plots a scatterplot of the relationship together with a fitted straight line.

Usage

```
corLinkcommCentrality(x, centrality = "degree", type = "commweight",
                      method = "spearman", plot = TRUE, pch = 20, ...)
```

Arguments

`x`	An object of class `linkcomm`.
`centrality`	A character string naming the classic centrality measure. Can be one of `"degree"`, `"betweenness"`, `"closeness"`, and `"constraint"`. Defaults to `"degree"`.
`type`	A character string naming the type of community centrality. Can be `"commweight"` or `"commconn"`, defaults to `"commweight"`.
`method`	A character string naming the correlation method. Can be one of `"spearman"`, `"pearson"`, or `"kendall"`. Defaults to `"spearman"`.
`plot`	Logical, whether to plot a scatterplot of the relationship, defaults to TRUE.
`pch`	An integer specifying the plot symbol (see `par`). Defaults to 20.
`...`	Additional arguments to be passed to `plot`.

Details

The correlation between community centrality and classic centrality measures, such as degree or betweenness, may reveal discrepancies, thereby indicating that community centrality scores provide a unique reflection of node importance.

Value

A correlation coefficient.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

getCommunityCentrality

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Correlate community centrality with degree centrality.
corLinkcommCentrality(lc)
```

---

cutDendrogramAt *Extract Meta-Communities*

Description

This function extracts meta-communities from a dendrogram of community relatedness based on a user-defined place at which to cut the dendrogram.

Usage

```
cutDendrogramAt(x, lc = NULL, cutat = NULL, plot = TRUE, col = TRUE,
                pal = brewer.pal(9, "Set1"), labels = FALSE, plotcut = TRUE,
                right = TRUE, verbose = TRUE, ...)
```

Arguments

x	An object of class hclust, usually generated by getClusterRelatedness.
lc	An object of class linkcomm. If included, the resulting plot will display additional information about the clusters. Defaults to NULL.
cutat	A numerical value at which to cut the dendrogram.

plot	Logical, whether to plot the dendrogram and the meta-communities, defaults to TRUE.
col	Logical, whether to colour the meta-communites.
pal	A character vector describing a colour palette to be used for colouring the meta-communites in the dendrogram plot. Defaults to `brewer.pal(9,"Set1")`.
labels	Logical, whether to put labels on the dendrogram. Defaults to FALSE.
plotcut	Logical, whether to display a horizontal line where the dendrogram is cut. Defaults to TRUE.
right	Logical, whether to orient the dendrogram to the right. Defaults to TRUE.
verbose	Logical, whether to display the progress of colouring the dendrogram. Defaults to TRUE.
...	Additional arguments to be passed to `plot`.

Details

Extracting meta-communities allows the user to explore community relatedness and structure at higher levels.

Value

A list of integer vectors, referring to meta-communities of link communities.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

`getClusterRelatedness`

Examples

```
## Generate graph, extract link communities, and cluster communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)
hc <- getClusterRelatedness(lc)

## Cut dendrogram at 1 and extract meta-communities.
cutDendrogramAt(hc, cutat = 1)
```

edge.duplicates *Find and Remove Edge Loops and Duplicates*

Description

This function finds and removes loops, edge duplicates, and bi-directional edges.

Usage

```
edge.duplicates(network, verbose = TRUE)
```

Arguments

network	An edge list, which is either a data frame or a character or integer matrix with two columns.
verbose	Logical, whether to display the progress of the function on the screen. Defaults to TRUE.

Details

This function removes loops, duplicate and bi-directional edges; the edges that occur closer to the end of the edge list will be removed.

Value

A list with the following elements: edges - a character matrix of the edges in the network with any loops or duplicate edges removed; inds - an integer vector of the edge indices of any loop or duplicate edges in the original network.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Examples

```
## Generate graph and remove loops and duplicate edges.
g <- swiss[,3:4]
g.dup <- edge.duplicates(g)
```

get.community.overlaps *Find Node Overlaps between Communities*

Description

This function returns lists of communities that share at least one node with each other.

Usage

```
get.community.overlaps(x)
```

Arguments

x	An object of class `linkcomm` or `OCG`.

Value

A list of length equal to the number of communities. Each element contains an integer vector giving the community IDs for communities sharing at least one node with each community. NAs indicate that a community shares no nodes with any other communities.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Get list of overlapping communities.
ov <- get.community.overlaps(lc)
```

`get.shared.nodes` *Get Nodes Shared by Communities*

Description

This function returns the nodes that are shared by specific sets of communities.

Usage

```
get.shared.nodes(x, comms)
```

Arguments

`x`	An object of class `linkcomm` or `OCG`.
`comms`	An integer vector giving the community IDs for which an intersecting set of shared nodes should be returned.

Value

A character vector giving the shared node names.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Get shared nodes for communities 1 and 7.
get.shared.nodes(lc, comms = c(1,7))
```

`getAllNestedComm` *Find Nested Communities*

Description

This function returns communities of nodes that are entirely nested within other larger communities of nodes.

Usage

```
getAllNestedComm(x, verbose = FALSE, plot = FALSE)
```

Arguments

`x`	An object of class `linkcomm`.
`verbose`	Logical, whether to print to the screen a warning that individual community IDs are not clustered in any other communities. Defaults to FALSE.
`plot`	Logical, whether to plot graphs of the nested communities. Defaults to FALSE.

Details

Nested community structures may reveal interesting relationships among sets of nodes.

Value

A named list of integer vectors; names are integers referring to nested communities, and the integer vectors are the communities that the named community is nested in.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

`getNestedHierarchies`

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Find nested communities.
getAllNestedComm(lc)
```

`getClusterRelatedness` *Hierarchichal Clustering of Link Communities*

Description

This function hierarchically clusters the link communities themselves and returns an object of class `hclust`.

Usage

```
getClusterRelatedness(x, clusterids = 1:x$numbers[3], hcmethod = "ward.D",
        cluster = TRUE, plot = TRUE, cutat = NULL, col = TRUE,
        pal = brewer.pal(11, "Spectral"), labels = FALSE, plotcut = TRUE,
        right = TRUE, verbose = TRUE, ...)
```

Arguments

`x`	An object of class `linkcomm`.
`clusterids`	An integer vector of community IDs. Defaults to all communities.
`hcmethod`	A character string naming the hierarchical clustering method to use. Can be one of `"ward.D"`, `"ward.D2"`, `"single"`, `"complete"`, `"average"`, `"mcquitty"`, `"median"`, or `"centroid"`. Defaults to `"ward.D"`.
`cluster`	Logical, whether to cluster the communities. If FALSE, the function returns the upper triangular dissimilarity matrix as a vector. Defaults to TRUE.
`plot`	Logical, whether to plot the cluster dendrogram.
`cutat`	A numerical value at which to cut the dendrogram. If NULL, the dendrogram is not cut and meta-communities are not returned. Defaults to NULL.
`col`	Logical, whether to colour the dendrogram. Defaults to TRUE.
`pal`	A character vector describing a colour palette to be used for colouring the meta-communites in the dendrogram plot. Defaults to `brewer.pal(11,"Spectral")`.
`labels`	Logical, whether to add labels to the dendrogram plot.
`plotcut`	Logical, whether to display a horizontal line where the dendrogram is cut. Defaults to TRUE.
`right`	Logical, whether to orient the dendrogram to the right. Defaults to TRUE.
`verbose`	Logical, whether to display the progress of the calculation on the screen. Defaults to TRUE.
`...`	Additional arguments to be passed to `plot`.

