Package ‘CommKern’

May 7, 2026

Version 1.0.1

Title Network-Based Communities and Kernel Machine Methods

Description Analysis of network community objects with applications to neuroimaging data. There are two main components to this package. The first is the hierarchical multimodal spinglass (HMS) algorithm, which is a novel community detection algorithm specifically tailored to the unique issues within brain connectivity. The other is a suite of semiparametric kernel machine methods that allow for statistical inference to be performed to test for potential associations between these community structures and an outcome of interest (binary or continuous).

Depends R (>= 4.0.0)

License GPL (>= 2)

Encoding UTF-8

URL https://github.com/aljensen89/CommKern

Language en-us

LazyData true

Imports ggnewscale, ggplot2, gridExtra, Matrix, RColorBrewer, reshape2

Suggests knitr, matrixcalc, pheatmap, rmarkdown

RoxygenNote 7.2.1

VignetteBuilder knitr

NeedsCompilation no

Author Alexandria Jensen [aut, cre] (ORCID: <https://orcid.org/0000-0001-6337-9079>),
Peter DeWitt [ctb] (ORCID: <https://orcid.org/0000-0002-6391-0795>),
Debashis Ghosh [ths] (ORCID: <https://orcid.org/0000-0001-6618-1316>)

Maintainer Alexandria Jensen <alexandria.jensen@cuanschutz.edu>

Repository CRAN

Date/Publication 2022-09-23 10:20:06 UTC

Contents

adj_RI . . . 3
CommKern . . . 4
community_allegiance . . . 4
community_plot . . . 5
compute_modularity_matrix . . . 7
compute_multimodal_mod . . . 7
consensus_similarity . . . 8
count_pairs . . . 9
degree . . . 10
entropy . . . 11
ext_distance . . . 11
find_start_temp . . . 12
get_weights . . . 13
group_adj_perturb . . . 14
group_network_perturb . . . 15
ham_distance . . . 17
heatbath_multimodal . . . 18
hms . . . 19
kernel . . . 20
matrix_plot . . . 21
matrix_to_df . . . 22
NMI . . . 23
purity . . . 24
SBM_net . . . 25
score_cont_nonparam . . . 25
score_cont_semiparam . . . 27
score_log_nonparam . . . 29
score_log_semiparam . . . 30
simasd_array . . . 32
simasd_comm_df . . . 32
simasd_covars . . . 33
simasd_hamil_df . . . 33
simnet_df_perturb . . . 34
sort_pairs . . . 35
subset_matrix_to_df . . . 36
tr . . . 36
up_low . . . 37
zrand . . . 37

Index **39**

adj_RI *Adjusted Rand Index (ARI)*

Description

Description of the adjusted Rand Index function.

Usage

```
adj_RI(a, b)
```

Arguments

a	a vector of classifications; this must be a vector of characters, integers, numerics, or a factor, but not a list.
b	a vector of classifications

Details

In information theory, the Rand Index (also called the Rand Measure) is a measure of the similarity between two data clusterings or classifications. If N is the set of elements and X and Y are the partition of N into n subsets, then the Rand Index is composed of four subsets: (a) the number of pairs of elements in N that are in the same subset in in X and the same subset in Y; (b) the number of pairs of elements in N that are in different subsets in X and different subsets in Y; (c) the number of pairs of elements in N that are in the same subset in X but different subsets in Y; and (d) the number of pairs of elements in N that are in different subsets in X but the same subset in Y. The adjusted Rand Index is the corrected-for-chance version of the Rand Index, which establishes a baseline by using the expected similarity of all pairwise comparisons between clusterings specified by a random model. The ARI can yield negative results if the index is less than the expected index.

Value

a scalar with the adjusted Rand Index (ARI)

See Also

NMI, purity

Examples

```
set.seed(7)
x <- sample(x = rep(1:3, 4), 12)

set.seed(18)
y <- sample(x = rep(1:3, 4), 12)

adj_RI(x,y)
```

CommKern *CommKern*

Description

The CommKern package provides a streamlined implementation in the design and analysis of network community structures with specific applications to neuroimaging data. The hierarchical multimodal spinglass (HMS) algorithm has been developed as a novel community detection algorithm, while the semiparametric kernel machine methods allow for statistical inference to be performed to test for potential associations between these community structures and an outcome of interest, whether binary or continuous.

This package was part of Alexandria Jensen's Ph.D. dissertation which was overseen by her advisor Debashis Ghosh. Peter DeWitt provided extensive mentorship on the creation of this package.

community_allegiance *Community Allegiance*

Description

Description of the community allegiance function.

Usage

```
community_allegiance(comm_matrix)
```

Arguments

comm_matrix	a matrix whose first column is the node label/id and all subsequent columns are different partitions

Details

This function calculates the community allegiance of each node in a network. For node i, the stability of its allegiance to community A is calculated as the number of times where node i belongs to community A, divided by the total number of runs. This measure is bounded in [0,1], where higher values of stability indicate that a node belong to a single community across a greater number of runs.

The function returns a square matrix whose values are bounded in [0,1], where higher values in the off diagonal indicate that the two nodes belong to the same community over a higher proportion of partitions.

Value

a matrix whose values are bounded in [0,1], where higher values in the off diagonal indicate that the two nodes belong to the same community over a higher proportion of runs.

Examples

```
set.seed(7)
x <- sample(x = rep(1:3, 4), 12)

set.seed(18)
y <- sample(x = rep(1:3, 4), 12)

set.seed(3)
z <- sample(x = rep(1:3, 4), 12)

xyz_comms <- data.frame(id=seq(1:length(x)),x_comm=x,y_comm=y,z_comm=z)
xyz_alleg <- community_allegiance(xyz_comms)

xyz_melt <- reshape2::melt(xyz_alleg)

ggplot2::ggplot(data = xyz_melt) +
 ggplot2::theme_minimal() +
 ggplot2::aes(x = as.factor(Var1), y = as.factor(Var2), fill = value) +
 ggplot2::geom_tile() +
 ggplot2::xlab('Node') + ggplot2::ylab('Node') +
 ggplot2::ggtitle('Community Allegiance Example') +
 ggplot2::scale_fill_gradient2(
   low  = 'navy',
   high = 'goldenrod1',
   mid  = 'darkturquoise',
   midpoint = 0.5,
   limit = c(0, 1),
   space = 'Lab',
   name='')
```

`community_plot` *Communities by layer plot*

Description

Generate a graphical representation of the communities and layers.

