Package ‘iGraphMatch’

May 17, 2024

Type Package
Title Tools for Graph Matching
Version 2.0.5
Description Versatile tools and data for graph matching analysis with various forms of prior information that supports working with 'igraph' objects, matrix objects, or lists of either.
URL https://github.com/dpmcsuss/iGraphMatch, https://dpmcsuss.github.io/iGraphMatch/
Depends R (>= 3.6.0)
Imports clue (>= 0.3-54), Matrix (>= 1.6-2), igraph (>= 2.0.0), irlba, methods, Rcpp
Suggests spelling, dplyr (>= 0.5.0), testthat (>= 2.0.0), knitr, rmarkdown, ggplot2, purrr, bookdown
VignetteBuilder knitr
License GPL (>= 2)
LazyData TRUE
RoxygenNote 7.3.1
Language en-US
Encoding UTF-8
LinkingTo Rcpp
Config/testthat/edition 3
BugReports https://github.com/dpmcsuss/iGraphMatch/issues
NeedsCompilation yes
Author Daniel Sussman [aut, cre], Zihuan Qiao [aut], Joshua Agterberg [ctb], Lujia Wang [ctb], Vince Lyzinski [ctb]
Maintainer Daniel Sussman <sussman@bu.edu>
Repository CRAN
Date/Publication 2024-05-17 19:10:03 UTC
### R topics documented:

- `best_matches` .......................................................... 2
- `C.Elegans` ............................................................ 3
- `center_graph` ........................................................ 4
- `check_graph` .......................................................... 5
- `check_seeds` ........................................................... 6
- `check_sim` ............................................................... 7
- `do_lap` ................................................................. 8
- `Enron` ................................................................. 9
- `get_perm_mat` ......................................................... 10
- `gm` ..................................................................... 11
- `graphMatch-class` .................................................... 12
- `init_start` ............................................................. 14
- `largest_common_cc` .................................................. 16
- `pad` ................................................................. 17
- `plot.igraph.igraph-method` ....................................... 17
- `sample_correlated_gnp_pair` ...................................... 19
- `sample_correlated_lgl_pair` ...................................... 20
- `sample_correlated_sbm_pair` ...................................... 21
- `split_igraph` ......................................................... 23
- `splrMatrix-class` ..................................................... 23
- `splr_sparse_plus_constant` ........................................ 24
- `summary,graphMatch-method` ...................................... 25
- `Transportation` ...................................................... 26
- `%*%`, `igraph`, `igraph-method` ................................... 27

### Index

<table>
<thead>
<tr>
<th>best_matches</th>
<th>Rank best matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>

### Description

Rank vertex-pairs in order of a goodness of matching metric

### Usage

```r
best_matches(A, B, match, measure, num = NULL, true_label = NULL)
```

### Arguments

- **A**
  - A matrix, an `igraph` object, or a list of either. See `check_graph`
- **B**
  - A matrix, an `igraph` object, or a list of either. See `check_graph`
- **match**
  - `graphMatch`, eg result of call to `gm`
- **measure**
  - One of "row_cor", "row_diff", or "row_perm_stat" or a function (see details). Measure for computing goodness of matching.
C.Elegans networks consist of the chemical synapses network and the electrical synapses network of the roundworm, where each of 279 nodes represents a neuron and each edge represents the intensity of synapses connections between two neurons.

Usage

data(C.Elegans)
Format

An object of class list of length 2.

Details

Two networks are weighted and directed graphs with self-loops. There are 2194 and 1031 edges in two graphs respectively and the empirical Pearson’s correlation between two graphs is 0.17. Two networks are stored in a list in the form of igraph objects, where the first network in the list is the chemical synapses network and the other one is the electrical synapses network.

References


Examples

data(C.Elegans)
g1 <- C.Elegans[[1]]
g2 <- C.Elegans[[2]]
plot(g1, g2)

center_graph

Center adjacency matrix

Description

Center the adjacency matrix by re-weighting edges according to a specified scheme

Usage

center_graph(A, scheme = c(-1, 1), use_splr = TRUE)

Arguments

A A matrix, an igraph object. Adjacency matrix.
scheme A character vector, number or pair of numbers. Default c(-1, 1). See Details.
use_splr A boolean indicating whether to use the splrMatrix object when storing the centered graph. Defaults to TRUE.
check_graph

Details

The options for scheme are

- "naive" Returns original A
- Integer: Returns \( A - A_{\text{scheme}} \) where \( A_{\text{scheme}} \) is the best rank-scheme approximation of A.
- A pair of scalars: Returns \( s \cdot A + a \) such that the minimum of the returned matrix is \( \min(\text{scheme}) \) and the maximum is \( \max(\text{scheme}) \).
- "center": Same as scheme=c(-1,1)

Value

centered adjacency matrix as a splrMatrix if useSplr = TRUE, otherwise as a Matrix object.

