Package 'NIMAA'

September 13, 2021

Title Nominal Data Mining Analysis

Version 0.1.0

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 7.1.1

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], Cheng Chen [aut, cre]

Maintainer Cheng Chen <cheng.chen@helsinki.fi>

Repository CRAN

Date/Publication 2021-09-13 08:00:05 UTC

R topics documented:

`analyseNetwork` *Analyze the network (graph)*

Description

This function analyzes the input graph object and returns several technical indicators.

Usage

```
analyseNetwork(graph)
```

Arguments

`graph`	A igraph graph object.

Details

This function mainly uses several methods in the igraph package to analyze the input igraph graph object, and summarize the indices as a result.

Value

A list containing various indicators. *vertices* is the number of nodes, *edges* is the number of edges, *general_stats* are other indicators.

See Also

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

Examples

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

# run analyseNetwork
analyseNetwork(g)
```

---

beatAML *beatAML*

Description

The Beat AML program publishes an extensive genomic, drug response, and clinical dataset on acute myeloid leukemia. We selected some data from it.

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

http://vizome.org/aml/

drugComb *drugComb*

Description

DrugComb collect datasets from several major drug combination studies, covering 437 923 drug combination experiments with 7 423 800 data points across 93 human cancer cell lines.

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`

For further details, see https://academic.oup.com/nar/article-abstract/47/W1/W43/5486743

Source

https://www.biorxiv.org/content/10.1101/2021.03.18.436040v3.full

el2IncMatrix *Convert Edge list to Incidence Matrix*

Description

This function will convert edge-list-format data (usually a dataframe concluding 2/3 columns stand for two parts and possible numerical relationship) to a matrix (columns and rows stand for two parts, values in the matrix are the 'relationship')

Usage

```
el2IncMatrix(el, index_nominal = c(1, 2), index_numeric = 3, print_skim = TRUE)
```

Arguments

el	A dataframe or matrix, edgelist data
index_nominal	A vector with two values. The indexes for the nominal columns (the first value indicating the rows objects and the second value indicating the column object).
index_numeric	An integer, the index for numeric values. (this is value for selecting the column which contains our numeric values and we change it to the matrix for missing value investigation and imputation).
print_skim	A Boolean value, If TRUE, then the funtion will print skim information in console.

Value

The incidence matrix

See Also

pivot_wider, column_to_rownames

Examples

```
# generate a edgelist
el <- data.frame(
  A = c(1, 2, 3, 4, 5, 6),
  B = c("a", "b", "c", "e", "f", "g"),
  m = c(1, 2, 1, 2, 1, 2)
)

# run el2IncMatrix() to convert edgelist to incidence matrix
inc_mat <- el2IncMatrix(el, print_skim = FALSE)
```

---

extractSubMatrix *Extract the percentage-cut-off non-missing sub-matrices in a given matrix.*

Description

This function arrange the input matrix, extract the sub-matrices without any missing value or with specific proportion of missing values (not for elements-max matrix). The result will also be showed 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 = TRUE
)
```

Arguments

x	A matrix including valid values and NAs.
shape	A string array indicating the shape of the output, by defalut is "All", other options are "Square", "Rectangular_row", "Rectangular_col", "Rectangular_element_max".
verbose	A Boolean value, If TRUE, the plot will be saved as the .png file in the working directory. By default is FALSE.
palette	A string or number. Color palette used for the heatmap. By default is 'Blues'. (Find the option in the ggplot2 manual 'Sequential, diverging and qualitative colour scales from ColorBrewer')
row.vars	A string, the name for the row variable.
col.vars	A string, the name for the column variable.
bar	A numeric value bigger than0 and less or equal to 0. The cut-off percentage, i.e., the proportion of non missing values. By default is 1, which means there is no missing value in sub-matrices. This argument is not applicable for elements-max sub-matrix.
plot_weight	A A Boolean value, If TRUE, then the funtion will print matrices with weights, otherwise it will print the matrices with all weights as 1.
print_skim	A Boolean value, If TRUE, then the funtion will print skim information in console.

Details

The extraction process has two types of data preprocessing, the difference is that the first one directly uses the original input matrix (row-wise), while the second one uses the transposed matrix (column-wise). After preprocessing, the matrix will be "three-step arrangement", the first step is row arranging, the second step is col arranging, and the third step is total rearranging. Then look for the largest possible matrix (with no missing values) in the four dimensions, output the result and print the visualization.

Value

A matrix or a list of matrices, with non(bar = 1) or few missing value inside.

