Package 'MRPC'

October 14, 2018

Type Package

Title Mendelian Randomization (MR) Based PC Algorithm

Version 2.0.0

Author Md Bahadur Badsha [aut,cre],Evan A Martin [ctb] and Audrey Qiuyan Fu [aut]

Maintainer Md Bahadur Badsha <mdbadsha@uidaho.edu>

Description Infer causal graphs using the principle of Mendelian randomization-based PC (MRPC) algorithm. See Badsha and Fu (2018) <doi:10.1101/171348>,Badsha et al. (2018) <arXiv:1806.01899>.

License GPL (>= 2)

Depends R (>= 3.0)

LazyData TRUE

Imports bnlearn,compositions,dynamicTreeCut,GGally,fastcluster,gtools,graph,graphics,Hmisc,methods,mice,network,pcalg,Rgra

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-13 22:50:16 UTC

R topics documented:

`aSHD` *Adjusted Structural Hamming Distance (aSHD)*

Description

The SHD as implemented in the R package pcalg (Kalisch et al., 2012) and bnlearn(Scutari, 2010), counts how many differences exist between two directed graphs. This distance is 1 if an edge exists in one graph but is missing in the other, or if the direction of an edge is different between the two graphs. The larger this distance is the more different the two graphs are. We adjusted the SHD to reduce the penalty of having the wrong direction of an edge to 0.5. For example, between two graphs V –> T1 <– T2 and V –> T1 –> T2, the SHD is 1 and the aSHD is 0.5.

Usage

```
aSHD(g1, g2, GV,edge.presence = 1.0, edge.direction = 0.5)
```

Arguments

`g1`	First graph object
`g2`	Second graph object
`GV`	The number of genetic variants (SNPs/indels/CNV/eQTL) in the input data matrix. For example, if the data has one genetic variant, first column, then GV = 1, if 2, 1st and 2nd column, then GV = 2, and so on.

edge.presence The weight for an edge being present.

edge.direction The weight for the edge direction.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

References

1.Kalisch, M., Machler, M., Colombo, D., Maathuis, M.H. & Buhlmann, P. Causal Inference Using Graphical Models with the R Package pcalg. J. Stat.Softw. 47, 26 (2012).

2.Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

Examples

```
# True model
# True graph (V1 --> T1 --> T2 --> T3)
tarmat_s1 <- matrix(0,
                    nrow = 4,
                    ncol = 4)

colnames(tarmat_s1) <- c("V1", "T1", "T2", "T3")

rownames(tarmat_s1) <- colnames(tarmat_s1)

# Create an adjacency matrix for the true graph
tarmat_s1[1, 2] <- 1
tarmat_s1[2, 3] <- 1
tarmat_s1[3, 4] <- 1

# Graph object of the true graph
Truth <- as(tarmat_s1,
            "graphNEL")


# Inferred graph (V1 --> T1 <-- T2 --> T3)
tarmat_s2 <- matrix(0,
                    nrow = 4,
                    ncol = 4)

colnames(tarmat_s2) <-c ("V1", "T1", "T2", "T3")

rownames(tarmat_s2) <- colnames(tarmat_s2)

# Create an adjacency matrix for the inferred graph
tarmat_s2[1, 2] <- 1
tarmat_s2[3, 2] <- 1
tarmat_s2[3, 4] <- 1

# Graph objects for the inferred graph
Inferred <- as(tarmat_s2,
```

```
                "graphNEL")

Distance <- aSHD(Truth,
                 Inferred,
                 GV = 1,
                 edge.presence = 1.0,
                 edge.direction = 0.5)
```

Case_1P *Case for One Parent Generating Simulated Data*

Description

The gene to have data generated has only one parent.

Usage

```
Case_1P(N, P1, b0.1, b1.1, sd.1)
```

Arguments

N	Number of observations
P1	Data vector of the parent gene P1.
b0.1	Intercept of b0.1 + b1.1*P1, where P1 is the parent of the corresponding gene.
b1.1	Slope of P1 for b0.1 + b1.1*P1, where P1 is the parent of the corresponding gene.
sd.1	Standard deviation for corresponding data generated genes.

Value

Vector

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
Case_1P(N = 10^3, P1 = 1, b0.1 = 0, b1.1 = 1, sd.1 = 1)
```

`Case_2P` *Case for Two Parents Generating Simulated Data*

Description

The gene to have data generated has two parents.

Usage

```
Case_2P(N, P1, P2, b0.1, b1.1, b1.2, sd.1)
```

Arguments

`N`	Number of observations
`P1`	Data vector of the parent gene, P1.
`P2`	Data vector of the parent gene, P2.
`b0.1`	Intercept of b0.1 + b1.1*P1 + b1.2*P2, where P1 and P2 are the parents of the corresponding gene.
`b1.1`	Slope of P1 for b0.1 + b1.1*P1+ b1.2*P2, where P1 and P2 are the parents of the corresponding gene.
`b1.2`	Slope of P2 for b0.1 + b1.1*P1 + b1.2*P2, where P1 and P2 are the parents of the corresponding gene.
`sd.1`	Standard deviation for corresponding data generated genes.

Value

Vector

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
Case_2P(N = 10^3, P1 = 1, P2 = 1,
        b0.1 = 0, b1.1 = 1,
        b1.2 = 1, sd.1 = 1)
```

Case_3P *Case for Three Parent Generating Simulated Data*

Description

The gene to have data generated has three parents.

Usage

```
Case_3P(N, P1, P2, P3, b0.1, b1.1, b1.2, b1.3, sd.1)
```

Arguments

N	Number of observations.
P1	Data vector of the parent gene, P1.
P2	Data vector of the parent gene, P2.
P3	Data vector of the parent gene, P3.
b0.1	Intercept of b0.1 + b1.1*P1 + b1.2*P2 + b1.3*P3, where P1, P2, and P3 are the parents of the corresponding gene.
b1.1	Slope of P1 for b0.1 + b1.1*P1 + b1.2*P2 + b1.3*P3, where P1, P2, and P3 are the parents of the corresponding gene.
b1.2	Slope of P2 for b0.1 + b1.1*P1 + b1.2*P2 + b1.3*P3, where P1, P2, and P3 are the parents of the corresponding gene.
b1.3	Slope of P3 for b0.1 + b1.1*P1 + b1.2*P2 + b1.3*P3, where P1, P2, and P3 are the parents of the corresponding gene.
sd.1	Standard deviation for corresponding data generated gene.

Value

Vector

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
Case_3P(N = 10^3, P1 = 1, P2 = 1, P3 = 1,
        b0.1 = 0, b1.1 = 1, b1.2 = 1,
        b1.3 = 1, sd.1 = 1)
```

Case_NP *Case for No Parent Generating Simulated Data*

Description

The gene to have data generated has no parents.

Usage

```
Case_NP(N, b0.1, sd.1)
```

Arguments

N	Number of observations
b0.1	Intercept of the corresponding simulated gene.
sd.1	Standard deviation for corresponding data generated gene.

Value

Vector

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
Case_NP(N = 10^3, b0.1 = 0, sd.1 = 1)
```

Cut_Modules *Cut the Modules*

Description

Similar to cut2 function with some modification.

Usage

```
Cut_Modules(x, cuts, m, g, levels.mean = FALSE, digits, minmax = TRUE,
    oneval = TRUE, onlycuts = FALSE)
```

Arguments

x	Numeric vector to classify into intervals.
cuts	Cut points
m	Desired minimum number of observations in a group.
g	Number of quantile groups.
levels.mean	Set to TRUE to make the new categorical vector have levels attribute that is the group means of x instead of interval endpoint labels.
digits	Number of significant digits to use in constructing levels. The default is 3, and 5 if levels.mean = TRUE.
minmax	If cuts is specified but min(x) < min(cuts) or max(x) > max(cuts) augments cuts to include min and max x.
oneval	If an interval contains only one unique value, the interval will be labeled with the formatted version of that value instead of the interval endpoints unless oneval = FALSE.
onlycuts	Set to TRUE to only return the vector of computed cuts. This consists of the interior values plus outer ranges.

Value

Vector

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

Data_GEUVADIS *Data from GEUVADIS*

Description

GEUVADIS (Lappalainen et al., 2013) data (i.e., gene expression measured in LCLs of a subset of individuals genotyped in the 1000 Genomes Project) from 373 Europeans and 89 Africans.

Details

Among the most stringent set of eQTLs, ~70 have more than one target gene. Finally, we found 62 unique eQTLs discovered at the most stringent criteria exhibit pleiotropy. We extracted the genotypes of these 62 eQTLs and the expression of the target genes in the 373 Europeans and 89 Africans.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

References

1. Lappalainen, T. et al. Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501, 506-511 (2013).

Examples

```
# Data for 373 Europeans of eQTL 1
Data_GEUVADIS$Data_Q1$Data_EUR

# Data for 89 Africans of eQTL 1
Data_GEUVADIS$Data_Q1$Data_AFR
```

Data_withoutlier *Example Data with outlier*

Description

This is the data with outliers.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

Examples

```
Data_withoutlier
```

Data_withoutoutlier *Example Data without outliers*

Description

This is the data without outliers.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

Examples

```
Data_withoutoutlier
```

DendroModuleGraph *Visualization of Nodes by Dendrogram with Modules and Inferred Graph*

Description

Clustering dendrogram of nodes with dissimilarity based on topological overlap, together with assigned module colors and graphical visualization.

