Package 'iGraphMatch'

January 27, 2021

Type Package

Title Tools Graph Matching

Version 1.0.1

Description Graph matching methods and analysis. The package
works for both 'igraph' objects and matrix objects. You provide the adjacency
matrices of two graphs and some other information you might know, choose the
graph matching method, and it returns the graph matching results. 'iGraphMatch'
also provides a bunch of useful functions you might need related to graph
matching.

Depends R (>= 3.3.1)

Imports clue (>= 0.3-54), Matrix (>= 1.2-11), igraph (>= 1.1.2),
irlba, methods, Rcpp

Suggests dplyr (>= 0.5.0), testthat (>= 2.0.0), knitr, rmarkdown

VignetteBuilder knitr

License GPL (>= 2)

LazyData TRUE

RoxygenNote 7.1.1

Language en-US

Encoding UTF-8

LinkingTo Rcpp

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 2021-01-27 16:00:02 UTC

R topics documented:

`bari_start` *Start matrix initialization*

Description

initialize the start matrix for graph matching iteration.

Usage

```
bari_start(nns, ns = 0, soft_seeds = NULL)

rds_sinkhorn_start(nns, ns = 0, soft_seeds = NULL, distribution = "runif")

rds_perm_bari_start(nns, ns = 0, soft_seeds = NULL, g = 1, is_splr = TRUE)

rds_from_sim_start(nns, ns = 0, soft_seeds = NULL, sim)
```

Arguments

nns	An integer. Number of non-seeds.
ns	An integer. Number of hard seeds.
soft_seeds	A vector, a matrix or a data frame. If there is no error in soft seeds, input can be a vector of soft seed indices in G_1. Or if there is error in soft seeds, 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. Note that if there are seeds in graphs, seeds should be put before non-seeds.
distribution	A character. Specify the distribution from which the random doubly stochastic matrix is sampled. Should input the name of the function for generating random deviates from that distribution.
g	A number. Specified in the range of [0, 1] to set weights to random permutation matrix and barycenter matrix.
is_splr	should we return a splr matrix? (default = TRUE)
sim	nns x nns non-negative matrix.

Value

`bari_start` returns a `nns-by-nns` matrix with 1's corresponding to the adaptive seeds and being bari-centered at other places.

`rds_sinkhorn_start` returns a `nns-by-nns` doubly stochastic matrix with 1's corresponding to adaptive seeds.

`rds_perm_bari` returns a `nns-by-nns` doubly stochastic matrix with 1's corresponding to adaptive seeds.

`rds_from_sim_start` returns a doubly stochastic Matrix given by sinkhorn algorithm applied to a matrix of iid log-normal with mu=sim. Note, this ignores soft seeds.

Examples

```
## Case without soft seeds
bari_start(3)

## Case with correct soft seeds and input is a vector
bari_start(nns=5, ns=3, soft_seeds=c(5, 7, 8))

## Case with erroneous soft seeds and the input is a matrix
```

```
bari_start(nns=5, soft_seeds=matrix(c(2, 4, 2, 3), nrow=2))

## Case without soft seeds
rds_sinkhorn_start(5)

## Case with soft seeds and the input is a data frame
rds_sinkhorn_start(nns=5,
   soft_seeds = as.data.frame(matrix(c(2, 4, 2, 3), nrow=2)),
   distribution = "rnorm")

## Case without soft seeds
rds_perm_bari_start(nns=5)

## Case with soft seeds and the input is a data frame
rds_perm_bari_start(nns=5, ns=0, soft_seeds=as.data.frame(matrix(c(2, 4, 2, 3), nrow=2)))

sim <- Matrix::rsparsematrix(10, 10, .4,
 rand.x = function(n) rep(1,n))
start_sparse <- rds_from_sim_start(10, sim = sim)
start_dense <- rds_from_sim_start(10, sim = as.matrix(sim))
```

`best_matches` *Choose best matches*

Description

Find a set of vertices pairs in the order of goodness of matching according to a specified measure.

Usage

```
best_matches(A, B, match, measure, num)
```

Arguments

`A`	A matrix, an 'igraph' object or a list of either. Adjacency matrix of G_1.
`B`	A matrix, an 'igraph' object or a list of either. Adjacency matrix of G_2.
`match`	Graph matching result see graph match methods.
`measure`	A character. Measure for computing goodness of matching.
`num`	An integer. Number of pairs of best matched vertices needed.