Details

Extracting meta-communities allows the user to explore community relatedness and structure at higher levels. Community relatedness is calculated using the Jaccard coefficient and the number of nodes that community i and j share:

$$S(i,j) = \frac{|n_i \cap n_j|}{|n_i \cup n_j|}$$

Value

Either a numerical vector (the upper triangular dissimilarity matrix - if `cluster` = `FALSE`), a list of integer vectors (the meta-communities - if `cutat` is not NULL), or an object of class `hclust` (if `cluster` is TRUE and `cutat` is NULL).

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

`meta.communities`, `cutDendrogramAt`, `hclust`

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Cluster the link communities.
getClusterRelatedness(lc)

## Cluster the link communities, cut the dendrogram, and return the meta-communities.
getClusterRelatedness(lc, cutat = 1)
```

`getCommunityCentrality` *Calculate Community Centrality Measures for Nodes*

Description

This function returns community-based node centrality measures.

Usage

```
getCommunityCentrality(x, nodes = names(x$numclusters), type = "commweight",
                       normalise = TRUE)
```

Arguments

x	An object of class linkcomm.
nodes	A character vector giving the names of nodes for calculating community centrality scores. Defaults to all nodes.
type	A character string naming the community centrality measure. Can be one of "commweight" or "commconn" (see Details below). Defaults to "commweight".
normalise	Logical, whether to normalise community connectedness for "commconn". Defaults to TRUE. Will be ignored for "commweight".

Details

Community-based measures of node centrality provide an alternative to classic measures of node centrality. "commweight" weights each community that a node belongs to by how similar that community is to each of the other communities to which the node also belongs. For node i the community centrality is

$$C_C(i) = \sum_{i \in j}^{N} \left(1 - \frac{1}{m} \sum_{i \in j \cap k}^{m} S(j,k) \right)$$

where the main sum is over the N communities to which node i belongs, and $S(j,k)$ refers to the similarity between community j and k, calculated as the Jaccard coefficient for the number of shared nodes between each community pair, and this is averaged over the m communities paired with community j and in which node i jointly belongs. "commconn" weights each community that a node belongs to by how many connections the community forms outside of itself relative to how many connections the community has within itself (the inverse of modularity), so that nodes that belong to more highly connecting communitites will receive a higher community centrality score. For node i the community centrality is

$$C_C(i) = \sum_{i \in j}^{N} e_{ij} \frac{\check{e}_{B(j)}}{\check{e}_{W(j)}}$$

where e_{ij} is the number of edges node i has in community j, $\check{e}_{B(j)} = \frac{e_{B(j)}}{n_j \bar{d}}$ is the number of edges community j makes outside of itself normalised by the number of nodes in community j multiplied by the average degree in the network, and $\check{e}_{W(j)} = \frac{e_{W(j)}}{n(n-1)/2}$ is the number of edges within community j normalised by the total number possible.

Value

A named numerical vector where the names are node names and the numbers are community centrality measures.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

getCommunityConnectedness

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Calculate community centrality.
cc <- getCommunityCentrality(lc)
```

---

getCommunityConnectedness

Calculate Community Connectedness or Modularity

Description

This function returns a measure of how relatively outwardly or inwardly connected a community is.

Usage

```
getCommunityConnectedness(x, clusterids = 1:x$numbers[3], conn = "conn",
                          normalise = TRUE, verbose = FALSE)
```

Arguments

x	An object of class linkcomm.
clusterids	An integer vector of community IDs. Defaults to all communities.
conn	A character string naming the connectedness measure to use. Can be one of "conn" or "mod" (see Details below). Defaults to "conn".
normalise	Logical, whether to normalise community connectedness measures by the number of nodes in individual communities. Defaults to TRUE.
verbose	Logical, whether to display the progress of the calculation on the screen. Defaults to FALSE.

Details

The connectedness and modularity of different communities indicates whether a particular community is bridging several other communities, or existing as a relatively isolated module. The modularity of community i is

$$M_i = \left(\frac{e_w(i)}{n_i(n_i - 1)/2} \right) \cdot \left(\frac{e_b(i)}{n_i \hat{d}} \right)^{-1}$$

where $e_w(i)$ is the number of edges within community i, $e_b(i)$ is the number of edges community i makes to other communities, n_i is the number of nodes in community i, and $\hat{d}$ is the average degree in the network. Community connectedness is the inverse of this value.

Value

A named numerical vector, where the names are community IDs and the numbers are community connectedness or modularity scores.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

getCommunityCentrality

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Get community connectedness.
cc <- getCommunityConnectedness(lc, normalise = FALSE)
```

---

getCommunityMatrix *Construct a Community Membership Matrix*

Description

This function returns a binary matrix with nodes as rows, communities as columns, and unit entries indicating membership in a community.

Usage

```
getCommunityMatrix(x, nodes = head(names(x$numclusters), 20))
```

Arguments

x	An object of class linkcomm.
nodes	A character vector containing the nodes for the community membership matrix. Defaults to the 20 (or less) nodes that belong to the most communities.

Value

A binary matrix with nodes as rows and communities as columns.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plot.linkcomm

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Get community membership matrix.
getCommunityMatrix(lc)
```

---

getEdgesIn *Extract Edge Indices from Communities*

Description

This function returns edge indices that belong to certain communities or that are incident upon certain nodes.

Usage

```
getEdgesIn(x, clusterids = 1, nodes = NULL, all = FALSE)
```

Arguments

x	An object of class linkcomm or OCG.
clusterids	An integer vector of community IDs. Defaults to community 1.
nodes	A character vector specifying node(s) for which edge indices should be returned. Overrides clusterids. Defaults to NULL.
all	Logical, whether the edges for all communities to which the named nodes belong should be returned. Will have an effect only if nodes is not NULL. If FALSE, edges that are directly incident upon the named nodes will be returned. Defaults to FALSE.

Value

An integer vector of edge indices.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Get edges from community 1.
getEdgesIn(lc)
```

---

getLinkCommunities *Extract Link Communities from a Network*

Description

This function extracts link communities from networks of arbitrary size and type.

Usage

