Package 'miic'

October 13, 2022

Title Learning Causal or Non-Causal Graphical Models Using Information Theory

Version 1.5.3

Description We report an information-theoretic method which learns a large class of causal or non-causal graphical models from purely observational data, while including the effects of unobserved latent variables, commonly found in many datasets. Starting from a complete graph, the method iteratively removes dispensable edges, by uncovering significant information contributions from indirect paths, and assesses edge-specific confidences from randomization of available data. The remaining edges are then oriented based on the signature of causality in observational data. This approach can be applied on a wide range of datasets and provide new biological insights on regulatory networks from single cell expression data, genomic alterations during tumor development and co-evolving residues in protein structures. For more information you can refer to:
Cabeli et al. PLoS Comp. Bio. 2020 <doi:10.1371/journal.pcbi.1007866>,
Verny et al. PLoS Comp. Bio. 2017 <doi:10.1371/journal.pcbi.1005662>.

License GPL (>= 2)

URL https://github.com/miicTeam/miic_R_package

BugReports https://github.com/miicTeam/miic_R_package/issues

Imports ppcor, Rcpp, scales, stats,

Suggests igraph, grDevices, ggplot2 (>= 3.3.0), gridExtra

LinkingTo Rcpp

SystemRequirements C++14

LazyData true

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation yes

Author Vincent Cabeli [aut, cre],
Honghao Li [aut],
Marcel Ribeiro Dantas [aut],

Nadir Sella [aut],
Louis Verny [aut],
Severine Affeldt [aut],
Hervé Isambert [aut]

Maintainer Vincent Cabeli <vincent.cabeli@curie.fr>

Repository CRAN

Date/Publication 2020-10-13 23:50:11 UTC

R topics documented:

`cosmicCancer` *Genomic and ploidy alterations in breast tumors*

Description

The dataset contains 807 samples without predisposing Brca1/2 germline mutations and includes 204 somatic mutations (from whole exome sequencing) and expression level information for 91 genes.

Usage

```
data(cosmicCancer)
```

Format

A data.frame object.

References

Forbes SA, Beare D, Gunasekaran P, Leung K, Bindal N, et al. (2015) Nucleic Acids Res 43:D805–D811. (PubMed link)

cosmicCancer_stateOrder *Genomic and ploidy alterations in breast tumors*

Description

The dataset contains 807 samples without predisposing Brca1/2 germline mutations and includes 204 somatic mutations (from whole exome sequencing) and expression level information for 91 genes, category order file.

Usage

```
data(cosmicCancer_stateOrder)
```

Format

A data.frame object.

References

Forbes SA, Beare D, Gunasekaran P, Leung K, Bindal N, et al. (2015) Nucleic Acids Res 43:D805–D811. (PubMed link)

discretizeMDL *Discretize a real valued distribution*

Description

This function performs minimum description length (MDL)-optimal histogram density estimation as described in Kontkanen and Myllymäki (2007) and returns the cutpoints found to give the best model according to the MDL principle.

Usage

```
discretizeMDL(x = NULL, max_bins = 20)
```

Arguments

x	[a vector] A vector that contains the distribution to be discretized.
max_bins	[an int] The maximum number of bins allowed by the algorithm.

Value

A list containing the cutpoints of the best discretization.

References

- Kontkanen P, Myllymäki P. MDL histogram density estimation. Artificial Intelligence and Statistics 2007 Mar 11 (pp. 219-226).

Examples

```
library(miic)
# Bimodal normal distribution
N <- 300
modes <- sample(1:2, size = N, replace = TRUE)
x <- as.numeric(modes == 1) * rnorm(N, mean = 0, sd = 1) +
     as.numeric(modes == 2) * rnorm(N, mean = 5, sd = 2)
MDL_disc <- discretizeMDL(x)
hist(x, breaks = MDL_disc$cutpoints)

N <- 2000
modes <- sample(1:2, size = N, replace = TRUE)
x <- as.numeric(modes == 1) * rnorm(N, mean = 0, sd = 1) +
     as.numeric(modes == 2) * rnorm(N, mean = 5, sd = 2)
MDL_disc <- discretizeMDL(x)
hist(x, breaks = MDL_disc$cutpoints)
```

`discretizeMutual` *Iterative dynamic programming for (conditional) mutual information through optimized discretization.*

Description

This function chooses cutpoints in the input distributions by maximizing the mutual information minus a complexity cost (computed as BIC or with the Normalized Maximum Likelihood). The (conditional) mutual information computed on the optimized discretized distributions effectively approaches the mutual information computed on the original continuous variables.

