Package 'NIMAA'

April 11, 2022

Title Nominal Data Mining Analysis

Version 0.2.1

Description
Functions for nominal data mining based on bipartite graphs, which build a pipeline for analysis and missing values imputation. Methods are mainly from the paper: Jafari, Mohieddin, et al. (2021) <doi:10.1101/2021.03.18.436040>, some new ones are also included.

License GPL (>= 3)

URL https://github.com/jafarilab/NIMAA

BugReports https://github.com/jafarilab/NIMAA/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Suggests knitr, utils, rmarkdown, htmltools, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports plotly, tidyr, bipartite, crayon, dplyr, ggplot2, igraph, purrr, skimr, bnstruct, RColorBrewer, fpc, mice, missMDA, networkD3, scales, softImpute, tibble, tidytext, visNetwork, stats

VignetteBuilder knitr

Depends R (>= 3.5.0)

NeedsCompilation no

Author Mohieddin Jafari [aut, cre], Cheng Chen [aut]

Maintainer Mohieddin Jafari <mohieddin.jafari@helsinki.fi>

Repository CRAN

Date/Publication 2022-04-11 14:12:45 UTC

R topics documented:

`analyseNetwork` *General properties of the network*

Description

Generic function for network properties, allowing you to get a quick overview of the network topology.

Usage

```
analyseNetwork(graph)
```

Arguments

`graph`	An `igraph` object.

Details

The following measurements are calculated in one step using the igraph package to analyze the input graph object: the Degree, Betweenness, Closeness, Kleinberg's hub score and Eigenvector centrality of nodes (vertices), the Betweenness centrality of edges, number of nodes, edges and components, the edge density, global Eigenvector centrality value and global Kleinberg's hub centrality score.

Value

A list containing `vertices` and `edges` centrality values as well as `general_stats` for the whole network.

See Also

`vcount`, `ecount`, `edge_density`, `count_components`, `degree`,`betweenness`, `edge_betweenness`,`closeness`, `eigen_centrality`,`hub_score`.

Examples

```
# generate a toy graph
g1 <- igraph::make_graph(c(1, 2, 3, 4, 1, 3), directed = FALSE)
igraph::V(g1)$name <- c("n1", "n2", "n3", "n4")

# generate random graph according to the Erdos-Renyi model
g2 <- igraph::sample_gnm(10, 23)
igraph::V(g2)$name <- letters[1:10]

# run analyseNetwork
analyseNetwork(g1)
analyseNetwork(g2)
```

`beatAML` *beatAML*

Description

The Beat AML program produced a comprehensive dataset on acute myeloid leukemia (AML) that included genomic data (i.e., whole-exome sequencing and RNA sequencing), ex vivo drug response, and clinical data. We have included only the drug response data from this program in this package.

Usage

```
beatAML
```

Format

A tibble with three variables:

`inhibitor` names of inhibitors

`patient_id` anonymous patient ID

`median` the median of drug response at corresponding `inhibitor` and `patient_id`

Source

Tyner, Jeffrey W., et al.(2018), Functional Genomic Landscape of Acute Myeloid Leukaemia. Nature 562 (7728): 526–31. http://vizome.org/aml/

drugComb *drugComb*

Description

DrugComb is a web-based portal for storing and analyzing drug combination screening datasets, containing 739,964 drug combination experiments across 2320 cancer cell lines and 8397 drugs. Single drug screening was extracted and provided here as a subset of drug combination experiments.

Usage

```
drugComb
```

Format

A tibble with three variables:

`inhibitor` names of inhibitors

`cell_line` cell_line for each `inhibitor`

`median` the median of drug response at corresponding `inhibitor` and `cell_line`

Source

Zheng, S., Aldahdooh, J., Shadbahr, T., Wang, Y., Aldahdooh, D., Bao, J., Wang, W., Tang, J. (2021). DrugComb update: a more comprehensive drug sensitivity data repository and analysis portal, Nucleic Acids Research, 49(W1), W174–W184. https://academic.oup.com/nar/article/49/W1/W174/6290546

extractSubMatrix *Extract the non-missing submatrices from a given matrix.*

Description

This function arranges the input matrix and extracts the submatrices with non-missing values or with a specific proportion of missing values (except for the elements-max submatrix). The result is also shown as `plotly` figure.

