Package ‘clustAnalytics’
February 19, 2024

Type Package
Title Cluster Evaluation on Graphs
Version 0.5.5
Date 2024-02-18
Author Martí Renedo Mirambell
Maintainer Martí Renedo Mirambell <marti.renedo@gmail.com>
Description Evaluates the stability and significance of clusters on ‘igraph’ graphs.
  Supports weighted and unweighted graphs. Implements the cluster evaluation methods
defined by Arratia A, Renedo M (2021) <doi:10.7717/peerj-cs.600>. Also includes an
implementation of the Reduced Mutual Information introduced by Newman et al. (2020)
License GPL (>= 3)
Imports Rcpp (>= 1.0.1), mcclust, mclust, truncnorm, boot, fossil,
aricode, dplyr, Rdpack
LinkingTo Rcpp
RdMacros Rdpack
RoxygenNote 7.3.1
Suggests igraphdata, knitr, rmarkdown, testthat,
VignetteBuilder knitr
Encoding UTF-8
URL https://github.com/martirm/clustAnalytics
BugReports https://github.com/martirm/clustAnalytics/issues
Depends R (>= 2.10), igraph
LazyData true
NeedsCompilation yes
Repository CRAN
Date/Publication 2024-02-18 23:10:03 UTC
R topics documented:

average_degree ........................................ 2
average_odf ............................................. 3
barabasi_albert_blocks ................................. 4
boot_alg_list ........................................... 5
conductance ............................................. 6
contingency_to_membership_vectors .................. 7
count_contingency_tables_log ......................... 7
coverage ............................................... 8
cut_ratio ............................................... 8
density_ratio .......................................... 9
deges_inside ........................................... 10
estimate_H_fraction_r_rows ......................... 11
evaluate_significance .................................. 11
evaluate_significance_r ................................ 12
expansion .............................................. 13
FOMD ..................................................... 14
g_forex .................................................. 15
igraph_to_edgelist ...................................... 15
internal_density ....................................... 16
make_graph_weighted .................................... 16
max_odf .................................................. 17
normalized_cut ........................................ 18
out_degree_fractions ................................... 18
reduced_mutual_information ............................. 19
relabel .................................................. 20
rewireCpp ............................................... 21
scoring_functions ....................................... 22
sort_matrix ............................................. 23
triangle_participation_ratio_communities .......... 23
weighted_clustering_coefficient ...................... 24
weighted_transitivity ................................ 24

Index 26

average_degree Average Degree

Description

Average degree (weighted degree, if the graph is weighted) of a graph's communities.

Usage

average_degree(g, com)
average_odf

Arguments

- **g**: Graph to be analyzed (as an igraph object). If the edges have a "weight" attribute, those will be used as weights.
- **com**: Community membership integer vector. Each element corresponds to a vertex.

Value

Numeric vector with the average degree of each community.

See Also

Other cluster scoring functions: `FOMD()`, `average_odf()`, `conductance()`, `coverage()`, `cut_ratio()`, `density_ratio()`, `edges_inside()`, `expansion()`, `internal_density()`, `max_odf()`, `normalized_cut()`, `scoring_functions()`, `weighted_clustering_coefficient()`, `weighted_transitivity()`

Examples

```r
data(karate, package="igraphdata")
average_degree(karate, membership(cluster_louvain(karate)))
```
Barabasi-Albert Graph with Community Structure

**See Also**

Other cluster scoring functions: `FOMD()`, `average_degree()`, `conductance()`, `coverage()`, `cut_ratio()`, `density_ratio()`, `edges_inside()`, `expansion()`, `internal_density()`, `max_odf()`, `normalized_cut()`, `scoring_functions()`, `weighted_clustering_coefficient()`, `weighted_transitivity()`