Usage

```
DendroModuleGraph (Adj_directed,minModuleSize,GV)
```

Arguments

Adj_directed	Adjacency matrix from directed graph
minModuleSize	Minimum module size.
GV	Number of SNPs/indels/CNV/eQTL in the input data matrix. For example, If the data has one SNPs/indels/CNV/eQTL (first column), then GV = 1, if 2 SNPs/indels/CNV/eQTL (1st and 2nd Column), then GV = 2, and so on.

Value

A list containing the graph objects as follows:

- obj: An object of class "graph" of the estimated graph.
- GroupMods: A list of modules containing the number of nodes in each module.
- Adjmatrixdirected: The adjacency matrix of the directed graph used to extract directed edges along with their nodes.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

MRPC for estimating a DAG using the Mendelian Randomization (MR) based PC (MRPC) algorithm; ModiSkeleton for estimating a skeleton using the modified skeleton function; EdgeOrientation for orientation rules used to determine the edges in MRPC; SimulatedData for simulated data generating function.

Examples

```
# Adjacency matrix from directed example graph
Adj_directed <- as(ExampleMRPC$complex$cont$withGV$graph,
                   "matrix")

# Plot of the graph
DendroModuleGraph(Adj_directed,
                  minModuleSize = 5,
                  GV = 14)
```

EdgeOrientation *Edge Orientation Rules for the MRPC Algorithm*

Description

This function performs the last step of the MRPC algorithm where it determines the edge direction of the undirected graph. The function first determines the edges between genetic variants and gene expression nodes based on MR. Then it orients the v-structures followed by the remaining edges weather MR is applicable or not. MR is a new way for edge direction determination based on four different cases. See below for the details.

Usage

```
EdgeOrientation(gInput, GV = GV, suffStat, FDR, indepTest = indepTest, verbose = FALSE)
```

Arguments

gInput	Object containing skeleton, marginal and conditional independence information.
GV	The number of genetic variants (SNPs/indels/CNV/eQTL) in the input data matrix. For example, if the data has one genetic variant, first column, then GV = 1, if 2, 1st and 2nd Column, then GV = 2, and so on.
suffStat	A list of sufficient statistics containing all necessary elements for the conditional independence tests in the function indepTest for gaussCItest. The sufficient statistics consist of the correlation matrix of the data and the sample size.

FDR	Need to specify pre-assigned level. If FDR = 0.05, that ensures FDR and mFDR remains below 0.05.
indepTest	A function for testing conditional independence. It is used to test the conditional independence of x and y given S, called as indepTest(x, y, S, suffStat). Where, x and y are variables, and S is a vector, possibly empty, of variables. suffStat is a list, see the argument above. The return value of indepTest is the p-value of the test for conditional independence. The different indepTest is used for different data types, for example, Gaussian data = gaussCItest, Discrete data = disCItest and Binary data = binCItest. See help(gaussCItest)
verbose	(optional) 1: detailed output is provided; 0: No output is provided

Details

The orientation of the edge directions based on Mendelian randomization using the four different cases. Here, we consider x is a genetic variant, y and z are the gene expression data. The 1st column of the input matrix will be the genetic variant and the remaining columns are the gene expression data.

Four different cases are as follows:

Case-1: Relation between x, genetic variant, and the other nodes. Then genetic variant will regulate the other node, genes, and direction will be genetic variant –> other node. Note that if the data has more than one genetic variant and two genetic variant have edges, then direction will be genetic variant <–> genetic variant, which indicates that there is evidence that the two genetic variants are not independent, but we do not have enough information to determine which genetic variant is the regulator and which is the target.

Case-2: If y and z are adjacent and, x and z are conditionally independent given y, then gene y will regulate the expression of gene z and the edge direction will be y –> z.

Case-3: If y and z are adjacent and, x and z are conditionally dependent given y, then gene z will regulate the expression of gene y and the edge direction will be z –> y.

Case-4: If y and z are adjacent with x and y conditionally dependent given z and x and z conditionally dependent given y, then the edge direction will be y <–> z.

Value

An object of class that contains an estimate of the equivalence class of the underlying DAG.

`call`: A call object: the original function call.

`n`: The sample size used to estimate the graph.

`max.ord`: The maximum size of the conditioning set used in the conditional independence tests of the first part of the algorithm.

`n.edgetests`: The number of conditional independence tests performed by the first part of the algorithm.

`sepset`: Separation sets.

`pMax`: A square matrix , where the (i, j)th entry contains the maximal p-value of all conditional independence tests for edge i–j.

`graph`: An object of class `"graph"`: The undirected or partially directed graph that was estimated.

`zMin`: Deprecated.

`test`: The number of tests that have been performed.

`alpha`: The level of significance for the current test.

`R`: A vector of all the decisions made so far from the tests that have been performed.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

MRPC for estimating a DAG using the Mendelian Randomization (MR) based (PC) algorithm; ModiSkeleton for estimating a skeleton using modified skeleton function; SimulatedData for simulated data generating function.

Examples

```
# Load packages
library(pcalg)  #library for existing pc

# Load predefined data
# Data pre-processing

# The 1st column of the input matrix will be the
# genetic variant and the remaining columns are the gene expression data.

# Model 1
Truth <- MRPCtruth$M1   #Truth for model 1
data <- simu.data_M1    #data load for model 1
n <- nrow (data)        #Number of row
V <- colnames(data)     #Column names

Rcor_R <- RobustCor(data,
                    0.005) #Robust correlation (Beta = 0.005)

suffStat_R <- list(C = Rcor_R$RR,
                   n = n)

# Estimate skeleton
Skel.fit <- ModiSkeleton(data, suffStat_R, FDR = 0.05,
                         indepTest = 'gaussCItest',
                         labels = V, verbose = TRUE)

# Edge Orientation
Edge_orientation <- EdgeOrientation(Skel.fit, GV = 1,
                                    suffStat_R, FDR = 0.05,
                                    indepTest = 'gaussCItest', verbose = 1)

# Plot the results
par(mfrow = c(1, 2))
plot(Truth,
```

```
     main = "(A) Truth")
plot(Edge_orientation,
     main = "(B) MRPC ")

# Other models are available and may be called as follows:
# Model 0
# Truth <- MRPCtruth$M0
# data <- simu.data_M0

# Model 2
# Truth <- MRPCtruth$M2
# data <- simu.data_M2

# Model 3
# Truth <- MRPCtruth$M3
# data <- simu.data_M3

# Model 4
# Truth <- MRPCtruth$M4
# data <- simu.data_M4

# Model Multiparent
# Truth <- MRPCtruth$Multiparent
# data <- simu.data_multiparent

# Model Star
# Truth <- MRPCtruth$Star
# data <- simu.data_starshaped

# Model Layered
# Truth <- MRPCtruth$Layered
# data <- simu.data_layered
```

`empty` *Check empty matrix*

Description

Need for check empty matrix.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

ExampleMRPC *Example Data for Simple and Complex Model*

Description

Examples with both continuous and discrete data.

Usage

```
data(ExampleMRPC)
```

Details

Contains a numeric data matrix and a graph for both continuous and discrete data.

For continuous data with genetic information: 1000 samples in row and 6 variables in column. First two columns are the genetic variants and remaning columns are gene expression.

Continuous data without genetic information: 1000 samples in row and 8 variables in column.

Discrete data with genetic information: 1000 samples in row and 6 variables in column. First column is the genetic variant and remaning columns are the gene expression.

Discrete data without genetic information: 1000 samples in row and 5 variables in column.

Continuous data with genetic information for complex model: 1000 samples in row and 22 variables in column. First 14 column is the genetic variants and remaning columns are the genes expression.

Value

A list that containing the numeric data matrix and components of a graph.

- `simple`: Simple model.
- `complex`: Complex model.
- `cont`: Continuous.
- `disc`: Discrete.
- `withGV`: With genetic information.
- `withoutGV`: Without genetic information.
- `data`: Data matrix.
- `graph`: Components of a graph.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

