Package 'MRPC'

January 24, 2021

Type Package

Version 2.2.2

Date 2021-01-08

Title PC Algorithm with the Principle of Mendelian Randomization

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

Maintainer Md Bahadur Badsha <mbbadshar@gmail.com>

Description A PC Algorithm with the Principle of Mendelian Randomization. This package implements the MRPC
(PC with the principle of Mendelian randomization) algorithm to infer causal graphs. It also contains functions to simulate data under a certain topology, to visualize a graph in different ways, and to compare graphs and quantify the differences.
See Badsha and Fu (2019) <doi:10.3389/fgene.2019.00460>,Badsha, Martin and Fu (2018) <arXiv:1806.01899>.

License GPL (>= 2)

Depends R (>= 3.0.0)

LazyData TRUE

Imports
bnlearn,compositions,dynamicTreeCut,GGally,fastcluster,gtools,graph,graphics,Hmisc,methods,mice,network,pcalg,psych,Rgraphviz,stats,sna,utils,WGCNA,plyr

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-24 06:40:02 UTC

R topics documented:

`AdjustMatrix` *Adjust the columns of the input matrix same as in the reference matrix*

Description

We adjusted the columns of the input matrix to have the same ordering as in the reference matrix.

Usage

```
AdjustMatrix(reference, input)
```

Arguments

reference	The reference matrix.
input	The input matrix.

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

Examples

```
# Reference matrix

reference <- matrix(0,nrow=4,ncol = 4)
colnames(reference) <- c("V1","T1","T2","T3")
rownames(reference) <- colnames(reference)

# Adjacency matrix for reference
reference[1,2] <- 1
reference[2,3] <- 1
reference[3,4] <- 1

# Input matrix
input <- matrix(0,
                nrow = 4,
                ncol = 4)
colnames(input) <- c("V1","T2","T3","T1")
rownames(input) <- colnames(input)

# Adjacency matrix for input
input[1,2] <- 1
input[2,3] <- 1
input[3,4] <- 1

# Adjust the columns of the input matrix same as in the reference matrix
AdjustMatrix <- AdjustMatrix (reference, input)
```

---

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 (mbbadshar@gmail.com)

References

1. Kalisch M, Machler M, Colombo D, Maathuis MH and Buhlmann P (2012). Causal Inference Using Graphical Models with the R Package pcalg. Journal of Statistical Software, 47, 26.

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

Examples

```
# True model (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)
```

`CompareMethodsNodeOrdering` *Comparison of inference accuracy using the same data but with different node orderings.*

Description

Investigate the performance of five methods on the same data but with different node orderings: MRPC (Badsha and Fu, 2019; Badsha et al., 2018), pc, implemented in `pcalg` (Kalisch et al., 2012), and pc.stable, mmpc, mmhc, and hc, the last four all implemented in `bnlearn` (Scutari, 2010). See details in Badsha et al., 2018.

Usage

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

Arguments

`N`	The number of observations.

signal	The signal strength which is the coefficient of the parent nodes in the linear model.
model	Either 'truth1' or 'truth2' to specify the model to generate data from.
n_data	The number of independent data sets to generate.
n_nodeordering	The number of times to reorder the nodes.

Details

We generated different data sets from each of the two graphs (V1–>T1–>T2–>T3 and V1–>T1<–T2–>T3), where V1 is the genetic varitant node and T1, T2 and T3 are the phenotype nodes. For each data set, we reordered the columns of the data matrix with different permutations of the T nodes, and thus generated new permuted data sets. We then applied all the methods to each of the data sets (restricting edge direction wherever necessary and possible), obtained different inferred graphs, and counted the number of unique graphs among the inferred graphs for each data set.

The output is the number of unique graphs inferred by each method across all permutations.The columns are indicates which methods (MRPC, pc, pc.stable, mmpc mmhc, and hc), and each rows are indicates the independent data sets under different node orderings. See details in Badsha et al., 2018.

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

References

1. Badsha MB and Fu AQ (2019). Learning causal biological networks with the principle of Mendelian randomization. Frontiers in Genetics, 10(460).

2. Badsha MB, Martin EA and Fu AQ (2018). MRPC: An R package for accurate inference of causal graphs. arXiv:1806.01899.

3. Kalisch M, Machler M, Colombo D, Maathuis MH and Buhlmann P (2012). Causal Inference Using Graphical Models with the R Package pcalg. Journal of Statistical Software, 47, 26.