**Examples**

```r
data(karate, package="igraphdata")
average_odf(karate, membership(cluster_louvain(karate)))
```

**Description**

Generates a Barabási-Albert graph with community structure

**Usage**

```r
barabasi_albert_blocks(
  m, p, B, t_max, G0 = NULL, t0 = NULL, G0_labels = NULL,
  sample_with_replacement = FALSE, type = "Hajek"
)
```

**Arguments**

- `m`: number of edges added at each step.
- `p`: vector of label probabilities. If they don’t sum 1, they will be scaled accordingly.
- `B`: matrix indicating the affinity of vertices of each label.
- `t_max`: maximum value of t (which corresponds to graph order)
- `G0`: initial graph
- `t0`: t value at which new vertex start to be attached. If G0 is provided, this argument is ignored and assumed to be gorder(G0)+1. If it isn’t, a G0 graph will be generated with order t0-1.
- `G0_labels`: labels of the initial graph. If NULL, they will all be set to 1.
- `sample_with_replacement`: If TRUE, allows parallel edges.
- `type`: Either "Hajek" or "block_first".
**Value**

The resulting graph, as an igraph object. The vertices have a "label" attribute.

**Examples**

```r
B <- matrix(c(1, 0.2, 0.2, 1), ncol=2)
G <- barabasi_albert_blocks(m=4, p=c(0.5, 0.5), B=B, t_max=100, type="Hajek",
                           sample_with_replacement = FALSE)
```

---

**Description**

Performs nonparametric bootstrap on a graph's by resampling its vertices and clustering the results using a list of clustering algorithms.

**Usage**

```r
boot_alg_list(
  alg_list = list(Louvain = cluster_louvain, 'label prop' = cluster_label_prop, walktrap = cluster_walktrap),
  g,
  R = 999,
  return_data = FALSE,
  type = "global"
)
```

**Arguments**

- `alg_list` List of igraph clustering algorithms
- `g` igraph graph object
- `R` Number of bootstrap replicates.
- `return_data` Logical. If TRUE, returns a list of "boot" objects with the full results. Otherwise, returns a table with the mean results.
- `type` Can be "global" (Variation of Information, Reduced Mutual Information, and adjusted Rand Index) or "cluster-wise" (Jaccard distance)

**Value**

If `return_data` is set to TRUE, returns a list of objects of class "boot" (see `boot`). Otherwise, returns as table with the mean distances from the clusters in the original graph to the resampled ones, for each of the algorithms.
Conductance

Description

Conductance of a graph’s communities, which is given by

\[
\frac{c_s}{2m_s + c_s}
\]

where \(c_s\) is the weight of the edges connecting the community \(s\) to the rest of the graph, and \(m_s\) is the internal weight of the community.

Usage

\[
\text{conductance}(g, \text{com})
\]

Arguments

g Graph to be analyzed (as an igraph object). If the edges have a "weight" attribute, those will be used as weights.

com community membership integer vector. Each element corresponds to a vertex.

Value

Numeric vector with the conductance of each community.

See Also

Other cluster scoring functions: FOMD(), average_degree(), average_odf(), coverage(), cut_ratio(), density_ratio(), edges_inside(), expansion(), internal_density(), max_odf(), normalized_cut(), scoring_functions(), weighted_clustering_coefficient(), weighted_transitivity()

Examples

data(karate, package="igraphdata")
conductance(karate, membership(cluster_louvain(karate)))
contingency_to_membership_vectors

Computes possible membership vectors from contingency table

Description
Given a contingency table, obtains a possible pair of corresponding labelings. That is, element $M[i,j]$ is the number of elements that belong to community $i$ in the first labeling and $j$ in the second.

Usage
contingency_to_membership_vectors(M)

Arguments
M the contingency table

Value
a list containing the two membership vectors

count_contingency_tables_log

Natural logarithm of the number of contingency tables

Description
Given a contingency table, returns the natural logarithm of the number of contingency tables that share the same column and row sums. This implementation combines a Markov Chain Monte Carlo approximation with an analytical formula. The input can be either $M$ a contingency table, or two vectors of labels $c1$ and $c2$ (in this case, we are counting contingency tables with the same column an row sums as the one produced by $c1$ and $c2$).