```
getLinkCommunities(network, hcmethod = "average", use.all.edges = FALSE,
                   edglim = 10^4, directed = FALSE, dirweight = 0.5,
                   bipartite = FALSE, dist = NULL, plot = TRUE,
                   check.duplicates = TRUE, removetrivial = TRUE,
                   verbose = TRUE)
```

Arguments

network	An edge list, which is a matrix or data frame with 2 or 3 columns. The first 2 columns contain the nodes that interact with each other, which can be character strings or integer values. The optional third column is a numerical vector of weights for each edge. Can also be a character string naming a file containing an edge list.
hcmethod	A character string naming the hierarchical clustering method to use. Can be one of "ward", "single", "complete", "average", "mcquitty", "median", or "centroid". Defaults to "average" (if the number of edges is greater than edglim then "single" is used).
use.all.edges	Logical, indicating whether edge similarities should be calculated for all pairs of edges (TRUE), or only for edge pairs that share a node (FALSE) as in the original Ahn et al. (2010) algorithm. Defaults to FALSE. If TRUE, networks are treated as undirected.
edglim	An integer value indicating the largest number of edges permissible for the hierarchical clustering to be handled in memory. Above this value the upper triangular dissimilarity matrix will be written to disk and read and written as clustering proceeds until the file size is 0 bytes (see Details below). Defaults to 10^4.
directed	Logical, whether the network is directed. Defaults to FALSE.
dirweight	A numerical value between 1 and 0 inclusive indicating the weight that will be attached to edges that share a node but are in the opposite orientation. Defaults to 0.5. Will be ignored if directed = FALSE.
bipartite	Logical, whether the input network is bi-partite. See Details for an explanation of how bi-partite networks are handled. Defaults to FALSE.
dist	An object of class "dist" representing a user-defined distance matrix for the network. Note, this must be the lower triangular matrix of an n*n distance matrix, where n is the number of edges in the network (make sure duplicated edges are removed). If NULL, then the distance matrix is calculated by the algorithm. Defaults to NULL.
plot	Logical, whether to plot summary output from the algorithm (dendrogram and partition density plot). Defaults to TRUE. Note, if there are more than 1500 but less than edglim edges then the dendrogram will be plotted without colour and in a separate panel from the partition density to avoid lengthy rendering times; when there are more than edglim edges then only the partition density will be plotted.
check.duplicates	Logical, whether to check for and remove loops, duplicate edges, and bi-directional edges. Defaults to TRUE. Note, if you wish to avoid this step by setting this parameter to FALSE then you must be certain that there are no duplicate edges in the network.
removetrivial	Logical, whether to remove trivial community clusters that contain 2 edges. Defaults to TRUE.
verbose	Logical, whether to display the progress of the algorithm on the screen. Defaults to TRUE.

Details

This is the main algorithm used for extracting link communities from networks of arbitrary size and type. Input networks may be directed, weighted, both directed and weighted, or neither. The algorithm used is the one outlined by Ahn et al. (2010). The similarity between links, e_{ik} and e_{jk}, that share a node, k, is calculated using the Jaccard coefficient

$$S(e_{ik}, e_{jk}) = \frac{|n_+(i) \cap n_+(j)|}{|n_+(i) \cup n_+(j)|}$$

where $n_+(i)$ refers to the first-order node neighbourhood of node i, which includes node i itself (inclusive neighbour set). After assigning pairwise similarities to all of the links in the network, the links are hierarchically clustered using single-linkage clustering, and the resulting dendrogram is cut at a point that maximises the density of links within the clusters normalising against the maximum and minimum numbers of links possible in each cluster, known as the partition density. For directed and weighted networks, the Tanimoto coefficient is used for assigning similarity between links

$$S(e_{ik}, e_{jk}) = \frac{\mathbf{a}_i.\mathbf{a}_j}{|\mathbf{a}_i|^2 + |\mathbf{a}_j|^2 - \mathbf{a}_i.\mathbf{a}_j}$$

where $\mathbf{a}_i$ refers to a vector describing the weights of links between node i and the nodes in the first-order neighbourhoods of both nodes i and j (equal to 0 in the event of an absent link). For directed networks, links to nodes shared by both node i and j are given a user-defined weight below 1 if they are in the opposite orientation.

For bi-partite networks, the set of neighbours (instead of the inclusive neighbour set) is used to count nodes for the edge similarity metric because node i and node j cannot share an edge in a bi-partite network. The partition density for bi-partite networks is calculated as:

$$D_c = \frac{2}{M} \sum_c m_c \frac{m_c + 1 - n_c}{2n_{c0}n_{c1} - 2(n_c - 1)}$$

where M is the total number of edges, m_c is the number of edges in subset c, n_c is the number of nodes in subset c, n_{c0} is the number of nodes in partition 0, and n_{c1} is the number of nodes in partition 1.

Value

An object of class `linkcomm`, which is a list containing the following components:

`numbers`	An integer vector with the number of edges, nodes, and communities.
`hclust`	An object of class `hclust`, which contains information about the hierarchical clustering of links.
`pdmax`	A numerical value indicating the height of the dendrogram at which the partition density is maximised.
`pdens`	A numerical matrix with 2 columns; the first is the heights at which clusters appear and the second is the partition density.
`nodeclusters`	A data frame consisting of 2 columns; the first contains node names, and the second contains single community IDs for each node. All communities and their nodes are represented, but not necessarily all nodes.

clusters	A list of integer vectors containing the link IDs that belong to each community. Community IDs are the numerical position of the communities in the list.
edges	A data frame with 3 columns; the first two contain nodes that interact with each other, and the third is an integer vector of community IDs indicating community membership for each link.
numclusters	A named integer vector. Names are node names and integer values are the number of communities to which each node belongs.
clustsizes	A named integer vector. Names are community IDs and integer values indicate the number of nodes that belong in each community.
igraph	An object of class igraph. The network is represented here as an igraph object.
edgelist	A character matrix with 2 columns containing the nodes that interact with each other.
directed	Logical indicating whether the network is directed.
bipartite	Logical indicating whether the network is bi-partite.

Note

When the number of links is less than edglim the hierarchical clustering will be handled in memory. Above this value the upper triangular dissimilarity matrix will be compressed and written to disk and read and written as clustering proceeds until the file size is 0 bytes using a compiled C++ function. In this case the hierarchical clustering method will always be "single" to enhance performance for large networks. The size of edglim can be modified to suit the computer resources available to the user. As a guide, a network with 10^4 links will require $((10^4)^2) * 8 = 800$ MB to be handled in an uncompressed format in the memory.

For directed networks, a pair of bidirectional interactions between two nodes cannot be assigned similarities and the edge that appears lower in the edge list for the network will be discarded.

When use.all.edges is TRUE, the algorithm may be slow as all pairs of edges will be compared (n^2 comparisons, where n is the number of edges).

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Ahn, Y.Y., Bagrow, J.P., and Lehmann, S. (2010). Link communities reveal multiscale complexity in networks. *Nature* **466**, 761-764.

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plot.linkcomm, newLinkCommsAt, meta.communities

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Extract communities by writing a temporary file to disk.
lc <- getLinkCommunities(g, edglim = 10)

## Use similarities between all pairs of edges.
lc <- getLinkCommunities(g, use.all.edges = TRUE)

## Directed network.
lc <- getLinkCommunities(g, directed = TRUE, dirweight = 0.8)

## Weighted network.
g <- cbind(swiss[,3:4], runif(nrow(swiss[,3:4])))
lc <- getLinkCommunities(g)

## Directed and weighted network.
lc <- getLinkCommunities(g, directed = TRUE, dirweight = 0.8)
```

---

getNestedHierarchies *Find Nested Structures in Communities*

Description

This function determines whether a particular community is nested within any other communities.

Usage

```
getNestedHierarchies(x, clusid = 1, verbose = TRUE, plot = TRUE, ids = FALSE)
```

Arguments

x	An object of class linkcomm.
clusid	An integer value indicating the community ID whose nesting structure will be tested. Defaults to 1.
verbose	Logical, whether to display a warning that a particular community is not nested in any other communities on the screen. Defaults to FALSE.
plot	Logical, whether to plot a graph layout of the nested community.
ids	Logical, whether to return only the community IDs that the community is nested in, or the node names also. Defaults to FALSE.

Value

Either a list of character vectors, each giving the nodes that the community is nested in, or an integer vector of community IDs that the community is nested in.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

getAllNestedComm

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Determine if community 1 is nested in any other communities.
getNestedHierarchies(lc, clusid = 1)
```

---

getNodesIn *Extract Nodes from Communities*

Description

This function returns node names that belong to sets of communities.

Usage

```
getNodesIn(x, clusterids = 1, type = "names")
```

Arguments

x	An object of class linkcomm or OCG.
clusterids	An integer vector of community IDs. Defaults to community 1.
type	A character string specifying how nodes are returned. Can be one of "names" or "indices".

Value

A character vector of node names (if type is "names") or a numerical vector of node indices (if type is "indices").

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Get nodes from community 1.
getNodesIn(lc)
```

`getOCG.clusters` *Generate Overlapping Cluster Generator (OCG) Communities*

Description

This function generates communities based on the OCG algorithm.