Value

`best_matches` returns a data frame with the indices of best matched vertices in G_1 named `A_best`, the indices of best matched vertices in G_2 named `B_best` and the values of measure for best matches.

Examples

```
cgnp_pair <- sample_correlated_gnp_pair(n = 50, corr =  0.3, p =  0.5)
g1 <- cgnp_pair$graph1
g2 <- cgnp_pair$graph2
seeds <- 1:50 <= 10
nonseeds <- !seeds
match <- graph_match_FW(g1, g2, seeds)

# Application: select best matched seeds from non seeds as new seeds, and do the
# graph matching iteratively to get higher matching accuracy
best_matches(A = g1, B = g2, match = match, measure = "row_perm_stat", num = 5)
```

C.Elegans *Chemical synapses and electrical synapses networks of roundworm*

Description

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. 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.

Usage

```
data(C.Elegans)
```

Format

An object of class `list` of length 2.

Examples

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

`center_graph` *Center adjacency matrix*

Description

Center the adjacency matrix, including center the adjacency matrix to entries equal to -1 or 1, center the adjacency matrix by using Universal Singular Value Thresholding.

Usage

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

Arguments

`A`	A matrix or 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.

Details

The options for scheme are

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

Value

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

Examples

```
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_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.
nv	An integer. Number of total vertices.
logical	An logical. TRUE indicates returns seeds in a vector of logicals where TRUE indicates the corresponding vertex is a seed. FALSE indicates returns 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)
```

`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 $\max_{v \in \Pi_n} \sum_{i=1}^{n} score_{i,v(i)}$ where Π_n denotes all permutations on n objects.

Usage

```
do_lap(score, method)
```

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-Volgenaut approach implemented in this package. lapmod use a version that exploits sparsity in the score matrix.

Value

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

Examples

```
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. 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.

Usage

```
data(Enron)
```

Format

An object of class `list` of length 2.

Examples

```
data(Enron)
g1 <- Enron[[1]]
g2 <- Enron[[2]]
```

get_perm *Get Permutation*

Description

Get an `m-by-n` permutation matrix according to the mapping correspondence.

Usage

```
get_perm(m, n, corr)
```

Arguments

`m`	An integer. Order of G_1.
`n`	An integer. Order of G_2.
`corr`	A matrix or a data frame. Matching correspondence with the first and second columns correspond to indices in G_1 and G_2 respectively.

Value

get_perm returns an m-by-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(4, 4, corr)

# submatrix is a permutation matrix: parts of graphs get matched
get_perm(5, 6, corr)
```

---

graph_match_convex *Frank-Wolfe Graph Matching Methods*

Description

Match two given graphs, returns a list of graph matching results, including matching correspondence vector of G_2 with respect to G_1, doubly stochastic matrix and permutation matrix.

Usage

```
graph_match_convex(
  A,
  B,
  seeds = NULL,
  similarity = NULL,
  start = "bari",
  max_iter = 100,
  tol = 1e-05,
  lap_method = NULL
)

graph_match_FW(
  A,
  B,
  seeds = NULL,
  similarity = NULL,
  start = "bari",
  max_iter = 20,
  lap_method = NULL
)

gm_indefinite(
```

```
  A,
  B,
  seeds = NULL,
  similarity = NULL,
  start = "bari",
  max_iter = 20,
  lap_method = NULL
)

graph_match_PATH(
  A,
  B,
  seeds = NULL,
  similarity = NULL,
  epsilon = 1,
  tol = 1e-05,
  max_iter = 20,
  lap_method = NULL
)
```

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-by-n matrix containing vertex similarities.
start	A matrix or a character. Any nns-by-nns matrix or character value like "bari" or "convex" to initialize the starting matrix.
max_iter	A number. Maximum number of replacing matches equals to max_iter times number of total vertices of G_1.
tol	A number. Tolerance of edge disagreements.
lap_method	Choice for lap method.
epsilon	A small number

Value

graph_match_FW, graph_match_convex and graph_match _PATH return a list of graph matching results, including the graph matching formula, a data frame containing the matching correspondence between G_1 and G_2 named corr_A and corr_B, the doubly stochastic matrix from the last iteration and the permutation matrix after projection, seeds and number of iterations.