Examples

```r
A <- sample_correlated_gnp_pair(n = 10, corr = .5, p = .5)$graph1
center_graph(A, scheme = "naive")
center_graph(A, scheme = "center")
center_graph(A, scheme = 2)
center_graph(A, scheme = c(-4, 2))
```

check_graph

Parameter checking for a graph-pair

Description

Function that checks that the pair of graphs passed to a matching-related functions satisfies necessary conditions and modifies them according to specified parameters. check_single_graph does similar checks and modifications but just for one graph or list of graphs.

Usage

```r
check_graph(
  A,
  B,
  same_order = TRUE,
  square = TRUE,
  as_list = TRUE,
  as_igraph = FALSE
)
```

```r
check_single_graph(A, square = TRUE, as_list = TRUE, as_igraph = FALSE)
```
Arguments

A matrix, an igraph object, or list of either.

B A matrix, an igraph object, or list of either.

same_order Whether the returned objects should have the same number of nodes. If the graphs start with different numbers of nodes the smaller graph is padded with isolated vertices. (default = TRUE)

square Whether the matrices need to be square. (default = TRUE) Currently non-square matrices are not supported.

as_list Whether to return the results as a matrix_list. (default = TRUE) If FALSE and A and B have length > 1

as_igraph Whether to return an igraph object. (default=FALSE) Only allowed if the original parameters are igraph objects. If FALSE, then this converts the objects to sparse matrices.

Details

If A and B are lists of matrices or igraph objects, then the lists must be the same length. Additionally, within each list the graphs need to have the same number of vertices but this does not need to be true across lists.

Value

List containing A and B modified according to the parameters and the number of vertices in each graph in totv1 and totv2.

check_seeds Standardize seeds input data type

Description

Convert the input seeds data into data frame type with the first column being the indices of $G_1$ and the second column being the corresponding indices of $G_2$.

Usage

cHECK_seeds(seeds, nv, logical = FALSE)

Arguments

seeds A vector of integers or logicals, a matrix or a data frame. Input in the form of a vector of integers denotes the indices of seeds which are identical in both graphs. Input in the form of a vector of logicals indicate the location of seeds with TRUE and the indices of seeds are identical in both graphs. Input in the form of a matrix or a data frame, with the first column being the indices of $G_1$ and the second column being the corresponding indices of $G_2$. 
check_sim

nv
An integer. Number of total vertices.

logical
A logical. TRUE indicates to return seeds in a vector of logicals where TRUE indicates the corresponding vertex is a seed. FALSE indicates to return a data frame.

Value
returns a data frame with the first column being the corresponding indices of $G_1$ and the second column being the corresponding indices of $G_2$ or a vector of logicals where TRUE indicates the corresponding vertex is a seed.

Examples

#input is a vector of logicals
check_seeds(1:10 <= 3, nv = 10)

#input is a vector of integers
check_seeds(c(1,4,2,7,3), nv = 10)

#input is a matrix
check_seeds(matrix(1:4, 2), nv = 10)

#input is a data frame
check_seeds(as.data.frame(matrix(1:4, 2)), nv = 10)

check_sim

Check the similarity matrix passed to a matching function

Description
Internal function that checks that a similarity matrix satisfies necessary conditions and modifies it for use in graph matching.

Usage
check_sim(sim, seeds, nonseeds, totv1, totv2, for_nonseeds = TRUE)

Arguments

sim
Similarity matrix

seeds
dataframe of seed matches from running check_seeds

nonseeds
dataframe of nonseed nodes from running check_seeds

totv1
total number of vertices in the first graph

totv2
total number of vertices in the second graph

for_nonseeds
Whether the similarities are between non-seed nodes only (default = TRUE), or if similarities among seed nodes are included (FALSE)
Details
The goal here is to be flexible in terms of the dimensions of the similarity matrix passed to gm. This is useful when the graphs have different orders in which case the function accepts matrices with dimensions equal to that of orders of the original graphs or the number of nonseeds.

Value
Standardized similarity matrix for similarities only between nonseeds across the two graphs, if for_nonseeds = TRUE, or between all nodes, if for_nonseeds = FALSE

do_lap
Linear (sum) assignment problem

Description
Compute the best bipartite matching using one of three methods. For an n x n score matrix it find $\max_{v \in \Pi_n} \sum_{i=1}^{n} \text{score}_{i,v(i)}$ where $\Pi_n$ denotes all permutations on n objects.

Usage
do_lap(score, method = "clue")

Arguments
score matrix of pairwise scores
method One of "lapjv", "lapmod", or "clue"

Details
Solves a linear assignment using one of three methods. "clue" uses solve_lsap from the clue package. "lapjv" uses the Jonker-Volgenant approach implemented in this package. "lapmod" use a modification of JV that exploits sparsity in the score matrix.

Scores do not need to be non-negative. For "clue" the scores are pre-translated to be non-negative which preserves the LAP solution.

Value
do_lap returns a vector which indicates the best matching column for each row.

References
**Enron**

**Examples**

```r
set.seed(12345)
cost <- Matrix::rsparsematrix(10, 10, .5)
cbind(
  do_lap(cost, "lapjv"),
  do_lap(cost, "lapmod"),
  do_lap(cost, "clue")
)
```

---

**Enron**  
*Email communication networks of Enron Corporation*

**Description**

The Enron network data consists of email messages between 184 employees of the Enron Corporation where each graph represents one week of emails and each edge indicates whether there is email sent from one employee to the other.