Usage

```
getOCG.clusters(network, init.class.sys = 3, max.class.card = 0,
             cent.class.sys = 1, min.class = 2, verbose = TRUE, keep.out = FALSE)
```

Arguments

`network`	Either a character string naming the file containing the network as an edge list, or a data frame/matrix object containing the edge list.
`init.class.sys`	An integer number specifying the Initial Class System: 1 - Maximal Cliques, 2 - Edges, or 3 - Centered Cliques. Defaults to 3.
`max.class.card`	An integer number specifying the maximum allowed class cardinality. Defaults to 0, which indicates no constraint.
`cent.class.sys`	A binary value indicating the choice of class system for centered cliques: 0 - Final class system, needs the expected minimum number of clusters and the maximum caldinality of the final clusters, or 1 - the class system that maximizes modularity. Defaults to 1.
`min.class`	An integer number specifying the minimum number of expected classes. Defaults to 2.
`verbose`	Logical, whether to display progress of the algorithm to the screen. Defaults to TRUE.
`keep.out`	Logical, whether to keep the OCG partition intermediate file on disk or not. Defaults to FALSE.

Value

An object of class OCG, which is a list containing the following elements:

numbers	An integer vector with the number of edges, nodes, and communities.
modularity	An integer number specifying the modularity of the network.
Q	A real number specifying the value of Q generated by the OCG algorithm.
nodeclusters	A data frame consisting of 2 columns; the first contains node names, and the second contains single community IDs for each node. All communities and their nodes are represented, but not necessarily all nodes.
numclusters	A named integer vector. Names are node names and integer values are the number of communities to which each node belongs.
igraph	An object of class igraph. The network is represented here as an igraph object.
edgelist	A character matrix with 2 columns containing the nodes that interact with each other.
clustsizes	A named integer vector. Names are community IDs and integer values indicate the number of nodes that belong in each community.

Note

For optimal results, the input network must contain at least one connected component (a subgraph in which any two vertices are connected by a path, which is not connected to additional vertices in the supergraph).

Author(s)

Alain Guenoche (main algorithm), and ported into R by Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Becker, E., Robisson, B., Chapple, C.E., Guenoche, A. and Brun, C. (2012) Multifunctional proteins revealed by overlapping clustering in protein interaction network. *Bioinformatics* **28**, 84-90.

Examples

```
## Generate graph and extract OCG communities.
g <- swiss[,3:4]
oc <- getOCG.clusters(g)
```

graph.feature *Make Node or Edge Graph Features*

Description

This function returns vectors of node sizes or edge widths for use in `plot.linkcomm`.

Usage

```
graph.feature(x, type = "nodes", clusterids = 1:length(x$clusters),
              nodes = NULL, indices, features, default = 15, showall = FALSE)
```

Arguments

`x`	An object of class `linkcomm`.
`type`	A character string specifying either `"nodes"` or `"edges"`.
`clusterids`	An integer vector of community IDs that will be plotted. Defaults to all communities.
`nodes`	A character vector specifying node(s) that will be plotted. Overrides `clusterids`. Defaults to NULL.
`indices`	An integer vector specifying the indices of the nodes or edges that will be given specific size or width values. See `getNodesIn` and `getEdgesIn` for ways to generate these indices. Also see examples in `vignette(topic = "linkcomm",package = "linkcomm")`.
`features`	An integer vector specifying the node or edge sizes for the nodes or edges that are to be changed. If there is a single value then this will be applied to all nodes or edges specified in `indices`, otherwise the `features` vector must be the same length as the `indices` vector and the values will be matched to each other.
`default`	An integer value specifying the node size or edge width that all nodes or edges not specified by `indices` will take. Defaults to 15.
`showall`	Logical, whether edges that don't belong to communities will also be plotted or not. Defaults to FALSE.

Value

A named integer vector of node sizes or edge widths. The names will be either node names or edge indices.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plotLinkCommGraph, getNodesIn, getEdgesIn, vignette(topic = "linkcomm",package = "linkcomm").

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Make node size vector for all nodes that belong to community 1.
graph.feature(lc, indices = getNodesIn(lc, type = "indices"), features = 20, default = 5)

## Make edge width vector for all edges that belong to community 1.
graph.feature(lc, type = "edges", indices = getEdgesIn(lc), features = 5, default = 1)
```

---

human_pp *Sample Human Protein Interactome*

Description

A set of 864 human proteins involved in 999 interactions.

Usage

```
human_pp
```

Format

Data frame with 2 columns.

References

Becker, E., Robisson, B., Chapple, C.E., Guenoche, A. and Brun, C. (2012) Multifunctional proteins revealed by overlapping clustering in protein interaction network. *Bioinformatics* **28**, 84-90.

---

integer.edgelist *Convert A Network to an Integer Edgelist*

Description

This function converts a character string edgelist into an integer edgelist.

Usage

```
integer.edgelist(network)
```

Arguments

network	An edge list, which is a matrix or data frame with 2 or 3 columns. The first 2 columns contain the nodes that interact with each other, which can be character strings or integer values. The optional third column is a numerical vector of weights for each edge, which is stripped from the output.

Value

A list containing the following components:

edges	A matrix with two columns containing the integer edgelist.
nodes	A named integer vector mapping node integer IDs to their character string equivalents.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

See Also

getLinkCommunities

Examples

```
## Generate graph and convert into an integer edgelist.
g <- data.frame(letters[1:10],letters[6:15])
gi <- integer.edgelist(g)
```

---

karate *Social Network in a Karate Club*

Description

A social network of friendships between 34 members of a karate club at a US university in the 1970s (Zachary 1977).

Usage

```
karate
```

Format

Data frame with 2 columns.

Source

https://github.com/gephi/gephi/wiki/Datasets

References

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* **33**, 452-473.

`layout.spencer.circle` *Calculate Node Coordinates for a Spencer Circle*

Description

This function returns the x-y coordinates for nodes in a Spencer circle together with community anchor positions.

Usage

```
layout.spencer.circle(x, clusterids = 1:x$numbers[3], verbose = TRUE,
                      jitter = 0.2)
```

Arguments

`x`	An object of class `linkcomm`.
`clusterids`	An integer vector of community IDs. Defaults to all communities.
`verbose`	Logical, whether to print the progress of the calculation to the screen. Defaults to TRUE.
`jitter`	A positive numerical value specifying the range (negative to positive) of random, uniformly distributed noise that will be added to nodes that have identical x-y coordinates. Defaults to 0.2.

Details

This algorithm anchors communities evenly around the circumference of a circle in their dendrogram order (to minimise crossing over of links) and positions nodes within the circle according to how many links they possess in each of the communities (Spencer, 2010). Thus, nodes that have links to a lot of communities will get pushed into the centre of the circle making this method well suited for representing ego networks where one or a small number of nodes belong to multiple communities.

Value

A list with the following components:

`nodes`	A numerical matrix with nodes as rows and with 2 columns; the first contains the x coordinates and the second the y coordinates.
`anchors`	A numerical matrix with communities as rows and with 2 columns of x and y coordinates.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Spencer, R. (2010). http://scaledinnovation.com/analytics/communities/comlinks.html

See Also

plot.linkcomm, plotLinkCommGraph

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Extract x-y coordinates for nodes in a Spencer circle.
layout.spencer.circle(lc)
```

---

lesmiserables *Co-Appearance Network from Les Miserables*

Description

The co-appearance network for Les Miserables (Knuth 1993). Involves 252 interactions among 77 nodes.

Usage

```
lesmiserables
```

Format

Data frame with 2 columns.