Usage

```
discretizeMutual(
  X,
  Y,
  matrix_u = NULL,
  maxbins = NULL,
  cplx = "nml",
  n_eff = NULL,
  sample_weights = NULL,
  is_discrete = NULL,
  plot = TRUE
)
```

Arguments

X	[a vector] A vector that contains the observational data of the first variable.
Y	[a vector] A vector that contains the observational data of the second variable.
matrix_u	[a numeric matrix] The matrix with the observations of as many columns as conditioning variables.
maxbins	[an int] The maximum number of bins desired in the discretization. A lower number makes the computation faster, a higher number allows finer discretization (by default : 5 * cubic root of N).
cplx	[a string] The complexity used in the dynamic programming. Either "mdl" for Minimum description Length or "nml" for Normalized Maximum Likelihood, which is less costly in the finite sample case and will allow more bins than mdl.
n_eff	[an int] The number of effective samples. When there is significant autocorrelation in the samples you may want to specify a number of effective samples that is lower than the number of points in the distribution.
sample_weights	[a vector of floats] Individual weights for each sample, used for the same reason as the effective sample number but with individual precision.
is_discrete	[a vector of booleans] Specify if each variable is to be treated as discrete (TRUE) or continuous (FALSE) in a logical vector of length ncol(matrix_u) + 2, in the order [X, Y, U1, U2...]. By default, factors and character vectors are treated as discrete, and numerical vectors as continuous.
plot	[a boolean] Specify if the XY joint space with discretization scheme is to be plotted or not (requires ggplot2 and gridExtra).

Details

For a pair of variables X and Y, the algorithm will in turn choose cutpoints on X then on Y, maximizing $I(X_d; Y_d) - cplx(X_d; Y_d)$ where $cplx(X_d; Y_d)$ is the complexity cost of the considered discretizations of X and Y (see Affeldt 2016 and Cabeli 2020). When the value $I(X_d; Y_d)$ is stable between two iterations the discretization scheme of X_d and Y_d is returned as well as $I(X_d; Y_d)$ and $I(X_d; Y_d) - cplx(X_d; Y_d)$.

With a set of conditioning variables U, the discretization scheme maximizes each term of the sum $I(X; Y|U) \sim 0.5 * (I(X_d; Y_d, U_d) - I(X_d; U_d) + I(Y_d; X_d, U_d) - I(Y_d; U_d))$.

Discrete variables can be passed as factors and will be used "as is" to maximize each term.

Value

A list that contains :

- two vectors containing the cutpoints for each variable : *cutpoints1* corresponds to /emphmyDist1, /emphcutpoints2 corresponds to /emphmyDist2.
- *niterations* is the number of iterations performed before convergence of the (C)MI estimation.
- *iterationN*, lists contatining the cutpoint vectors for each iteration.
- *info* and *infok*, the estimated (C)MI value and (C)MI minus the complexity cost.
- if $emphplot == TRUE, a plot object (requires ggplot2 and gridExtra).