**Usage**

```r
data(Enron)
```

**Format**

An object of class `list` of length 2.

**Details**

Two networks are unweighted and directed with self-loops. There are 488 and 482 edges in two networks respectively and the empirical Pearson’s correlation between two graphs is 0.85. Two email communication networks for two different weeks are stored in a list in the form of igraph objects.

**References**

Originally released by William Cohen at CMU. [More details](#) on the origins and research uses of the dataset.

**Examples**

```r
data(Enron)
g1 <- Enron[[1]]
g2 <- Enron[[2]]
plot(g1, g2)
```
get_perm_mat  

Get Permutation

Description

Get an \( m \times n \) permutation matrix according to the mapping correspondence.

Usage

\[
\text{get_perm_mat}(\text{match}, \text{dim} = \text{NULL}, \text{padded} = \text{FALSE}, \text{seeds} = \text{TRUE})
\]

Arguments

- **match**: Either a graphMatch object or 2-column matrix or data frame. The first and second columns correspond to indices in \( G_1 \) and \( G_2 \) respectively.
- **dim**: desired dimensions of the matrix. Note, this does not have to be square. If NULL and match is a graphMatch object then dim is set to dim(match)
- **padded**: If FALSE then this returns a square matrix the size of the larger of the two graph otherwise dim = dim(match). This is ignored if match is not a graphMatch object.
- **seeds**: Whether to keep the seed vertices (TRUE) from the match or to remove them (FALSE). Ignored if match is not a graphMatch object.

Value

get_perm_mat returns an \( m \times n \) sparse permutation matrix or whose submatrix is a permutation matrix if only parts of nodes from both graphs get matched or in the case of matching graphs of different order.

Examples

# returns a permutation matrix: m=n, all the nodes get matched
corr <- data.frame(corr_A = c(1,2,3,4), corr_B = c(1,4,2,3))
get_perm_mat(corr, c(4, 4))

# submatrix is a permutation matrix: parts of graphs get matched
get_perm_mat(corr, c(5, 6))
Description

gm is used to match a pair of given graphs, with specifications of the adjacency matrices of for a pair of graphs, possible prior knowledge, and a graph matching method.

Usage

gm(A, B, seeds = NULL, similarity = NULL, method = "indefinite", ...)

Arguments

A  
A matrix, igraph object, or list of either.

B  
A matrix, igraph object, or list of either.

seeds  
A vector of integers or logicals, a matrix or a data frame. If the seed pairs have the same indices in both graphs then seeds can be a vector. If not, seeds must be a matrix or a data frame, with the first column being the indices of \( G_1 \) and the second column being the corresponding indices of \( G_2 \).

similarity  
A matrix. An \( n \times n \) matrix containing vertex similarities. Mandatory for the "IsoRank" method.

method  
Choice for graph matching methods. One of "indefinite", "convex", "PATH", "percolation", "IsoRank", "Umeyama", or a user-defined graph matching function. Please check Details and Examples sections for instructions on how to define your own function.

...  
Arguments passed to graph matching methods. Please refer to Details section for more information.

Details

If method is a function, it should take two matrices or igraph objects, seeds and similarity scores as arguments for minimum. Additionally, it can also take other arguments if needed. The self-defined function should return a graphMatch class object with matching correspondence, sizes of two input graphs, matching formula, and other algorithm hyperparameter details.

The method argument can also take one of the implemented algorithms, including "indefinite", "convex", "PATH", "percolation", "IsoRank", and "Umeyama". In this case, one can pass additional arguments to the gm function according to the specified method. For a detailed list of additional arguments for each one of the implemented method, please click on the corresponding method name for its help page.

Most graph matching functions include as list elements additional details about the match. Call names() on a graphMatch object to see the available details. As an example, PATH, IsoRank, Umeyama, Indefinite, and Convex each include soft, which is the matrix found by the algorithm prior to projection onto the set of permutation matrices. Similarly, PATH, Indefinite, and Convex return iter, the number of iterations, and IsoRank (with greedy LAP) and Percolation return match_order, the order that the node-pairs were added to the match.
Value

gm returns an object of class "graphMatch". See graphMatch-class and links therein for details on
the graphMatch class.

Please also refer to the help page for each implemented method, i.e. "indefinite", "convex", "PATH",
"percolation", "IsoRank", and "Umeyama" for details on the corresponding returned list.