Source

https://github.com/gephi/gephi/wiki/Datasets

References

Knuth, D. E. (1993). *The Stanford GraphBase: A Platform for Combinatorial Computing*, Addison-Wesley, Reading, MA.

`linkcomm2clustnsee` *Write a Partition File for Clust&See*

Description

This function writes out a partition file which can be imported into the Cytoscape plug-in Clust&See.

Usage

```
linkcomm2clustnsee(x, file = "temp.cns", network.name = NULL)
```

Arguments

`x`	An object of class `linkcomm` or `OCG`.
`file`	A character string naming a Clust&See partition file (.cns extension). Defaults to `"temp.cns"`.
`network.name`	A character string providing a name for the network. This name must correspond to the file name of the network that will be imported into Cytoscape. If NULL, the object name, `x`, is used. Defaults to NULL.

Details

Cytoscape is an open source platform for complex-network analysis and visualization, and Clust&See (Spinelli et al. 2013) is a Cytoscape plug-in used for visualizing and manipulating the clusters produced by various network clustering algorithms.

Value

Used for its side-effect of writing a Clust&See partition file to disk.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Becker, E. et al. (2012) Multifunctional proteins revealed by overlapping clustering in protein interaction network. *Bioinformatics* **28**, 84-90.

Gambette, P. and Guenoche, A. (2011) Bootstrap clustering for graph partitioning. *RAIRO-Operations Research* **45**, 339-352.

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Shannon, P. et al. (2003) Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research* **13**, 2498-2504.

Spinelli, L. et al. (2013) Clust&See: a Cytoscape plugin for the identification, visualization, and manipulation of network clusters. *BioSystems* **113**, 91-95.

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Write a partition file to disk.
## Not run:
linkcomm2clustnsee(lc)

## End(Not run)
```

`linkcomm2cytoscape` *Write an Edge Attribute File for Cytoscape*

Description

This function writes out an edge attribute file for visualising the link communities in Cytoscape.

Usage

```
linkcomm2cytoscape(x, interaction = "pp", ea = "temp.ea")
```

Arguments

`x`	An object of class `linkcomm`.
`interaction`	A character string indicating the type of interaction between nodes. Defaults to `"pp"` for protein-protein interaction.
`ea`	A character string indicating the file for writing the edge attributes. Defaults to `"temp.ea"`.

Details

Cytoscape is an open source platform for complex-network analysis and visualization (Shannon et al. 2003).

Value

Used for its side-effect of writing an edge attribute file to disk.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Shannon, P. et al. (2003) Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research* **13**, 2498-2504.

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Write an edge attribute file to disk.
## Not run:
linkcomm2cytoscape(lc)

## End(Not run)
```

LinkDensities *Calculate Link Community Link Densities*

Description

This function calculates link densities for link communities.

Usage

```
LinkDensities(x, clusterids = 1:x$numbers[3])
```

Arguments

x	An object of class linkcomm.
clusterids	An integer vector of community IDs. Defaults to all communities.

Details

The link density of community i is

$$D_i = \frac{e_i - n_i + 1}{(n_i(n_i - 1)/2) - n_i + 1}$$

where e_i is the number of edges in community i and n_i is the number of nodes in community i.

Value

A named numerical vector, where the names are community IDs and the numbers are link densities.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Ahn, Y.Y., Bagrow, J.P., and Lehmann, S. (2010). Link communities reveal multiscale complexity in networks. *Nature* **466**, 761-764.

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plot.linkcomm, plotLinkCommSummComm

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Calculate link densities.
ld <- LinkDensities(lc)
```

---

meta.communities *Produce a Set of Meta-Communities*

Description

This function returns meta-communities by hierarchically clustering link communities based on their number of shared nodes.

Usage

```
meta.communities(x, hcmethod = "ward.D", deepSplit = FALSE)
```

Arguments

x	An object of class linkcomm or OCG.
hcmethod	A character string naming the hierarchical clustering method to use. Can be one of "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", or "centroid". Defaults to "ward.D".
deepSplit	Logical or integer value from 0 to 4 indicating how fine-grained the clusters should be with larger values giving increased cluster structure. Defaults to FALSE.

Details

Extracting meta-communities allows the user to explore community relatedness and structure at higher levels. Community relatedness is calculated using the Jaccard coefficient and the number of nodes that community i and j share:

$$S(i,j) = \frac{|n_i \cap n_j|}{|n_i \cup n_j|}$$

A hierarchical clustering dendrogram is generated based on the above distance metric and then an algorithm is used to automatically extract clusters (Langfelder et al. 2008). This function works best on large networks with a large number of link communities.

Value

An object of class `linkcomm`, which is a list containing the following components:

`numbers`	An integer vector with the number of edges, nodes, and communities.
`hclust`	An object of class `hclust`, which contains information about the hierarchical clustering of links.
`pdmax`	A numerical value indicating the height of the dendrogram at which the partition density is maximised.
`pdens`	A numerical matrix with 2 columns; the first is the heights at which clusters appear and the second is the partition density.
`nodeclusters`	A data frame consisting of 2 columns; the first contains node names, and the second contains single community IDs for each node. All communities and their nodes are represented, but not necessarily all nodes.
`clusters`	A list of integer vectors containing the link IDs that belong to each community. Community IDs are the numerical position of the communities in the list.
`edges`	A data frame with 3 columns; the first two contain nodes that interact with each other, and the third is an integer vector of community IDs indicating community membership for each link.
`numclusters`	A named integer vector. Names are node names and integer values are the number of communities to which each node belongs.
`clustsizes`	A named integer vector. Names are community IDs and integer values indicate the number of nodes that belong in each community.
`igraph`	An object of class `igraph`. The network is represented here as an `igraph` object.
`edgelist`	A character matrix with 2 columns containing the nodes that interact with each other.
`directed`	Logical indicating whether the network is directed.
`bipartite`	Logical indicating whether the network is bi-partite.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Langfelder, P., Zhang, B., and Horvath, S. (2008) Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. *Bioinformatics* **24**, 719-720.

See Also

`getClusterRelatedness`

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Produce meta-communities.
## Not run: mc <- meta.communities(lc)
```

---

newLinkCommsAt *User-Defined Link Communities*

Description

This function allows the user to extract link communities by cutting the dendrogram at a specified height.

Usage

```
newLinkCommsAt(x, cutat = 0.5)
```

Arguments

`x`	An object of class `linkcomm`.
`cutat`	A numerical value indicating the height at which to cut the dendrogram. Defaults to 0.5.

Details

Users may wish to explore the communities formed by cutting the dendrogram higher or lower than the maximum partition density height. After cutting at a new height, the `pdmax` value in the resulting object will be set to the `cutat` value used by the user, only to distinguish the new object from the one generated by the link communities algorithm (the global partition density maximum does not change).

Value

An object of class `linkcomm`, which is a list containing the following components:

`numbers`	An integer vector with the number of edges, nodes, and communities.
`hclust`	An object of class `hclust`, which contains information about the hierarchical clustering of links.
`pdmax`	A numerical value indicating the height of the dendrogram at which the partition density is maximised.
`pdens`	A numerical matrix with 2 columns; the first is the heights at which clusters appear and the second is the partition density.
`nodeclusters`	A data frame consisting of 2 columns; the first contains node names, and the second contains single community IDs for each node. All communities and their nodes are represented, but not necessarily all nodes.
`clusters`	A list of integer vectors containing the link IDs that belong to each community. Community IDs are the numerical position of the communities in the list.
`edges`	A data frame with 3 columns; the first two contain nodes that interact with each other, and the third is an integer vector of community IDs indicating community membership for each link.
`numclusters`	A named integer vector. Names are node names and integer values are the number of communities to which each node belongs.
`clustsizes`	A named integer vector. Names are community IDs and integer values indicate the number of nodes that belong in each community.
`igraph`	An object of class `igraph`. The network is represented here as an `igraph` object.
`edgelist`	A character matrix with 2 columns containing the nodes that interact with each other.
`directed`	Logical indicating whether the network is directed.
`bipartite`	Logical indicating whether the network is bipartite.