Usage
count_contingency_tables_log(c1, c2, M = NULL, monte_carlo_only = FALSE)

Arguments
c1, c2 membership vectors
M contingency table
monte_carlo_only
   Uses only the Monte Carlo approximation
coverage

Coverage

Description
Computes the coverage (fraction of internal edges with respect to the total number of edges) of a graph and its communities.

Usage
coverage(g, com)

Arguments
- **g**: Graph to be analyzed (as an igraph object).
- **com**: Community membership integer vector. Each element corresponds to a vertex of the graph, and contains the index of the community it belongs to.

Value
Numeric value of the coverage of g and com.

See Also
Other cluster scoring functions: FOMD(), average_degree(), average_odf(), conductance(), cut_ratio(), density_ratio(), edges_inside(), expansion(), internal_density(), max_odf(), normalized_cut(), scoring_functions(), weighted_clustering_coefficient(), weighted_transitivity()

Examples
```r
data(karate, package="igraphdata")
coverage(karate, membership(cluster_louvain(karate)))
```

cut_ratio

Cut Ratio

Description
The cut ratio of a graph’s community is the total edge weight connecting the community to the rest of the graph divided by number of unordered pairs of vertices such that one belongs to the community and the other does not.

Usage
cut_ratio(g, com)
density_ratio

Arguments

- **g**: Graph to be analyzed (as an igraph object). If the edges have a "weight" attribute, those will be used as weights.
- **com**: Community membership integer vector. Each element corresponds to a vertex.

Value

Numeric vector with the cut ratio of each community.

See Also

Other cluster scoring functions: `FOMD()`, `average_degree()`, `average_odf()`, `conductance()`, `coverage()`, `density_ratio()`, `edges_inside()`, `expansion()`, `internal_density()`, `max_odf()`, `normalized_cut()`, `scoring_functions()`, `weighted_clustering_coefficient()`, `weighted_transitivity()`

Examples

```r
data(karate, package="igraphdata")
cut_ratio(karate, membership(cluster_louvain(karate)))
```

density_ratio

Description

Density ratio of a graph’s communities.

Usage

```r
density_ratio(g, com, type = "local")
```

Arguments

- **g**: Graph to be analyzed (as an igraph object). If the edges have a "weight" attribute, those will be used as weights.
- **com**: Community membership integer vector. Each element corresponds to a vertex.
- **type**: Can either be "local" or "global"

Value

Numeric vector with the internal density of each community.

See Also

Other cluster scoring functions: `FOMD()`, `average_degree()`, `average_odf()`, `conductance()`, `coverage()`, `cut_ratio()`, `edges_inside()`, `expansion()`, `internal_density()`, `max_odf()`, `normalized_cut()`, `scoring_functions()`, `weighted_clustering_coefficient()`, `weighted_transitivity()`
edges_inside

Examples

data(karate, package="igraphdata")
density_ratio(karate, membership(cluster_louvain(karate)))

Edges Inside

Description

Number of edges inside a graph’s communities, or their accumulated weight if the graph’s edges are weighted.

Usage

edges_inside(g, com)

Arguments

g  Graph to be analyzed (as an igraph object). If the edges have a "weight" attribute, those will be used as weights.
com  community membership integer vector. Each element corresponds to a vertex.

Value

Numeric vector with the internal edge weight of each community

See Also

Other cluster scoring functions: FOMD(), average_degree(), average_odf(), conductance(), coverage(), cut_ratio(), density_ratio(), expansion(), internal_density(), max_odf(), normalized_cut(), scoring_functions(), weighted_clustering_coefficient(), weighted_transitivity()

Examples

data(karate, package="igraphdata")
edges_inside(karate, membership(cluster_louvain(karate)))
**estimate_H_fraction_r_rows**

Estimates $|H_0|/|H_{r*}|$

**Description**

This is the total number of contingency tables (of the same margins as M) divided by the number that match M until the r-th row (included, 0-indexed). Note that if r==0, this is always 1 by definition.