Usage

```
extractSubMatrix(
  x,
  shape = "All",
  verbose = FALSE,
  palette = "Greys",
  row.vars = NULL,
```

```
  col.vars = NULL,
  bar = 1,
  plot_weight = FALSE,
  print_skim = FALSE
)
```

Arguments

x	A matrix.
shape	A string array indicating the shape of the submatrix, by default is "All", other options are "Square", "Rectangular_row", "Rectangular_col", "Rectangular_element_max".
verbose	A logical value, If TRUE, the plot is saved as the .png file in the working directory. By default, it is FALSE.
palette	A string or number. Color palette used for the visualization. By default, it is 'Blues'.
row.vars	A string, the name for the row variable.
col.vars	A string, the name for the column variable.
bar	A numeric value. The cut-off percentage, i.e., the proportion of non-missing values. By default, it is set to 1, indicating that no missing values are permitted in the submatrices. This argument is not applicable to the elements-max submatrix.
plot_weight	A logical value, If TRUE, then the function prints submatrices with weights, otherwise it prints the submatrices with all weights as 1.
print_skim	A logical value, If TRUE, then the function prints skim information in console. By default, it is FALSE.

Details

This function performs row- and column-wise preprocessing in order to extract the largest submtrices. The distinction is that the first employs the original input matrix (row-wise), whereas the second employs the transposed matrix (column-wise). Following that, this function performs a "three-step arrangement" on the matrix, the first step being row-by-row arrangement, the second step being column-by-column arrangement, and the third step being total rearranging. Then, using four strategies, namely "Square", "Rectangular row", "Rectangular col", and "Rectangular element max", this function finds the largest possible submatrix (with no missing values), outputs the result, and prints the visualization. "Square" denotes the square submatric with the same number of rows and columns. "Rectangular_row" indicates the submatrices with the most rows. "Rectangular_col" denotes the submatrices with the most columns. "Rectangular_element_max" indicates the submatrices with the most elements which is typically a rectangular submatrix.

Value

A matrix or a list of matrices with non-missing (bar = 1) or a few missing values inside. Also, a specific heat map plot is generated to visualize the topology of missing values and the submatrix sub-setting from the original incidence matrix. Additionally, the nestedness temperature is included to indicate whether the original incidence matrix should be divided into several incidence matrices beforehand.

See Also

arrange, arrange_if

Examples

```
# load part of the beatAML data
beatAML_data <- NIMAA::beatAML[1:10000,]

# convert to incidence matrix
beatAML_incidence_matrix <- nominalAsBinet(beatAML_data)

# extract submatrices with non-missing values
sub_matrices <- extractSubMatrix(beatAML_incidence_matrix, col.vars = "patient_id",
 row.vars = "inhibitor")
```

---

findCluster *Find clusters in projected unipartite networks*

Description

This function looks for the clusters in the projected unipartite networks of the bipartite network (the incidence matrix) that was given to it.

Usage

```
findCluster(
  inc_mat,
  part = 1,
  method = "all",
  normalization = TRUE,
  rm_weak_edges = TRUE,
  rm_method = "delete",
  threshold = "median",
  set_remaining_to_1 = TRUE,
  extra_feature = NULL,
  comparison = TRUE
)
```

Arguments

inc_mat	An incidence matrix.
part	An integer, 1 or 2, indicating which unipartite projection should be used. The default is 1.
method	A string array indicating the clustering methods. The defalut is "all", which means all available clustering methods in this function are utilized. Other options are combinations of "walktrap", "multi level", "infomap", "label propagation", "leading eigenvector", "spinglass", and "fast greedy".

`normalization` A logical value indicating whether edge weights should be normalized before the computation proceeds. The default is TRUE.

`rm_weak_edges` A logical value indicating whether weak edges should be removed before the computation proceeds. The default is TRUE.