Usage

```
community_plot(x, ...)
```

Arguments

`x`	a `spinglass_net` object created by `hms`
`...`	additional arguments from other methods

Details

This is an ancillary function that creates the plots seen in the manuscript, with a heatmap-style plot on the top panel, derived from a network adjacency matrix, and a community assignment plot on the bottom panel, separated by layer.

Value

a `gtable` object

See Also

`link{hms}`

Examples

```
data(SBM_net)


# plot with max of two layers
SBM_netcomm <- hms(
  input_net  = SBM_net,
  spins      = 4,
  alpha      = 0,
  coolfact   = 0.90,
  tol  = 0.05,
  max_layers = 2
  )

community_plot(SBM_netcomm)



# plot with three layers
# don't run automatically on CRAN; > 5 seconds
SBM_netcomm <- hms(
  input_net  = SBM_net,
  spins      = 4,
  alpha      = 0,
  coolfact   = 0.90,
  tol  = 0.05,
  max_layers = 3
  )

community_plot(SBM_netcomm)
```

compute_modularity_matrix

Compute modularity matrix

Description

Description of the compute modularity matrix function.

Usage

```
compute_modularity_matrix(net)
```

Arguments

net	a spinglass_net object (see matrix_to_df for more details)

Details

Calculates the modularity matrix, which is the difference between the observed adjacency matrix and the expected adjacency matrix (from a null model). This is only computed for the main component of network information, not accounting for the guidance. For neuroimaging application, this function would be computing the modularity matrix for the functional connectivity aspect of the network object. The function takes in a network object and returns the modularity matrix.

Value

mod_matrix

See Also

matrix_to_df

compute_multimodal_mod

Compute multimodal modularity matrix

Description

Description of the compute multimodal modularity matrix function.

Usage

```
compute_multimodal_mod(mod_matrix, net, communities, alpha)
```

Arguments

mod_matrix	the modularity matrix output from the compute_modularity_matrix function
net	a network object in list form (see the matrix_to_df function for more details)
communities	the vector of node assignments to communities
alpha	a double parameter balancing the use of the guidance matrix in modularity calculation

Details

Calculates the multimodal version of the modularity matrix, which is detailed in the accompanying manuscript as the following:

$$\sum_{i \neq j} M_{ij} \delta(C_i, C_j) - \alpha \sum_{i \neq j} S_{ij} \delta(C_i, C_j).$$

This function incorporates both the modularity matrix calculated from the compute_modularity_matrix function and adds the additional component of a guidance matrix. The alpha parameter controls the extent to which the guidance matrix influences the modularity, where alpha=0 means the function reverts to the typical modularity calculation and alpha > 0 allows for some influence of the guidance matrix. The guidance matrix will not penalize the modularity if two nodes are not connected within it; it will only decrease the modularity if the two nodes have guidance information. The function takes in a network object, the mod_matrix output from compute_modularity_matrix, a vector of communities, and a parameter alpha and returns the multimodal modularity matrix.

Value

multimodal modularity matrix

See Also

matrix_to_df, compute_modularity_matrix

---

consensus_similarity *Consensus Similarity*

Description

Description of the consensus similarity function.

Usage

```
consensus_similarity(comm_matrix)
```

Arguments

comm_matrix	a matrix whose columns are different partition and whose rows are nodes within a network

Details

This function identifies a single representative partition from a set of partitions that is the most similar to the all others. Here, similarity is taken to be the z-score of the Rand coefficient.

Value

the consensus partition determined by the maximum average pairwise similarity

Examples

```
set.seed(7183)
x <- sample(x = rep(1:3, 4), 12)

y <- sample(x = rep(1:3, 4), 12)

z <- sample(x = rep(1:3, 4), 12)

xyz_comms_mat <- matrix(c(x,y,z),nrow=length(x),ncol=3)
consensus_similarity(xyz_comms_mat)
```

---

count_pairs *Count pairs*

Description

Description of the count pairs function.

Usage

```
count_pairs(a, b, order)
```

Arguments

a	a vector of classifications
b	a vector of classifications
order	a vector of permutations (coming from the order() function in base R)

Details

A function to count pairs of integers or factors and identify the pair counts

Value

a list of five different vectors:

- `pair_nb`: a vector containing counts of nodes within all possible classification pairs from partitions a and b
- `pair_a`: a vector of the same length as pair_nb, specifying the order of classifications in pair_nb from partition a
- `pair_b`: a vector of the same length as pair_nb, specifying the order of classifications in pair_nb from partition b
- `a_nb`: a vector containing counts of nodes within each class for partition a
- `b_nb`: a vector containing counts of nodes within each class for partition b

---

`degree` *Node degree calculation*

Description

Description of the node degree calculation function.

Usage

```
degree(adj_matrix_func, adj_matrix_str, vertex_df)
```

Arguments

`adj_matrix_func`	the adjacency matrix for functional connectivity
`adj_matrix_str`	the adjacency matrix for structural connectivity
`vertex_df`	a data frame of node (or vertex) information

Details

This is an ancillary function that calculates the functional and structural degree of each network node using the functional and structural adjacency matrices, respectively.

Value

a data frame to be incorporated into the network object

entropy *Entropy*

Description

Description of the entropy function.