See Also

arrange, arrange_if

Examples

```
# load some data from beatAML dataset
data <- NIMAA::beatAML[1:10000,]

# convert to incidence matrix
inc_mat <- NIMAA::el2IncMatrix(data, print_skim = FALSE)

# extract sub-matricess without missing value
sub_matrices <- extractSubMatrix(
inc_mat,
print_skim = FALSE,
col.vars = "patient_id",
row.vars = "inhibitor",
verbose = FALSE
)
```

`findCluster` *Find clusters in one-partite graph*

Description

This function will extract the clusters in one projection of the bipartite graph of the given incidence matrix.

Usage

```
findCluster(
  inc_mat,
  dim = 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`	A matrix including valid values and NAs.
`dim`	An integer, 1 or 2, indicating which one-partite projection should be used. Default is 1
`method`	A string array indicating the clustering methods. Defalut is "all" which means all clutering methods in this function will be used, other options are conbinations of "walktrap", "multi level", "infomap", "label propagation", "leading eigenvector", "spinglass", "fast greedy".

normalization	A logical, whether to normalize the weights. Default is TRUE.
rm_weak_edges	A logical, whether to remove the weak edges. Default is TRUE.
rm_method	A string indicating the weak edges removing method, if 'rm_weak_edges' is False, then this argument will be ignored. Default is 'delete', which means delete weak edges from graph, other option is 'as_zero', set the weak edges' weights to 0.
threshold	A string indicating the weak edges threshold selection method, if 'rm_weak_edges' is False, then this argument will be ignored.. Default is 'median', other option is 'keep_connected', removing edges in ascending order of weight until the last one that keep the graph connected.
set_remaining_to_1	A logical, whether to set the remaining edges' weight to 1. Default is TRUE.
extra_feature	A dataframe has only one column indicating the membership of each nodes (rownames).
comparison	A logical, whether to compare different clustering methods' result. Default is TRUE.

Details

This function will perform optional pre-processing on the input incidence matrix, such as normalization. Then use the matrix to perform bipartite graph projection, and perform optional pre-processing in one of the specified parts, such as removing edges with lower weights, that is, weak edges. The removal method and threshold selection can also be specified, and for the remaining You can choose to keep the original weight or set all of them to 1. For the graphs obtained after processing, implement some clustering methods in igraph to obtain the classification results. In addition, if there is an input of external features (prior knowledge), the function will also compare the clustering results obtained with external features regard similarity.

Value

A list containing the clustering results.

Examples

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

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

herbIngredient *herbIngredient*

Description

This data set used the second version of the TCMID database, as the largest dataset in TCM herbs field, which contains richer experimental data originating from ingredient-specific and herbal mass spectrometry spectra.

Usage

```
herbIngredient
```

Format

A tibble with two variables:

`herb` names of herbs

`pubchem_id` unique id of ingredients in pubchemn for `herb`

Source

https://www.frontiersin.org/articles/10.3389/fphar.2020.01319/full#h6

imputeMissingValue *Impute the missing value in the given matrix.*

Description

This function will call several data imputation methods, where the columns of the matrix are different objects, and the rows represent multiple observations.

Usage

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

Arguments

`inc_mat`	A matrix containing missing values, represented by NAs.
`method`	A string or list of string, can be one of them, or belong to the interpolation method suitable for numerical data in the MICE package.Default is a list, c('svd','median','als','CA'), other options could be in MICE or 'knn', 'FAMD', 'PCA', 'pmm'.

Details

First, this function will convert the column name and row name to avoid possible interpolation failures caused by the special characters of the column name and row name. Then it will perform a variety of numerical imputation according to the user's input, and return all the data that does not contain any missing data, a list of matrices. 'median' will replace the missing values with the median of each rows(observations), 'knn' is the method in package `bnstruct`, 'als' and 'svd' are methods from package `softImpute`, 'CA', 'PCA' and 'FAMD' are from package `missMDA`, others are from the famous `mice`.

Value

A list of matrices without missing values

See Also

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

Examples

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

# convert to incidence matrix
inc_mat <- el2IncMatrix(data, print_skim = FALSE)

# impute
imputeMissingValue(inc_mat)
```

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

Description

The NIMAA package provides 12 main functions that build a pipeline for nominal data mining.

---

plotBipartite *Plot the graph based on input data*

Description

This function converts the input incidence matrix into a bipartite graph, and uses the igraph package to draw an figure.