`rm_method` A string indicating the weak edges removing method. If `rm_weak_edges` is False, then this argument is ignored. The default is `delete`, which means deleting weak edges from the network. The other option is `as_zero`, which sets the weak edges' weights to 0.

`threshold` A string indicating the weak edge threshold selection method. If `rm_weak_edges` is False, then this argument is ignored. By default, `median` is used. The other option is `keep_connected`, which prevents the network from being unconnected and removes edges in ascending order of weights.

`set_remaining_to_1` A logical value indicating whether the remaining edges' weight should be set to 1. The default is TRUE.

`extra_feature` A data frame object that shows the group membership of each node based on prior knowledge.

`comparison` A logical value indicating whether clustering methods should be compared to each other using internal measures of clustering, including modularity, average silluoutte width, and coverage. The default value is TRUE.

Details

This function performs optional preprocessing, such as normalization, on the input incidence matrix (bipartite network). The matrix is then used to perform bipartite network projection and optional preprocessing on one of the projected networks specified, such as removing edges with low weights (weak edges). Additionally, the user can specify the removal method, threshold value, or binarization of the weights. For the networks obtained after processing, this function implements some clustering methods in igraph such as "walktrap" and "infomap", to detect the communities within the network. Furthermore, if external features (prior knowledge) are provided, the function compares the clustering results obtained with the external features in terms of similarity as an external validation of clustering. Otherwise, several internal validation criteria such as modularity and coverage are only represented to compare the clustering results.

Value

A list containing the igraph object of the projected network, the clustering results of each method on the projected network separately, along with a comparison between them. The applied clustering arguments and the network's distance matrix are also included in this list for potential use in the next steps. In the case of weighted projected networks, the distance matrix is obtained by inverting the edge weights. The comparison of selected clustering methods is also presented as bar plots simultaneously.

Examples

```
# generate an incidence matrix
data <- matrix(c(1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0), nrow = 3)
colnames(data) <- letters[1:5]
```

```
rownames(data) <- LETTERS[1:3]

# run findCluster() to do clustering
cls <- findCluster(
  data,
  part = 1,
  method = "all",
  normalization = FALSE,
  rm_weak_edges = TRUE,
  comparison = TRUE
)
```

---

herbIngredient *herbIngredient*

Description

The second version of the TCMID database was used as one of the large datasets in the TCM herb field. TCMID includes additional ingredient-specific experimental data based on herbal mass spectrometry spectra. The herb and ingredient associations from this database are provided here.

Usage

```
herbIngredient
```

Format

A tibble with two variables:

`herb` Scientific Latin names of available herbs in TCM database

`pubchem_id` unique compound id (CID) of ingredients based on PubChem database for available `herb` in TCMID database

Source

Huang, L., Xie, D., Yu, Y., Liu, H., Shi, Y., Shi, T., & Wen, C. (2018). TCMID 2.0: a comprehensive resource for TCM. Nucleic acids research, 46(D1), D1117–D1120. https://academic.oup.com/nar/article/46/D1/D1117/4584630

NIMAA *NIMAA: A package for Nominal Data Mining Analysis.*

Description

The NIMAA package contains 13 main functions that together provide a pipeline for nominal data mining. This package also includes four datasets that can be used to perform various nominal data mining analyses.

`nominalAsBinet` *Convert nominal data to a bipartite network*

Description

This function converts nominal data, which is represented in a data frame format as an edge list, to a bipartite network in the incidence matrix format. Nominal data is typically a data frame with two (or three) columns representing two nominal variables labels that NIMAA considers as starting and ending nodes (labels) in an edge list (or with a numeric value for each pairwise relationship of labels). The elements in the incidence matrix are the binary or numeric values of the pairwise relationships.