Examples

```
# Continuous data with genetic information
# load the data
data("ExampleMRPC")

# extract the sample size
n <- nrow(ExampleMRPC$simple$cont$withGV$data)

# extract the node/column names
V <- colnames(ExampleMRPC$simple$cont$withGV$data)

# calculate robust correlation matrix
Rcor_R <- RobustCor(ExampleMRPC$simple$cont$withGV$data,
                    Beta = 0.005)

suffStat_R <- list(C = Rcor_R$RR, n = n)

data.mrpc.cont.withGV <- MRPC(data = ExampleMRPC$simple$cont$withGV$data,
                              suffStat = suffStat_R, GV = 2,
                              FDR = 0.05, indepTest = 'gaussCItest',
                              labels = V, verbose = TRUE)

par(mfrow = c(1, 2))
# plot the true graph
plot(ExampleMRPC$simple$cont$withGV$graph,
     main = "truth")
# plot the inferred graph
plot(data.mrpc.cont.withGV,
     main = "inferred")


# Continuous data without genetic information
# load the data
data("ExampleMRPC")

# extract the sample size
n <- nrow(ExampleMRPC$simple$cont$withoutGV$data)

# extract the node/column names
V <- colnames(ExampleMRPC$simple$cont$withoutGV$data)

# calculate robust correlation matrix
Rcor_R <- RobustCor(ExampleMRPC$simple$cont$withoutGV$data,
                    Beta = 0.005)

suffStat_R <- list(C = Rcor_R$RR, n = n)

data.mrpc.cont.withoutGV <- MRPC(data = ExampleMRPC$simple$cont$withoutGV$data,
                                 suffStat = suffStat_R, GV = 0,
                                 FDR = 0.05, indepTest = 'gaussCItest',
                                 labels = V, verbose = TRUE)
```

```
par(mfrow = c(1, 2))
# plot the true graph
plot(ExampleMRPC$simple$cont$withoutGV$graph,
     main = "truth")
# plot the inferred graph
plot(data.mrpc.cont.withoutGV,
     main = "inferred")

# Discrete data with genetic information
# load the data
data("ExampleMRPC")

# extract the sample size
n <- nrow(ExampleMRPC$simple$disc$withGV$data)

# extract the node/column names
V <- colnames(ExampleMRPC$simple$disc$withGV$data)

# calculate robust correlation matrix
Rcor_R <- RobustCor(ExampleMRPC$simple$disc$withGV$data,
                    Beta = 0.005)

suffStat_R <- list(C = Rcor_R$RR, n = n)

data.mrpc.disc.withGV <- MRPC(data = ExampleMRPC$simple$disc$withGV$data,
                              suffStat = suffStat_R, GV = 1,
                              FDR = 0.05, indepTest = 'gaussCItest',
                              labels = V, verbose = TRUE)

par (mfrow = c(1, 2))
# plot the true graph
plot(ExampleMRPC$simple$disc$withGV$graph,
     main = "truth")
# plot the inferred causal graph
plot(data.mrpc.disc.withGV,
     main = "inferred")

# Discrete data without genetic information
# load the data
data("ExampleMRPC")

# extract the sample size
n <- nrow (ExampleMRPC$simple$disc$withoutGV$data)

# extract the node/column names
V <- colnames(ExampleMRPC$simple$disc$withoutGV$data)

# calculate robust correlation matrix
Rcor_R <- RobustCor(ExampleMRPC$simple$disc$withoutGV$data,
                    Beta = 0.005)

suffStat_R <- list(C = Rcor_R$RR, n = n)
```

```
data.mrpc.disc.withoutGV <- MRPC(data = ExampleMRPC$simple$disc$withoutGV$data,
                                 suffStat = suffStat_R, GV = 1,
                                 FDR = 0.05, indepTest = 'gaussCItest',
                                 labels = V, verbose = TRUE)

par(mfrow = c(1, 2))
# plot the true graph
plot(ExampleMRPC$simple$disc$withoutGV$graph,
     main = "truth")
# plot the inferred graph
plot(data.mrpc.disc.withoutGV,
     main = "inferred")

# Continuous data with genetic information for complex model
# load the data
data("ExampleMRPC")

# graph without clustering
plot(ExampleMRPC$complex$cont$withGV$graph)

Adj_directed <- as(ExampleMRPC$complex$cont$withGV$graph,
                   "matrix")

# dendrogram and graph with clustering
DendroModuleGraph(Adj_directed,
                  minModuleSize = 5,
                  GV = 14)
```

`Example_Outlier` *Impact of outliers on graph inference*

Description

The data contain two genotype nodes, V1 and V2, and 3 phenotype nodes, T1, T2 and T3. The genotype nodes are discrete, whereas the phenotype nodes are continuous. We randomly added 10 artificial outliers to our original data matrix.First, we generated uniform distribution for outlying expression and add with the original data. Then we check the robustness by three methods (MRPC, mmhc, and pc) in the presence of outliers.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

Examples

```
library(pcalg) #for pc
library(bnlearn) #for mmhc

#Truth without outlier
```

```
tarmat <- matrix(0,
                 nrow = ncol(Data_withoutoutlier),
                 ncol = ncol(Data_withoutoutlier))

colnames(tarmat) <- colnames(Data_withoutoutlier)

rownames(tarmat) <- colnames(Data_withoutoutlier)

tarmat[1,2] <- 1
tarmat[2,1] <- 1
tarmat[2,3] <- 1
tarmat[4,3] <- 1
tarmat[4,5] <- 1

Truth <- as(tarmat,
            "graphNEL")

# Data without outliers
n <- nrow(Data_withoutoutlier)     # Number of rows
V <- colnames(Data_withoutoutlier) # Column names

# Classical correlation
suffStat_C1 <- list(C = cor(Data_withoutoutlier),
                    n = n)

# Robust correlation (Beta = 0.005)
Rcor_R1 <- RobustCor(Data_withoutoutlier,
                     0.005)

suffStat_R1 <- list(C = Rcor_R1$RR,
                    n = n)

# Estimated graph by MRPC using gaussCItest and beta = 0.005
MRPC.fit_withoutoutlier <- MRPC(Data_withoutoutlier, suffStat_R1,
                                GV = 2, FDR = 0.05,
                                indepTest ='gaussCItest',
                                labels = V, verbose = TRUE)

# Estimated graph by mmhc
data <- data.frame(Data_withoutoutlier)
mmhc_withoutoutlier <- mmhc(data)

# Estimated graph by pc with alpha = 0.05
pc.fit_withoutoutlier <- pc(suffStat_C1,
                            indepTest = gaussCItest,
                            alpha = 0.05, labels = V,
                            verbose = TRUE)

# Data with outliers
n <- nrow (Data_withoutlier)    # Number of rows
V <- colnames(Data_withoutlier) # Column names

# Classical correlation
```

```
suffStat_C1 <- list(C = cor(Data_withoutlier),
                    n = n)

# Robust correlation (Beta = 0.005)
Rcor_R1 <- RobustCor(Data_withoutlier,
                     0.005)

suffStat_R1 <- list(C = Rcor_R1$RR,
                    n = n)

# Estimated graph by MRPC using gaussCItest and beta = 0.005
MRPC.fit_withoutlier <- MRPC(Data_withoutlier, suffStat_R1,
                             GV = 2, FDR = 0.05,
                             indepTest ='gaussCItest',
                             labels = V, verbose = TRUE)

# Estimated graph by mmhc
data <- data.frame(Data_withoutlier)
mmhc_withoutlier <- mmhc(data)

# Estimated graph by pc with alpha = 0.05
pc.fit_withoutlier <- pc(suffStat_C1,
                         indepTest = gaussCItest,
                         alpha = 0.05, labels = V,
                         verbose = TRUE)

# Plot the results
# Show the estimated graph
par(mfrow = c(2, 4))
plot(Truth,
     main = "Truth")
plot(MRPC.fit_withoutoutlier@graph,
     main = "MRPC")
graphviz.plot(mmhc_withoutoutlier,
              main = "mmhc")
plot(pc.fit_withoutoutlier,
     main = "pc")
plot(Truth,
     main = " ")
plot(MRPC.fit_withoutlier@graph,
     main = " ")
graphviz.plot(mmhc_withoutlier,
              main = " ")
plot(pc.fit_withoutlier,
     main = " ")
```

InferenceAccuracyExample

Comparison of inference accuracy with and without a v-structure

---

Description

This is the example comparison of inference accuracy with and without a v-structure by recall and precision of three methods, MPRC, mmhc, and pc, across 1000 data sets. Signal values indicate the signal strength: the larger the value the stronger the signal. The sample size in each simulated data set is 1000. See Badsha and Fu, 2018 for the comparison of inference accuracy with sample sizes of 50, 200, and 500 .

Usage

```
InferenceAccuracyExample(N = 1000, signal, model, ita = 1000)
```

Arguments

`N`	Number of observation
`signal`	The coefficient of parent nodes in the linear model. For example, strong = 1.0, moderate = 0.5, and weak = 0.2 signal strengths.
`model`	The model data will be simulated for. For example, if you want comparison of inference accuracy without a v-structure you would type 'model1' into the function. If you want comparison of inference accuracy with a v-structure you would type 'model2' into the function.
`ita`	Number of different data sets

Details

The output is a matrix, where the rows are the three methods, MRPC, mmhc, and pc, and the colums are mean Recall, sd Recall, mean Precision, and sd Precision, respectively.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

References

1. Badsha MB, Fu AQ (2018). Learning causal biological networks with the principle of Mendelian randomization." bioRxiv. doi:10.1101/171348.

See Also

Recall_Precision: Performance Evaluation by Recall and Precision in MRPC.

Examples

```
# Comparison of inference accuracy without a v-structure
# across 10 data sets and sample size in each simulated
# data set is set to 100 with signal = 1.0.
Result1 <- InferenceAccuracyExample(N = 100, signal = 1.0,
                                    'model1', ita = 10)

# Comparison of inference accuracy with a v-structure
# across 10 data sets and sample size in each simulated
# data set is set to 100 with signal=1.0.
Result2 <- InferenceAccuracyExample(N = 100, signal = 1.0,
                                    'model2', ita = 10)
```

`ModiSkeleton` *Draw Undirected Graph using the Modified Skeleton Function*

Description

The Modiskeleton function is used to estimate the undirected graph, skeleton, using the modified skeleton function. The main difference is that the ModiSkeleton function uses alpha, significance level, as a function of the prior outcomes, reject or accept hypothesis, but the existing skeleton function used alpha as a fixed pre-assigned significance level. See details in **Choice of Significance Level** below.

Usage

```
ModiSkeleton(data, suffStat, FDR, indepTest, labels, p,
    method = c("stable", "original", "stable.fast"),
    m.max = Inf, fixedGaps = NULL, fixedEdges = NULL,
    NAdelete = TRUE, verbose = FALSE)
```

Arguments

Since, this is the modification of the existing skeleton function, therefore, most of the terms and sentences are taken from skeleton (Spirtes et al., 2000). Also, most of the arguments are identical to MRPC and pc.

Data matrix, where the rows are the observations and the columns are the genes. The columns are ordered from single-nucleotide polymorphism (SNPs), indels, copy number variation (CNV), or expression quantitative trait loci (eQTL) to genes. If we consider one genetic variant, then the first column of the input matrix is the genetic variant and the remaining columns are the gene expression data and so on.