Examples

# match G_1 & G_2 with some known node pairs as seeds
set.seed(123)
cgnp_pair <- sample_correlated_gnp_pair(n = 10, corr = 0.5, p = 0.5)
g1 <- cgnp_pair$graph1
g2 <- cgnp_pair$graph2
seeds <- 1:10 <= 4
m_rds <- gm(g1, g2, seeds, method = "indefinite", start = "rds", max_iter = 20)
summary(m_rds, g1, g2, true_label = 1:10)

# match two multi-layer graphs
set.seed(123)
gp_list <- replicate(3, sample_correlated_gnp_pair(10, .3, .5), simplify = FALSE)
A <- lapply(gp_list, function(gp)gp[[1]])
B <- lapply(gp_list, function(gp)gp[[2]])

m_perco <- gm(A, B, seeds, method = "percolation", ExpandWhenStuck = FALSE)
summary(m_perco, A, B)

sim <- as.matrix(init_start(start = "bari", nns = 20, soft_seeds = 1:5))
m_Iso <- gm(A, B, similarity = sim, method = "IsoRank", lap_method = "greedy")
summary(m_Iso, A, B)

# customized graph matching algorithm
graph_match_rand <- function(A, B, seeds = NULL, similarity = NULL, rand_seed){
  nm <- min(nrow(A), nrow(B))
  set.seed(rand_seed)
  m <- data.frame(sample(nrow(A), nm), corr_B = sample(nrow(B), nm))
  m <- as.graphMatch(m)
  m$rand_seed <- rand_seed
  m
}
m_self <- gm(g1, g2, method = graph_match_rand, rand_seed = 123)
summary(m_self, g1, g2)
Description

An S4 class for the results of a graph matching function

Usage

```r
graphMatch(corr, nnodes, call = NULL, detail = list())
as.graphMatch(from)
```

Arguments

- `corr`: data.frame indicating the correspondence between two graphs
- `nnodes`: dimensions of the original two graphs
- `call`: The call to the graph matching function
- `detail`: List with other more detailed information
- `from`: object to convert to graphMatch object

Details

graphMatch objects are returned by any of the graph matching methods implemented in the iGraph-Match package. These objects are primarily to represent the found correspondence between the two vertex sets. This is represented by a data.frame with two columns indicating the aligned vertex-pairs across the two graphs.

Value

graphMatch object

Slots

- `corr`: data.frame indicating the correspondence between two graphs
- `nnodes`: of the original two graphs
- `call`: The call to the graph matching function

See Also

graphMatch_methods, graphMatch_summary, graphMatch_operators, graphMatch_plot

Examples

```r
# sample a pair of correlated random graphs from G(n,p)
set.seed(123)
cgnp_pair <- sample_correlated_gnp_pair(n = 10, corr = 0.3, p = 0.5)
g1 <- cgnp_pair$graph1
g2 <- cgnp_pair$graph2

# match g1 & g2 using percolation algorithm with some known node pairs as seeds
match <- gm(A = g1, B = g2, seeds = 1:3, method = 'indefinite')
```
# graphMatch object
match

match$corr_A # matching correspondence in the first graph
match$corr_B # matching correspondence in the second graph
match$seeds # vector of logicals indicating seeded nodes

as.data.frame(match)
match[]
dim(match)
length(match)

# matching details unique to the FW methodology with indefinite relaxation
match$iter # number of iterations
match$soft # doubly stochastic matrix from the last iteration, can be used to extract soft matching
match$lap_method # method for solving lap

# create a graphMatch object from a data.frame or matrix
as.graphMatch(data.frame(1:5, 1:5))
as.graphMatch(1:5)

---

**init_start**  
*Initialization of the start matrix*

## Description
Initialize the start matrix for graph matching iteration.

## Usage

```
init_start(start, nns, ns = 0, soft_seeds = NULL, seeds = NULL, ...)
```

## Arguments

- **start**
  - A matrix, character, or function. A nns-by-nns matrix, start method like "bari", "convex" or "rds", or a function to initialize the start matrix. If a function, it must have at least the arguments nns, ns, and softs_seeds.

- **nns**
  - An integer. Number of non-seeds.

- **ns**
  - An integer. Number of seeds.

- **soft_seeds**
  - A vector, a matrix or a data frame indicating entries of the start matrix that will be initialized at 1 to indicate . See `check_seeds`.

- **seeds**
  - A vector, a matrix or a data frame. Indicating hard seeds. These are used for "convex" start but otherwise are ignored.

- **...**
  - Arguments passed to other start functions. See details in Values section.
Details

When `start` is a character, there are five options.

- "bari" initializes at the barycenter.
- "rds_perm_bari" gives a random linear combination of barycenter and a random permutation matrix, \((1-a) B + a P\). The argument \(g\) controls \(a\) with \(a\) being sampled as \(g \times \text{runif()}\).
- "rds" gives a random doubly stochastic matrix. Users can specify a random deviates generator to the distribution argument, and the default is \(\text{runif}()\). A random matrix with iid entries from distribution and the the Sinkhorn algorithm is applied to produce the output.
- "rds_from_sim" gives a random doubly stochastic matrix derived from similarity scores. One needs to input a similarity score matrix to the `sim` argument for this method. The procedure is the same as "rds" but before the Sinkhorn algorithm is applied, the entries of the random matrix are scaled by `sim`.
- "convex" returns the doubly stochastic matrix from the last iteration of running the Frank-Wolfe algorithm with convex relaxation initialized at the barycenter. For this method, one needs to input two graphs \(A\) and \(B\), as well as seeds if applicable.