References

- Verny et al., *PLoS Comp. Bio. 2017.* https://doi.org/10.1371/journal.pcbi.1005662
- Cabeli et al., *PLoS Comp. Bio. 2020.* https://doi.org/10.1371/journal.pcbi.1007866
- Affeldt et al., *Bioinformatics 2016*

Examples

```
library(miic)
N <- 1000
# Dependence, conditional independence : X <- Z -> Y
Z <- runif(N)
X <- Z * 2 + rnorm(N, sd = 0.2)
Y <- Z * 2 + rnorm(N, sd = 0.2)
res <- discretizeMutual(X, Y, plot = FALSE)
message("I(X;Y) = ", res$info)
res <- discretizeMutual(X, Y, matrix_u = matrix(Z, ncol = 1), plot = FALSE)
message("I(X;Y|Z) = ", res$info)


# Conditional independence with categorical conditioning variable : X <- Z -> Y
Z <- sample(1:3, N, replace = TRUE)
X <- -as.numeric(Z == 1) + as.numeric(Z == 2) + 0.2 * rnorm(N)
Y <- as.numeric(Z == 1) + as.numeric(Z == 2) + 0.2 * rnorm(N)
res <- miic::discretizeMutual(X, Y, cplx = "nml")
message("I(X;Y) = ", res$info)
res <- miic::discretizeMutual(X, Y, matrix(Z, ncol = 1), is_discrete = c(FALSE, FALSE, TRUE))
message("I(X;Y|Z) = ", res$info)


# Independence, conditional dependence : X -> Z <- Y
X <- runif(N)
Y <- runif(N)
Z <- X + Y + rnorm(N, sd = 0.1)
res <- discretizeMutual(X, Y, plot = TRUE)
message("I(X;Y) = ", res$info)
res <- discretizeMutual(X, Y, matrix_u = matrix(Z, ncol = 1), plot = TRUE)
message("I(X;Y|Z) = ", res$info)
```

`getIgraph` *Igraph plotting function for miic*

Description

This functions returns an igraph object built from the result returned by `miic`.

Usage

```
getIgraph(miic.res)
```

Arguments

`miic.res`	[a miic graph object] The graph object returned by the miic execution.

Details

Edges attributes are passed to the igraph graph and can be accessed with e.g. `E(g)$partial_correlation`. See `miic` for more details on edge parameters. By default, edges are colored according to the partial correlation between two nodes conditioned on the conditioning set (negative is blue, null is gray and positive is red) and their width is based on the conditional mutual information minus the complexity cost.

Value

An igraph graph object.

See Also

`miic` for details on edge parameters in the returned object, `igraph.plotting` for the detailed description of the plotting parameters and `layout` for different layouts.

`hematoData` *Early blood development: single cell binary gene expression data*

Description

Binarized expression data of 33 transcription factors involved in early differentiation of primitive erythroid and endothelial cells (3934 cells).

Usage

```
data(hematoData)
```

Format

A data.frame object.

References

Moignard et al. (2015) Nat Biotechnol 33(3):269-76 (PubMed link)

`miic` *MIIC, causal network learning algorithm including latent variables*

Description

MIIC (Multivariate Information based Inductive Causation) combines constraint-based and information-theoretic approaches to disentangle direct from indirect effects amongst correlated variables, including cause-effect relationships and the effect of unobserved latent causes.

Usage

```
miic(
  input_data,
  state_order = NULL,
  true_edges = NULL,
  black_box = NULL,
  n_threads = 1,
  cplx = c("nml", "mdl"),
  orientation = TRUE,
  ori_proba_ratio = 1,
  propagation = TRUE,
  latent = c("no", "yes", "orientation"),
  n_eff = -1,
  n_shuffles = 0,
  conf_threshold = 0,
  sample_weights = NULL,
  test_mar = TRUE,
  consistent = c("no", "orientation", "skeleton"),
  max_iteration = 100,
  consensus_threshold = 0.8,
  verbose = FALSE
)
```

Arguments

`input_data`	[a data frame] A n*d data frame (n samples, d variables) that contains the observational data. Each column corresponds to one variable and each row is a sample that gives the values for all the observed variables. The column names correspond to the names of the observed variables. Numeric columns will be treated as continuous values, factors and character as categorical.
`state_order`	[a data frame] An optional d*(2-3) data frame giving the order of the ordinal categorical variables. It will be used during post-processing to compute the signs of the edges using partial linear correlation. If specified, the data frame must have at least a "var_names" column, containing the names of each variable as specified by colnames(input_data). A "var_type" column may specify if each variable is to be considered as discrete (0) or continuous (1). And the "levels_increasing_order" column contains a single character string with all of the

unique levels of the ordinal variable in increasing order, delimited by a comma. If the variable is categorical but not ordinal, the "levels_increasing_order" column may instead contain NA.