Usage

```
plotBipartite(
  inc_mat,
  dim = 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 graph.
`dim`	An integer value, chosen between plotting the original bipartite graph or its projection. Default is 0 which plot the original bipartite graph. other options: 1, 2
`verbose`	A logical, if TRUE, the plot is saved as the .png file in the working directory. Default is FALSE.
`vertex.label.display`	A Boolean var, if TRUE then show the label of each vertex. Default is FALSE.
`layout`	A function from igraph package, plot layout. Default is layout.bipartite, thus the nodes on the top side are the variables in rows, nodes on the bottom side are those in columns.
`vertex.shape`	A vector, shapes for two different sets of vertices. Default is c("square", "circle"), the first one is for the nodes in rows, the second one is for nodes in columns, if the `dim` is not 0, then only the first one will be used.
`vertix.color`	A vector, colors for two different sets of vertices. Default is c("steel blue", "orange"), the first one is for the nodes in rows, the second one is for nodes in columns, if the `dim` is not 0, then only the first one will be used..
`vertex.label.cex`	A numeric, vertex labels' size. Default is 0.3.
`vertex.size`	A numeric, vertex size. Default is 4.
`edge.width`	A numeric, edge width. Default is 0.1.
`edge.color`	A string, edge color. Default is pink.

Value

An igraph graph object.

Examples

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

# plot beatAML data and convert to incidence matrix
beatAML_incidence_matrix <- el2IncMatrix(beatAML_data, print_skim = FALSE)

# 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 graph figure

---

Description

This function converts the input incidence matrix into a bipartite graph, and uses the visNetwork package to draw an interactive figure.

Usage

```
plotBipartiteInteractive(inc_mat)
```

Arguments

inc_mat	A matrix, the incidence matrix of graph.

Details

This function realizes interactive visualization. The user can input a simple incidence matrix, and then get a dynamic graph with a bipartite graph layout, in which two parts use different colors and shapes to represent nodes.

Value

An visNetwork interactive figure, details in visNetwork

See Also

visNetwork

Examples

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

# plot beatAML data and convert to incidence matrix
beatAML_incidence_matrix <- el2IncMatrix(beatAML_data, print_skim = FALSE)

plotBipartiteInteractive(inc_mat = beatAML_incidence_matrix)
```

plotCluster *Plot the clusters in one projection of the bipartite graph.*

Description

This function will visualize the input igraph cluster objects and igraph graph objects with membership properties. Different memberships will be represented by dots of different colors.

Usage

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

Arguments

graph	A igraph graph object.
cluster	A igraph cluster object, usually got from findCluster
...	Pass to forceNetwork

Value

A networkD3 object, the plot of cluster.

See Also

forceNetwork

Examples

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

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

plotInput *Plot the given data with heatmap figure.*

Description

This function plot the input data in heatmap format. The row and column are two nominal variables. The color/value of heatmap could be the weights or logical value showing valid/missing connection.

Usage

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

Arguments

x	A a dataframe containing the missing values, at least 2 nominal columns.
index_nominal	A vector containing two numbers, which are the indexes for the nominal columns (the first value indicating the rows objects and the second value indicating the column object). By default is c(1,2), i.e., the first two columns are nomianl columns.
index_numeric	An integer, the index for numeric values. (this is value for selecting the column which contains our numeric values and we change it to the matrix for missing value investigation and imputation).
palette	A string or number. Color palette used for the heatmap. By default is 'Blues'. (Find the option in the manual of scale_fill_distiller()).
verbose	A Boolean value, If TRUE, the plot will be saved as the .png file in the working directory. By default is FALSE.
plot_weight	A Boolean value, If TRUE, the plot will have corresponding colors based on weights, otherwise plot the binary/logical matrix figure. By default is FALSE.
print_skim	A Boolean value, If TRUE, then the funtion will print skim information in console.

Details

This function mainly convert data in the form of edge list into matrix data and visualize it. In addition, it will also calculate some basic matrix properties and the proportion of missing data.

Value

A matrix, the incidence matrix got from input data.

See Also

el2IncMatrix

Examples

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

beatAML_incidence_matrix <- plotInput(beatAML_data,index_numeric=3)
```

---

robertson *robertson*

Description

The author listed 1429 animal species visiting flowers of 456 plant species that grew in a small area in soutWhwestern Illinois, USA. Marlin and LaBerge (2001) describe Robertson's methods.

Usage

```
robertson
```

Format

A tibble with two variables:

`plant` names of plants

`pollinator` names of pollinators for `plant`

Source

http://www.ecologia.ib.usp.br/iwdb/html/robertson_1929.html

scoreCluster *Score the clusters in one projection of the bipartite graph.*

Description

This function will use the community object, graph object and distance matrix to analyze, mainly using the cluster.stats function in the fpc package, in addition, it also calculates a 'coverage' indicator.

Usage