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

Examples

```
# We will generate 10 different data sets from truth1 (V1-->T1-->T2-->T3)
# with signal = 1.0, N = 100 and 6 different T nodes orderings. Therefore, we
# will get 10 by 6 output matrix, where the rows are 10 independent data
# sets and columns are for six methods MRPC, pc, pc.stable, mmpc
# mmhc, and hc respectively.

library(MRPC)    #  MRPC
library(pcalg)   #  pc
```

```
library(bnlearn) #  pc.stable, mmpc, mmhc, and hc

# Run
Output <- CompareMethodsNodeOrdering(N = 100,
                                     signal = 1.0,
                                     model = 'truth1',
                                     n_data = 10,
                                     n_nodeordering = 6)
```

`CompareMethodsVStructure` *Comparison of inference accuracy of different methods on data with and without a v-structure*

Description

This function compares inference accuracy on graphs with and without a v-structure in terms of recall and precision by six methods MRPC, pc, pc.stable, mmpc, mmhc, and hc, across multiple data sets. See details in Badsha and Fu, 2019 and Badsha et al., 2018.

Usage

```
CompareMethodsVStructure(N, signal, model, includeGV, ita)
```

Arguments

`N`	Number of observations.
`signal`	The coefficient of parent nodes in the linear model. For example, strong = 1.0, moderate = 0.5, and weak = 0.2.
`model`	The graph from which the data is generated. Specifically, two graphs are considered here: 'model 1' (V1->T1->T2), which does not contain a v-structure, and 'model 2' (V1->T1<-T2), which is a v-structure.
`includeGV`	If TRUE, include edges involving genetic variants (GVs) when comparing the true and inferred graphs. If FALSE, exclude such edges.
`ita`	Number of independent data sets to simulate.

Details

The output is a matrix, where the rows are the six methods: MRPC, pc, pc.stable, mmpc, mmhc, and hc, and the columns are the mean of recall, sd of recall, mean of precision, and sd of precision, respectively. Mean and sd are calculated across all the simulated data sets. For methods from the bnlearn package (pc.stable, mmpc, mmhc, and hc), we apply the blacklist argument to exclude edges pointing at the genetic variant, and therefore evaluate recall and precision including the edges involving these edges (i.e., includeGV = TRUE).

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

References

1. Badsha MB and Fu AQ (2019). Learning causal biological networks with the principle of Mendelian randomization. Frontiers in Genetics, 10(460).

2. Badsha MB, Martin EA and Fu AQ (2018). MRPC: An R package for accurate inference of causal graphs. arXiv:1806.01899.

3. Kalisch M, Machler M, Colombo D, Maathuis MH and Buhlmann P (2012). Causal Inference Using Graphical Models with the R Package pcalg. Journal of Statistical Software, 47, 26.

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

See Also

RecallPrecision: Performance evaluation in terms of recall and precision.

Examples

```
# For demonstration purposes, only 10 data sets
# with a sample size of 100 and signal is 1, are simulated here.

# Comparison of inference accuracy on model 1 without a v-structure
Result1 <- CompareMethodsVStructure(N = 100,
                                    signal = 1.0,
                                    'model1',
                                    includeGV = TRUE,
                                    ita = 10)

# Comparison of inference accuracy with a v-structure
Result2 <- CompareMethodsVStructure(N = 100,
                                    signal = 1.0,
                                    'model2',
                                    includeGV = TRUE,
                                    ita = 10)
```

CutModules *Cut a numeric variable into intervals*

Description

Similar to cut2 function with some modification.