**Usage**

```r
estimate_H_fraction_r_rows(M, r, error = 0.1)
```

**Arguments**

- `M`: contingency table
- `r`: row index
- `error`: error for the convergence of the method

---

**evaluate_significance**  
Evaluates significance of cluster algorithm results on a graph

**Description**

Given a graph and a list of clustering algorithms, computes several scoring functions on the clusters found by each of the algorithms.

**Usage**

```r
evaluate_significance(
  g,
  alg_list = list(Louvain = cluster_louvain, 'label prop' = cluster_label_prop, walktrap = cluster_walktrap),
  no_clustering_coef = FALSE,
  gt_clustering = NULL,
  w_max = NULL
)
```

**Arguments**

- `g`: Graph to be analyzed (as an igraph object)
- `alg_list`: List of clustering algorithms, which take an igraph graph as input and return an object of the communities class.
evaluate_significance_r

- no_clustering_coef
  - Logical. If TRUE, skips the computation of the clustering coefficient, which is the most computationally costly of the scoring functions.

- gt_clustering
  - Vector of integers that correspond to labels of the ground truth clustering. The scoring functions will be evaluated on it.

- w_max
  - Numeric. Upper bound for edge weights. Should be generally left as default (NULL).

Value

A data frame with the values of scoring functions (see scoring_functions) of the clusters obtained by applying the clustering algorithms to the graph.

Examples

```r
data(karate, package="igraphdata")
evaluate_significance(karate)
```

Description

Computes community scoring functions to the communities obtained by applying the given clustering algorithms to a graph. These are compared to the same scores for randomized versions of the graph obtained by a switching algorithm that rewire edges.

Usage

```r
evaluate_significance_r(
  g,
  alg_list = list(Louvain = cluster_louvain, 'label prop' = cluster_label_prop, walktrap = cluster_walktrap),
  no_clustering_coef = FALSE,
  gt_clustering = NULL,
  table_style = "default",
  ignore_degenerate_cl = TRUE,
  Q = 100,
  lower_bound = 0,
  weight_sel = "const_var",
  n_reps = 5,
  w_max = NULL
)
```
**expansion**

**Description**

Given a graph (possibly weighted) split into communities, the expansion of a community is the sum of all edge weights connecting it to the rest of the graph divided by the number of vertices in the community.

**Usage**

```
expansion(g, com)
```
Arguments

g  Graph to be analyzed (as an igraph object). If the edges have a "weight" attribute, those will be used as weights.
com  community membership integer vector. Each element corresponds to a vertex.

Value

Numeric vector with the expansion of each community.

See Also

Other cluster scoring functions: FOMD(), average_degree(), average_odf(), conductance(), coverage(), cut_ratio(), density_ratio(), edges_inside(), internal_density(), max_odf(), normalized_cut(), scoring_functions(), weighted_clustering_coefficient(), weighted_transitivity()

Examples

data(karate, package="igraphdata")
expansion(karate, membership(cluster_louvain(karate)))

FOMD (Fraction Over Median Degree)

Description

Given a weighted graph and a partition into communities, returns the fraction of nodes of each community whose internal degree (i.e. the degree accounting only intra-community edges) is greater than the median degree of the whole graph.

Usage

FOMD(g, com, edgelist = NULL)

Arguments

g  Graph to be analyzed (as an igraph object). If the edges have a "weight" attribute, those will be used as weights.
com  Community membership integer vector. Each element corresponds to a vertex.
edgelist  alternatively, the edgelist of the graph, as a matrix where the first two columns to the vertices and the third is the weight of each edge.