`true_edges` [a data frame] An optional E*2 data frame containing the E edges of the true graph for computing performance after the run.

`black_box` [a data frame] An optional E*2 data frame containing E pairs of variables that will be considered as independent during the network reconstruction. In practice, these edges will not be included in the skeleton initialization and cannot be part of the final result. Variable names must correspond to the *input_data* data frame.

`n_threads` [a positive integer] When set greater than 1, n_threads parallel threads will be used for computation. Make sure your compiler is compatible with openmp if you wish to use multithreading.

`cplx` [a string; *c("nml", "mdl")*] In practice, the finite size of the input dataset requires that the 2-point and 3-point information measures should be *shifted* by a *complexity* term. The finite size corrections can be based on the Minimal Description Length (MDL) criterion (set the option with "mdl"). In practice, the MDL complexity criterion tends to underestimate the relevance of edges connecting variables with many different categories, leading to the removal of false negative edges. To avoid such biases with finite datasets, the (universal) Normalized Maximum Likelihood (NML) criterion can be used (set the option with "nml"). The default is "nml" (see Affeldt *et al.*, UAI 2015).

`orientation` [a boolean value] The miic network skeleton can be partially directed by orienting and propagating edge directions, based on the sign and magnitude of the conditional 3-point information of unshielded triples. The propagation procedure relyes on probabilities; for more details, see Verny *et al.*, PLoS Comp. Bio. 2017). If set to FALSE the orientation step is not performed.

`ori_proba_ratio` [a floating point between 0 and 1] When orienting an edge according to the probability of orientation, the threshold to accept the orientation. For a given edge, denote by $p > 0.5$ the probability of orientation, the orientation is accepted if $(1 - p) / p <$ ori_proba_ratio. 0 means reject all orientations, 1 means accept all orientations.

`propagation` [a boolean value] If set to FALSE, the skeleton is partially oriented with only the v-structure orientations. Otherwise, the v-structure orientations are propagated to downstream undirected edges in unshielded triples following the orientation method

`latent` [a string; *c("no", "yes", "orientation")*] When set to "yes", the network reconstruction is taking into account hidden (latent) variables. When set to "orientation", latent variables are not considered during the skeleton reconstruction but allows bi-directed edges during the orientation. Dependence between two observed variables due to a latent variable is indicated with a '6' in the adjacency matrix and in the network edges.summary and by a bi-directed edge in the (partially) oriented graph.

`n_eff` [a positive integer] The n samples given in the *input_data* data frame are expected to be independent. In case of correlated samples such as in time series or

Monte Carlo sampling approaches, the effective number of independent samples *n_eff* can be estimated using the decay of the autocorrelation function (Verny *et al.*, PLoS Comp. Bio. 2017). This *effective* number *n_eff* of *independent* samples can be provided using this parameter.

`n_shuffles` [a positive integer] The number of shufflings of the original dataset in order to evaluate the edge specific confidence ratio of all inferred edges.

`conf_threshold` [a positive floating point] The threshold used to filter the less probable edges following the skeleton step. See Verny *et al.*, PLoS Comp. Bio. 2017.

`sample_weights` [a numeric vector] An optional vector containing the weight of each observation.

`test_mar` [a boolean value] If set to TRUE, distributions with missing values will be tested with Kullback-Leibler divergence : conditioning variables for the given link $X \rightarrow YZ$ will be considered only if the divergence between the full distribution and the non-missing distribution $KL(P(X,Y)|P(X,Y)_{!NA})$ is low enough (with $P(X,Y)_{!NA}$ as the joint distribution of X and Y on samples which are not missing on Z. This is a way to ensure that data are missing at random for the considered interaction and to avoid selection bias. Set to TRUE by default

`consistent` [a string; *c("no", "orientation", "skeleton")*] if "orientation": iterate over skeleton and orientation steps to ensure consistency of the network; if "skeleton": iterate over skeleton step to get a consistent skeleton, then orient edges and discard inconsistent orientations to ensure consistency of the network. See (Li *et al.*, NeurIPS 2019) for details.