Usage

```
nominalAsBinet(
  el,
  index_nominal = c(1, 2),
  index_numeric = 3,
  print_skim = FALSE
)
```

Arguments

`el`	A data frame or matrix object as an edge list.
`index_nominal`	A vector with two values represents the indices for the columns containing nominal variables. The first value indicates the row objects and the second value indicates the column objects in the incidence matrix output.
`index_numeric`	An integer, the index for numeric values. This is the value used to pick the column containing the numeric values corresponding to the pairwise relationship of nominal variable labels. This column is used for missing value investigation and imputation steps.
`print_skim`	A logical value, If `TRUE`, then the function prints `skim` information in console.

Value

The incidence matrix representing the corresponding bipartite network. If `print_skim` set to `TRUE`, a summary of the matrix is also provided.

See Also

`pivot_wider`, `column_to_rownames`

Examples

```
# generate a data frame with two nominal variables without numeric values
el1 <- data.frame(
  nominal_var_1 = c("d", "e", "e", "b", "a", "a"),
  nominal_var_2 = c("J", "N", "O", "R", "R", "L")
)

# generate a data frame with two nominal variables with numeric values
el2 <- data.frame(
  nominal_var_1 = c("d", "e", "e", "b", "a", "a"),
  nominal_var_2 = c("J", "N", "O", "R", "R", "L"),
  numeric_val = c(4, 5, 5, 8, 7, 7)
)

# run nominalAsBinet() to convert the edge list to the incidence matrix
inc_mat1 <- nominalAsBinet(el1)
inc_mat2 <- nominalAsBinet(el2)
```

---

plotBipartite *Plot the bipartite network and the corresponding projected networks*

Description

This function converts the incidence matrix into an igraph object and provides network visualization in multiple ways.

Usage

```
plotBipartite(
  inc_mat,
  part = 0,
  verbose = FALSE,
  vertex.label.display = FALSE,
  layout = layout.bipartite,
  vertex.shape = c("square", "circle"),
  vertix.color = c("steel blue", "orange"),
  vertex.label.cex = 0.3,
  vertex.size = 4,
  edge.width = 0.4,
  edge.color = "pink"
)
```

Arguments

inc_mat	A matrix, the incidence matrix of bipartite network.

part An integer value indicates whether the original bipartite network or its projection should be plotted. The default value of 0 represents the original bipartite network. Other possibilities are 1 and 2 for two projected networks.

verbose A logical value, if it is TRUE, the plot is saved as the .png file in the working directory. The default value is FALSE.

vertex.label.display
A logical value, if it is TRUE, then the label of each vertex is shown in the output graph. The default value is FALSE.

layout A function from igraph package. The default is layout.bipartite, so the nodes on the top side are the variables in rows, and the nodes on the bottom side are those in columns.

vertex.shape A string vector to define the shapes for two different sets of vertices. The default value is c("square","circle"), the first string value is for the nodes in rows, and the second one is for nodes in columns. If the part is not 0, then only the first string value is used.

vertix.color A string vector to define the colors for two different sets of vertices. The default value is c("steel blue","orange"), the first string value is for the nodes in rows, and the second one is for nodes in columns. If the part is not 0, then only the first string value is used.

vertex.label.cex
A numeric value used to define the size of vertex labels. The default value is 0.3.

vertex.size A numeric value used to define the size of vertex. The default value is 4.

edge.width A numeric value used to define the edge width. The default value is 0.1.

edge.color A string used to define the color of edges. The default value is "pink".

Value

An igraph network object with visualization.

Examples

```
# load part of the beatAML data
beatAML_data <- NIMAA::beatAML[1:10000,]

# convert to incidence matrix
beatAML_incidence_matrix <- nominalAsBinet(beatAML_data)

# plot with the vertex label showing
plotBipartite(inc_mat = beatAML_incidence_matrix, vertex.label.display = TRUE)
```

plotBipartiteInteractive *Use the incidence matrix to plot an interactive bipartite network*

Description

This function converts the input incidence matrix into a bipartite network, and generates a customized interactive bipartite network visualization.

Usage

```
plotBipartiteInteractive(inc_mat)
```

Arguments

inc_mat	A matrix, the incidence matrix of bipartite network.