Usage

```
CutModules(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 (mbbadshar@gmail.com)

---

data_examples *Example data under a simple and complex models*

Description

Example data under the simple and complex graphs. Data may be continuous or discrete.

Usage

```
data(data_examples)
```

Details

For each model, the graph and a simulated data matrix are available 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 (mbbadshar@gmail.com)

Examples

```
# Continuous data with genetic varitant (GV)
# load the data
data("data_examples")

# Extract the sample size
n <- nrow(data_examples$simple$cont$withGV$data)

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

# Calculate Pearson correlation
suffStat_C <- list(C = cor(data_examples$simple$cont$withGV$data),
                   n = n)
```

```
# Infer the graph by MRPC
data.mrpc.cont.withGV <- MRPC(data = data_examples$simple$cont$withGV$data,
                              suffStat = suffStat_C,
                              GV = 2,
                              FDR = 0.05,
                              indepTest = 'gaussCItest',
                              labels = V,
                              FDRcontrol = TRUE,
                              verbose = FALSE)
# Plot the results
par(mfrow = c(1, 2))
# plot the true graph
plot(data_examples$simple$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("data_examples")

# Extract the sample size
n <- nrow(data_examples$simple$cont$withoutGV$data)

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

# Calculate Pearson correlation
suffStat_C <- list(C = cor(data_examples$simple$cont$withoutGV$data),
                   n = n)

# Infer the graph by MRPC
data.mrpc.cont.withoutGV <- MRPC(data = data_examples$simple$cont$withoutGV$data,
                                 suffStat = suffStat_C,
                                 GV = 0,
                                 FDR = 0.05,
                                 indepTest = 'gaussCItest',
                                 labels = V,
                                 FDRcontrol = TRUE,
                                 verbose = FALSE)
# Plot the results
par(mfrow = c(1, 2))
# plot the true graph
plot(data_examples$simple$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("data_examples")

# Extract the sample size
n <- nrow(data_examples$simple$disc$withGV$data)

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

# Calculate Pearson correlation
suffStat_C <- list(C = cor(data_examples$simple$disc$withGV$data),
                   n = n)

# Infer the graph by MRPC
data.mrpc.disc.withGV <- MRPC(data = data_examples$simple$disc$withGV$data,
                              suffStat = suffStat_C,
                              GV = 1,
                              FDR = 0.05,
                              indepTest = 'gaussCItest',
                              labels = V,
                              FDRcontrol = TRUE,
                              verbose = FALSE)
# Plot the results
par (mfrow = c(1, 2))
# plot the true graph
plot(data_examples$simple$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("data_examples")

# Extract the sample size
n <- nrow (data_examples$simple$disc$withoutGV$data)

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

# Calculate Pearson correlation
suffStat_C <- list(C = cor(data_examples$simple$disc$withoutGV$data),
                   n = n)

# Infer the graph by MRPC
data.mrpc.disc.withoutGV <- MRPC(data = data_examples$simple$disc$withoutGV$data,
                                 suffStat = suffStat_C,
                                 GV = 1,
                                 FDR = 0.05,
                                 indepTest = 'gaussCItest',
                                 labels = V,
                                 FDRcontrol = TRUE,
```

```
                                    verbose = FALSE)
# Plot the results
par(mfrow = c(1, 2))
# plot the true graph
plot(data_examples$simple$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("data_examples")

# Graph without clustering
plot(data_examples$complex$cont$withGV$graph)

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

# Plot of dendrogram with modules colors of nodes
PlotDendrogramObj <- PlotDendrogram(Adj_directed,
                                    minModuleSize = 5)