Usage

```
entropy(a, b)
```

Arguments

a	a vector of classifications; this must be a vector of characters, integers, numerics, or a factor, but not a list.
b	a vector of classifications

Details

A function to compute the empirical entropy for two vectors of classifications and the joint entropy

Value

a list of four objects:

- uv the joint entropy
- u the conditional entropy of partition a
- v the conditional entropy of partition b
- sort_pairs the output from the sort_pairs function

See Also

sort_pairs

ext_distance *Extrinsic evaluation distance matrix creation*

Description

Description of the extrinsic evaluation distance matrix creation function.

Usage

```
ext_distance(comm_df, variant = c("NMI", "adj_RI", "purity"))
```

Arguments

comm_df	a data frame whose columns are different partitions. All partitions must have the same set of nodes in order for this function to work and this data frame should exclude a node ID column for ease of computation.
variant	a string in ('NMI', 'Adj_RI', 'purity') that calculates different extrinsic cluster evaluation metrics.

Details

This function creates a distance matrix using the community output values from any community detection algorithm, such as the hierarchical multimodal spinglass algorithm. Because extrinsic evaluation metrics for clustering algorithms use the underlying idea of similarity, distance is calculated as (1-similarity). The use of distance ensures that the distance matrix will be positive and semi-definite, a requirement for its use in the kernel function.

Value

A m x m (m is the number of partitions) extrinsic evaluation distance matrix to be used as input for the kernel function

See Also

adj_RI, NMI, and purity

Examples

```
x <- c(2,2,3,1,3,1,3,3,2,2,1,1)
y <- c(3,3,2,1,1,1,1,2,2,3,2,3)
z <- c(1,1,2,3,2,3,2,1,1,2,3,3)

xyz_comms <- data.frame(x_comm = x, y_comm = y, z_comm = z)
ext_distance(xyz_comms, variant = 'NMI')
ext_distance(xyz_comms, variant = 'adj_RI')
ext_distance(xyz_comms, variant = 'purity')
```

---

find_start_temp *Starting temperature*

Description

Description of the starting temperature function.

Usage

```
find_start_temp(net, mod_matrix, spins, alpha, ts)
```

Arguments

net	a spinglass_net object
mod_matrix	mod_matrix
spins	spins
alpha	a double parameter balancing the use of the guidance matrix in modularity calculation
ts	the starting temperature for the search, set to 1 within the algorithm

Details

Within the spinglass algorithm, we would like to start from a temperature with at least 95 of all proposed spin changes accepted in 50 sweeps over the network. The function returns the temperature found.

Value

the starting temperature that meets the criteria specified above

---

get_weights *Simulated network edge weights*

Description

Description of the simulated network edge weights function.

Usage

```
get_weights(network_df, wcr, bcr, bfcr = NA, fuzzy_comms = NA)
```

Arguments

network_df	a data frame containing information about network nodes, their community assignment, and all node dyads, coming from simnet_df_perturb
wcr	within community edge weights, sampled from a beta distribution; for example, c(8,8) will ask for the within community edge weights to be sampled from a Beta(8,8) distribution
bcr	between community edge weights, sampled from a beta distribution; for example, c(1,8) will ask for the between community edge weights to be sampled from a Beta(1,8) distribution
bfcr	fuzzy community edge weights, sampled from a beta distribution; for example, c(4,8) will ask for the fuzzy community edge weights to be sampled from a Beta (4,8) distribution
fuzzy_comms	the communities for which their distinction is 'fuzzy,' or not as distinct; fuzzy communities tend to have higher between community edge weights; for example, c('comm_a','comm_c') will create a fuzzy distinction between communities a and c

Details

This is an ancillary function that creates a vector of edge weights sampled from Beta distributions. Within and between community edge weights are each sampled from a distinct Beta distribution. If 'fuzzy' communities wish to be created, a third Beta distribution is specified and the communities for which their distinction is 'fuzzy' also needs to be specified. This vector of edge weights is then passed to `group_network_perturb` to create the final simulated network object.

Value

a vector of edge weights associated with the node dyads from the network data frame

---

`group_adj_perturb` *Group adjacency matrices*

Description

Description of the simulated group adjacency matrices function.

Usage

```
group_adj_perturb(group_network_list, n_nets, n_nodes)
```

Arguments

`group_network_list`	the output from `group_network_perturb`, which is a list of data frames detailing nodes, community assignments of each node, and edge weights between each dyad of nodes
`n_nets`	the number of networks simulated
`n_nodes`	the number of nodes in each simulated network (will be the same across all networks)

Details

This function takes the output from the `group_network_perturb` function, which is a list of data frames summarizing each simulated network, and creates an array of adjacency matrices. These adjacency matrices can then be used as input to any community detection algorithm (such as the hierarchical multimodal spinglass algorithm, `hms`).

Value

an array of adjacency matrices of dimension (n_nets x n_nodes x n_nodes)

See Also

`group_network_perturb`, `hms`

Examples

```
# Example 1
sim_nofuzzy <-
  group_network_perturb(
    n_nodes = 45,
    n_comm = 3,
    n_nets = 3,
    perturb_prop = 0.1,
    wcr = c(8, 8),
    bcr = c(1.5, 8)
  )

nofuzzy_adj <-
  group_adj_perturb(sim_nofuzzy, n_nets = 3, n_nodes = 45)

if (require(pheatmap)) {
  pheatmap::pheatmap(
    nofuzzy_adj[1,,],
    treeheight_row = FALSE,
    treeheight_col = FALSE
  )
}

# Example 2
sim_fuzzy <-
  group_network_perturb(
    n_nodes = 45,
    n_comm = 3,
    n_nets = 3,
    perturb_prop = 0.1,
    wcr = c(8, 8),
    bcr = c(1.5, 8),
    bfcr = c(3, 8),
    fuzzy_comms = c('comm_b','comm_c')
  )

fuzzy_adj <-
  group_adj_perturb(sim_fuzzy, n_nets = 3, n_nodes = 45)

if (require(pheatmap)) {
  pheatmap::pheatmap(
    fuzzy_adj[2,,],
    treeheight_row = FALSE,
    treeheight_col = FALSE
  )
}
```

`group_network_perturb` *Simulated group networks*

Description

Description of the simulated group networks function.