Value

Numeric vector with the FOMD of each community.
See Also

Other cluster scoring functions: `average_degree()`, `average_odf()`, `conductance()`, `cut_ratio()`, `density_ratio()`, `edges_inside()`, `expansion()`, `internal_density()`, `max_odf()`, `normalized_cut()`, `scoring_functions()`, `weighted_clustering_coefficient()`, `weighted_transitivity()`

Examples

```r
data(karate, package="igraphdata")
FOMD(karate, membership(cluster_louvain(karate)))
```

---

**Description**

Network built from correlations between time series of exchange rate returns. It was built from the 13 most traded currencies and with data of January 2009. It is a complete graph of 78 vertices (corresponding to pairs of currencies) and has edge weights bounded between 0 and 1.

**Usage**

g_forex

**Format**

An igraph object with 78 vertices and 3003 weighted edges

---

**igraph_to_edgelist**

Returns edgelist with weights from a weighted igraph graph

**Description**

This function is just used internally for testing the package

**Usage**

`igraph_to_edgelist(g, sort = TRUE)`

**Arguments**

- `g`: igraph graph with weighted edges
- `sort`: sorts the edge list lexicographically before returning

**Value**

A matrix where the first two columns indicate the incident vertices, and the third is the weight of the corresponding edge.
internal_density  Internal Density

Description
Internal density of a graph’s communities. That is, the sum of weights of their edges divided by the number of unordered pairs of vertices (which is the number of potential edges).

Usage
internal_density(g, com)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>Graph to be analyzed (as an igraph object). If the edges have a &quot;weight&quot; attribute, those will be used as weights.</td>
</tr>
<tr>
<td>com</td>
<td>Community membership integer vector. Each element corresponds to a vertex.</td>
</tr>
</tbody>
</table>

Value
Numeric vector with the internal density of each community.

See Also
Other cluster scoring functions: FOMD(), average_degree(), average_odf(), conductance(), coverage(), cut_ratio(), density_ratio(), edges_inside(), expansion(), max_odf(), normalized_cut(), scoring_functions(), weighted_clustering_coefficient(), weighted_transitivity()

Examples
```r
data(karate, package="igraphdata")
internal_density(karate, membership(cluster_louvain(karate)))
```

make_graph_weighted  Make graph weighted

Description
Given a graph, create a "weight" attribute set to 1 for the edges if it doesn’t exist already.

Usage
make_graph_weighted(g)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>Igraph graph</td>
</tr>
</tbody>
</table>
**Value**

igraph graph with either all edge weights set to 1 (if the original graph was unweighted), or to their original weights if they already existed (in this case, the graph isn’t modified at all).

<table>
<thead>
<tr>
<th>max_odf</th>
<th>Max Out Degree Fraction</th>
</tr>
</thead>
</table>

**Description**

Computes the Maximum Out Degree Fraction (Max ODF) of a graph (which can be weighted) and its communities.

Computes the Flake Out Degree Fraction (Max ODF) of a graph (which can be weighted) and its communities.

**Usage**

```r
max_odf(g, com)
max_odf(g, com)
```

**Arguments**

- `g`  
  Graph to be analyzed (as an igraph object). If the edges have a “weight” attribute, those will be used as weights (otherwise, all edges are assumed to be 1).

- `com`  
  Community membership integer vector. Each element corresponds to a vertex of the graph, and contains the index of the community it belongs to.

**Value**

Numeric vector with the Max ODF of each community.

**See Also**

Other cluster scoring functions: `FOMD()`, `average_degree()`, `average_odf()`, `conductance()`, `coverage()`, `cut_ratio()`, `density_ratio()`, `edges_inside()`, `expansion()`, `internal_density()`, `normalized_cut()`, `scoring_functions()`, `weighted_clustering_coefficient()`, `weighted_transitivity()`

Other cluster scoring functions: `FOMD()`, `average_degree()`, `average_odf()`, `conductance()`, `coverage()`, `cut_ratio()`, `density_ratio()`, `edges_inside()`, `expansion()`, `internal_density()`, `normalized_cut()`, `scoring_functions()`, `weighted_clustering_coefficient()`, `weighted_transitivity()`