`suffStat` A list of sufficient statistics, containing all necessary elements for the conditional independence tests in the function indepTest for gaussCItest. The sufficient statistics consist of the correlation matrix of the data and the sample size.

FDR	Need to specify pre-assigned level.If FDR=0.05, that ensures FDR and mFDR remains below 0.05.
indepTest	A function for testing conditional independence. It is used to test the conditional independence of x and y given S, called as indepTest(x, y, S, suffStat). Where, x and y are variables, and S is a vector, possibly empty, of variables. suffStat is a list, see the argument above. The return value of indepTest is the p-value of the test for conditional independence. The different indepTest is used for different data types, for example, Gaussian data = gaussCItest, Discrete data = disCItest and Binary data = binCItest. See help(gaussCItest) The ci.test (Marco Scutari, 2010) is also used for testing conditional independence and return value of indepTest is the p-value. If none is specified, the default test statistic is the mutual information for categorical variables, the Jonckheere-Terpstra test for ordered factors and the linear correlation for continuous variables.See help(ci.test) Remember that need to specify the which indepTest would like for independence testing. For example, if you would like to use gaussCItest you would type indepTest = 'gaussCItest' into the function otherwise indepTest = 'citest'. Note that, we used gaussCItest to compare our MRPC with the existing pc, because of ci.test is not robust. See details in example.
labels	A character vector of variable, or node, names. All variables are denoted in column in the input matrix.
p	(optional) The number of variables, or nodes. May be specified if the labels are not provided, in which case the labels are set to 1:p.
method	(optional) Character string specifying method. The default, "stable" provides an order-independent skeleton.
m.max	(optional) Maximum size of the conditioning sets that are considered in the conditional independence tests.
fixedGaps	(optional) A logical matrix of dimension p*p. If entry [x, y], [y, x], or both are TRUE, the edge x—y is removed before starting the algorithm. Therefore, this edge is guaranteed to be absent in the resulting graph.
fixedEdges	(optional) A logical matrix of dimension p*p. If entry [x, y], [y, x], or both are TRUE, the edge x—y is never considered for removal. Therefore, this edge is guaranteed to be present in the resulting graph.
NAdelete	(optional) If indepTest returns NA and this option is TRUE, the corresponding edge is deleted. If this option is FALSE, the edge is not deleted.
verbose	(optional) If TRUE, detailed output is provided. Default is FALSE for no output details

Details

We incorporated sequential hypothesis testing to draw the undirected graph, skeleton, by the ModiSkeleton function this is similar to the pc algorithm skeleton function but, the difference is that the ModiSkeleton function uses alpha, significance level, as a function of the prior outcomes, reject or accept hypothesis. The skeleton function uses alpha as a fixed pre-assigned significance level. Also, the ModiSkeleton function takes a robust correlation approach (Badsha et al., 2013) which reduces the impact of outliers. While the skeleton function uses classical correlation which is highly

influnced by outliers. See details in in ModiSkeleton and skeleton). The other terms are the same as skeleton.

The skeleton function starts with a complete undirected graph. Then a series of conditional independence tests is done and edges are deleted in the following way.

First, all pairs of nodes are tested for marginal independence. If two nodes x and y are judged to be marginally independent at level alpha, we adjusted the significance level at every test, see below for the details **Choice of Significance Level**, the edge between them is deleted and the empty set is saved as separation sets S[x, y] and S[y, x]. After all pairs have been tested for marginal independence and some edges might have been removed.

Second, all pairs of nodes (x, y) are tested for conditional independence given any single node or all possible subsets of the remaining nodes. If there is any node z such that x and y are conditionally independent given z, the edge between x and y is removed and node z is saved as separation set, sepset, S[x, y] and S[y, x]. The algorithm continues in this way by increasing the size of the conditioning set step by step. The algorithm stops if all adjacency sets in the current graph are smaller than the size of the conditioning set. The result is the skeleton in which every edge is still undirected.

Now, each triple of vertices (x, z, y) such that the pairs (x, z) and (y, z) are each adjacent in the skeleton but (x, y) are not, such a triple is called an 'unshielded triple', is oriented based on the information saved in the conditioning sepset S[x, y] and S[y, x]. More precisely, an unshielded triple x—z—y is oriented as x —> z <— y, called a v-structure, if z is not in S[x, y] = S[y, x] i.e., x and y are conditionally dependent in given z.

Choice of Significance Level: One of the most important parts of our method is the choice of significance level. To draw the undirected skeleton, the significance level (number in (0,1)) treated as a tuning parameter for the marginal and conditional independence tests. At each step of the test, we must decide whether to reject or accept hypothesis, without having access to the number of hypotheses, potentially infinite, or the future p-values. Hypotheses with p-values below a significance level, typically 0.05, are considered to be statistically significant. While this controls type I errors for single testing problems, in the case of testing multiple hypotheses we need to adjust the significance level to control other metrics such as family-wise error rate (FWER) or false discovery rate (FDR) or marginal FDR (mFDR). FWER is the probability of rejecting at least one true null hypothesis, which is very conservative for multiple testing. On the other hand, FDR is the expected proportion of false positives among rejected null hypotheses (Benjamini and Hochberg, 1995) and mFDR is the expected number of false discoveries to the expected number of discoveries. FDR controls a property of one realized set of tests, while mFDR is the ratio of two expectations over many realizations.

To draw the undirected skeleton, the skeleton algorithm (Spirtes et al., 2000) used alpha as a fixed pre-assigned significance level, which does not control the FDR and mFDR. Kalisch and Buhlmann, 2007 also used alpha as a fixed pre-assigned value based on Structural Hamming Distance (SHD) suggested by Tsamardinos et al., (2006) for estimating high-dimensional DAG with the pc algorithm. But, the problem is that to calculate SHD, we have to need the true graph, which is unknown in our study.

Another problem of using fixed pre-assigned significance level, alpha, is that if the p-value is close to the value of alpha, then it is difficult to accept or reject null hypotheses. For example, if p-value = 0.02312 and if we used fixed alpha = 0.05, then it is considered to be statistically significant. But, if we used alpha = 0.01, then it is considered to be statistically insignificant, see examples in MRPC for model 0 and model 3. Now, a question is how can we choose the appropriate value of alpha.

To avoid these problems, we need to adjust the alpha value at each step of the test. (Benjamini and Hochberg, 1995) also proposed a sequential testing procedure to control FDR in multiple testing problems assuming that all the p-values are given a priori but it does not address the scenario described above.

Therefore, the ModiSkeleton function uses the LOND algorithm for alpha, level of significance, as a function of the prior outcomes, reject or accept, that control FDR and mFDR in an online manner (Javanmard and Montanari, 2015), which ensures that FDR remains below a pre-assigned alpha level.

Value

An object containing an estimate of the skeleton of the underlying DAG as follow:

`call`: A call object: the original function call.

`n`: The sample size used to estimate the graph.

`max.ord`: The maximum size of the conditioning set used in the conditional independence tests of the first part of the algorithm.

`n.edgetests`: The number of conditional independence tests performed by the first part of the algorithm.

`sepset`: Separation sets.

`pMax`: A square matrix , where the (i, j)th entry contains the maximum p-value of all conditional independence tests for edge i–j.

`graph`: Object of class "`graph`": The undirected or partially directed graph that was estimated.

`zMin`: Deprecated.

`test`: The number of tests that have been performed.

`alpha`: The level of significance for the current test.

`R`: All of the decisions made so far from tests that have been performed.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

References

1.Badsha, M. B. Mollah, M. N. Jahan, N. and Kurata, H. (2013). Robust complementary hierarchical clustering for gene expression data analysis by beta-divergence. J Biosci Bioeng 116(3): 397-407.

2. Benjamini, Y. and Y Hochberg(1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing, J. R. Statist. Soc. B, B, 57, 289-300.

3. Javanmard and Montanari (March 5, 2015) On Online Control of False Discovery Rate. arXiv:150206197 [statME].

4. Kalisch, M. and Buhlmann, P. (2007) Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm, Journal of Machine Learning Research, 8, 613-636.

5.Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

6.Tsamardinos, I. Brown, L.E. and Aliferis, C.F. (2006). The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. JMLR 65, 31-78.

7. Spirtes, P. Glymour, C and Scheines, R (2000). Causation, Prediction, and Search, 2nd edition. The MIT Press.

See Also

MRPC for estimating a DAG using the Mendelian Randomization (MR) based PC (MRPC) algorithm; EdgeOrientation for orientation rules for edges in MRPC; SimulatedData simulated data generating function.

Examples

```
# Load packages
library(pcalg)   # library for pc algorithm
library(bnlearn) # library for ci.test

# Load predefined simulated data
# Data pre-processing

# The 1st column of the input matrix will be the
# genotype of the expression quantitative trait loci
# (eQTL)/Copy number variation (CNVs) and the remaining
# columns are the gene expression data.
# We used pre-assigned level alpha = 0.05 that ensures
# FDR and mFDR remains below 0.05.

# Model 1
# Estimate Skeleton
data <- simu.data_M1 # data for model 1
n <- nrow(data)      # Number of row
V <- colnames(data)  # Column names

Rcor_R <- RobustCor(data,
                    0.005) # Robust correlation (Beta = 0.005)

suffStat_R <- list(C = Rcor_R$RR, n = n)

Skel.fit <- ModiSkeleton(data, suffStat_R,
                         FDR = 0.05, indepTest = 'gaussCItest',
                         labels = V, verbose = TRUE)

# Plot the results
# Show estimated skeleton
plot(Skel.fit@graph,
     main ="Estimated Skeleton")

# Other models are available and may be called as follows:
# Model 0
# data <- simu.data_M0

# Model 2
```

```
# data <- simu.data_M2

# Model 3
# data <- simu.data_M3

# Model 4
# data <- simu.data_M4

# Model Multiparent
# data <- simu.data_multiparent

# Model Star
# data <- simu.data_starshaped

# Model Layered
# data <- simu.data_layered
```

`mpinv` *Function of Calculate Inverse Matrix*

Description

Calculate inverse of the matrix, when the matrix is not square. This function is used to calculate the robust correlation matrix.