References

M. Zaslavskiy, F. Bach and J. Vert (2009), *A Path following algorithm for the graph matching problem*. IEEE Trans Pattern Anal Mach Intell, pages 2227-2242.

Examples

```
cgnp_pair <- sample_correlated_gnp_pair(n = 10, corr =  0.3, p =  0.5)
g1 <- cgnp_pair$graph1
g2 <- cgnp_pair$graph2
# match G_1 & G_2 with no seeds
graph_match_FW(g1, g2)
seeds <- 1:10 <= 3

graph_match_convex(g1, g2, seeds)



# match G_1 & G_2 with some known node pairs as seeds
seeds <- 1:10 <= 3
graph_match_FW(g1, g2, seeds, start = "bari")

# match G_1 & G_2 with some incorrect seeds
hard_seeds <- matrix(c(4,6,5,4),2)
seeds <- rbind(as.matrix(check_seeds(seeds, nv = 10)$seeds),hard_seeds)
graph_match_FW(g1, g2, seeds, start = "convex")

 gp_list <- replicate(3, sample_correlated_gnp_pair(20, .3, .5), simplify = FALSE)
 A <- lapply(gp_list, function(gp)gp[[1]])
 B <- lapply(gp_list, function(gp)gp[[2]])
 match <- graph_match_FW(A, B, seeds = 1:10, start = "bari", max_iter = 20)
 match$corr

# match G_1 & G_2 using PATH algorithm
graph_match_PATH(g1, g2)
```

---

graph_match_ExpandWhenStuck

Percolation Graph Matching Methods

Description

Percolation Graph Matching Methods

Usage

```
graph_match_ExpandWhenStuck(A, B, seeds, similarity = NULL, r = 2)

graph_match_percolation(A, B, seeds, similarity = NULL, r = 2)
```

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-by-n matrix containing vertex similarities.
r	A number. Threshold of neighboring pair scores.

Value

graph_match_percolation and graph_match_ExpandWhenStuck returns a list of graph matching results, including the graph matching formula, a data frame containing the matching correspondence between G_1 and G_2 named corr_A and corr_B, seeds and the order of nodes getting matched.

References

E. Kazemi, S. H. Hassani, and M. Grossglauser (2015), *Growing a graph matching from a handful of seeds*. Proc. of the VLDB Endowment, 8(10):1010–1021.

L. Yartseva and M. Grossglauser (2013), *On the performance of percolation graph matching*. COSN, Boston, MA, USA, pages 119–130.

Examples

```
cgnp_pair <- sample_correlated_gnp_pair(n = 10, corr =  0.3, p =  0.5)
g1 <- cgnp_pair$graph1
g2 <- cgnp_pair$graph2
# match G_1 & G_2 using Expand When Stuck graph matching method
seeds <- 1:5
graph_match_ExpandWhenStuck(g1, g2, seeds, r = 2)

# match G_1 & G_2 using percolation graph matching method
graph_match_percolation(g1, g2, seeds, r = 2)
```

---

graph_match_IsoRank *Spectral Graph Matching Methods: IsoRank Algorithm*

Description

Spectral Graph Matching Methods: IsoRank Algorithm

Usage

```
graph_match_IsoRank(
  A,
  B,
  seeds = NULL,
  similarity,
```

```
  max_iter = 50,
  method = "greedy"
)
```

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-by-n matrix containing vertex similarities.
max_iter	A number. Maximum number of replacing matches equals to max_iter times number of total vertices of G_1.
method	A character. Choice of method to extract mapping from score matrix, including greedy method and the Hungarian algorithm.

Value

graph_match_IsoRank returns a list of graph matching results, including the graph matching formula, a data frame containing the matching correspondence between G_1 and G_2 named corr_A and corr_B and seeds. If choose the greedy method to extract mapping, the order of nodes getting matched will also be returned.

References

R. Singh, J. Xu, B. Berger (2008), *Global alignment of multiple protein interaction networks with application to functional orthology detection*. Proc Natl Acad Sci. USA, pages 12763-12768.