```
scoreCluster(community, graph, distance_matrix)
```

Arguments

community	An igraph community object.
graph	An igraph graph object.
distance_matrix	A matrix, the distance of graph, usually got from findCluster

Value

A list of various scores.

See Also

cluster.stats, findCluster

Examples

```
# load part of the beatAML data
data <- NIMAA::beatAML[1:1000,]
# convert to incidence matrix
inc_mat <- el2IncMatrix(data, print_skim = FALSE)

# run findCluster() to do clustering
cls <- findCluster(
  inc_mat,
  dim = 1,
  method = "infomap",
  normalization = FALSE,
  rm_weak_edges = TRUE,
  comparison = FALSE
)
# get the scoring result
scoreCluster(
  community = cls$infomap,
  graph = cls$graph,
  distance_matrix = cls$distance_matrix
)
```

validateCluster *Validate the accuracy of our clustering of the projection comparing to users' input.*

Description

This function will verify the similarity between different clustering methods and external features (prior knowledge), which is to measure the results of clustering methods.

Usage

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

Arguments

dist_mat	A matrix, the distance of graph, usually got from
extra_feature	A dataframe has only one column indicating the membership of each nodes (rownames).
community	An igraph community object.

Value

A list containing the measured differences between clustering methods and reference clustering, corrected rand index and Jaccard similarity.

Examples

```
# load part of the beatAML data
data <- NIMAA::beatAML
# convert to incidence matrix
inc_mat <- el2IncMatrix(data, print_skim = FALSE)

# run findCluster() to do clustering
cls <- findCluster(
inc_mat,
dim = 1,
method = c('infomap','walktrap'),
normalization = FALSE,
rm_weak_edges = TRUE,
comparison = FALSE)

# generate some 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))
```

```
# validation
validateCluster(dist_mat = cls$distance_matrix,
community = cls$walktrap,
extra_feature = external_feature)
```

`validateImputation` *Validate different imputation methods' result comparing to user's input.*

Description

This function will cluster the input imputation results of no missing data, and the clustering method will be exactly the same as the reference clustering result (prior knowledge) based on the input. The results of the two clusters were then analyzed to compare the effects of imputation.

Usage

```
validateImputation(imputation, refer_community, clustering_args)
```

Arguments

`imputation`	A list ro a matrix, the results of different imputation method(s).
`refer_community`	An igraph community object, usually got from `findCluster`.
`clustering_args`	A list indicating the clutering arguments used in `findCluster`, usually got from `findCluster`.

Value

A list containing the following indicators, Jaccard similarity/ Dice similarity coefficient/ Rand index/ Minkowski(inversed)/ Fowlkes-Mallows index

Examples

```
# load part of the beatAML data and get the incidence matrix
beatAML_data <- NIMAA::beatAML[1:10000,]
beatAML_incidence_matrix <- el2IncMatrix(beatAML_data, print_skim = FALSE)

# do clustering
cls <- findCluster(
beatAML_incidence_matrix, # the sub-matrix
dim = 1)

# impute
imputations <- imputeMissingValue(beatAML_incidence_matrix)

# validate the imputation
validation_of_imputation <- validateImputation(
```

```
imputation = imputations,
refer_community = cls$fast_greedy,
clustering_args = cls$clustering_args
)
```

`visualClusterInBipartite` *Plot the bipartite graph with color coding for different cluster in different projections.*

Description

This function will use sankey to draw two graphs, the left and right parts are two projections, and in each projection, a node corresponds to a group in the clustering result.

Usage

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

Arguments

`data`	A a dataframe containing the missing values, at least 2 nominal columns.
`community_left`	An igraph community object, one projection of the graph, will be showed on the left side.
`community_right`	An igraph community object, the other projection of the graph, will be showed on the right side.
`name_left`	A string, the name of left community.
`name_right`	A string, the name of right community.

Value

A Data frame containing plotting datas.

See Also

`plot_ly`

Examples

```
# load part of the beatAML data and get the incidence matrix
beatAML_data <- NIMAA::beatAML[1:1000,]
beatAML_incidence_matrix <- el2IncMatrix(beatAML_data)

# extract the sub-matrix
sub_matrices <- extractSubMatrix(
beatAML_incidence_matrix,
shape = c("Rectangular_element_max") # the shapes you want to extract
)

# do clustering
cls <- findCluster(
sub_matrices$Rectangular_element_max, # the sub-matrix
dim = 1
)

cls2 <- findCluster(
sub_matrices$Rectangular_element_max, # the sub-matrix
dim = 2
)

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

Index