Usage

```
group_network_perturb(
  n_nodes,
  n_comm,
  n_nets,
  perturb_prop,
  wcr,
  bcr,
  bfcr = NA,
  fuzzy_comms = NA
)
```

Arguments

n_nodes	the number of nodes in each simulated network (will be the same across all networks)
n_comm	the number of communities to be simulated in each network (will be the same across all networks)
n_nets	the number of networks to simulate
perturb_prop	the proportion of network nodes to randomly alter their community assignment within each network
wcr	within community edge weights, sampled from a beta distribution; for example, c(8,8) will ask for the within community edge weights to be sampled from a Beta(8,8) distribution
bcr	between community edge weights, sampled from a beta distribution; for example, c(1,8) will ask for the between community edge weights to be sampled from a Beta(1,8) distribution
bfcr	fuzzy community edge weights, sampled from a beta distribution; for example, c(4,8) will ask for the fuzzy community edge weights to be sampled from a Beta (4,8) distribution
fuzzy_comms	the communities for which their distinction is 'fuzzy,' or not as distinct; fuzzy communities tend to have higher between community edge weights; for example, c('comm_a','comm_c') will create a fuzzy distinction between communities a and c

Details

This function creates a list of simulated networks, of which each network is in a data.frame format, which describes describes the community assignment for each node in the network, and simulates the edge weights based on whether the node dyad is: (a) within the same community; (b) between different communities, or (c) between different communities, but designated as 'fuzzy' in their distinction from one another.

The function returns a list of data.frames detailing the nodes, node dyads, community assignments, and edge weights for all dyads in each simulated network.

Value

a list of network data.frames containing nodes, their community assignment, node dyads, and edge weights

Examples

```
sim_nofuzzy <-
  group_network_perturb(
    n_nodes = 45,
    n_comm = 3,
    n_nets = 3,
    perturb_prop = 0.1,
    wcr = c(8, 8),
    bcr = c(1.5, 8)
  )
head(sim_nofuzzy[[1]])

sim_fuzzy <-
  group_network_perturb(
    n_nodes = 45,
    n_comm = 3,
    n_nets = 3,
    perturb_prop = 0.1,
    wcr = c(8, 8),
    bcr = c(1.5, 8),
    bfcr = c(3, 8),
    fuzzy_comms = c('comm_b', 'comm_c')
  )
head(sim_fuzzy[[2]])
```

---

ham_distance *Hamiltonian distance matrix creation*

Description

Description of the Hamiltonian distance matrix creation function.

Usage

```
ham_distance(hamil_df)
```

Arguments

hamil_df	a data frame containing two columns, one for network ID and another containing Hamiltonian values

Details

This function creates a distance matrix using the Hamiltonian output values from a community detection algorithm that implements a Hamiltonian value, such as the hierarchical multimodal spinglass algorithm. To ensure a positive, semi-definite matrix (as required for the kernel function), the absolute difference between Hamiltonian values is calculated.

The function returns an m x m matrix (where m is the number of networks) to be used as input for the kernel function.

Value

the Hamiltonian distance matrix to be used as input for the kernel function

See Also

hms

Examples

```
hamil_df <- data.frame(id  = seq(1:8),
                       ham = c(-160.5375, -167.8426, -121.7128, -155.7245,
                               -113.9834, -112.5262, -117.9724, -171.374))

ham_distance(hamil_df)
```

---

heatbath_multimodal *Multimodal heatbath algorithm*

Description

Description of the multimodal heatbath algorithm function.

Usage

```
heatbath_multimodal(net, mod_matrix, spins, alpha, temp, max_sweeps)
```

Arguments

net	a hms_network object
mod_matrix	mod_matrix
spins	spins
alpha	a double parameter balancing the use of the guidance matrix in modularity calculation
temp	a double parameter found using the find_start_temp() function
max_sweeps	an integer parameter of the maximum number of sweeps allowed at each temperature

Details

This is one of the two workhorse functions for the algorithm. The heatbath algorithm selects a network node at random, calculates the multimodal modularity for the current configuration, and then switches its community assignment to each possible community. If the modularity of this iterated configuration is less than the current configuration, the new configuration is accepted and the algorithm moves on to the next randomly chosen node. If this is not the case, the node is moved to the new community assignment with some probability, which is a function of the current modularity value, the iterated value, and the system's temperature. Once the algorithm finishes with the randomly chosen node, this counts as a sweep. A new sweep occurs, with the same steps taken as above, until the sweep number maxes out (usually set to 50 to balance computation time with robustness).

Value

acceptance value of the algorithm for the given temperature

---

hms — *Hierarchical multimodal spinglass algorithm*

Description

Description of the hierarchical multimodal spinglass algorithm function.

Usage

```
hms(input_net, spins, alpha, coolfact, tol, max_layers)
```

Arguments

`input_net`	a `spinglass_net` object (see `matrix_to_df` for more details)
`spins`	an integer indicating the maximum number of spins, or communities, that can be used
`alpha`	a double parameter balancing the use of the guidance matrix in modularity calculation
`coolfact`	a double parameter that indicates how quickly (or slowly) to cool the heatbath algorithm, typically set to be 0.95-0.99
`tol`	a double parameter that indicates the tolerance level of accepting the proposed changes within a temperature; at the end of each sweep, the number of proposed changes to the partition is assessed to see if it exceeds a threshold determined as a function of tol and spins, typically set to be 0.01-0.05
`max_layers`	an integer parameter that specifies the maximum number of layers of communities within the network

Details

This is the main function of the algorithm. After running checks on the input parameters, the algorithm begins on the first layer of the network, finding the optimal configuration of nodes to communities using the heatbath algorithm. Once the community assignments have been finalized, the set of nodes within each of these communities is broken up and become their own subnetworks, on which the algorithm is applied again to get further subnetwork community assignments. This continues until the maximum number of layers is reached.