Value

`init_start` returns a \(nns\)-by-\(nns\) doubly stochastic matrix as the start matrix in the graph matching iteration. If conduct a soft seeding graph matching, returns a \(nns\)-by-\(nns\) doubly stochastic matrix with 1's corresponding to the soft seeds and values at the other places are derived by different start method.

Examples

```r
ss <- matrix(c(5, 4, 4, 3), nrow = 2)
# initialize start matrix without soft seeds
init_start(start = "bari", nns = 5)
init_start(start = "rds", nns = 3)
init_start(start = "rds_perm_bari", nns = 5)
init_start(start = "rds_from_sim", nns = 3, sim = matrix(runif(9), 3))

# initialize start matrix with soft seeds
init_start(start = "bari", nns = 5, ns = 1, soft_seeds = ss)
init_start(start = "rds", nns = 5, soft_seeds = ss)
init_start(start = "rds_perm_bari", nns = 5, soft_seeds = ss)

# initialize start matrix for convex graph matching
cgnp_pair <- sample_correlated_gnp_pair(n = 10, corr = 0.3, p = 0.5)
g1 <- cgnp_pair$graph1
g2 <- cgnp_pair$graph2
seeds <- 1:10 <= 2
init_start(start = "convex", nns = 8, A = g1, B = g2, seeds = seeds)

# FW graph matching with incorrect seeds to start at convex start
init_start(start = "convex", nns = 8, ns = 2, soft_seeds = ss, A = g1, B = g2, seeds = seeds)
```
largest_common_cc

Find the largest common connected subgraph (LCCS) of two graphs

Description

Find the largest common connected subgraphs of two matched graphs, which is an induced connected subgraph of both graphs that has as many vertices as possible. The largest_cc function returns the largest connected subgraph of a single graph.

Usage

largest_common_cc(A, B, min_degree = 1)
largest_cc(A)

Arguments

A  A matrix or an igraph object. See check_graph. Must be single-layer.
B  A matrix or an igraph object. See check_graph. Must be single-layer.
min_degree  A number. Defines the level of connectedness of the obtained largest common connected subgraph. The induced subgraph is a graph with a minimum vertex-degree of at least min_degree.

Value

largest_common_cc returns the common largest connected subgraphs of two aligned graphs in the igraph object form and a logical vector indicating which vertices in the original graphs remain in the induced subgraph.

Examples

cgnp_pair <- sample_correlated_gnp_pair(n = 10, corr = 0.7, p = 0.2)
g1 <- cgnp_pair$graph1
g2 <- cgnp_pair$graph2
# put no constraint on the minimum degree of the common largest connected subgraph
lccs1 <- largest_common_cc(g1, g2, min_degree = 1)
# induced subgraph
lccs1$g1
lccs1$g2
# label of vertices of the induced subgraph in the original graph
igraph::V(g1)[lccs1$keep]

# obtain a common largest connected subgraph with each vertex having a minimum degree of 3
lccs3 <- largest_common_cc(g1, g2, min_degree = 3)
g <- igraph::sample_gnp(100, .01)
lcc <- largest_cc(g)
# induced subgraph
lcc$g
# label of vertices of the induced subgraph in the original graph
igraph::V(g)[lcc$keep]

---

**pad**

*Pad a matrix object with extra rows/columns of 0s.*

### Description

Attempts are made to make this padding efficient by employing sparse graphs.

### Usage

pad(m, nr, nc = nr)

### Arguments

- **m**: matrix
- **nr**: number of rows to add
- **nc**: number of columns to add. (default = nr)

### Value

m padded with nr rows and nc columns of zeros.

---

**plot,igraph,igraph-method**

*Plotting methods for visualizing matches*

### Description

Two functions are provided, `match_plot_igraph` which makes a ball and stick plot from igraph objects and `match_plot_matrix` which shows an adjacency matrix plot.

### Usage

```r
## S4 method for signature 'igraph,igraph'
plot(x, y, match = NULL, color = TRUE, linetype = TRUE, ...)
```

```r
## S4 method for signature 'Matrix,Matrix'
plot(x, y, match = NULL, col.regions = NULL, at = NULL, colorkey = NULL, ...)
```
Arguments

- **x**: First graph, either an igraph object or a Matrix
- **y**: Second graph, either an igraph object or a Matrix
- **match**: Result from a match call. Requires element `corr` as a data.frame with names `corr_A`, `corr_B`.
- **color**: Whether to color edges according to which graph(s) they are in.
- **linetype**: Whether to set edge line types according to which graph(s) they are in.
- **...**: Additional parameters passed to either the igraph plot function or the Matrix image function.
- **col.regions**: NULL for default colors, otherwise see image-methods.
- **at**: NULL for default at values for at (ensures zero is grey), otherwise see image-methods.
- **colorkey**: NULL for default colorkey, otherwise see image-methods.

Details

Grey edges/pixels indicate common edges, blue indicates edges only in graph A and red represents edges only graph B. The corresponding linetypes are solid, long dash, and short dash.