`max_iteration` [a positive integer] When the *consistent* parameter is set to "skeleton" or "orientation", the maximum number of iterations allowed when trying to find a consistent graph. Set to 100 by default.

`consensus_threshold`

[a floating point between 0.5 and 1.0] When the *consistent* parameter is set to "skeleton" or "orientation", and when the result graph is inconsistent, or is a union of more than one inconsistent graphs, a consensus graph will be produced based on a pool of graphs. If the result graph is inconsistent, then the pool is made of [max_iteration] graphs from the iterations, otherwise it is made of those graphs in the union. In the consensus graph, the status of each edge is determined as follows: Choose from the pool the most probable status. For example, if the pool contains [A, B, B, B, C], then choose status B, if the frequency of presence of B (0.6 in the example) is equal to or higher than [consensus_threshold], then set B as the status of the edge in the consensus graph, otherwise set undirected edge as the status. Set to 0.8 by default.

`verbose` [a boolean value] If TRUE, debugging output is printed.

Details

Starting from a complete graph, the method iteratively removes dispensable edges, by uncovering significant information contributions from indirect paths, and assesses edge-specific confidences from randomization of available data. The remaining edges are then oriented based on the signature of causality in observational data.

The method relies on an information theoretic based (conditional) independence test which is described in (Verny *et al.*, PLoS Comp. Bio. 2017), (Cabeli *et al.*, PLoS Comp. Bio. 2020). It deals

with both categorical and continuous variables by performing optimal context-dependent discretization. As such, the input data frame may contain both numerical columns which will be treated as continuous, or character / factor columns which will be treated as categorical. For further details on the optimal discretization method and the conditional independence test, see the function discretizeMutual. The user may also choose to run miic with scheme presented in (Li *et al.*, NeurIPS 2019) to improve the end result's interpretability by ensuring consistent separating set during the skeleton iterations.

Value

A *miic-like* object that contains:

- all.edges.summary: a data frame with information about the relationship between each pair of variables
  - *x:* X node
  - *y:* Y node
  - *type:* contains 'N' if the edge has been removed or 'P' for retained edges. If a true edges file is given, 'P' becomes 'TP' (True Positive) or 'FP' (False Positive), while 'N' becomes 'TN' (True Negative) or 'FN' (False Negative).
  - *ai:* the contributing nodes found by the method which participate in the mutual information between *x* and *y*, and possibly separate them.
  - *info:* provides the pairwise mutual information times *Nxyi* for the pair (*x*, *y*).
  - *info_cond:* provides the conditional mutual information times *Nxy_ai* for the pair (*x*, *y*) when conditioned on the collected nodes *ai*. It is equal to the *info* column when *ai* is an empty set.
  - *cplx:* gives the computed complexity between the (*x*, *y*) variables taking into account the contributing nodes *ai*. Edges that have have more conditional information *info_cond* than *cplx* are retained in the final graph.
  - *Nxy_ai:* gives the number of complete samples on which the information and the complexity have been computed. If the input dataset has no missing value, the number of samples is the same for all pairs and corresponds to the total number of samples.
  - *log_confidence:* represents the *info* - *cplx* value. It is a way to quantify the strength of the edge (*x*, *y*).
  - *confidenceRatio:* this column is present if the confidence cut is > 0 and it represents the ratio between the probability to reject the edge (*x*, *y*) in the dataset versus the mean probability to do the same in multiple (user defined) number of randomized datasets.
  - *infOrt:* the orientation of the edge (*x*, *y*). It is the same value as in the adjacency matrix at row *x* and column *y* : 1 for unoriented, 2 for an edge from X to Y, -2 from Y to X and 6 for bidirectional.
  - *trueOrt:* the orientation of the edge (*x*, *y*) present in the true edges file if provided.
  - *isOrtOk:* information about the consistency of the inferred graph's orientations with a reference graph is given (i.e. if true edges file is provided). Y: the orientation is consistent; N: the orientation is not consistent with the PAG (Partial Ancestor Graph) derived from the given true graph.
  - *sign:* the sign of the partial correlation between variables *x* and *y*, conditioned on the contributing nodes *ai*.