**Examples**

```r
data(karate, package="igraphdata")
max_odf(karate, membership(cluster_louvain(karate)))
data(karate, package="igraphdata")
max_odf(karate, membership(cluster_louvain(karate)))
```
Description

Normalized cut of a graph’s communities, which is given by

\[ \frac{c_s}{2m_s + c_s} + \frac{c_s}{2(m - m_s) + c_s} \]

, where \( c_s \) is the weight of the edges connecting the community \( s \) to the rest of the graph, \( m_s \) is the internal weight of the community, and \( m \) is the total weight of the network.

Usage

normalized_cut(g, com)

Arguments

g Graph to be analyzed (as an igraph object). If the edges have a "weight" attribute, those will be used as weights.

com community membership integer vector. Each element corresponds to a vertex.

Value

Numeric vector with the normalized cut of each community.

See Also

Other cluster scoring functions: FOMD(), average_degree(), average_odf(), conductance(), coverage(), cut_ratio(), density_ratio(), edges_inside(), expansion(), internal_density(), max_odf(), scoring_functions(), weighted_clustering_coefficient(), weighted_transitivity()

Examples

data(karate, package="igraphdata")
normalized_cut(karate, membership(cluster_louvain(karate)))
**out_degree_fractions**  
*Maximum, Average, and Flake Out Degree Fractions of a Graph Partition*

**Description**

Given a weighted graph and a partition into communities, returns the maximum, average and flake out degree fractions of each community.

**Usage**

```r
out_degree_fractions(g, com, edgelist)
```

**Arguments**

- `g` Graph to be analyzed (as an igraph object)
- `com` Community membership vector. Each element corresponds to a vertex of the graph, and contains the index of the community it belongs to.
- `edgelist` alternatively, the edgelist of the graph

**Value**

A numeric matrix where each row corresponds to a community, and the columns contain the max, average and flake ODFs respectively.

---

**reduced_mutual_information**  
*Reduced Mutual Information*

**Description**

Computes the Newman’s Reduced Mutual Information (RMI) as defined in (Newman et al. 2020).

**Usage**

```r
reduced_mutual_information(
  c1,
  c2,
  base = 2,
  normalized = FALSE,
  method = "approximation2",
  warning = TRUE
)
```
Arguments

c1, c2 membership vectors
base base of the logarithms used in the calculations. Changing it only scales the final value. By default set to e=exp(1).
normalized If true, computes the normalized version of the corrected mutual information.
method Can be "hybrid" (default, combines Monte Carlo with analytical formula), "monte_carlo", "approximation1" (appropriate for partitions into many very small clusters), or "approximation2" (for partitions into few larger clusters).
warning set to false to ignore the warning.

Details

The implementation is based on equations 23 (25 for the normalized case) and 29 in (Newman et al. 2020). The evaluations of the $\Gamma$ functions can get too large and cause overflow issues in the intermediate steps, so the following term of equation 29:

$$\frac{1}{2} \log \frac{\Gamma(\mu R)\Gamma(\nu S)}{(\Gamma(\nu)\Gamma(R))^S(\Gamma(\mu)\Gamma(S))^R}$$

is rewritten as

$$\frac{1}{2} (\log(\mu R) + \log(\nu S) - S \log(\nu) - S \log(\mu) - R \log(\nu S) - R \log(\mu R))$$

, and then the function lgamma is used instead of gamma.

Value

The value of Newman’s RMI (a scalar).

References


relabel Relabels membership vector

Description

Takes a vector of vertex ids indicating community membership, and relabels the communities to have consecutive values from 1 to the number of communities.

Usage

relabel(c)
Arguments

- **c**: numeric vector of vertex ids, not necessarily consecutive

Value

- A numeric vector of consecutive vertex ids starting from one

---

**rewireCpp**

Randomizes a weighted graph while keeping the degree distribution constant.