Examples

```
cgnp_pair <- sample_correlated_gnp_pair(n = 10, corr =  0.3, p =  0.5)
g1 <- cgnp_pair$graph1
g2 <- cgnp_pair$graph2
# match G_1 & G_2 using IsoRank algorithm
startm <- matrix(0, 10, 10)
diag(startm)[1:4] <- 1
GM_IsoRank <- graph_match_IsoRank(g1, g2, similarity = startm, method = "greedy")
```

graph_match_Umeyama *Spectral Graph Matching Methods: Umeyama Algorithm*

Description

Spectral Graph Matching Methods: Umeyama Algorithm

Usage

```
graph_match_Umeyama(A, B, seeds = NULL, similarity = NULL)
```

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-by-n matrix containing vertex similarities.

Value

graph_match_Umeyama returns a list of graph matching results, including the graph matching formula, a data frame containing the matching correspondence between G_1 and G_2 named corr_A and corr_B and seeds.

References

S. Umeyama (1988), *An eigendecomposition approach to weighted graph matching problems*. IEEE TPAMI. USA, pages 695-703.

Examples

```
# match G_1 & G_2 using Umeyama algorithm
G <- sample_correlated_gnp_pair(10, .9, .5)
g1 <- G$graph1
g2 <- G$graph2
startm <- matrix(0, 10, 10)
diag(startm)[1:4] <- 1
graph_match_Umeyama(g1, g2, similarity = startm)
```

`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, ...)
```

Arguments

`start`	A matrix or a character. Any `nns-by-nns` doubly stochastic matrix or start method like "bari", "convex" or "rds" to initialize the start matrix.
`nns`	An integer. Number of non-seeds.
`ns`	An integer. Number of seeds.
`soft_seeds`	A vector, a matrix or a data frame. If there is no error in the soft seeds, input can be a vector of soft seed indices in G_1. Or if there is error in soft seeds, input should be 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. Note that if there are seeds in graphs, seeds should be put before non-seeds.
`...`	Arguments passed to other start functions

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

```
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)

# initialize start matrix with soft seeds
init_start(start = "bari", nns = 5, ns = 3, soft_seeds = c(5, 7, 8))
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)
```

`innerproduct` *Matrix inner products*

Description

Matrix inner products

Usage

```
innerproduct(x, y)

## S4 method for signature 'splrMatrix,splrMatrix'
innerproduct(x, y)

## S4 method for signature 'splrMatrix,Matrix'
innerproduct(x, y)

## S4 method for signature 'Matrix,splrMatrix'
innerproduct(x, y)

## S4 method for signature 'matrix_list,matrix_list'
innerproduct(x, y)
```

Arguments

`x`	matrix like object
`y`	matrix like object

Value

inner product <x, y>

lapjv — *Solves the linear assignment problem using the Jonker-Vogenant algorithm*

Description

Find a set of vertices pairs in the order of goodness of matching according to a specified measure.

Usage

```
lapjv(cost, maximize = FALSE)
```

Arguments

cost	A non-negative matrix-like object that can be coerced to a matrix
maximize	If FALSE (default) then costs are minimized and if TRUE the costs are maximized

Details

The C++ code for this method is modified from code in the python lapjv package.

Value

The assignment of rows to columns as an integer vector

lapmod — *Solves the linear assignment problem using the LAPMOD algorithm*

Description

Find a set of vertices pairs in the order of goodness of matching according to a specified measure.

Usage

```
lapmod(cost, maximize = FALSE)
```

Arguments

cost	A non-negative CsparseMatrix object from the 'Matrix' package
maximize	If FALSE (default) then costs are minimized and if TRUE the costs are maximized

Details

The 'C++' code for this method is modified from code in the python lapjv package.

Value

The assignment of rows to columns as an integer vector

`largest_common_cc` *Find the largest common connected subgraph(LCCS)*

Description

Assume two aligned graphs, find the largest common connect subgraph of these two graphs, which is an induced connected subgraph of both graphs that has as many vertices as possible.

Usage

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

Arguments

`A`	A matrix or an 'igraph' object. Adjacency matrix of G_1.
`B`	A matrix or an 'igraph' object. Adjacency matrix of G_2.
`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 degree of vertices more than 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 conncect 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 connect subgraph with each vertex having a minimum degree of 3
lccs3 <- largest_common_cc(g1, g2, min_degree = 3)
```

matched_adjs *document*

Description

Return aligned versions of A and B according to a result of match method

Usage

```
matched_adjs(match, A, B)
```

Arguments

match	Result from a a graph matching method.
A	A matrix, 'igraph' object, or list of either. Likely used in the call for creating match.
B	A matrix, 'igraph' object, or list of either. Likely used in the call for creating match.