# Visualization of inferred graph with modules colors
PlotGraphWithModulesObj <- PlotGraphWithModules(Adj_directed,
                                                PlotDendrogramObj,
                                                GV = 14,
                                                node.size = 8,
                                                arrow.size = 5,
                                                label.size = 3,
                                                alpha = 1)
# plot
plot(PlotGraphWithModulesObj)
```

`data_GEUVADIS` *GEUVADIS data with 62 eQTL-gene sets*

Description

The GEUVADIS (Lappalainen et al., 2013) data (i.e., gene expression) measured in Lymphoblastoid Cell Lines (LCLs) on a subset of individuals from the 1,000 Genomes Project including 373 Europeans and 89 Africans.

Details

The GEUVADIS (Genetic European Variation in Disease) project identified eQTLs across the human genome. Among these eQTLs, ~70 have more than one target gene. Additionally, we found 62 unique eQTLs which exhibit pleiotropy. We extracted the genotypes of these 62 eQTLs and the expression of the target genes for 373 Europeans and 89 Africans (see Badsha and Fu, 2019).

Value

A list that contains 62 eQTL-gene sets data for 373 Europeans and 89 Africans.

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

References

1. Badsha MB and Fu AQ (2019). Learning causal biological networks with the principle of Mendelian randomization. Frontiers in Genetics, 10(460).

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

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_GEUVADIS_combined`

Combined genotype and gene expression data from 62 eQTL-gene sets in 373 Europeans from GEUVADIS

Description

The genotype and gene expression data of 62 eQTL-gene sets in 373 Europeans from the GEUVADIS consortium (Lappalainen et al., 2013) are combined into one data matrix. Each of these eQTLs has been identified to be associated with more than one gene (see details in Badsha and Fu, 2019).

Details

The data set contains 373 samples in rows and 194 variables (62 eQTLs and 132 genes) in columns. Specifically, the columns are: eQTL1, gene1 for eQTL1, gene2 for eQTL1, eQTL2, gene1 for eQTL2, gene2 for eQTL2 and so on.

For analysis, we account for potential confounding variables as additional nodes in the graph. To do so, we first perform Principal Component Analysis (PCA) on the entire gene expression matrix from the European samples in GEUVADIS, and extract the top 10 PCs as potential confounding variables. We next examine the statistical association between each of the top PCs and the eQTL-gene sets, and identify statistically significant associations (accounting for multiple testing with the q vlaue method). We then apply MRPC to each eQTL-gene set with its associated PCs. See details in the examples below. Also see Badsha and Fu (2019) and Badsha et al. (2018).

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

References

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

2. Badsha MB and Fu AQ (2019). Learning causal biological networks with the principle of Mendelian randomization. Frontiers in Genetics, 10(460).

3. Badsha MB, Martin EA and Fu AQ (2018). MRPC: An R package for accurate inference of causal graphs. arXiv:1806.01899.

See Also

data_GEUVADIS

Examples

```
# Examining principal components (PCs) as potential confounders in analysis of the GEUVADIS data

library(MRPC) # MRPC

# Load genomewide gene expression data in GEUVADIS
# 373 individuals
# 23722 genes
data_githubURL <- "https://github.com/audreyqyfu/mrpc_data/raw/master/data_GEUVADIS_allgenes.RData"
load(url(data_githubURL))

# Run PCA
library(stats) # prcomp
PCs <- prcomp(data_GEUVADIS_allgenes,scale=TRUE)
# Extract the PCs
PCs_matrix <- PCs$x

# Load the 62 eQTL-gene sets
# 373 individuals
# 194 variables (eQTLs=62 and genes=132)
data("data_GEUVADIS_combined")

# Identify PCs that are significantly associated with eQTL-gene sets
# Compute the correlation and corresponding p values between the top PCs and the eQTLs and genes
library(psych) # to use corr.test
no_PCs <- 10
corr_PCs <- corr.test(PCs_matrix[,1:no_PCs],data_GEUVADIS_combined)
# The correlation matrix
```

```
corr_matrix <- corr_PCs$r
# The p values
Pvalues <- corr_PCs$p
# Apply the q value method at FDR of 0.05
library(WGCNA) # qvalue
qobj <- qvalue(Pvalues, fdr.level=0.05,robust = TRUE)

# Significant associations
Significant_asso <- qobj$significant
List_significant_asso <- which(Significant_asso, arr.ind = TRUE, useNames = TRUE)
# 1st column contains the PCs
# 2nd column contains the associated eQTLs or genes
List_significant_asso[1:10,]

# Examples of eQTLs or genes that are significantly associated with selected PCs
# PC1
eqtl.genes_PC1 <- colnames(data_GEUVADIS_combined)[List_significant_asso
                           [which(List_significant_asso[,1]=="1"),2]]
print(eqtl.genes_PC1)
# PC2
eqtl.genes_PC2 <- colnames(data_GEUVADIS_combined)[List_significant_asso
                           [which(List_significant_asso[,1]=="2"),2]]
print(eqtl.genes_PC2)
# PC3
eqtl.genes_PC3 <- colnames(data_GEUVADIS_combined)[List_significant_asso
                           [which(List_significant_asso[,1]=="3"),2]]
print(eqtl.genes_PC3)


#-------------
# Example 1
# Gene SBF2-AS1 is significantly associated with PC2
print(eqtl.genes_PC2[24])

# Gene SBF2-AS1 is in the eQTL-gene set #50 with snp rs7124238 and gene SWAP70
data_GEU_Q50 <- data_GEUVADIS$Data_Q50$Data_EUR
colnames(data_GEU_Q50) <- c("rs7124238","SBF2-AS1","SWAP70")