Usage

```
mpinv(X)
```

Arguments

`X`	Data Matrix

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

Examples

```
Inversematrix <- mpinv(simu.data_M0)
```

MRPC *Estimate a causal network using the MRPC Algorithm*

Description

The MRPC () function is used to estimate a causal network (or causal graph) with directed and undirected edges from observational data, by modifying the existing pc algorithm (Spirtes et al., 2000). Existing pc algorithm is used for estimating the DAG from observational data and several variants are implemented in the R package called pcalg. Several challenges remain when the pc algorithm is applied to noisy genomic data. To tackle these challenges, we developed a novel machine learning algorithm called Mendelian Randomization (MR) based PC (MRPC) to efficiently learn a causal gene regulatory network (or a causal graph of genes) using genotype and gene expression data. Unlike existing pc algorithm, the four major updates in the MRPC algorithm are that (i) we incorporate a sequential testing method to control the False Discovery Rate (FDR), (ii) take a robust approach to reduce the impact of outliers, (iii) improve v-structure identification and (iv) implement a new way for edge direction determination based on the Principle of Mendelian Randomization (PMR) (Badsha and Fu, 2018 and Badsha et al., 2018). See details below.

Usage

```
MRPC(data, suffStat, GV, FDR = 0.05, indepTest, labels, p,
    fixedGaps = NULL, fixedEdges = NULL,
    NAdelete = TRUE, m.max = Inf,
    u2pd = c("relaxed", "rand", "retry"),
    skel.method = c("stable", "original", "stable.fast"),
    conservative = FALSE,
    maj.rule = FALSE, solve.confl = FALSE,
    verbose = FALSE)
```

Arguments

	The MRPC algorithm is a modification of the existing pc algorithm. Therefore, most of the terms and sentences are taken from pc (Spirtes et al., 2000). Data matrix, where the rows are the observations and the columns are the genes. The columns are ordered from single-nucleotide polymorphism (SNPs), indels, copy number variation (CNV), or expression quantitative trait loci (eQTL) to genes. If we consider one genetic variant, then the first column of the input matrix is the genetic variant and the remaining columns are the gene expression data and so on.
data suffStat	A list of sufficient statistics, containing all necessary elements for the conditional independence tests in the function indepTest for gaussCItest. The sufficient statistics consist of the correlation matrix of the data and the sample size.
GV	The number of genetic variants (SNPs/indels/CNV/eQTL) in the input data matrix. For example, if the data has one SNPs/indels/CNV/eQTL, first column, then GV = 1, if 2 SNPs/indels/CNV/eQTL, 1st and 2nd Column, then GV = 2, if no GV then GV = 0, and so on.

FDR	Need to specify pre-assigned level. If FDR = 0.05 this ensures FDR and mFDR remains below 0.05.
indepTest	A function for testing conditional independence. It is used to test the conditional independence of x and y given S, called as indepTest(x, y, S, suffStat). Where, x and y are variables, and S is a vector, possibly empty, of variables. suffStat is a list, see the argument above. The return value of indepTest is the p-value of the test for conditional independence. The different indepTest is used for different data types, for example, Gaussian data = gaussCItest, Discrete data = disCItest and Binary data = binCItest. See help(gaussCItest) The ci.test (Marco Scutari, 2010) is also used for testing conditional independence and return value of indepTest is the p-value. If none is specified, the default test statistic is the mutual information for categorical variables, the Jonckheere-Terpstra test for ordered factors and the linear correlation for continuous variables.See help(ci.test) Remember that need to specify the which indepTest would like for independence testing. For example, if you would like to use gaussCItest you would type indepTest = 'gaussCItest' into the function otherwise indepTest = 'citest'. Note that, we used gaussCItest to compare our MRPC with the existing pc, because of ci.test is not robust. See details in example.
labels	A character vector of variable, or node, names. All variables are denoted in column in the input matrix.
p	(optional) The number of variables, or nodes. May be specified if the labels are not provided, in which case the labels are set to 1:p.
fixedGaps	(optional) A logical matrix of dimension p*p. If entry [x, y], [y, x], or both are TRUE, the edge x—y is removed before starting the algorithm. Therefore, this edge is guaranteed to be absent in the resulting graph.
fixedEdges	(optional) A logical matrix of dimension p*p. If entry [x, y], [y, x], or both are TRUE, the edge x—y is never considered for removal. Therefore, this edge is guaranteed to be present in the resulting graph.
NAdelete	(optional) If indepTest returns NA and this option is TRUE, the corresponding edge is deleted. If this option is FALSE, the edge is not deleted.
m.max	(optional) Maximum size of the conditioning sets that are considered in the conditional independence tests.
u2pd	(optional) Character string specifying the method for dealing with conflicting information.
skel.method	(optional) Character string specifying method; the default, "stable" provides an order-independent skeleton.
conservative	(optional) Logical. Indicates if the conservative PC algorithm is used. In this case, only option u2pd = "relaxed" is supported. Note that the resulting object might not be extendable to a DAG. See details for more information.
maj.rule	(optional) Logical. Indicates that the triplets will be checked for ambiguity using a majority rule idea, which is less strict than the conservative PC algorithm. For more information, see details.

`solve.confl` (optional) If TRUE, the orientation of the v-structures and the orientation rules work with lists for candidate sets and allow bi-directed edges to resolve conflicting edge orientations. In this case, only option u2pd = "relaxed"" is supported. Note that the resulting object might not be a CPDAG because bi-directed edges might be present. See details for more information.

`verbose` (optional) If TRUE, detailed output is provided. The default is FALSE which does not print output details.

Details

Nodes are used as major building blocks of the models, which represent random variables and edges, which encode conditional dependence relations of the enclosing vertices (Kalisch and Buhlmann, 2007). The structure of conditional independence among the random variables can be explored using the Markov properties. The directed edges show the presence and direction of direct causal effects. A bidirected edge means that the edge orientation should be forward (—>) or bacward (<—). The PC algorithm is computationally efficient for learning the underlying DAG (Kalisch and Buhlmann, 2007). The MPRC algorithm improves on the existing PC algorithm in the following aspects:

(i) the genotype data at genetic variants (e.g., SNPs and copy number variation) provide additional information that helps distinguish the casual direction between two genes; this is the rationale behind Mendelian Randomization (MR). MR can greatly reduce the space of possible graphs and increase the inference efficiency for inference of genomic data but PC does not rely on MR and determination of the edge direction can be tricky when the graph is large.

(ii) the number of statistical tests is unknown beforehand in all existing variants of the PC algorithm. It is unclear how these algorithms control the false discovery rate (FDR) or marginal FDR (mFDR), because it is used alpha as a fixed pre-assigned significance level for testing the independence test, but results are sensitive to the choice of alpha, which is an important issue for multiple hypotheses testing.

(iii) the gene expression data often have outliers, even after normalization, which may drastically alter the topology of the inferred graph.

(iv) to find the v-structure the PC algorithm assumes that if there is no evidence that two nodes are conditionally independent, then the nodes are conditionally dependent, although we don't have any evidence about their conditional dependence. In that case the v-structure is worng.

To overcome the above challenges, we propose a novel algorithm, MRPC (MR-based PC), which can be applied to genotype and gene expression data and efficiently learn a causal graph of genes. **First**, we take a robust approach (Badsha et al., 2013) and calculate the robust correlation matrix on which the series of hypothesis testing is performed. **Second**, we adopt a sequential method LOND (significance Levels based On Number of Discoveries) (Javanmard and Montanari, 2015) that controls the FDR or marginal FDR (mFDR) in an online manner, where level of significance is used a function of the previous decision made so far (we adjusted at each step in the test) and that ensures that FDR and mFDR remains below alpha. **Third**, we improve the v-structure identification using the addtional conditional test. **Finally**, we implement a new way for edge direction determination based on Mendelian randomization.

The two main steps in the MRPC algorithm are as follows:

Step-1: We incorporated sequential hypothesis testing to draw the undirected graph (skeleton) by ModiSkeleton function (similar as the pc algorithm by skeleton function but, difference is that the

ModiSkeleton function is used alpha (significance level) as a function of the prior outcomes (reject or accept hypothesis) but the skeleton function used alpha as a fixed pre-assigned significance level and ModiSkeleton take a robust correlation approach reduces the impact of outliers but the skeleton used classical correlation which is highly influnced by outliers. See details in in ModiSkeleton and skeleton).

Step-2: We implemented a new way for edge direction determination based on Mendelian randomization (MR). We consider the first column of the input matrix will be the genotype of the SNPs/indels/CNV/eQTL and the remaining column are the gene expression data. See details in EdgeOrientation.

All statistical inference is done in **Step 1**, while **Step-2** is just application of deterministic rules on the results of **Step-1**. If the first part is done correctly, the second part will never fail. If, however, there are errors in **Step-1**, **Step-2** will be more sensitive since it depends on the inferential results of **Step-1**.

Value

An object of class that contains an estimate of the equivalence class of the underlying DAG.

`call`: a call object: the original function call.

`n`: The sample size used to estimate the graph.

`max.ord`: The maximum size of the conditioning set used in the conditional independence tests in the first part of the algorithm.