Note

After cutting at a new height, the `pdmax` value in the resulting object will be set to the `cutat` value used by the user, only to distinguish the new object from the one generated by the link communities algorithm (the global partition density maximum does not change).

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

getLinkCommunities

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## User defined communities.
lc2 <- newLinkCommsAt(lc, cutat = 0.8)
```

---

numberEdgesIn *Extract Node Community Membership by Edges*

Description

This function returns the number of edges node(s) contain in each community.

Usage

```
numberEdgesIn(x, clusterids = 1:x$numbers[3], nodes)
```

Arguments

x	An object of class linkcomm or OCG.
clusterids	An integer vector of community IDs. Defaults to all communities.
nodes	A character vector specifying node(s) for which edge membership should be returned.

Value

A named list of named integer vectors specifying the number of edges in each community a node belongs in. Names of the integer vectors are community IDs, and names of the list are node names.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Examples

```
## Generate graph and extract OCG communities.
g <- swiss[,3:4]
oc <- getOCG.clusters(g)

## Get edges from community 1.
numberEdgesIn(oc, nodes = 1)
```

`orderCommunities` *Order Link Communities According to the Dendrogram*

Description

This function returns link communities in the same order as in the hierarchical clustering dendrogram.

Usage

```
orderCommunities(x, clusterids = 1:x$numbers[3], verbose = TRUE)
```

Arguments

`x`	An object of class `linkcomm`.
`clusterids`	An integer vector of community IDs. Defaults to all communities.
`verbose`	Logical, whether to print progress of the calculation to the screen. Defaults to TRUE.

Details

Ordering link communities according to the dendrogram can aid in visualization when plotting them as a Spencer circle because it minimses crossing over between links.

Value

A list with the following components:

`ordered`	A list of integer vectors. These are the ordered communities of links.
`clusids`	An integer vector of community IDs in their new order.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plot.linkcomm, plotLinkCommGraph

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Order communities according to the dendrogram.
orderCommunities(lc)
```

---

plot.linkcomm *The* linkcomm *Plotting Function*

Description

This function plots various different linkcomm graphs.

Usage

```
## S3 method for class 'linkcomm'
plot(x, type = "", ...)
```

Arguments

x	An object of class linkcomm.
type	A character string specifying the type of plot. Can be one of "summary", "members", "graph", "commsumm", and "dend". See Details below.
...	Additional arguments to be passed to plot.

Details

"summary" plots the dendrogram and partition density plot side-by-side;
"members" plots a community membership matrix;
"graph" plots a graph layout of the network with coloured link communities;
"commsumm" plots a bar graph or pie chart summarising community modularity or connectedness for each community;
"dend" plots a dendrogram with coloured link communities.

See the individual plotting functions for details of arguments that can be passed to plot.linkcomm: plotLinkCommSumm, plotLinkCommMembers, plotLinkCommGraph, plotLinkCommSummComm, and plotLinkCommDend.

Value

Plots to the current device.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plotLinkCommSumm, plotLinkCommMembers, plotLinkCommGraph, plotLinkCommSummComm, plotLinkCommDend

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Plot a graph of link communities.
plot(lc, type = "graph")
```

---

plot.OCG *The* OCG *Plotting Function*

Description

This function plots various different OCG graphs.

Usage

```
## S3 method for class 'OCG'
plot(x, type = "", ...)
```

Arguments

x	An object of class OCG.
type	A character string specifying the type of plot. Can be one of "members" or "graph". See Details below.
...	Additional arguments to be passed to plot.

Details

"members" plots a community membership matrix;
"graph" plots a graph layout of the network with coloured link communities.

See the OCG plotting function for details of arguments that can be passed to plot.OCG: plotOCGraph, plotLinkCommMembers.

Value

Plots to the current device.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plotOCGraph, plotLinkCommMembers

Examples

```
## Generate graph and extract OCG communities.
g <- swiss[,3:4]
oc <- getOCG.clusters(g)

## Plot a graph of OCG clusters.
plot(oc, type = "graph")
```

---

plotLinkCommDend *Plot a Coloured Dendrogram of Link Communities*

Description

This function is called by `plot.linkcomm` to plot a dendrogram of coloured link communities.

Usage

```
plotLinkCommDend(x, col = TRUE, pal = brewer.pal(9, "Set1"),
        height = x$pdmax, right = FALSE, labels = FALSE, plotcut = TRUE,
        droptrivial = TRUE, leaflab = "none", verbose = TRUE, ...)
```

Arguments

x	An object of class `linkcomm`.
col	Logical, whether to add community-specific colours. Defaults to TRUE.
pal	A character vector describing a colour palette to be used for colouring the communites in the dendrogram plot. Defaults to `brewer.pal(9,"Set1")`.
height	A numerical value specifying the height at which the dendrogram is cut. Defaults to the maximum partition density height.

right	Logical, whether to orient the dendrogram to the right. Defaults to FALSE.
labels	Logical, whether to include labels in the dendrogram. Defaults to FALSE.
plotcut	Logical, whether to display a horizontal line where the dendrogram is cut. Defaults to TRUE.
droptrivial	Logical, whether to not colour communities of size 2. Defaults to TRUE.
leaflab	A character string describing the leaf labels on the dendrogram. Can be one of "none", "perpendicular", or "textlike". Defaults to "none".
verbose	Logical, whether to display the progress of colouring the dendrogram on screen. Defaults to TRUE.
...	Additional arguments to be passed to plot.

Details

Here we describe the parameters for plotting coloured dendrograms using:
`plot(x,type = "dend")`

Value

A dendrogram plot.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plot.linkcomm

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Plot a coloured dendrogram.
plot(lc, type = "dend")
```

plotLinkCommGraph *Plot a Graph Layout of Link Communities*

Description

This function is called by `plot.linkcomm` to plot a graph layout of the link communities.

Usage

```
plotLinkCommGraph(x, clusterids = 1:length(x$clusters), nodes = NULL,
        layout = layout.fruchterman.reingold, pal = brewer.pal(7, "Set2"),
       random = TRUE, node.pies = TRUE, pie.local = TRUE, vertex.radius = 0.03,
        scale.vertices = 0.05, edge.color = NULL, vshape = "none", vsize = 15,
        ewidth = 3, margin = 0, vlabel.cex = 0.8, vlabel.color = "black",
        vlabel.family = "Helvetica", vertex.color = "palegoldenrod",
        vlabel = TRUE, col.nonclusters = "black", jitter = 0.2, circle = TRUE,
        printids = TRUE, cid.cex = 1, shownodesin = 0, showall = FALSE,
        verbose = TRUE, ...)
```