Details

This function creates customized interactive visualization for bipartite networks. The user can enter a simple incidence matrix to generate a dynamic network with a bipartite network layout, in which two parts use different colors and shapes to represent nodes. This function relies on the visNetwork package.

Value

An visNetwork object for interactive figure.

See Also

visNetwork

Examples

```
# load part of the beatAML data
beatAML_data <- NIMAA::beatAML[1:10000,]

# convert to incidence matrix
beatAML_incidence_matrix <- nominalAsBinet(beatAML_data)

# plot with the interactive bipartite network

plotBipartiteInteractive(inc_mat = beatAML_incidence_matrix)
```

plotCluster *Plot the clusters in one projection of the bipartite network*

Description

This function makes an interactive network figure that shows nodes in which nodes belonging to the same cluster are colored the same and nodes belonging to other clusters are colored differently. This function has been customized to use the output of `findCluster`.

Usage

```
plotCluster(graph, cluster, ...)
```

Arguments

graph	An igraph object.
cluster	An igraph cluster object.
...	Pass to `forceNetwork`

Value

A networkD3 object.

See Also

`forceNetwork`

Examples

```
# generate an incidence matrix
data <- matrix(c(1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0), nrow = 3)
colnames(data) <- letters[1:5]
rownames(data) <- LETTERS[1:3]

# run findCluster() to do clustering
cls <- findCluster(
  data,
  part = 1,
  method = "all",
  normalization = FALSE,
  rm_weak_edges = TRUE,
  comparison = FALSE
)
# plot the cluster with Louvain method
plotCluster(graph = cls$graph, cluster = cls$louvain)
```

`plotIncMatrix` *Plot the incidence matrix.*

Description

This function converts a nominal data edge list to an incidence matrix and provides some information about that.

Usage

```
plotIncMatrix(
  x,
  index_nominal = c(1, 2),
  index_numeric = NULL,
  palette = "Blues",
  verbose = FALSE,
  plot_weight = FALSE,
  print_skim = FALSE
)
```

Arguments

`x`	A data frame containing at least 2 nominal columns and an optional numeric column.
`index_nominal`	A vector made up of two numbers that serve as nominal columns. The first value indicates the incidence matrix's row name, while the second value represents the incidence matrix's column name. It is c (1,2) by default, which implies that the first column in the x data frame is the incidence matrix's row, and the second column in the x data frame is the incidence matrix's column.
`index_numeric`	An integer, the index of a numeric variable. This is the value used to select the column that contains weight score values for pairwise relationships, and it is used to fill the elements of the incidence matrix. These values are also utilized for investigating missing data and predicting edges via imputation.
`palette`	A string or number. Color palette used for the heat map. By default, it sets to "Blues". (Find more option in the manual of `scale_fill_distiller()`).
`verbose`	A logical value, If it is set to `TRUE`, the plot is saved as the .png file in the working directory. By default, it is set to `FALSE`.
`plot_weight`	A logical value, If it is set to `TRUE`, the plot is displayed with the corresponding colors based on weight scores, otherwise the binary matrix image is displayed. By default, it is set to `FALSE`.
`print_skim`	A logical value, If `TRUE`, then the function prints `skim` information in console.

Details

This function mainly converts data in the form of edge list into matrix data. It also returns the incidence matrix object, the dimensions and proportion of missing values, as well as the matrix's image for visualization.

Value

An incidence matrix object and image with the dimension and missing value proportion.

See Also

nominalAsBinet

Examples

```
# load part of the beatAML data
beatAML_data <- NIMAA::beatAML[1:1000,]

# visualize the input dataset
beatAML_incidence_matrix <- plotIncMatrix(beatAML_data,
index_numeric= 3)
```

---

predictEdge *Edge prediction of weighted bipartite network.*

Description

This function utilizes several data imputation methods in order to predict the existence of a link between two nodes by imputing the edges' weight in a weighted bipartite network of nominal data.