The plots can be recreated from the output with the code:

```r
plot(g)
for g <- match_plot_igraph(...) and
col <- colorRampPalette(c("#AA4444", "#888888", "#44AA44"))
image(m, col.regions = col(256))
for m <- match_plot_match(...).
```

This only plots and returns the matched vertices.

Value

Both functions return values invisibly. `match_plot_igraph` returns the union of the matched graphs as an igraph object with additional edge attributes `edge_match`, `color`, and `lty`. `match_plot_matrix` returns the difference between the matched graphs.

Examples

```r
set.seed(123)
graphs <- sample_correlated_gnp_pair(20, .9, .3)
A <- graphs$graph1
B <- graphs$graph2
res <- gm(A, B, 1:4, method = "percolation")

plot(A, B, res)
plot(A[,], B[], res)
```
**sample_correlated_gnp_pair**

Sample correlated G(n,p) random graphs

**Description**

Sample a pair of correlated G(n,p) random graphs with correlation between two graphs being corr and edge probability being p.

**Usage**

```r
sample_correlated_gnp_pair(n, corr, p, ncore = n, permutation = 1:n, ...)
```

**Arguments**

- `n`: An integer. Number of total vertices for the sampled graphs.
- `corr`: A number. The target Pearson correlation between the adjacency matrices of the generated graphs. It must be in [0,1] interval.
- `p`: A number. Edge probability between two vertices. It must be in open [0,1] interval.
- `ncore`: An integer. Number of core vertices.
- `permutation`: A numeric vector to permute second graph.
- `...`: Passed to `sample_gnp`.

**Value**

`sample_correlated_gnp_pair` returns a list of two igraph object, named `graph1` and `graph2`, whose adjacency matrix entries are correlated with `corr`. If sample two graphs with junk vertices, the first `ncore` vertices are core vertices and the rest are junk vertices.

**References**


**See Also**

`sample_correlated_sbm_pair`, `sample_correlated_rdpg_pair`

**Examples**

```r
sample_correlated_gnp_pair(n=50, corr=0.3, p=0.5, ncore=40)
sample_correlated_gnp_pair(n=5, corr=0.3, p=0.5, permutation=c(1,3,2,4,5))
```
Sample graphs from edge probability matrix and correlation matrix

Description

Sample a pair of graphs with specified edge probability and correlation between each pair of vertices.

Usage

```r
sample_correlated_ieg_pair(
  n,
  p_mat,
  c_mat,
  ncore = n,
  directed = FALSE,
  loops = FALSE,
  permutation = 1:n
)
```

```r
sample_correlated_rdpg_pair(X, corr, ncore = nrow(X), ...)
```

Arguments

- `n`: An integer. Number of total vertices for the sampled graphs.
- `p_mat`: An `n`-by-`n` matrix. Edge probability matrix, each entry should be in the open `(0,1)` interval.
- `c_mat`: An `n`-by-`n` matrix. The target Pearson correlation matrix, each entry should be in the open `(0,1)` interval.
- `ncore`: An integer. Number of core vertices.
- `directed`: Logical scalar, whether to generate directed graphs.
- `loops`: Logical scalar, whether self-loops are allowed in the graph.
- `permutation`: A numeric vector, permute second graph.
- `X`: A matrix. Dot products matrix, each entry must be in open `(0,1)` interval.
- `corr`: A number. The target Pearson correlation between the adjacency matrices of the generated graphs. It must be in open `(0,1)` interval.
- `...`: Passed to `sample_correlated_ieg_pair`.

Value

`sample_correlated_ieg_pair` returns two igraph objects named `graph1` and `graph2`. If sample two graphs with junk vertices, the first `ncore` vertices are core vertices and the rest are junk vertices.

`sample_correlated_rdpg_pair` returns two igraph objects named `graph1` and `graph2` that are sampled from random dot product graphs model. If sample two graphs with junk vertices, the first `ncore` vertices are core vertices and the rest are junk vertices.
References


See Also

sample_correlated_gnp_pair, sample_correlated_sbm_pair

Examples

```r
n <- 50
p_mat <- matrix(runif(n^2),n)
c_mat <- matrix(runif(n^2),n)
sample_correlated_ieg_pair(n,p_mat,c_mat,ncore=40)

## sample a pair of igraph objects from random dot product graphs model with dimension 3 and scale 8
n <- 50
xdim <- 3
scale <- 8
X <- matrix(rgamma(n*(xdim+1),scale,1),n,xdim+1)
X <- X/rowSums(X)
X <- X[,1:xdim]
sample_correlated_rdpg_pair(X,corr=0.5,ncore=40)
```

---

sample_correlated_sbm_pair

*Sample graphs pair from stochastic block model*

Description

Sample a pair of random graphs from stochastic block model with correlation between two graphs being `corr` and edge probability being `p`.