Arguments

`x`	An object of class `linkcomm`.
`clusterids`	An integer vector of community IDs. Defaults to all communities.
`nodes`	A character vector naming the nodes to be plotted. If NULL, then community IDs are used instead. Defaults to NULL.
`layout`	A character string or function identifying the layout algorithm to be used for positioning nodes in the graph. Defaults to `layout.fruchterman.reingold`. See details for alternative layouts.
`pal`	A character vector describing a colour palette to be used for colouring the link communities in the graph. Defaults to `brewer.pal(7,"Set2")`.
`random`	Logical, whether to randomise the link colours. Defaults to TRUE.
`node.pies`	Logical, whether to plot node pies showing as a pie chart the fraction of each node's edges which belong to each of its communities. Defaults to TRUE.
`pie.local`	Logical, whether to include pie segments for just the communities to which the chosen nodes belong, or for all communities. In the latter case, communities that are not present among the chosen nodes will appear as empty segments in the node pies. Will be ignored if node.pies is FALSE. Defaults to TRUE.
`vertex.radius`	A real number specifying the vertex radius. Defaults to 0.03. Will be ignored if node.pies is FALSE.
`scale.vertices`	A real number specifying the fraction of increase in vertex radius for each community membership. If NULL then all vertices are the same size. Defaults to 0.05. Will be ignored if node.pies is FALSE.
`edge.color`	A character string specifying the colour of edges. Defaults to `"grey"`. Will be ignored if node.pies is FALSE.

vshape	A character string specifying the shape of the nodes. Can be one of "none", "circle", "square", "csquare", "rectangle", "crectangle", and "vrectangle". Defaults to "none".
vsize	An integer vector of node sizes. If there is a single value this will be used for all nodes. If there are multiple values, it must be the same length as the number of nodes in the network to be visualized. See graph.feature for generating these values. This argument only has an effect when vshape is not set to "none". Defaults to 15.
ewidth	An integer vector of edge widths. If there is a single value this will be used for all edges. If there are multiple values, it must be the same length as the number of edges in the network to be visualized. See graph.feature for generating these values. Defaults to 3.
margin	A numerical value specifying the amount of empty space around the graph. Negative values will zoom into the graph. Defaults to 0.
vlabel.cex	A numerical value specifying the size of node labels. Defaults to 0.8.
vlabel.color	A character string specifying the color of node labels. Defaults to "black".
vlabel.family	A character string specifying the font family for node labels. Defaults to "Helvetica".
vertex.color	A character string specifying the colour of nodes. If this is a character vector then the colours will be recycled. Defaults to "palegoldenrod".
vlabel	Logical, whether node labels are to be added. Defaults to TRUE.
col.nonclusters	A character string specifying the colour of edges that do not belong to any communities. Will only have an effect if showall is TRUE. Defaults to "black".
jitter	A numerical value specifying the range (negative to positive) of random noise that will be added to nodes that have identical x-y coordinates. Defaults to 0.2. Only used for Spencer circle layouts.
circle	Logical, whether to display a circle for a Spencer circle layout. Defaults to TRUE.
printids	Logical, whether to display community IDs at their anchor points around the Spencer circle. Defaults to TRUE.
cid.cex	A numerical value specifying the size of community IDs around the Spencer circle. Defaults to 1.
shownodesin	An integer value specifying the number of communities a node must belong to before it will be displayed. If 0 then all nodes are displayed. Defaults to 0.
showall	Logical, whether to display all links in the network regardless of whether they belong to communities or not. Defaults to FALSE.
verbose	Logical, whether to print the progress of the calculation to the screen. Defaults to TRUE.
...	Additional arguments to be passed to plot.

Details

Here we describe the parameters for plotting link community graphs using:
`plot(x,type = "graph",layout = layout)`

Various graph layouts are available:

1. "spencer.circle"
2. layout.random
3. layout.circle
4. layout.sphere
5. layout.fruchterman.reingold
6. layout.kamada.kawai
7. layout.spring
8. layout.reingold.tilford
9. layout.fruchterman.reingold.grid
10. layout.lgl
11. layout.graphopt
12. layout.mds
13. layout.svd
14. layout.norm

All of these, except the "spencer.circle", are described in more detail in the igraph package. The "spencer.circle" is described in layout.spencer.circle.

Value

A graph plot.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plot.linkcomm, layout.spencer.circle, graph.feature, igraph.plotting

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Plot a graph of link communities.
plot(lc, type = "graph")
```

```
## Plot a graph of link communities using a Spencer circle layout.
plot(lc, type = "graph", layout = "spencer.circle")
```

---

plotLinkCommMembers *Plot a Community Membership Matrix for Link Communities*

Description

This function is called by `plot.linkcomm` to plot a community membership matrix for the link communities.

Usage

```
plotLinkCommMembers(x, nodes = head(names(x$numclusters), 10),
        pal = brewer.pal(11, "Spectral"), shape = "rect", total = TRUE,
        fontsize = 11, nspace = 3.5, maxclusters = 20)
```

Arguments

`x`	An object of class `linkcomm`.
`nodes`	A character vector specifying the node names that will be included in the plot. Defaults to the 10 nodes that belong to the most communities.
`pal`	A character vector describing a colour palette to be used for community-specific colouring. Defaults to `brewer.pal(11,"Spectral")`.
`shape`	A character string specifying the shape of matrix entries. Can be one of `"rect"` or `"circle"`. Defaults to `"rect"`.
`total`	Logical, whether to display the number of communities each node belongs to and the number of nodes in each community. Defaults to TRUE.
`fontsize`	A numerical value specifying font size for the node names. Defaults to 11.
`nspace`	A numerical value specifying how much space to leave at the left for fitting in node names. Defaults to 3.5.
`maxclusters`	An integer value specifying the maximum number of communities to display. Defaults to 20.

Value

A community membership matrix plot.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plot.linkcomm

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Plot a community membership matrix.
plot(lc, type = "members")
```

---

plotLinkCommSumm *Plot a Summary of the Link Community Algorithm Output*

Description

This function is called by `plot.linkcomm` to plot a summary of the output of the `linkcomm` algorithm.

Usage

```
plotLinkCommSumm(x, col = TRUE, pal = brewer.pal(9, "Set1"), right = TRUE,
                 droptrivial = TRUE, verbose = TRUE, ...)
```

Arguments

`x`	An object of class `linkcomm`.
`col`	Logical, whether to colour link communities in the dendrogram. Defaults to TRUE.
`pal`	A character vector describing a colour palette to be used for colouring the link community dendrogram. Defaults to `brewer.pal(9,"Set1")`.
`right`	Logical, whether to orient the dendrogram to the right. Defaults to TRUE.
`droptrivial`	Logical, whether to not colour communities of size 2. Defaults to TRUE.
`...`	Additional arguments to be passed to `plot`.
`verbose`	Logical, whether to display the progress of colouring the dendrogram on the screen. Defaults to TRUE.

Details

Here we describe the parameters for plotting link community summaries using:
`plot(x,type = "summary")`

Value

A summary plot of the output from the `linkcomm` algorithm.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

`plot.linkcomm`

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Plot the modularity of the link communities.
plot(lc, type = "summary")
```

---

plotLinkCommSummComm *Plot a Summary of the Link Communities*

Description

This function is called by `plot.linkcomm` to plot either connectedness or modularity of individual link communities.

Usage

```
plotLinkCommSummComm(x, clusterids = 1:x$numbers[3], summary = "conn",
        pie = FALSE, col = TRUE, pal = brewer.pal(11, "Spectral"),
        random = FALSE, verbose = TRUE, ...)
```

Arguments

x	An object of class linkcomm.
clusterids	An integer vector of community IDs. Defaults to all communities.
summary	A character string specifying the community summary. Can be one of "conn", "mod", "ld" for connectedness, modularity, and link densities respectively. Defaults to "conn".
pie	Logical, whether to plot a pie graph. If FALSE, a bar plot is plotted. Defaults to FALSE.
col	Logical, whether to colour each community differently. Defaults to TRUE.
pal	A character vector describing a colour palette to be used for colouring the link communities. Defaults to brewer.pal(11,"Spectral").
random	Logical, whether to randomise the link colours. Defaults to FALSE.
verbose	Logical, whether to print the progress of the calculation to the screen. Defaults to TRUE.
...	Additional arguments to be passed to plot.

Details

Here we describe the parameters for plotting link community summaries using:
plot(x,type = "commsumm",type = "mod")

Value

A bar graph or pie chart.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plot.linkcomm

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)


## Plot the modularity of the link communities.
plot(lc, type = "commsumm", summary = "mod")
```

plotOCGraph *Plot a Graph Layout of OCG Communities*

Description

This function is called by `plot.OCG` to plot a graph layout of the OCG communities.