  - *partial_correlation:* value of the partial correlation for the edge (x, y) conditioned on the contributing nodes ai.
  - *isCausal:* details about the nature of the arrow tip for a directed edge. A directed edge in a causal graph does not necessarily imply causation but it does imply that the cause-effect relationship is not the other way around. An arrow-tip which is itself downstream of another directed edge suggests stronger causal sense and is marked by a 'Y', or 'N' otherwise.
  - *proba:* probabilities for the inferred orientation, derived from the three-point mutual information (cf Affeldt & Isambert, UAI 2015 proceedings) and noted as p(x->y);p(x<-y).

- retained.edges.summary: a data frame in the format of all.edges.summary containing only the inferred edges.

- orientations.prob: this data frame lists the orientation probabilities of the two edges of all unshielded triples of the reconstructed network with the structure: node1 – mid-node – node2:
  - node1: node at the end of the unshielded triplet
  - p1: probability of the arrowhead node1 <- mid-node
  - p2: probability of the arrowhead node1 -> mid-node
  - mid-node: node at the center of the unshielded triplet
  - p3: probability of the arrowhead mid-node <- node2
  - p4: probability of the arrowhead mid-node -> node2
  - node2: node at the end of the unshielded triplet
  - NI3: 3 point (conditional) mutual information * N

- AdjMatrix: the adjacency matrix is a square matrix used to represent the inferred graph. The entries of the matrix indicate whether pairs of vertices are adjacent or not in the graph. The matrix can be read as a (row, column) set of couples where the row represents the source node and the column the target node. Since miic can reconstruct mixed networks (including directed, undirected and bidirected edges), we will have a different digit for each case:
  - 1: (x, y) edge is undirected
  - 2: (x, y) edge is directed as x -> y
  - -2: (x, y) edge is directed as x <- y
  - 6: (x, y) edge is bidirected

References

- Verny et al., *PLoS Comp. Bio. 2017.* https://doi.org/10.1371/journal.pcbi.1005662
- Cabeli et al., *PLoS Comp. Bio. 2020.* https://doi.org/10.1371/journal.pcbi.1007866
- Li et al., *NeurIPS 2019* http://papers.nips.cc/paper/9573-constraint-based-causal-structure-learning-with-consistent-separating-sets.pdf

See Also

`discretizeMutual` for optimal discretization and (conditional) independence test.

Examples

```
library(miic)

# EXAMPLE HEMATOPOIESIS
data(hematoData)

# execute MIIC (reconstruct graph)
miic.res <- miic(
  input_data = hematoData[1:1000,], latent = "yes",
  n_shuffles = 10, conf_threshold = 0.001
)

# plot graph
if(require(igraph)) {
 plot(miic.res, method="igraph")
}


# write graph to graphml format. Note that to correctly visualize
# the network we created the miic style for Cytoscape (http://www.cytoscape.org/).

miic.write.network.cytoscape(g = miic.res, file = file.path(tempdir(), "temp"))

# EXAMPLE CANCER
data(cosmicCancer)
data(cosmicCancer_stateOrder)
# execute MIIC (reconstruct graph)
miic.res <- miic(
  input_data = cosmicCancer, state_order = cosmicCancer_stateOrder, latent = "yes",
  n_shuffles = 100, conf_threshold = 0.001
)

# plot graph
if(require(igraph)) {
 plot(miic.res)
}

# write graph to graphml format. Note that to correctly visualize
# the network we created the miic style for Cytoscape (http://www.cytoscape.org/).
miic.write.network.cytoscape(g = miic.res, file = file.path(tempdir(), "temp"))

# EXAMPLE OHNOLOGS
data(ohno)
data(ohno_stateOrder)
# execute MIIC (reconstruct graph)
miic.res <- miic(
  input_data = ohno, latent = "yes", state_order = ohno_stateOrder,
  n_shuffles = 100, conf_threshold = 0.001
)

# plot graph
if(require(igraph)) {
```

```
 plot(miic.res)
}

# write graph to graphml format. Note that to correctly visualize
# the network we created the miic style for Cytoscape (http://www.cytoscape.org/).
miic.write.network.cytoscape(g = miic.res, file = file.path(tempdir(), "temp"))
```

`miic.export` *Export miic result to different plotting methods*

Description

This function creates an object built from the result returned by `miic` that is ready to be fed to different plotting methods.