Value

A list of aligned graphs named A_m and B_m.

match_plot_igraph *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

```
match_plot_igraph(A, B, match, color = TRUE, linetype = TRUE, ...)

match_plot_matrix(
  A,
  B,
  match,
  col.regions = NULL,
  at = NULL,
  colorkey = NULL,
  ...
)
```

Arguments

A	First graph. For match_plot_igraph must be an 'igraph' object.
B	First graph. For match_plot_igraph must be an 'igraph' object.
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 linetypes 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, red indicates edges only in graph A and green represents edges only graph B. The corresponding linetypes are solid, short dash, and long dash.

The plots can be recreated from the output with the code
`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,lty`. match_plot_matrix returns the difference between the matched graphs.

Examples

```
set.seed(123)
graphs <- sample_correlated_gnp_pair(20, .5, .3)
A <- graphs$graph1
B <- graphs$graph2
res <- graph_match_percolation(A, B, 1:4)

match_plot_igraph(A, B, res)
match_plot_matrix(A, B, res)
```

`match_report` *Matching performance summary*

Description

Get a summary of the matching result and measures of the matching performance based on several evaluation metrics associated with nodes and edges of two graphs.

Usage

```
match_report(match, A, B, true_label = NULL, directed = NULL)

edge_match_info(corr, A, B, directed = NULL)
```

Arguments

`match`	Graph matching result see graph match methods.
`A`	A matrix or an 'igraph' object. Adjacency matrix of G_1.
`B`	A matrix or an 'igraph' object. Adjacency matrix of G_2.
`true_label`	A vector. NULL if the true correspondence between two graphs is unknown. A vector indicating the true correspondence in the second graph if the true correspondence is known.
`directed`	Whether the graphs should be treated as directed or undirected. NULL defaults to !isSymmetric(A).
`corr`	Correspondence data frame as given by match$corr

Details

For multilayered graphs information is given per layer. For weighted graphs the counts are based on non-zero entries. Equality of weights is not tested. If you want to ignore seeds in the edge match info you must remove them from corr/match$corr.

Value

`match_report` returns the match object evaluation metrics including number of matches, true matches, and a data frame with edge correctness information. `edge_match_info` returns this data frame with columns for number of common edges, missing edges, extra edges, and common non-edges, and Frobenius norm.

TODO

support weighted? loops? ...?

Examples

```
graphs <- sample_correlated_gnp_pair(10, .5, .3)
A <- graphs$graph1
B <- graphs$graph2
res <- graph_match_percolation(A, B, 1:4)
match_report(res, A, B)

gp_list <- replicate(3,
   sample_correlated_gnp_pair(100, .8, .3),
   simplify = FALSE)
A <- lapply(gp_list, function(gp)gp[[1]])
B <- lapply(gp_list, function(gp)gp[[2]])
corr <- data.frame(corr_A = 1:100, corr_B = 1:100)
edge_match_info(corr, A, B)
```

---

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.

row_cor *Measure functions*

Description

Measures for computing the goodness of matching for each vertex.

Usage

```
row_cor(g1, g2)

row_diff(g1, g2)

row_perm_stat(g1, g2, exact = TRUE)
```

Arguments

g1	A matrix or an 'igraph' object. Adjacency matrix of G_1.
g2	A matrix or an 'igraph' object. Adjacency matrix of G_2 after adjusting rows and columns according to the correlation of matching between two graphs.
exact	A logical. If g1 and g2 are binary, then set exact=TRUE, if g1 and g2 are weighted graphs, then set exact=FALSE.

Value

row_cor returns a vector, each element is 1 minus the row correlation value for the corresponding vertex.

row_diff returns a vector, each element is the row difference value for the corresponding vertex.

row_perm_stat returns a vector, each element is the row permutation statistics value for the corresponding vertex.