Description

Converts the graph to a weighted edge list in NumericMatrix, which is compatible with Rcpp. The Rcpp function "randomize" is called, and then the resulting edge list is converted back into an igraph object.

Usage

```r
rewireCpp(
  g,
  Q = 100,
  weight_sel = "max_weight",
  lower_bound = 0,
  upper_bound = NULL
)
```

Arguments

- **g**: igraph graph, which can be weighted.
- **Q**: Numeric. Parameter that controls the number of iterations, which will be Q times the order of the graph.
- **weight_sel**: can be either "const_var" or "max_weight".
- **lower_bound**, **upper_bound**: Bounds to the edge weights. The randomization process will avoid steps that would make edge weights fall outside these bounds. Set to NULL for no bound. By default, 0 and NULL respectively.

Value

The rewired graph.
scoring_functions

Scoring Functions of a Graph Partition

Description

Computes the scoring functions of a graph and its clusters.

Usage

scoring_functions(
  g,
  com,
  no_clustering_coef = TRUE,
  type = "local",
  weighted = TRUE,
  w_max = NULL
)

Arguments

g               Graph to be analyzed (as an igraph object). If the edges have a "weight" attribute, those will be used as weights (otherwise, all edges are assumed to be 1).
com             Community membership integer vector. Each element corresponds to a vertex of the graph, and contains the index of the community it belongs to.
no_clustering_coef
                Logical. If TRUE, skips the computation of the clustering coefficient (which can be slow on large graphs).
type            can be "local" for a cluster by cluster analysis, or "global" for a global analysis of the whole graph partition.
weighted        Is the graph weighted? If it is, doesn’t compute TPR score.
w_max           Numeric. Upper bound for edge weights. Should be generally left as default (NULL). Only affects the computation of the clustering coefficient.

Value

If type="local", returns a dataframe with a row for each community, and a column for each score. If type="global", returns a single row with the weighted average scores.

See Also

Other cluster scoring functions: FOMD(), average_degree(), average_odf(), conductance(), coverage(), cut_ratio(), density_ratio(), edges_inside(), expansion(), internal_density(), max_odf(), normalized_cut(), weighted_clustering_coefficient(), weighted_transitivity()
sort_matrix

Examples

```r
data(karate, package="igraphdata")
scoring_functions(karate, membership(cluster_louvain(karate)))
```

descriptions

sort_matrix

Sort matrix

Description

Given a matrix, rearranges rows and columns so that row sums and col sums end up in ascending order.

Usage

```r
sort_matrix(M)
```

Arguments

- `M` matrix

Value

rearranged matrix

triangle_participation_ratio_communities

Triangle Participation Ratio (community-wise)

Description

Computes the triangle participation ratio (proportion of vertices that belong to a triangle). The computation is done to the subgraphs induced by each of the communities in the given partition.

Usage

```
triangle_participation_ratio_communities(g, com)
```

Arguments

- `g` The input graph (as an igraph object). Edge weights and directions are ignored.
- `com` Community membership vector. Each element corresponds to a vertex of the graph, and contains the index of the community it belongs to.

Value

A vector containing the triangle participation ratio of each community.
weighted_clustering_coefficient

*Weighted clustering coefficient of a weighted graph.*

**Description**

Weighted clustering Computed using the definition given by McAssey, M. P. and Bijma, F. in "A clustering coefficient for complete weighted networks" (2015).

**Usage**

`weighted_clustering_coefficient(g, upper_bound = NULL)`

**Arguments**

- `g` igraph graph
- `upper_bound` upper bound to the edge weights used to compute the integral

**Value**

The weighted clustering coefficient of the graph (a scalar).