`n.edgetests`: The number of conditional independence tests performed by the first part of the algorithm.

`sepset`: Separation sets.

`pMax`: A numeric square matrix , where the (i, j)th entry contains the maximal p-value of all conditional independence tests for edge i–j.

`graph`: Object of class "`graph`": the undirected or partially directed graph that was estimated.

`zMin`: Deprecated.

`test`: The number of tests that have been performed.

`alpha`: The level of significance for the current test.

`R`: All of the decisions made so far from tests that have been performed.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

References

1.Badsha, M. B. Mollah, M. N. Jahan, N. and Kurata, H. (2013). Robust complementary hierarchical clustering for gene expression data analysis by beta-divergence. J Biosci Bioeng 116(3): 397-407.

2. Javanmard and Montanari (March 5, 2015) On Online Control of False Discovery Rate. arXiv:150206197 [statME].

3.Kalisch, M., Machler, M., Colombo, D., Maathuis, M.H. & Buhlmann, P. Causal Inference Using Graphical Models with the R Package pcalg. J. Stat. Softw. 47, 26 (2012).

4. Kalisch, M. and Buhlmann, P. (2007) Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm, Journal of Machine Learning Research, 8, 613-636.

5.Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

6. Md Bahadur Badsha and Audrey Qiuyan Fu. Learning causal biological networks with the principle of Mendelian randomization. bioRxiv, 2018. doi:10.1101/171348.

7. Md Bahadur Badsha, Evan A Martin, and Audrey Qiuyan Fu. MRPC: An R package for accurate inference of causal graphs. arXiv, 2018. arXiv:1806.01899

8. Spirtes,P. Glymour, C. and Scheines, R (2000). Causation, Prediction, and Search, 2nd edition. The MIT Press.

See Also

ModiSkeleton for estimating a skeleton using modified skeleton function; EdgeOrientation for orientation rules determination for edges in MRPC; SimulatedData for simulated data generating function.

Examples

```
# Load packages
# We compare different simulated data across three methods: MRPC,
# PC in pcalg (Kalisch et al., 2012), and mmhc in bnlearn (Marco Scutari, 2010)
library(pcalg)    # library for pc algorithm
library(bnlearn)  # library for mmhc

# Load predefined simulated data
# Data pre-processing

# The 1st column of the input matrix will be the genotype of the
# expression quantitative trait loci (eQTL)/Copy number variation (CNV)
# and the remaining columns are the gene expression data.

# We used pre-assigned level alpha = 0.05 that ensures FDR and mFDR
# will remain below 0.05.

# Model 1
Truth <- MRPCtruth$M1   # Truth for model 1
data <- simu.data_M1    # load data for model 1
n <- nrow (data)        # Number of rows
V <- colnames(data)     # Column names

# Classical correlation
suffStat_C <- list(C = cor(data),
                   n = n)

# Robust correlation (Beta = 0.005)
Rcor_R <- RobustCor(data, 0.005)
suffStat_R <- list(C = Rcor_R$RR,
                   n = n)
```

```
# Estimated graph by MRPC using gaussCItest and beta = 0.005
MRPC.fit<- MRPC(data,
                suffStat_R,
                GV = 1,
                FDR = 0.05,
                indepTest = 'gaussCItest',
                labels = V,
                verbose = TRUE)

# Estimated graph by mmhc
mmhc.fit <- mmhc(data)

# Estimated graph by PC with alpha = 0.05
pc.fit <- pc(suffStat_C,
             indepTest = gaussCItest,
             alpha = 0.05,
             labels = V,
             verbose = TRUE)

# Plot the results
# Show estimated graph
par(mfrow = c(2, 2))
plot(Truth,
     main = "(A) Truth")
plot(MRPC.fit,
     main = "(B) MRPC")
graphviz.plot(mmhc.fit,
              main = "(C) mmhc")
plot(pc.fit,
     main ="(D) pc")

# Another option for plot of the results. First fig is the nodes
# dendrogram with colored modules. Second fig is the plot of the graph
# with color based on modules.
# To idendify modules and complex graph (Suitable if you have many nodes)
# Adjacency matrix from directed graph
Adj_directed <- as(MRPC.fit@graph,
                   "matrix")

# Plot of the graph with colored modules.
DendroModuleGraph(Adj_directed,
                  minModuleSize = 1,
                  GV = 1)

# Other models are available and may be called as follows:
# Model 0
# Truth <- MRPCtruth$M0
# data <- simu.data_M0

# Model 2
# Truth <- MRPCtruth$M2
# data <- simu.data_M2
```

```
# Model 3
# Truth <- MRPCtruth$M3
# data <- simu.data_M3

# Model 4
# Truth <- MRPCtruth$M4
# data <- simu.data_M4

# Model Multiparent
# Truth <- MRPCtruth$Multiparent
# data <- simu.data_multiparent

# Model Star
# Truth <- MRPCtruth$Star
# data <- simu.data_starshaped

# Model Layered
# Truth <- MRPCtruth$Layered
# data <- simu.data_layered
```

`MRPCclass-class` *Class of MRPC Algorithm Results*

Description

This class of objects is returned by the functions `ModiSkeleton` and `MRPC` to represent the (ModiSkeleton) of an estimated DAG similarly from `pcAlgo-class`. Objects of this class have methods for the functions plot, show and summary.

Usage

```
## S4 method for signature 'MRPCclass,ANY'
plot(x, y, main = NULL,
     zvalue.lwd = FALSE, lwd.max = 7, labels = NULL, ...)
## S3 method for class 'MRPCclass'
print(x, amat = FALSE, zero.print = ".", ...)

## S4 method for signature 'MRPCclass'
summary(object, amat = TRUE, zero.print = ".", ...)
## S4 method for signature 'MRPCclass'
show(object)
```

Arguments

`x, object`	a "`MRPCclass`" object.
`y`	(generic `plot()` argument; unused).
`main`	main title for the plot (with an automatic default).

zvalue.lwd	logical indicating if the line width (lwd) of the edges should be made proportional to the entries of matrix zMin (originally) or derived from matrix pMax.
lwd.max	maximal lwd to be used, if zvalue.lwd is true.
labels	if non-NULL, these are used to define node attributes nodeAttrs and attrs, passed to agopen() from package **Rgraphviz**.
amat	logical indicating if the adjacency matrix should be printed as well.
zero.print	String for printing 0 ('zero') entries in the adjacency matrix.
...	(optional) Further arguments passed from and to methods.

Creation of objects

Objects are typically created as result from skeleton() or pc(), but could be be created by calls of the form new("MRPCclass", ...).

Slots

The slots call, n, max.ord, n.edgetests, sepset, pMax, graph, zMin, test, alpha and R are inherited class.

In addition, "MRPCclass" has slots

call: a call object: the original function call.

n: The sample size used to estimate the graph.

max.ord: The maximum size of the conditioning set used in the conditional independence tests of the first part of the algorithm.

n.edgetests: The number of conditional independence tests performed by the first part of the algorithm.

sepset: Separation sets.

pMax: A square matrix , where the (i, j)th entry contains the maximum p-value of all conditional independence tests for edge i–j.

graph: Object of class "graph": The undirected or partially directed graph that was estimated.

zMin: Deprecated.

test: The number of tests that have been performed.

alpha: The level of significance for the current test.

R: All of the decisions made so far from tests that have been performed.

Methods

plot signature(x = "MRPCclass"): Plot the resulting graph. If argument "zvalue.lwd" is true, the linewidth an edge reflects zMin, so that thicker lines indicate more reliable dependencies. The argument "lwd.max" controls the maximum linewidth.

show signature(object = "MRPCclass"): Show basic properties of the fitted object

summary signature(object = "MRPCclass"): Show details of the fitted object

Author(s)

Md. Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

MRPC, ModiSkeleton

Examples

```
showClass("MRPCclass")

# generate a MRPCclass object
data <- simu.data_M1
n <- nrow(data)      # Number of rows
V <- colnames(data)  # Column names

# Classical correlation
suffStat_C <- list(C = cor(data), n = n)

# Robust correlation (Beta = 0.005)
Rcor_R <- RobustCor(data,
                    0.005)

suffStat_R <- list(C = Rcor_R$RR, n = n)

# Estimated graph by MRPC using gaussCItest and beta = 0.005
MRPC.fit <- MRPC(data, suffStat_R, GV = 1,
                 FDR = 0.05, indepTest ='gaussCItest',
                 labels = V, verbose = TRUE)


# use methods of class MRPCclass
show(MRPC.fit)

plot(MRPC.fit)
summary(MRPC.fit)

# access slots of this object
(g  <- MRPC.fit@graph)
str(ss <- MRPC.fit@sepset, max = 1)
```

MRPCtruth *Truth Models for MRPC Examples*

Description

Truth of the five basic topologies and three common topologies in biology, multi-parents, star and layered.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

Examples

```
data("MRPCtruth") #load data

par(mfrow = c(2, 4))
plot(MRPCtruth$M0,
     main = "Model0")
plot(MRPCtruth$M1,
     main = "Model1")
plot(MRPCtruth$M2,
     main = "Model2")
plot(MRPCtruth$M3,
     main = "Model3")
plot(MRPCtruth$M4,
     main = "Model4")
plot(MRPCtruth$Multiparent,
     main = "Multiparent")
plot(MRPCtruth$Star,
     main = "Star")
plot(MRPCtruth$Layered,
     main = "Layered")
```

Recall_Precision *Performance Evaluation by Recall and Precision in MRPC*

Description

Recall = (# edges correctly identified in inferred graph) / (# edges in true graph); Precision = (# edges correctly identified in inferred graph) / (# edges in inferred graph).For example, when the true graph is V–>T1–>T2, and the inferred graphs are i) V–>T1–>T2, and V–>T2; ii) V–>T1–T2; and iii) V–>T1<–T2, the number of correctly identified edges is then 2, 1.5, and 1.5, respectively. Recall is calculated to be 1, 0.75, and 0.75 respectively, whereas precision is 2/3 = 0.67, 1.5/2 = 0.75, and 1.5/2 = 0.75, respectively.