Examples

```
cgnp_pair <- sample_correlated_gnp_pair(n = 50, corr =  0.3,p =  0.5)
g1 <- cgnp_pair$graph1
g2 <- cgnp_pair$graph2
match <- graph_match_FW(g1, g2)
g2m <- g2[match$corr$corr_B, match$corr$corr_B]
g1 <- g1[]
row_cor(g1, g2m)
row_diff(g1, g2m)
row_perm_stat(g1, g2m)
```

rperm *Sample random permutation matrix*

Description

Sample an n-by-n random permutation matrix.

Usage

```
rperm(n)
```

Arguments

n	An integer. Dimension of the permutation matrix.

Value

rperm returns an n-by-n permutation matrix.

Examples

```
rperm(3)
```

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 rho and edge probability being p.

Usage

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

sample_correlated_gnp_pair_w_junk(
  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 open (0,1) interval.
p	A number. Edge probability between two vertices. It must be in open (0,1) interval.
permutation	A numeric vector,permute second graph.
...	Passed to sample_correlated_gnp_pair and sample_correlated_gnp_pair_w_junk.
ncore	An integer. Number of core vertices.

Value

sample_correlated_gnp_pair returns a list of two 'igraph' object, named graph1 and graph2, which are two graphs whose adjacency matrix entries correlated with rho.

sample_correlated_gnp_pair_w_junk returns a list of two 'igraph' object, named graph1 and graph2, which are two graphs whose adjacency matrix entries correlated with rho and with first ncore vertices being core vertices and the rest being junk vertices.

Examples

```
sample_correlated_gnp_pair(50, 0.3, 0.5)
sample_correlated_gnp_pair_w_junk(50, 0.3, 0.5, 40)
```

---

sample_correlated_ieg_pair

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

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

sample_correlated_rdpg(X, rho, nc = 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.
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.
rho	A number. The target Pearson correlation between the adjacency matrices of the generated graphs. It must be in open (0,1) interval.
nc	An integer. Number of core vertices.
...	Passed to sample_correlated_rdpg_pair.

Value

sample_correlated_ieg_pair returns two 'igraph' objects named graph1 and graph2.

sample_correlated_rdpg returns two 'igraph' objects named graph1 and graph2 that are sampled from random dot product graphs model.

Examples

```
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)

## 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(X,rho=0.5)
```

`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 `rho` and edge probability being `p`.

Usage

```
sample_correlated_sbm_pair(
  n,
  pref.matrix,
  block.sizes,
  rho,
  permutation = 1:n,
  ...
)

sample_correlated_sbm_pair_w_junk(
  n,
  pref.matrix,
  block.sizes,
  rho,
  core.block.sizes,
  permutation = 1:n,
  ...
)
```

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.
`rho`	A number. The target Pearson correlation between the adjacency matrices of the generated graphs. It must be in open (0,1) interval.
`permutation`	A numeric vector, permute second graph.
`...`	Passed to `sample_correlated_sbm_pair` and `sample_correlated_sbm_pair_w_junk`.
`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`.

Value

A list of two 'igraph' object, named graph1 and graph2.

Examples

```
pm <- cbind( c(.1, .001), c(.001, .05) )
sample_correlated_sbm_pair(1000, pref.matrix=pm, block.sizes=c(300,700), rho=0.5)
sample_correlated_sbm_pair_w_junk(1000, pref.matrix=pm, block.sizes=c(300,700), rho=0.5,
core.block.sizes=c(200,500))
```

---

split_igraph *Split an 'igraph' object into aligned graphs by attribute*

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 attributed.

Usage

```
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

Examples

```
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

```
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 `splr`.

splr_sparse_plus_constant *Add a constant to a splrMatrix object*

Description

Add a constant to a splrMatrix object

Usage

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

Arguments

x	splrMatrix object
a	scalar

Value

new splrMatrix object x + a

splr_to_sparse *Convert splr 'Matrix' to Sparse*

Description

Convert splr 'Matrix' to Sparse

Usage

```
splr_to_sparse(data)
```

Arguments

data	splrMatrix

Value

sparse Matrix equal to x + a

See Matrix.

Index