**See Also**

Other cluster scoring functions: `FOMD()`, `average_degree()`, `average_odf()`, `conductance()`, `coverage()`, `cut_ratio()`, `density_ratio()`, `edges_inside()`, `expansion()`, `internal_density()`, `max_odf()`, `normalized_cut()`, `scoring_functions()`, `weighted_transitivity()`

**Examples**

```r
data(karate, package="igraphdata")
weighted_clustering_coefficient(karate)
```

---

weighted_transitivity

*Weighed transitivity of a weighted graph.*

**Description**

Computed using the definition given by McAssey, M. P. and Bijma, F. in "A clustering coefficient for complete weighted networks" (2015).

**Usage**

`weighted_transitivity(g, upper_bound = NULL)`

**See Also**

Other cluster scoring functions: `FOMD()`, `average_degree()`, `average_odf()`, `conductance()`, `coverage()`, `cut_ratio()`, `density_ratio()`, `edges_inside()`, `expansion()`, `internal_density()`, `max_odf()`, `normalized_cut()`, `scoring_functions()`, `weighted_clustering_coefficient()`
weighted_transitivity

Arguments

- `g`: igraph graph
- `upper_bound`: upper bound to the edge weights used to compute the integral

Value

The weighted transitivity of the graph (a scalar).

See Also

Other cluster scoring functions: `FOMD()`, `average_degree()`, `average_odf()`, `conductance()`, `coverage()`, `cut_ratio()`, `density_ratio()`, `edges_inside()`, `expansion()`, `internal_density()`, `max_odf()`, `normalized_cut()`, `scoring_functions()`, `weighted_clustering_coefficient()`

Examples

```r
data(karate, package="igraphdata")
weighted_transitivity(karate)
```
Index

* cluster scoring functions
  average_degree, 2
  average_odf, 3
  conductance, 6
  coverage, 8
  cut_ratio, 8
  density_ratio, 9
  edges_inside, 10
  expansion, 13
  FOMD, 14
  internal_density, 16
  max_odf, 17
  normalized_cut, 18
  scoring_functions, 22
  weighted_clustering_coefficient, 24
  weighted_transitivity, 24

* datasets
  g_forex, 15
  average_degree, 2, 4, 6, 8–10, 14–18, 22, 24, 25
  average_odf, 3, 3, 6, 8–10, 14–18, 22, 24, 25
  barabasi_albert_blocks, 4
  boot, 5
  boot_alg_list, 5
  conductance, 3, 4, 6, 8–10, 14–18, 22, 24, 25
  contingency_to_membership_vectors, 7
  count_contingency_tables_log, 7
  coverage, 3, 4, 6, 8, 9, 10, 14–18, 22, 24, 25
  cut_ratio, 3, 4, 6, 8, 9, 10, 14–18, 22, 24, 25
  density_ratio, 3, 4, 6, 8, 9, 9, 10, 14–18, 22, 24, 25
  edges_inside, 3, 4, 6, 8, 9, 10, 14–18, 22, 24, 25
  estimate_H_fraction_r_rows, 11

  evaluate_significance, 11
  evaluate_significance_r, 12
  expansion, 3, 4, 6, 8–10, 13, 15–18, 22, 24, 25
  FOMD, 3, 4, 6, 8–10, 14, 14, 16–18, 22, 24, 25
  g_forex, 15
  gamma, 20
  igraph_to_edgelist, 15
  internal_density, 3, 4, 6, 8–10, 14, 15, 16, 17, 18, 22, 24, 25
  lgamma, 20
  make_graph_weighted, 16
  max_odf, 3, 4, 6, 8–10, 14–16, 17, 18, 22, 24, 25
  normalized_cut, 3, 4, 6, 8–10, 14–17, 18, 22, 24, 25
  out_degree_fractions, 19
  reduced_mutual_information, 19
  relabel, 20
  rewireCpp, 13, 21
  scoring_functions, 3, 4, 6, 8–10, 12, 14–18, 22, 24, 25
  sort_matrix, 23
  triangle_participation_ratio_communities, 23
  weighted_clustering_coefficient, 3, 4, 6, 8–10, 14–18, 22, 24, 25
  weighted_transitivity, 3, 4, 6, 8–10, 14–18, 22, 24, 25