Usage

```
predictEdge(inc_mat, method = c("svd", "median", "als", "CA"))
```

Arguments

inc_mat	An incidence matrix containing missing values (edge weights), represented by NAs.
method	A string or list of string. By default, it is set to this list: c("svd","median","als","CA"). Other available methods in MICE, knn, FAMD, PCA, and pmm, can be called to perform at a single step.

Details

This function performs a variety of numerical imputations according to the user's input, and returns a list of imputed data matrices based on each method separately, such as median which replaces the missing values with the median of each rows (observations), and knn which uses the k-Nearest Neighbour algorithm to impute missing values.

Value

A list of matrices with original and imputed values by different methods.

See Also

knn.impute, softImpute, imputeCA, imputeFAMD, imputePCA, mice.

Examples

```
# load part of the beatAML data
beatAML_data <- NIMAA::beatAML[1:10000,]

# convert to incidence matrix
beatAML_incidence_matrix <- nominalAsBinet(beatAML_data)

# predict the edges by imputation the weights
predictEdge(beatAML_incidence_matrix)
```

---

robertson *robertson*

Description

Charles Robertson's detailed collections provide as an amazing resource for long-term comparisons of numerous terrestrial plant and insect species collected in a single location. From 1884 through 1916, Robertson observed and collected 1429 insect species that visited 456 flower of plant species. He gathered this dataset in Macoupin County, Illinois, USA, within a 16-kilometer radius of Carlinville.

Usage

```
robertson
```

Format

A tibble with two variables:

`plant` Scientific Latin names of plants

`pollinator` Scientific Latin names of pollinators for the `plant`

Source

Marlin, J. C., & LaBerge, W. E. (2001). The native bee fauna of Carlinville, Illinois, revisited after 75 years: a case for persistence. Conservation Ecology, 5(1). http://www.ecologia.ib.usp.br/iwdb/html/robertson_1929.html

scoreCluster *Score the cluster analysis in a projected network based on additional internal measures.*

Description

This function provides additional internal cluster validity measures such as entropy and coverage. The concept of scoring is according to the weight fraction of all intra-cluster edges relative to the total weights of all edges in the graph. This function requires the community object, igraph object and distance matrix returned by `findCluster` to analyze.

Usage

```
scoreCluster(community, graph, dist_mat)
```

Arguments

community	An igraph community object.
graph	An igraph graph object.
dist_mat	A matrix containing the distance of nodes in the network. This matrix can be retrieved by the output of `findCluster` to analyze.

Value

A list containing internal cluster validity scores.

See Also

`cluster.stats`, `findCluster`

Examples

```
# load part of the beatAML data
beatAML_data <- NIMAA::beatAML[1:10000,]

# convert to incidence matrix
beatAML_incidence_matrix <- nominalAsBinet(beatAML_data)

# do clustering
cls <- findCluster(beatAML_incidence_matrix,
  part = 1, method = "infomap", normalization = FALSE,
  rm_weak_edges = TRUE, comparison = FALSE)

# get the scoring result
scoreCluster(community = cls$infomap, graph = cls$graph,
  dist_mat = cls$distance_matrix)
```

validateCluster *Validate the cluster analysis in a projected network based on additional external measures.*

Description

This function calculates the similarity of a given clustering method to the provided ground truth as external features (prior knowledge). This function provides external cluster validity measures including `corrected.rand` and jaccard similarity. This function requires the community object, igraph object and distance matrix returned by `findCluster` to analyze.

Usage

```
validateCluster(community, extra_feature, dist_mat)
```

Arguments

`community`	An igraph community object.
`extra_feature`	A data frame object that shows the group membership of each node based on prior knowledge.
`dist_mat`	A matrix containing the distance of nodes in the network. This matrix can be retrieved by the output of `findCluster` to analyze.

Value

A list containing the similarity measures for the clustering results and the ground truth represented as an external features, i.e., corrected Rand and Jaccard indices.