Usage

```
plotOCGraph(x, clusterids = 1:x$numbers[3], nodes = NULL, pie.local = TRUE,
        incident = TRUE, layout = layout.fruchterman.reingold,
        vertex.radius = 0.03, scale.vertices = 0.05, edge.color = "grey",
        vertex.label.color = "black", vertex.label.cex = 0.8,
        pal = brewer.pal(7,"Set2"), shownodesin = 0, vlabel = TRUE,
        random = TRUE, ...)
```

Arguments

`x`	An object of class `OCG`.
`clusterids`	An integer vector of community IDs. Defaults to all communities.
`nodes`	A character vector naming the nodes to be plotted. If NULL, then community IDs are used instead. Defaults to NULL.
`pie.local`	Logical, whether to include pie segments for just the communities to which the chosen nodes belong, or for all communities. In the latter case, communities that are not present among the chosen nodes will appear as empty segments in the node pies. Defaults to TRUE.
`incident`	Logical, whether to include just the communities of the named node(s), or the community membership of all nodes that interact with the named node(s). Defaults to TRUE.
`layout`	A character string or function identifying the layout algorithm to be used for positioning nodes in the graph. Defaults to `layout.fruchterman.reingold`. See details for alternative layouts.
`vertex.radius`	A real number specifying the vertex radius. Defaults to 0.03.
`scale.vertices`	A real number specifying the fraction of increase in vertex radius for each community membership. If NULL then all vertices are the same size. Defaults to 0.05.
`edge.color`	A character string specifying the colour of edges. Defaults to `"grey"`.
`vertex.label.color`	A character string specifying the color of node labels. Defaults to `"black"`.
`vertex.label.cex`	A numerical value specifying the size of the node labels. Defaults to 0.8.
`pal`	A character vector describing a colour palette to be used for colouring the link communities in the graph. Defaults to `brewer.pal(7,"Set2")`.

shownodesin	An integer value specifying the number of communities a node must belong to before it will be displayed. If 0 then all nodes are displayed. Defaults to 0.
vlabel	Logical, whether node labels are to be added. Defaults to TRUE.
random	Logical, whether to randomise the link colours. Defaults to TRUE.
...	Additional arguments to be passed to plot.

Details

Here we describe the parameters for plotting OCG community graphs using:
`plot(x,type = "graph",layout = layout)`

Various graph layouts are available:

1. layout.random
2. layout.circle
3. layout.sphere
4. layout.fruchterman.reingold
5. layout.kamada.kawai
6. layout.spring
7. layout.reingold.tilford
8. layout.fruchterman.reingold.grid
9. layout.lgl
10. layout.graphopt
11. layout.mds
12. layout.svd
13. layout.norm

All of these are described in more detail in the igraph package.

Value

A graph plot.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

See Also

plot.OCG, igraph.plotting

Examples

```
## Generate graph and extract OCG communities.
g <- swiss[,3:4]
oc <- getOCG.clusters(g)

## Plot a graph of OCG communities.
plot(oc, type = "graph")
```

`pp_rnapol` *Sample Yeast Protein Interactome*

Description

A set of 56 yeast proteins involved in 651 interactions related to transcription (Yu et al. 2008).

Usage

```
pp_rnapol
```

Format

Data frame with 2 columns.

Source

http://interactome.dfci.harvard.edu

References

Yu, H., *et al*. (2008). High-quality binary protein interaction map of the yeast interactome network. *Science* **322**, 104-110.

`print.linkcomm` *Print a Summary of a* `linkcomm` *Object*

Description

This function prints summary statistics for a `linkcomm` object to the screen.

Usage

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

Arguments

x	An object of class linkcomm.
...	Further arguments passed to or from other methods.

Value

Prints summary data to the screen.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Examples

```
## Generate graph and extract link communities.
g <- swiss[,3:4]
lc <- getLinkCommunities(g)

## Print summary statistics to the screen.
print(lc)
```

---

print.OCG *Print a Summary of an* OCG *Object*

Description

This function prints summary statistics for an OCG object to the screen.

Usage

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

Arguments

x	An object of class OCG.
...	Further arguments passed to or from other methods.

Value

Prints summary data to the screen.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Examples

```
## Generate graph and extract OCG communities.
g <- swiss[,3:4]
oc <- getOCG.clusters(g)

## Print summary statistics to the screen.
print(oc)
```

---

read.OCG *Read an OCG Partition File into R*

Description

This function reads in an OCG partition file and converts it into an OCG object for use in R.

Usage

```
read.OCG(file, elfile = NULL, verbose = FALSE, keep.out = FALSE)
```

Arguments

file	A character string naming the OCG partition file.
elfile	A character string naming the file containing the network that the OCG partition is based upon.
verbose	Logical, whether to print progress to the screen. Defaults to FALSE.
keep.out	Logical, whether to keep the intermediate files written when reading in the OCG partition. Defaults to FALSE.

Value

An object of class OCG, which is a list containing the following elements:

numbers	An integer vector with the number of edges, nodes, and communities.
modularity	An integer number specifying the modularity of the network.
Q	A real number specifying the value of Q generated by the OCG algorithm.

nodeclusters	A data frame consisting of 2 columns; the first contains node names, and the second contains single community IDs for each node. All communities and their nodes are represented, but not necessarily all nodes.
numclusters	A named integer vector. Names are node names and integer values are the number of communities to which each node belongs.
igraph	An object of class igraph. The network is represented here as an igraph object.
edgelist	A character matrix with 2 columns containing the nodes that interact with each other.
clustsizes	A named integer vector. Names are community IDs and integer values indicate the number of nodes that belong in each community.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Examples

```
## Read an OCG partition file into R.
## Not run: oc <- read.OCG(file = "OCG_partition.txt", elfile = "network.txt")
```

---

weighted *Sample Gene Co-Expression Network*

Description

A sample of 200 links from a Drosophila gene co-expression network illustrating the input required for a weighted network (Tomancak et al. 2007).

Usage

```
weighted
```

Format

Data frame with 3 columns.

Source

http://www.fruitfly.org/cgi-bin/ex/insitu.pl

References

Tomancak, P. *et al*. (2007). Global analysis of patterns of gene expression during *Drosophila* embryogenesis. *Genome Biol* **8**, 145.1-145.34.

`which.communities` *Extract Community Membership for Nodes*

Description

This function returns the community IDs of the communities to which one or more nodes belong.

Usage

```
which.communities(x, nodes)
```

Arguments

`x`	An object of class `linkcomm` or `OCG`.
`nodes`	A character vector specifying the nodes.

Value

An integer vector of community IDs.

Author(s)

Alex T. Kalinka <alex.t.kalinka@gmail.com>

References

Kalinka, A.T. and Tomancak, P. (2011). linkcomm: an R package for the generation, visualization, and analysis of link communities in networks of arbitrary size and type. *Bioinformatics* **27**, 2011-2012.

Examples

```
## Generate graph and extract OCG communities.
g <- swiss[,3:4]
oc <- getOCG.clusters(g)

## Get edges from community 1.
which.communities(oc, nodes = 1)
```

Index