Usage

```
miic.export(miic.res, method = NULL)
```

Arguments

`miic.res`	[a miic graph object] The graph object returned by the miic execution.
`method`	A string representing the plotting method. Currently only "igraph" is supported.

Details

See the details of specific function for each method. For igraph, see `getIgraph`.

Value

A graph object adapted to the method.

See Also

`getIgraph` for details on the igraph exported object.

Examples

```
library(miic)
data(hematoData)

# execute MIIC (reconstruct graph)
miic.res <- miic(
  input_data = hematoData, latent = "yes",
  n_shuffles = 10, conf_threshold = 0.001
)
```

```
# Using igraph
if(require(igraph)) {
g = miic.export(miic.res, "igraph")
plot(g) # Default visualisation, calls igraph::plot.igraph()

# Specifying layout (see ?igraph::layout_)
l <-layout_with_kk(g)
plot(g, layout=l)

# Override some graphical parameters
plot(g, edge.curved = .2)
plot(g, vertex.shape="none", edge.color="gray85", vertex.label.color="gray10")
}
```

`miic.write.network.cytoscape`
GraphML converting function for miic graph

Description

Convert miic graph to GraphML format.

Usage

```
miic.write.network.cytoscape(g, file, layout = NULL)
```

Arguments

`g`	The graph object returned by miic.
`file`	A string. Path to the output file containing file name without extension (.graphml will be appended).
`layout`	An optional data frame of 2 (or 3) columns containing the coordinate `x` and `y` for each node. The optional first column can contain node names. If node names is not given, the order of the input file will be assigned to the list of positions.

miic.write.style.cytoscape

Style writing function for the miic network

Description

This function writes the miic style for a correct visualization using the cytoscape tool (http://www.cytoscape.org/).

Usage

```
miic.write.style.cytoscape(file)
```

Arguments

file	[a string] The file path of the output file (containing the file name without extension).

Details

The style is written in the xml file format.

ohno

Tetraploidization in vertebrate evolution

Description

20,415 protein-coding genes in the human genome from Ensembl (v70) and information on the retention of duplicates originating either from the two whole genome duplications at the onset of vertebrates ('ohnolog') or from subsequent small scale duplications ('SSD') as well as copy number variants ('CNV').

Usage

```
data(ohno)
```

Format

A data.frame object.

References

Verny et al., PLoS Comp. Bio. 2017.

ohno_stateOrder *Tetraploidization in vertebrate evolution*

Description

20,415 protein-coding genes in the human genome from Ensembl (v70) and information on the retention of duplicates originating either from the two whole genome duplications at the onset of vertebrates ('ohnolog') or from subsequent small scale duplications ('SSD') as well as copy number variants ('CNV'), category order.

Usage

```
data(ohno_stateOrder)
```

Format

A data.frame object.

References

Verny et al., PLoS Comp. Bio. 2017.

plot.miic *Basic plot function of a miic network inference result*

Description

This function calls `miic.export` to build a plottable object from the result returned by `miic` and plot it.

Usage

```
## S3 method for class 'miic'
plot(x, method = "igraph", ...)
```

Arguments

x	[a miic graph object] The graph object returned by `miic`.
method	A string representing the plotting method. Default to "igraph". Currently only "igraph" is supported.
...	Additional plotting parameters. See the corresponding plot function for the complete list. For igraph, see `igraph.plotting`.

Details

See the documentation of `miic.export` for further details.

See Also

miic.export for generic exports, getIgraph for igraph export, igraph.plotting

Index