Examples

```
# load part of the beatAML data
beatAML_data <- NIMAA::beatAML[1:10000,]

# convert to incidence matrix
beatAML_incidence_matrix <- nominalAsBinet(beatAML_data)

# do clustering
cls <- findCluster(beatAML_incidence_matrix,
  part = 1, method = c('infomap','walktrap'),
  normalization = FALSE, rm_weak_edges = TRUE,
  comparison = FALSE)

# generate a random external_feature
external_feature <- data.frame(row.names = cls$infomap$names)
external_feature[,'membership'] <- paste('group',
sample(c(1,2,3,4), nrow(external_feature),
replace = TRUE))
```

```
# validate clusters using random external feature
validateCluster(community = cls$walktrap,
extra_feature = external_feature,
dist_mat = cls$distance_matrix)
```

`validateEdgePrediction` *Validate and compare edge prediction methods.*

Description

This function compares the imputation approaches for predicting edges using the clustering result of a submatrix with non-missing values as a benchmark. This function performs the same analysis as the `findCluster` function on every imputed incidence matrices independently. Then, using different similarity measures, all imputation approaches are compared to each other, revealing how edge prediction methods affects network communities (clusters). The best method should result in a higher degree of similarity (common node membership) to the non-missing submatrix as a benchmark.

Usage

```
validateEdgePrediction(imputation, refer_community, clustering_args)
```

Arguments

`imputation`	A list or a matrix containing the results of imputation method (s).
`refer_community`	An igraph community object obtained through `findCluster` using a given method.
`clustering_args`	A list indicating the clustering arguments values used in `findCluster` for a given method. This list is retrievable from the output of `findCluster`.

Value

A list containing the following indices: Jaccard similarity, Dice similarity coefficient, Rand index, Minkowski (inversed), and Fowlkes-Mallows index. The higher value indicates the greater similarity between the imputed dataset and the benchmark.

Examples

```
# load part of the beatAML data
beatAML_data <- NIMAA::beatAML[1:10000,]

# convert to incidence matrix
beatAML_incidence_matrix <- nominalAsBinet(beatAML_data)

# do clustering
cls <- findCluster(beatAML_incidence_matrix, part = 1)
```

```
# predict the edges by imputation the weights
imputed_beatAML <- predictEdge(beatAML_incidence_matrix)

# validate the edge prediction
validateEdgePrediction(imputation = imputed_beatAML,
refer_community = cls$fast_greedy,
clustering_args = cls$clustering_args
)
```

`visualClusterInBipartite` *Plot the bipartite graph with color coding for different clusters in both parts*

Description

The Sankey diagram is used to depict the connections between clusters within each part of the bipartite network. The display is also interactive, and by grouping nodes within each cluster as "summary" nodes, this function emphasizes how clusters of each part are connected together.

Usage

```
visualClusterInBipartite(
  data,
  community_left,
  community_right,
  name_left = "Left",
  name_right = "Right"
)
```

Arguments

`data`	A data frame or matrix object as an edge list.
`community_left`	An igraph community object, one projection of the bipartite network to be showed on the left side.
`community_right`	An igraph community object, the other projection of the bipartite network to be showed on the right side.
`name_left`	A string value, the name of left community.
`name_right`	A string value, the name of right community.

Value

A customized Sankey plot with a data frame containing the cluster pairwise relationship with the sum of weight values in the weighted bipartite network.

See Also

plot_ly

Examples

```
# load part of the beatAML data
beatAML_data <- NIMAA::beatAML[1:1000,]

# convert to incidence matrix
beatAML_incidence_matrix <- nominalAsBinet(beatAML_data)

# extract the Recetengular_element_max submatrix
sub_matrices <- extractSubMatrix(beatAML_incidence_matrix,
col.vars = "patient_id", row.vars = "inhibitor",
shape = c("Rectangular_element_max"))

# do clustering analysis
cls1 <- findCluster(sub_matrices$Rectangular_element_max,
part = 1, comparison = FALSE)

cls2 <- findCluster(sub_matrices$Rectangular_element_max,
part = 2, comparison = FALSE)

visualClusterInBipartite(data = beatAML_data,
community_left = cls2$leading_eigen,
community_right = cls1$fast_greedy,
name_left = 'patient_id',
name_right = 'inhibitor')
```

Index