Value

a list of two components: comm_layers_tree, a dataframe whose first column is the node id and all subsequent columns are the partitioning of the nodes to communities across the number of pre-specified layers; and best_hamiltonian, a vector of the optimized Hamiltonian values for each run of the algorithm

See Also

matrix_to_df, community_plot

Examples

```
hms_object <-
  hms(input_net = SBM_net,
      spins = 4,
      alpha = 0,
      coolfact = 0.90,
      tol = 0.05,
      max_layers = 1)

str(hms_object)
str(hms_object$comm_layers_tree)
str(hms_object$net)

identical(SBM_net, hms_object$net)
hms_object$net$vertexes

community_plot(hms_object)
```

---

kernel *Distance-based kernel*

Description

Description of the distance-based kernel function

Usage

```
kernel(mat, rho)
```

Arguments

`mat`	a distance-based matrix
`rho`	a bandwidth/scaling parameter whose optimal value is solved for within the larger score function

Details

This is an ancillary function that is passed into the score function, used for calculating the distance-based kernel.

The function returns an m x m matrix (where m is the number of networks) to be used as input for the kernel function.

Value

the value of the kernel

`matrix_plot` *Functional and Structural Matrix Plot*

Description

Provide a graphical representation of the functional and structural matrices within a `spinglass_net` object.

Usage

```
matrix_plot(x, ...)
```

Arguments

`x`	a `spinglass_net` object
`...`	additional arguments from other methods

Value

a `gtable` object

Examples

```
data(SBM_net)

matrix_plot(SBM_net)
```

`matrix_to_df` *Convert matrices to dataframe list for network*

Description

Description of the convert matrices to data frame list for network function.

Usage

```
matrix_to_df(func_mat, str_mat)
```

Arguments

`func_mat`	a square, symmetric matrix to be used as the main input for the `hms` algorithm. For brain connectivity, this will be a representation of functional (e.g., BOLD) connectivity.
`str_mat`	a square, symmetric matrix to be used as the guidance input for the `hms` algorithm. For brain connectivity, this will be a representation of structural (e.g., white matter) connectivity.

Details

This is an ancillary function that creates a data frame list for the initial network. This is the form of the network used for the spinglass algorithm

Value

A list containing the functional matrix, structural matrix, a data frame of the functional edge weights, a data frame of the structural edge weights, and nodal information (functional degree, structural degree, community assignment, and label information)

Examples

```
# Using the example data SBM_net$func_matrix and SBM_net$str_mat
net <- matrix_to_df(SBM_net$func_mat, SBM_net$str_mat)
str(net)
identical(net, SBM_net)
```

NMI *Normalized mutual information (NMI)*

Description

Description of the normalized mutual information function.

Usage

```
NMI(a, b, variant = c("max", "min", "sqrt", "sum", "joint"))
```

Arguments

a	a vector of classifications; this must be a vector of characters, integers, numerics, or a factor, but not a list.
b	a vector of classifications
variant	a string in ('max', 'min', 'sqrt', 'sum', 'joint') that calculates different variants of the NMI. The default use is 'max'.

Details

In information theory, the mutual information (MI) of two random variables is a measure of the mutual dependence between two variables, or the quantification of the 'amount of information' obtained about one random variable by observing the other random variable. The normalization of the MI score scales the results between 0 (no mutual information) and 1 (perfect correlation). The five options for the variant - max, min, square root, sum, and joint - all relate to the denominator of the NMI = MI / D.

Value

a scalar with the normalized mutual information (NMI).

See Also

adj_RI, purity

Examples

```
x <- c(1, 3, 1, 2, 3, 3, 3, 2, 1, 2, 1, 2)
y <- c(1, 1, 2, 3, 2, 1, 3, 1, 2, 3, 3, 2)

NMI(x, y, variant = 'max')
NMI(x, y, variant = 'min')
NMI(x, y, variant = 'sqrt')
NMI(x, y, variant = 'sum')
NMI(x, y, variant = 'joint')

x <- c("A", "A", "A", "A", "B", "C", "A", "B", "B", "C")
```

```
y <- c("B", "A", "A", "A", "C", "C", "B", "C", "D", "D")
NMI(x, y, variant = 'max')
NMI(x, y, variant = 'min')
NMI(x, y, variant = 'sqrt')
NMI(x, y, variant = 'sum')
NMI(x, y, variant = 'joint')
```

---

purity *Purity*

Description

Description of the purity function.

Usage

```
purity(a, b)
```

Arguments

a	a vector of classifications; this must be a vector of characters, integers, numerics, or a factor, but not a list.
b	a vector of classifications

Details

In information theory, purity is an external evaluation criterion of cluster quality. It is the percent of the total number of objects (data points) that were classified in the range of [0,1]. Because we lack a ground truth partition, a harmonic mean is calculated, where we consider partition a to be the ground truth and then consider partition b to be the ground truth.

Value

a scalar with the harmonic mean of the purity

See Also

adj_RI, NMI

Examples

```
set.seed(7)
x <- sample(x = rep(1:3, 4), 12)

set.seed(18)
y <- sample(x = rep(1:3, 4), 12)

purity(x,y)
```

SBM_net — *Simulated functional and structural connectivity with nested hierarchical community structure*

Description

A dataset containing multimodal network information simulated to emulate functional and structural brain connectivity data with a nested hierarchical community structure. This dataset is a list containing five components in a format used as an input to the `hms` function. The components, and their associated variables, are as follows:

Usage

```
SBM_net
```

Format

A list containing five components:

func_edges a dataframe containing 1233 rows and 3 columns: func_start_node, func_end_node, and func_weight. This dataframe describes the pairwise functional edge weights between nodes.

str_edges a dataframe containing 453 rows and 3 columns: str_start_node, str_end_node, and str_weight. This dataframe describes the pairwise structural edge weights between nodes. There are fewer rows to this dataframe than func_edges as structural connectivity tends to be sparser than functional connectivity.

vertexes a dataframe containing 80 rows and 5 columns: node_id, node_label, func_degree, str_degree, and community. The degree of a node is the sum of all edge weights connected to the node. In this simulated network, node_label is left as NA but, for other networks, a specific label may be used to denote additional information about the node. The community variable is left blank but is used by the `hms` algorithm.

func_matrix an 80 x 80 matrix in the style of a network adjacency matrix. It contains the same information as func_edges, just in a wide, rather than long, format.

str_matrix an 80 x 80 matrix in the style of a network adjacency matrix. It contains the same information as str_edges, just in a wide, rather than long, format.

score_cont_nonparam — *Nonparametric score function for distance-based kernel and continuous outcome.*

Description

Description of the nonparametric score function for distance-based kernel function and continuous outcome.