Usage

```r
sample_correlated_sbm_pair(
  n,
  pref.matrix,
  block.sizes,
  corr,
  core.block.sizes = NULL,
  permutation = 1:n,
  ...
)
```
sample_correlated_sbm_pair

Arguments

n  An integer. Number of vertices in the graph.
pref.matrix  The matrix giving the Bernoulli rates. This is a K-by-K matrix, where k is the number of groups. The probability of creating an edge between vertices from groups i and j is given by element i,j. For undirected graphs, this matrix must be symmetric.
block.sizes  A numeric vector. Give the number of vertices in each group. The sum of the vector must match the number of vertices.
corr  A number. The target Pearson correlation between the adjacency matrices of the generated graphs. It must be in open (0,1) interval.
core.block.sizes  A numeric vector. Give the number of core vertices in each group. Entries should be smaller than block.sizes and the vector length should be the same as block.sizes.
permutation  A numeric vector, permute second graph.
...  Passed to sample_sbm.

Value

Returns a list of two igraph object, named graph1 and graph2. If sample two graphs with junk vertices, in each corresponding block the first core.block.sizes vertices are core vertices and the rest are junk vertices.

References


See Also

sample_correlated_gnp_pair, sample_correlated_rdpg_pair

Examples

pm <- cbind( c(.1, .001), c(.001, .05) )
sample_correlated_sbm_pair(n=1000, pref.matrix=pm, block.sizes=c(300,700), corr=0.5)
sample_correlated_sbm_pair(n=1000, pref.matrix=pm, block.sizes=c(300,700), corr=0.5,
core.block.sizes=c(200,500))
**Description**

Given an igraph object and an edge attribute, this function finds all unique values of the edge attribute in the graph and returns a list of igraph objects on the same vertex set where each element of the list has a graph containing only those edges with specified attribute.

**Usage**

```r
split_igraph(g, e_attr, strip_vertex_attr = FALSE)
```

**Arguments**

- `g` An igraph object
- `e_attr` the name of an edge attribute in `g`
- `strip_vertex_attr` Whether to remove all vertex attribute from the new graphs

**Value**

A named list of igraph objects with names corresponding to the values of the edge attributes.

**Examples**

```r
g <- igraph::sample_gnm(20, 60)
igraph::E(g)$color <- sample(c("red", "green"), 60, replace = TRUE)
split_igraph(g, "color")
```

---

**splrMatrix-class**

**Sparse Plus Low-Rank Matrices**

**Description**

An "S4" class for efficient computation with sparse plus low-rank matrices. Stores sparse plus low-rank matrices (e.g. from matrix factorization or centering graphs) of the form \( x + a \%\% t(b) \) for faster computation.

**Usage**

```r
splr(x, a = NULL, b = NULL, rank = NULL, dimnames = list(NULL, NULL), ...)
## S4 method for signature 'Matrix,Matrix,Matrix'
splr(x, a = NULL, b = NULL, rank = NULL, dimnames = list(NULL, NULL), ...)
```
Arguments

- **x**: as in "Matrix"
- **a**: as in "Matrix"
- **b**: as in "Matrix"
- **rank**: rank of the matrix to be factorized.
- **dimnames**: optional - the list of names for the matrix
- **...**: as in "Matrix"

Value

- splrMatrix object
- splrMatrix object

Slots

- **x**: a sparse matrix
- **a**: a low-rank factor or a matrix
- **b**: optional. a low-rank factor for a %*% t(b). if b is not provided, a will be factorized using irlba provided factorize = TRUE

See Also

Methods are documented in splrMatrix_method. Other relevant methods are splr_sparse_plus_constant and

---

```r
splr_sparse_plus_constant
```

*Add a constant to a splrMatrix object*

---

Description

Add a constant to a splrMatrix object

Usage

```r
splr_sparse_plus_constant(x, a)
```

Arguments

- **x**: sparse Matrix object
- **a**: scalar

Value

- new splrMatrix object x + a
**summary,graphMatch-method**

**Summary methods for graphMatch objects**

**Description**

Summary methods for graphMatch objects

**Usage**

```r
## S4 method for signature 'graphMatch'
summary(object, A = NULL, B = NULL, true_label = NULL, directed = NULL)
```

**Arguments**

- **object**: graphMatch object
- **A**: igraph or matrix-like object
- **B**: igraph or matrix-like object
- **true_label**: the true correspondence (if available)
- **directed**: whether to treat the graphs as directed (TRUE) or not directed (FALSE) default is NULL which will treat the graphs as directed if either adjacency matrix is not symmetric.

**Value**

summary returns the graph matching formula, and a summary of graph matching results including the number of matches, the number of correct matches (if the true correspondence is available), and common edges, missing edges, extra edges, common non-edges and the objective function value.

**Examples**

```r
set.seed(123)
graphs <- sample_correlated_gnp_pair(20, .9, .3)
A <- graphs$graph1
B <- graphs$graph2
match <- gm(A, B, 1:4, method = "percolation")
summary(match, A, B)
summary(match, A, B, true_label = 1:20) # also output the number of correct matches
```
Description

The Britain Transportation Network reflects the transportation connections in the UK, with five layers representing ferry, rail, metro, coach, and bus.

Usage

data(Transportation)

Format

A list of length 3, corresponding to the template graph, world graph, and candidate data frame with first column indicating template node ID’s and second column indicating world node ID’s. The template graph and world graph are stored as lists of five adjacency matrices, representing ferry, rail, metro, coach, and bus transportation connections respectively.