Usage

```
Recall_Precision(g1, g2, GV, edge.presence = 1.0, edge.direction = 0.5)
```

Arguments

g1	First graph object, from the true graph
g2	Second graph object, from the inferred graph
GV	The number of genetic variants (SNPs/indels/CNV/eQTL) in the input data. For example, if the data has one genetic variant, first column, then GV = 1, if 2, 1st and 2nd Column, then GV = 2, and so on.

`edge.presence` The weight for an edge being present.

`edge.direction` The weight for the edge direction.

Details

We consider it more important to be able to identify the presence of an edge than to also get the direct correct. Therefore, we assign 1 to an edge with the correct direction and 0.5 to an edge with the wrong direction or no direction (Badsha et al., 2018).

Value

A list of object that containing the following:

- `Matrix`: Results store for TP and FP
- `TP`: Total found edges in the inferred graph and edge exists in the true graph.
- `FP`: Total found edges in the inferred graph but no edge exists in the true graph.
- `Recall`: Power, or sensitivity measures how many edges from the true graph a method can recover.
- `Precision`: Measures how many correct edges are recovered in the inferred graph.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

References

1.Md. Bahadur Badsha, Audrey Qiuyan Fu: Learning causal biological networks with the principle of Mendelian randomization (Nat Comn; under reveiw 2018),

See Also

aSHD: adjusted Structural Hamming Distance (aSHD)

Examples

```
# True model
# True graph (V1 --> T1 --> T2 --> T3)
tarmat_s1 <- matrix(0,
                    nrow = 4,
                    ncol = 4)

colnames(tarmat_s1) <- c("V1", "T1", "T2", "T3")

rownames(tarmat_s1) <- colnames(tarmat_s1)

# Create an adjacency matrix for the true graph
tarmat_s1[1, 2] <- 1
tarmat_s1[2, 3] <- 1
tarmat_s1[3, 4] <- 1
```

```
# Graph object of the true graph
Truth <- as(tarmat_s1,
            "graphNEL")

# Inferred graph (V1 --> T1 <-- T2 --> T3)
tarmat_s2 <- matrix(0,
                    nrow = 4,
                    ncol = 4)

colnames(tarmat_s2) <-c ("V1", "T1", "T2", "T3")

rownames(tarmat_s2) <- colnames(tarmat_s2)

# Create an adjacency matrix for the inferred graph
tarmat_s2[1, 2] <- 1
tarmat_s2[3, 2] <- 1
tarmat_s2[3, 4] <- 1

# Graph objects for the inferred graph
Inferred <- as(tarmat_s2,
               "graphNEL")

#Recall and Precision
Recall_Precision(Truth,
                 Inferred,
                 GV = 1,
                 edge.presence = 1.0,
                 edge.direction = 0.5)
```

RobustCor *Robust Correlation Matrix*

Description

Robust correlation matrix based on beta value. The value of beta plays a key role in the performance of the robust method, which controls the tradeoff between the robustness and efficiency of the estimators.

Usage

```
RobustCor(xx, Beta, plot = FALSE)
```

Arguments

xx	Data matrix
Beta	Tuning parameter, between 0 and 1, if 0 then equal to nonrobust, classical method. We suggest using, Beta = 0.005 in both without and with outliers in simulation study. This value should reflect the amount of outliers in the data.

Whereas a large value increases robustness, it reduces sensitivity of identifying an edge. We need a more principled way to determine this value.

`plot` To set no plotting as the default for weight vs gene index.

Details

We take a robust approach and calculate the robust correlation matrix (Badsha et al., 2013) on which the series of hypothesis testing is performed. The performance of the robust correlation method depends on the values of the tuning parameter beta. It controls the tradeoff between robustness and efficiency of estimators. This method shows high performance for a wide range of beta. The values of beta lies between 0 and 1, such that a large value of beta decreases the efficiency, while it increases the robustness of an estimator, and vice-versa for a small value of beta. Thus, we need to select an optimal beta to obtain both high robustness and efficiency, while it depends on the initialization of model parameters, data contamination rates, types of data contamination, types of datasets, and so on. We used the beta value from Badsha et al., 2013. The robust method reduces to the classical method (Biased estimator) with the tuning parameter beta –>0. When the data matrix contains missing values, we perform imputation using the R package mice (Buuren and Groothuis-Oudshoorn, 2011).

Value

list of objects as follows:

- `RR`: Robust correlation matrix.
- `M`: Robust mean vector.
- `V`: Robust covariance matrix.
- `Wt`: Weight for each observation.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

References

1.Badsha, M. B. Mollah, M. N. Jahan, N. and Kurata, H. (2013). Robust complementary hierarchical clustering for gene expression data analysis by beta-divergence. J Biosci Bioeng 116(3): 397-407. 2. Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software, 45(3), 1-67. http://www.jstatsoft.org/v45/i03/

Examples

```
R <- RobustCor(simu.data_M0,
               Beta = 0.005,
               plot = FALSE)

RCor <- R$RR  # Correlation matrix
```

seqDiff *Deviation of the Two Sequnces*

Description

Deviation of the inferred graph from the true graph.

Usage

```
seqDiff(g1, g2)
```

Arguments

g1	Adjacency matrix from the first graph object.
g2	Adjacency matrix from the second graph object.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

Examples

```
# True model
# True graph (V1 --> T1 --> T2 --> T3)
tarmat_s1 <- matrix(0,
                    nrow = 4,
                    ncol = 4)

colnames(tarmat_s1) <- c("V1", "T1", "T2", "T3")

rownames(tarmat_s1) <- colnames(tarmat_s1)

# Create an adjacency matrix for the true graph
tarmat_s1[1, 2] <- 1
tarmat_s1[2, 3] <- 1
tarmat_s1[3, 4] <- 1

# Inferred graph (V1 --> T1 <-- T2 --> T3)
tarmat_s2 <- matrix(0,
                    nrow = 4,
                    ncol = 4)

colnames(tarmat_s2) <-c ("V1", "T1", "T2", "T3")

rownames(tarmat_s2) <- colnames(tarmat_s2)

# Create an adjacency matrix for the inferred graph
tarmat_s2[1, 2] <- 1
```

```
tarmat_s2[3, 2] <- 1
tarmat_s2[3, 4] <- 1

# Deviation of the inferred graph from the true graph.
seqDiff(tarmat_s2,
        tarmat_s1)
```

SeqFDR *Sequential FDR*

Description

Sequential FDR method that controls the FDR and mFDR in an online manner.

Usage

```
SeqFDR(m, FDR, a=2, R)
```

Arguments

m	The number of current the test.
FDR	FDR level.
a	A constant.
R	All of the decisions from the tests that have already been performed.

Details

We used the LOND (significance Levels based On Number of Discoveries) algorithm that controls FDR and mFDR in an online manner (Javanmard and Montanari, 2015). Where the significance level, alpha, is based on the total number of discoveries made so far. Which is similar to the algorithm called alpha-investing rules introduced by (Foster and Staine, 2007) to control only mFDR in an online manner.

Value

The value of alpha.

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

References

[1] Javanmard and Montanari (March 5, 2015): On Online Control of False Discovery Rate. arXiv:150206197 [statME].

[2] Foster, D. P. and Stine, R. A. (2007): Alpha-investing: A procedure for sequential control of expected false discoveries. http://gosset.wharton.upenn.edu/research/edc.pdf,.

See Also

MRPC for estimating a DAG using the Mendelian Randomization (MR) based (MRPC) algorithm; ModiSkeleton for estimating a skeleton using modified skeleton function.

`simu.data_layered` *Data for the Layered Model*

Description

Simulate Data for the Layered Model.

Details

The column order of the input matrix is the genotype from Single-nucleotide polymorphism (SNPs), indels, copy number variation (CNV), or Expression quantitative trait loci (eQTL) to genes. If we consider one SNPs/CNV/indels/CNV/eQTL then the first column of the input matrix is the SNPs/CNV/indels/CNV/eQTL and the remaining columns are the gene expression data.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
## Layered Model
simu.data_layered <- SimulatedData(N = 10^3, p = 0.45,
                                   'layered', b0.1 = 0,
                                   b1.1 = 1, b1.2 = 1,
                                   b1.3 = 1, sd.1 = 1)
```

simu.data_M0 *Data for Model 0*

Description

Simulate Data for Model 0.

Details

The column order of the input matrix is the genotype from Single-nucleotide polymorphism (SNPs), indels, copy number variation (CNV), or Expression quantitative trait loci (eQTL) to genes. If we consider one SNPs/CNV/indels/CNV/eQTL then the first column of the input matrix is the SNPs/CNV/indels/CNV/eQTL and the remaining columns are the gene expression data.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
## Model 0
simu.data_M0 <- SimulatedData(N = 10^3, p = 0.45,
                              'model0', b0.1 = 0,
                              b1.1 = 1, b1.2 = 1,
                              b1.3 = 1, sd.1 = 1)
```

simu.data_M1 *Data for Model 1*

Description

Simulate Data for Model 1.