Usage

```
score_cont_nonparam(outcome, dist_mat, grid_gran = 5000)
```

Arguments

`outcome`	a numeric vector containing the continuous outcome variable (in the same ID order as dist_mat)
`dist_mat`	a square distance matrix
`grid_gran`	a numeric value specifying the grid search length, preset to 5000

Details

This is the main function that calculates the p-value associated with a nonparametric kernel test of association between the kernel and continuous outcome variable. A null model (where the kernel is not associated with the outcome) is initially fit. Then, the variance of $Y_i|X_i$ is used as the basis for the score test,

$$S(\rho) = \frac{Q_\tau\left(\hat{\beta}_0, \rho\right) - \mu_Q}{\sigma_Q}.$$

However, because ρ disappears under the null hypothesis, we run a grid search over a range of values of ρ (the bounds of which were derived by Liu et al. in 2008). This grid search gets the upper bound for the score test's p-value. This function is tailored for the underlying model

$$y_i = h(z_i) + e_i,$$

where $h(\cdot)$ is the kernel function, z_i is a multidimensional array of variables, and y_i is a continuous outcome taking values in in the real numbers.

The function returns an numeric p-value for the kernel score test of association.

Value

the score function p-value

References

Liu D, Ghosh D, and Lin X (2008) "Estimation and testing for the effect of a genetic pathway on a disease outcome using logistic kernel machine regression via logistic mixed models." BMC Bioinformatics, 9(1), 292. ISSN 1471-2105. doi:10.1186/147121059292.

See Also

`hms`, `ext_distance`, `ham_distance` `score_log_semiparam` for semiparametric score function of distance-based kernel functions and binary outcome. `score_log_nonparam` for nonparametric score function of distance-based kernel functions and binary outcome. `score_cont_semiparam` for semiparametric score function of distance-based kernel function and continuous outcome.

Examples

```
data(simasd_hamil_df)
data(simasd_covars)

hamil_matrix <- ham_distance(simasd_hamil_df)


score_cont_nonparam(
  dist_mat = hamil_matrix,
  outcome = simasd_covars$verbal_IQ,
  grid_gran = 5000
)
```

`score_cont_semiparam` *Semiparametric score function for distance-based kernel and continuous outcome.*

Description

Description of the semiparametric score function for distance-based kernel function and continuous outcome.

Usage

```
score_cont_semiparam(outcome, covars, dist_mat, grid_gran = 5000)
```

Arguments

`outcome`	a numeric vector containing the continuous outcome variable (in the same ID order as dist_mat)
`covars`	a data frame containing the covariates to be modeled parametrically (should NOT include an ID variable)
`dist_mat`	a square distance matrix
`grid_gran`	a numeric value specifying the grid search length, preset to 5000

Details

This is the main function that calculates the p-value associated with a semiparametric kernel test of association between the kernel and continuous outcome variable. A null model (where the kernel is not associated with the outcome) is initially fit. Then, the variance of $Y_i|X_i$ is used as the basis for the score test,

$$S(\rho) = \frac{Q_\tau\left(\hat{\beta}_0, \rho\right) - \mu_Q}{\sigma_Q}.$$

However, because ρ disappears under the null hypothesis, we run a grid search over a range of values of ρ (the bounds of which were derived by Liu et al. in 2008). This grid search gets the upper bound for the score test's p-value. This function is tailored for the underlying model

$$y_i = x_i^T \beta + h(z_i) + e_i,$$

where $h(\cdot)$ is the kernel function, z_i is a multidimensional array of variables, x_i is a vector or matrix of covariates, β is a vector of regression coefficients, and y_i is a continuous outcome taking values in the real numbers.

Value

the score function p-value for the kernel score test of association.

References

Liu D, Ghosh D, and Lin X (2008) "Estimation and testing for the effect of a genetic pathway on a disease outcome using logistic kernel machine regression via logistic mixed models." BMC Bioinformatics, 9(1), 292. ISSN 1471-2105. doi:10.1186/147121059292.

See Also

hms, ext_distance, ham_distance score_log_semiparam for semiparametric score function of distance-based kernel functions and binary outcome. score_log_nonparam for nonparametric score function of distance-based kernel functions and binary outcome. score_cont_nonparam for nonparametric score function of distance-based kernel function and continuous outcome.

Examples

```
data(simasd_hamil_df)
data(simasd_covars)

hamil_matrix <- ham_distance(simasd_hamil_df)
covars_df <- simasd_covars[,3:4]


score_cont_semiparam(
  outcome   = simasd_covars$verbal_IQ,
  covars    = covars_df,
  dist_mat  = hamil_matrix,
  grid_gran = 5000
  )
```

`score_log_nonparam` *Nonparametric score function for distance-based kernel and binary outcome*

Description

Description of the nonparametric score function for distance-based kernel function and a binary outcome.