Details

The data consists of a smaller template graph with 53 nodes and 56 connections across five layers, a larger world graph with candidates of the template graph with 2075 nodes and 8368 connections, and a list of candidate matches for each template node, where the true correspondence is guaranteed to be among the candidates.

The template graph was constructed based on a random walk starting from a randomly chosen hub node, a node that has connections in all the layers. All edges in the template are common edges shared by two graphs, where 40%, 24.1%, 37.5%, 31.7% and 25.6% of edges in the world graph are in template for each layer. All graphs are unweighted, directed, and do not have self-loops.

References


See Also

The original Britain Transportation Network data is found here math.bu.edu/people/sussman/data/Transportation.rda. The template graph and world graph in the ‘Transportation’ data are induced subgraphs of the original graphs, keeping only the candidate nodes.
Examples

```r
tm <- Transportation[[1]]
cm <- Transportation[[2]]
candidate <- Transportation[[3]]
tn <- nrow(tm[[1]])
wn <- nrow(cm[[1]])
similarity <- with(candidate, Matrix::sparseMatrix(i = tem, j = wor, x = 1,
dims = c(tn,wn)))
```

Description

Methods to use `graphMatch` objects as operators on igraph and matrix-like objects.

Usage

```r
## S4 method for signature 'graphMatch,ANY'
x %*% y

## S4 method for signature 'ANY,graphMatch'
x %*% y

## S4 method for signature 'graphMatch,Matrix'
x %*% y

## S4 method for signature 'Matrix,graphMatch'
x %*% y

## S4 method for signature 'graphMatch,igraph'
x %*% y

## S4 method for signature 'igraph,graphMatch'
x %*% y
```

Arguments

- **x** Either `graphMatch` object or a matrix-like object
- **y** Either `graphMatch` object or a matrix-like object

Value

These methods return an object of the same type as the non-graphMatch object. If `m` is the match of `g1` to `g2` (both `igraph` objects), then `m` permuted so as to match with `g1`. Conversely, `g1` returns `g1` permuted so as to match with `g2`. 
Examples

```r
set.seed(123)
cgnp_pair <- sample_correlated_gnp_pair(n = 10, corr = 0.3, p = 0.5)
g1 <- cgnp_pair$graph1
g2 <- cgnp_pair$graph2

# match g1 & g2 using FW methodology with indefinite relaxation
match <- gm(A = g1, B = g2, seeds = 1:3, method = 'indefinite')

# permute the second graph according to the match result: P %*% g2 %*% P^T
match %*% g2 # return an igraph object
# equivalent to the matrix operation
match[] %*% g2[] %*% t(match[])

match %*% g2[] # return a matrix
# equivalent to:
P <- match[]
P %*% g2[] %*% Matrix::t(P)

# the inverse operations are performed via right multiplication
all(g1[] %*% match == t(P) %*% g1[] %*% P)
```
Index

* datasets
  C.Elegans, 3
  Enron, 9
  Transportation, 26

%*, ANY, graphMatch-method
  (%*, graphMatch, ANY-method), 27
%*, Matrix, graphMatch-method
  (%*, graphMatch, ANY-method), 27
%*, graphMatch, Matrix-method
  (%*, graphMatch, ANY-method), 27
%*, graphMatch, igraph-method
  (%*, graphMatch, ANY-method), 27
%*, igraph, graphMatch-method
  (%*, graphMatch, ANY-method), 27
%*, graphMatch, ANY-method, 27

as.graphMatch (graphMatch-class), 12
best_matches, 2

C.Elegans, 3
center_graph, 4
check_graph, 2, 5, 16
check_seeds, 6, 7, 14
check_sim, 7
check_single_graph (check_graph), 5
do_lap, 8

Enron, 9

gt_perm_mat, 10
gm, 2, 8, 11
graphMatch, 2, 12, 27
graphMatch (graphMatch-class), 12
graphMatch-class, 12, 12
graphMatch_methods, 13
graphMatch_operators, 13
graphMatch_plot, 13
graphMatch_summary, 13

image-methods, 18
init_start, 14
irlba, 24

largest_cc (largest_common_cc), 16
largest_common_cc, 16
pad, 17
plot, igraph, igraph-method, 17
plot, Matrix, Matrix-method
  (plot, igraph, igraph-method), 17
sample_correlated_gnp_pair, 19, 21, 22
sample_correlated_ieg_pair, 20
sample_correlated_rdpg_pair, 19, 22
sample_correlated_rdpg_pair
  (sample_correlated_ieg_pair), 20
sample_correlated_sbm_pair, 19, 21, 21
split_igraph, 23
splr (splrMatrix-class), 23
splr, Matrix, Matrix, Matrix-method
  (splrMatrix-class), 23
splr_sparse_plus_constant, 24, 24
splrMatrix, 4, 5
splrMatrix-class, 23
splrMatrix_method, 24
summary, graphMatch-method, 25

Transportation, 26