Details

The column order of the input matrix is the genotype from Single-nucleotide polymorphism (SNPs), indels, copy number variation (CNV), or Expression quantitative trait loci (eQTL) to genes. If we consider one SNPs/CNV/indels/CNV/eQTL then the first column of the input matrix is the SNPs/CNV/indels/CNV/eQTL and the remaining columns are the gene expression data.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
## Model 1
simu.data_M1 <- SimulatedData(N = 10^3, p = 0.45,
                              'model1', b0.1 = 0,
                              b1.1 = 1, b1.2 = 1,
                              b1.3 = 1, sd.1 = 1)
```

---

simu.data_M2 *Data for Model 2*

Description

Simulate Data for Model 2.

Details

The column order of the input matrix is the genotype from Single-nucleotide polymorphism (SNPs), indels, copy number variation (CNV), or Expression quantitative trait loci (eQTL) to genes. If we consider one SNPs/CNV/indels/CNV/eQTL then the first column of the input matrix is the SNPs/CNV/indels/CNV/eQTL and the remaining columns are the gene expression data.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
## Model 2
simu.data_M2 <- SimulatedData(N = 10^3, p = 0.45,
                              'model2', b0.1 = 0,
                              b1.1 = 1, b1.2 = 1,
                              b1.3 = 1, sd.1 = 1)
```

`simu.data_M3` *Data for Model 3*

Description

Simulate Data for Model 3.

Details

The column order of the input matrix is the genotype from Single-nucleotide polymorphism (SNPs), indels, copy number variation (CNV), or Expression quantitative trait loci (eQTL) to genes. If we consider one SNPs/CNV/indels/CNV/eQTL then the first column of the input matrix is the SNPs/CNV/indels/CNV/eQTL and the remaining columns are the gene expression data.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
## Model 3
simu.data_M3 <- SimulatedData(N = 10^3, p = 0.45,
                              'model3', b0.1 = 0,
                              b1.1 = 1, b1.2 = 1,
                              b1.3 = 1, sd.1 = 1)
```

simu.data_M4 *Data for Model 4*

Description

Simulate Data for Model 4.

Details

The column order of the input matrix is the genotype from Single-nucleotide polymorphism (SNPs), indels, copy number variation (CNV), or Expression quantitative trait loci (eQTL) to genes. If we consider one SNPs/CNV/indels/CNV/eQTL then the first column of the input matrix is the SNPs/CNV/indels/CNV/eQTL and the remaining columns are the gene expression data.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
## Model 4
simu.data_M4 <- SimulatedData(N = 10^3, p = 0.45,
                              'model4', b0.1 = 0,
                              b1.1 = 1, b1.2 = 1,
                              b1.3 = 1, sd.1 = 1)
```

simu.data_multiparent *Data for Multiple Parent Model*

Description

Simulate data for the multiple parent model, where the parents can be eQTL/CNVs along with several gene expression nodes.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
## Multiple Parent Model
simu.data_multiparent <- SimulatedData(N = 10^3, p = 0.45,
                                       'multiparent', b0.1 = 0,
                                       b1.1 = 1, b1.2 = 1,
                                       b1.3 = 1, sd.1 = 1)
```

---

simu.data_starshaped *Data for Star Model*

Description

Simulate data for the start model, where one gene has more than two children.

Details

The column order of the input matrix is the genotype from Single-nucleotide polymorphism (SNPs), indels, copy number variation (CNV), or Expression quantitative trait loci (eQTL) to genes. If we consider one SNPs/CNV/indels/CNV/eQTL then the first column of the input matrix is the SNPs/CNV/indels/CNV/eQTL and the remaining columns are the gene expression data.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

SimulatedData for simulated data generating function.

Examples

```
## Star Model
simu.data_starshaped <- SimulatedData(N = 10^3, p = 0.45,
                                      'starshaped', b0.1 = 0,
                                      b1.1 = 1, b1.2 = 1,
                                      b1.3 = 1, sd.1 = 1)
```

SimulatedData *Generating Simulated Data*

Description

Function for simulating data.

Usage

```
SimulatedData(N, p, model, b0.1, b1.1, b1.2, b1.3, sd.1)
```

Arguments

N	The number of observations.
p	Population frequency of the reference allele. Real number between 0 to 1, which is the number of a particular allele is present.
model	The model for which data will be simulated. For example, if you want to generate data for model 0 you would type 'model0' into the function.
b0.1	Intercept of b0.1 + b1.1*P1 + b1.2*P2 + b1.3*P3, where P1, P2, and P3 are the parents of the corresponding gene.
b1.1	Slope of P1 for b0.1 + b1.1*P1 + b1.2*P2 + b1.3*P3, where P1, P2, and P3 are the parents of the corresponding gene.
b1.2	Slope of P2 for b0.1 + b1.1*P1 + b1.2*P2 + b1.3*P3, where P1, P2, and P3 are the parents of the corresponding gene.
b1.3	Slope of P3 for b0.1 + b1.1*P1 + b1.2*P2 + b1.3*P3, where P1, P2, and P3 are the parents of the corresponding gene.
sd.1	Standard deviation for corresponding data generated gene.

Details

The first column of the input matrix is the genotype of the expression quantitative trait loci (eQTL)/Copy number variation (CNVs) and the remaining columns are the gene expression data.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

See Also

MRPC for estimating a DAG using the modified pc (MRPC) algorithm; ModiSkeleton for estimating a undirected graph using modified skeleton function; EdgeOrientation for orientation rules for edges in MRPC algorithm; Case_NP the gene to have data generated has no parent; Case_1P the gene to have data generated has one parent; Case_2P the gene to have data generated has two parents.

Examples

```
# Data pre-processing

# If you use only one genotype of the expression quantitative trait loci
# (eQTL)/Copy number variation (CNV), the 1st column of
# the input matrix will be #eQTL/CNV and the remaining
# columns are the gene expression data.


## Model 0
simu.data_M0 <- SimulatedData(N = 10^3, p = 0.45,
                              'model0', b0.1 = 0,
                              b1.1 = 1, b1.2 = 1,
                              b1.3 = 1, sd.1 = 1)

## Model 1
simu.data_M1 <- SimulatedData(N = 10^3, p = 0.45,
                              'model1', b0.1 = 0,
                              b1.1 = 1, b1.2 = 1,
                              b1.3 = 1, sd.1 = 1)

## Model 2
simu.data_M2 <- SimulatedData(N = 10^3, p = 0.45,
                              'model2', b0.1 = 0,
                              b1.1 = 1, b1.2 = 1,
                              b1.3 = 1, sd.1 = 1)

## Model 3
simu.data_M3 <- SimulatedData(N = 10^3, p = 0.45,
                              'model3', b0.1 = 0,
                              b1.1 = 1, b1.2 = 1,
                              b1.3 = 1, sd.1 = 1)

## Model 4
simu.data_M4 <- SimulatedData(N = 10^3, p = 0.45,
                              'model4', b0.1 = 0,
                              b1.1 = 1, b1.2 = 1,
                              b1.3 = 1, sd.1 = 1)

## Multiple Parent Model
simu.data_multiparent <- SimulatedData(N = 10^3, p = 0.45,
                                       'multiparent', b0.1 = 0,
                                       b1.1 = 1, b1.2 = 1,
```

```
                                        b1.3 = 1, sd.1 = 1)

## Star Model
simu.data_starshaped <- SimulatedData(N = 10^3, p = 0.45,
                                      'starshaped', b0.1 = 0,
                                      b1.1 = 1, b1.2 = 1,
                                      b1.3 = 1, sd.1 = 1)

## Layered Model
simu.data_layered <- SimulatedData(N = 10^3, p = 0.45,
                                    'layered', b0.1 = 0,
                                    b1.1 = 1, b1.2 = 1,
                                    b1.3 = 1, sd.1 = 1)
```

`SimulationDemo` *Same data but different node orderings.*

Description

Investigate the impact of the same data but with different node orderings across three methods: MRPC (Badsha et al., 2018), the mmhc method (implemented in bnlearn; Scutari, 2010) and the pc method (implemented in pcalg; Kalisch et al., 2012).

Usage

```
SimulationDemo(N, model, signal, n_data, n_nodeordering)
```

Arguments

`N`	The number of observation.
`model`	Either 'truth1' or 'truth2' to specify the model you want to generate data from.
`signal`	The signal strength which is the coefficient of the parent nodes in the linear model.
`n_data`	The number of data sets you want to generate for testing the results of the three methods.
`n_nodeordering`	The number of times you want to reorder the nodes for testing the results of the three methods.

Details

The output is a matrix where each row indicates the data sets and the columns are the results of the different node orderings by three methods MRPC, mmhc, and pc respectively.

Value

Matrix

Author(s)

Md Bahadur Badsha (mdbadsha@uidaho.edu)

References

1.Md. Bahadur Badsha, Audrey Qiuyan Fu: Learning causal biological networks with generalized Mendelian randomization.bioRxiv (2017), 171348.

2.Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

3.Kalisch, M., Machler, M., Colombo, D., Maathuis, M.H. & Buhlmann, P. Causal Inference Using Graphical Models with the R Package pcalg. J. Stat.Softw. 47, 26 (2012).

Examples

```
# We want to generate 2 different data sets for truth1 with
# signal = 1, N = 100 and 6 different node orderings. Therefore,
# you will get 2 by 18 output matrix, where first and second row
# are the data 1 and 2 respectively. Column 1:6 is the result of
# MRPC with node order 1, 2, ..., 6, then column 7:12 is for mmhc and
# column 13:18 is for pc.

library(bnlearn) # for mmhc
library(pcalg)   # for pc

# Run
Output <- SimulationDemo(N = 100, 'truth1',
                         signal = 1, n_data = 2,
                         n_nodeordering = 6)
```

Index