Usage

```
score_log_nonparam(outcome, dist_mat, grid_gran = 5000)
```

Arguments

`outcome`	a numeric vector containing the binary outcome variable, 0/1 (in the same ID order as dist_mat)
`dist_mat`	a square distance matrix
`grid_gran`	a numeric value specifying the grid search length, preset to 5000

Details

This is the main function that calculates the p-value associated with a nonparametric kernel test of association between the kernel and binary outcome variable. A null model (where the kernel is not associated with the outcome) is initially fit. Then, the variance of $Y_i|X_i$ is used as the basis for the score test,

$$S(\rho) = \frac{Q_\tau\left(\hat{\beta}_0, \rho\right) - \mu_Q}{\sigma_Q}.$$

However, because ρ disappears under the null hypothesis, we run a grid search over a range of values of ρ (the bounds of which were derived by Liu et al. in 2008). This grid search gets the upper bound for the score test's p-value. This function is tailored for the underlying model

$$y_i = h(z_i) + e_i,$$

where

$$h(\cdot)$$

is the kernel function, z_i is a multidimensional array of variables, and y_i is a binary outcome taking values in 0, 1.

The function returns an numeric p-value for the kernel score test of association.

Value

the score function p-value

References

Liu D, Ghosh D, and Lin X (2008) "Estimation and testing for the effect of a genetic pathway on a disease outcome using logistic kernel machine regression via logistic mixed models." BMC Bioinformatics, 9(1), 292. ISSN 1471-2105. doi:10.1186/147121059292.

See Also

hms, ext_distance, ham_distance score_log_semiparam for semiparametric score function of distance-based kernel functions and binary outcome. score_cont_nonparam for nonparametric score function of distance-based kernel function and continuous outcome. score_cont_semiparam for semiparametric score function of distance-based kernel function and continuous outcome.

Examples

```
data(simasd_hamil_df)
data(simasd_covars)

hamil_matrix <- ham_distance(simasd_hamil_df)


score_log_nonparam(
  outcome   = simasd_covars$dx_group,
  dist_mat  = hamil_matrix,
  grid_gran = 5000
  )
```

---

score_log_semiparam *Semiparametric score function for distance-based kernel*

Description

Description of the semiparametric score function for distance-based kernel function and binary outcome.

Usage

```
score_log_semiparam(outcome, covars, dist_mat, grid_gran = 5000)
```

Arguments

outcome	a numeric vector containing the binary outcome variable, 0/1 (in the same ID order as dist_mat)
covars	a dataframe containing the covariates to be modeled parametrically (should NOT include an ID variable)
dist_mat	a square distance matrix
grid_gran	a numeric value specifying the grid search length, preset to 5000

Details

This is the main function that calculates the p-value associated with a semiparametric kernel test of association between the kernel and binary outcome variable. A null model (where the kernel is not associated with the outcome) is initially fit. Then, the variance of $Y_i|X_i$ is used as the basis for the score test,

$$S(\rho) = \frac{Q_\tau\left(\hat{\beta}_0, \rho\right) - \mu_Q}{\sigma_Q}$$

. However, because ρ disappears under the null hypothesis, we run a grid search over a range of values of ρ (the bounds of which were derived by Liu et al. in 2008). This grid search gets the upper bound for the score test's p-value. This function is tailored for the underlying model

$$y_i = x_i^T \beta + h(z_i) + e_i,$$

where $h(\cdot)$ is the kernel function, z_i is a multidimensional array of variables, x_i is a vector or matrix of covariates, β is a vector of regression coefficients, and y_i is a binary outcome taking values in 0, 1.

The function returns an numeric p-value for the kernel score test of association.

Value

the score function p-value

References

Liu D, Ghosh D, and Lin X (2008) "Estimation and testing for the effect of a genetic pathway on a disease outcome using logistic kernel machine regression via logistic mixed models." BMC Bioinformatics, 9(1), 292. ISSN 1471-2105. doi:10.1186/147121059292.

See Also

hms, ext_distance, ham_distance score_log_nonparam for nonparametric score function of distance-based kernel functions and binary outcome. score_cont_nonparam for nonparametric score function of distance-based kernel function and continuous outcome. score_cont_semiparam for semiparametric score function of distance-based kernel function and continuous outcome.

Examples

```
data(simasd_hamil_df)
data(simasd_covars)

hamil_matrix <- ham_distance(simasd_hamil_df)
covars_df <- simasd_covars[,3:4]


score_log_semiparam(
  outcome   = simasd_covars$dx_group,
  covars    = covars_df,
  dist_mat  = hamil_matrix,
  grid_gran = 5000
```

```
)
```

`simasd_array` *Simulated Array*

Description

A dataset containing an array of simulated adjacency matrices. The dimensions of each matrix is 80 x 80, for a total of 49 simulated networks. This simulated array is the basis of the simasd_hamil_df and simasd_comm_df datasets and is complementary to the simasd_covars dataframe.

Usage

```
simasd_array
```

Format

An array of dimensions 49 x 80 x 80, denoting matrices for 49 simulated networks, with each network's matrix corresponding to an adjacency matrix for an 80 node network.

`simasd_comm_df` *Simulated partitions of nodes to communities from HMS algorithm*

Description

A dataset of partitions of nodes to communities from simulated group-level networks with community structures. This dataset is complementary to the simasd_covars dataset, which contains the demographic information related to this dataset. For more information on how these group-level networks were simulated, please refer to the example script titled "beta_simulation_data.set.R". The variables are as follows:

Usage

```
simasd_comm_df
```

Format

A dataframe with 80 rows and 49 columns, where rows correspond to nodes within the simulated networks and columns correspond to the subject ID.

simasd_covars — *Simulated demographics dataset modeled of a subset of the preprocessed ABIDE database*

Description

A dataset of demographics generated based on summary statistics for a subset of the ABIDE preprocessed database (http://preprocessed-connectomes-project.org/abide/). The variables are as follows:

Usage

```
simasd_covars
```

Format

A dataframe with 49 rows and 8 columns:

id a generic ID, an integer value

dx_group diagnostic group (0=control, 1=Autism Spectrum Disorder (ASD)

sex subject sex (0=male, 1=female)

age subject age in years

handedness subject handedness category, a factor with three level (0=right, 1=left, 2=ambidextrous)

fullscale_IQ fullscale IQ score, simulated as if administered from the Wechsler Abbreviated Scales of Intelligence (WASI), an integer value in (50,160)

verbal_IQ verbal IQ component, simulated as if administered from the Wechsler Abbreviated Scales of Intelligence (WASI), an integer value in (55,160)

nonverbal_IQ nonverbal IQ component, simulated as if administered from the Wechsler Abbreviated Scales of Intelligence (WASI), an integer value in (53,160)

simasd_hamil_df — *Simulated Hamiltonian values from HMS algorithm*

Description

A dataset of Hamiltonian values from simulated group-level networks with community structure. This dataset is complementary to the simasd_covars dataset, which contains the demographic information related to this dataset. For more information on how these group-level networks were simulated, please refer to the example script titled "beta_simulation_data.set.R". The variables are as follows:

Usage

```
simasd_hamil_df
```

Format

A dataframe with 49 rows and 2 columns:

id a generic ID, corresponding to the id variable in simasd_covars

hamil Hamiltonian value calculated from running the simulated network through the HMS algorithm, a numeric value

---

simnet_df_perturb *Simulated network data frame*

Description

Description of the simulated network data frame function.

Usage

```
simnet_df_perturb(n_nodes, n_comm, n_nets, perturb_prop)
```

Arguments

n_nodes	the number of nodes in each simulated network (will be the same across all networks)
n_comm	the number of communities to be simulated in each network (will be the same across all networks)
n_nets	the number of networks to simulate
perturb_prop	the proportion of network nodes to randomly alter their community assignment within each network

Details

This is an ancillary function that creates a list of data frames, of which each data frame describes the community assignment for each node in the network. These data frames are used as a starting point for the edge weights to be added between nodes (see group_network_perturb and get_weights for more information).

Value

a list of network data frames containing nodes, their community assignment, and node dyads

`sort_pairs` *Sort pairs*

Description

Description of the sort pairs function.

Usage

```
sort_pairs(a, b)
```

Arguments

a	a vector of classifications
b	a vector of classifications

Details

A function to sort pairs of integers or factors and identify the pairs

Value

a list of six objects used as the basis to calculate many cluster evaluation metrics, like NMI, ARI, and the Rand z-score.

- levelsa list of the classes within each of the partitions a and b
- n_ija vector containing counts of nodes within all possible classification pairs from partitions a and b
- n_i.a vector of the same length as pair_nb, specifying the order of classifications in pair_nb from partition a
- n_.ja vector of the same length as pair_nb, specifying the order of classifications in pair_nb from partition b
- pair_aa vector containing counts of nodes within each class for partition a
- pair_ba vector containing counts of nodes within each class for partition b

subset_matrix_to_df *Convert matrices to list of data frames for subnetworks*

Description

Description of the convert matrices to data frame list for subnetworks function.

Usage

```
subset_matrix_to_df(func_matrix, str_matrix)
```

Arguments

func_matrix	a square, symmetric matrix to be used as the main input for the hms algorithm. For brain connectivity, this will be a representation of functional (e.g., BOLD) connectivity.
str_matrix	a square, symmetric matrix to be used as the guidance input for the hms algorithm. For brain connectivity, this will be a representation of structural (e.g., white matter) connectivity.

Details

This is an ancillary function that creates a data frame list for the subnetworks created using the multimodal hierarchical spinglass algorithm.

Value

A list of data frame containing the functional matrix, structural matrix, a data frame of the functional edge weights, a data frame of the structural edge weights, and nodal information (functional degree, structural degree, community assignment, and label information)

tr *Trace*

Description

Return the trace of a square matrix

Usage

```
tr(x, ...)
```

Arguments

x	a square matrix
...	arguments passed to sum

Value

The trace, the sum of the diagonal elements, of the square matrix `x`

---

up_low — *Bounds of grid search function*

Description

Description of the bounds of grid search function.

Usage

```
up_low(dist_mat)
```

Arguments

`dist_mat`	a square distance matrix

Details

This ancillary function finds the upper and lower bounds of the grid search implemented in the kernel score test.

The function returns an m x m matrix (where m is the number of networks) to be used as input for the kernel function.

Value

a square matrix of the same dimensions of the input matrix, comprised of the sum square differences.

---

zrand — *Rand z-score*

Description

Description of the Rand z-score function.

Usage

```
zrand(part1, part2)
```

Arguments

`part1`	a partition of nodes to communities or clusters
`part2`	a partition of nodes to communities or clusters

Details

This is an ancillary function that calculates the Rand z-score between two partitions, which is used in the consensus similarity function

Value

the Rand z-score between two partitions

Index

∗ **datasets**
SBM_net, 25
simasd_array, 32
simasd_comm_df, 32
simasd_covars, 33
simasd_hamil_df, 33

adj_RI, 3, *12*, *23, 24*

CommKern, 4
community_allegiance, 4
community_plot, 5, *20*
compute_modularity_matrix, 7, *8*
compute_multimodal_mod, 7
consensus_similarity, 8
count_pairs, 9

degree, 10

entropy, 11
ext_distance, 11, *26*, *28*, *30, 31*

find_start_temp, 12

get_weights, 13, *34*
group_adj_perturb, 14
group_network_perturb, *14*, 15, *34*

ham_distance, 17, *26*, *28*, *30, 31*
heatbath_multimodal, 18
hms, *5*, *14*, *18*, 19, *22*, *25, 26*, *28*, *30, 31*, *36*

kernel, 20

matrix_plot, 21
matrix_to_df, *7, 8*, *19, 20*, 22

NMI, *3*, *12*, 23, *24*

purity, *3*, *12*, *23*, 24

SBM_net, 25
score_cont_nonparam, 25, *28*, *30, 31*
score_cont_semiparam, *26*, 27, *30, 31*
score_log_nonparam, *26*, *28*, 29, *31*
score_log_semiparam, *26*, *28*, *30*, 30
simasd_array, 32
simasd_comm_df, 32
simasd_covars, 33
simasd_hamil_df, 33
simnet_df_perturb, *13*, 34
sort_pairs, *11*, 35
subset_matrix_to_df, 36
sum, *36*

tr, 36

up_low, 37

zrand, 37