# Analyze the eQTL-gene set without PC2
n <- nrow (data_GEU_Q50)        # Number of rows
V <- colnames(data_GEU_Q50)     # Column names

# Calculate Pearson correlation
suffStat_C_Q50 <- list(C = cor(data_GEU_Q50, use = 'pairwise.complete.obs'),
                       n = n)

# Infer the graph by MRPC
MRPC.fit_withoutPC_GEU_Q50 <- MRPC(data_GEU_Q50,
                                   suffStat = suffStat_C_Q50,
                                   GV = 1,
                                   FDR = 0.05,
                                   indepTest = 'gaussCItest',
                                   labels = V,
```

```
                                  FDRcontrol = TRUE,
                                  verbose = FALSE)

# Analyze the eQTL-gene set with PC2
data_withPC_Q50 <- cbind(data_GEU_Q50,PCs_matrix[,2])
colnames(data_withPC_Q50)[4] <- "PC2"

n <- nrow (data_withPC_Q50)        # Number of rows
V <- colnames(data_withPC_Q50)     # Column names

# Calculate Pearson correlation
suffStat_C_withPC_Q50 <- list(C = cor(data_withPC_Q50, use = 'pairwise.complete.obs'),
                              n = n)

# Infer the graph by MRPC
MRPC.fit_withPC_GEU_Q50 <- MRPC(data_withPC_Q50,
                                suffStat = suffStat_C_withPC_Q50,
                                GV = 1,
                                FDR = 0.05,
                                indepTest = 'gaussCItest',
                                labels = V,
                                FDRcontrol = TRUE,
                                verbose = FALSE)

# Plot inferred graphs
par(mfrow=c(1,2))
plot(MRPC.fit_withoutPC_GEU_Q50,
     main = "Without PC" )
plot(MRPC.fit_withPC_GEU_Q50,
     main = "Without PC")

#-------------
# Example 2
# Gene LCMT2 is significantly associated with PC1
print(eqtl.genes_PC1[8])

# Gene LCMT2 is in the eQTL-gene set #29 with snp rs2278858 and gene ADAL
data_GEU_Q29 <- data_GEUVADIS$Data_Q29$Data_EUR
colnames(data_GEU_Q29) <- c("rs2278858", "LCMT2", "ADAL")

# Analyze the eQTL-gene set without PC1
n <- nrow (data_GEU_Q29)        # Number of rows
V <- colnames(data_GEU_Q29)     # Column names

# Calculate Pearson correlation
suffStat_C_Q29 <- list(C = cor(data_GEU_Q29, use = 'pairwise.complete.obs'),
                       n = n)

# Infer the graph by MRPC
MRPC.fit_withoutPC_GEU_Q29 <- MRPC(data_GEU_Q29,
                                   suffStat = suffStat_C_Q29,
                                   GV = 1,
                                   FDR = 0.05,
```

```
                                    indepTest = 'gaussCItest',
                                    labels = V,
                                    FDRcontrol = TRUE,
                                    verbose = FALSE)

# Analyze the eQTL-gene set with PC1
data_withPC_Q29 <- cbind(data_GEU_Q29,PCs_matrix[,1])
colnames(data_withPC_Q29)[4] <- "PC1"

n <- nrow (data_withPC_Q29)        # Number of rows
V <- colnames(data_withPC_Q29)     # Column names

# Calculate Pearson correlation
suffStat_C_withPC_Q29 <- list(C = cor(data_withPC_Q29, use = 'pairwise.complete.obs'),
                              n = n)

# Infer graph by MRPC
MRPC.fit_withPC_GEU_Q29 <- MRPC(data_withPC_Q29,
                                suffStat = suffStat_C_withPC_Q29,
                                GV = 1,
                                FDR = 0.05,
                                indepTest = 'gaussCItest',
                                labels = V,
                                FDRcontrol = TRUE,
                                verbose = FALSE)

# Plot inferred graphs
par(mfrow=c(1,2))
plot(MRPC.fit_withoutPC_GEU_Q29,
     main = "Without PC" )
plot(MRPC.fit_withPC_GEU_Q29,
     main = "With PC")

#-------------
# Example 3
# Genes SERPINB8 and HMSD are significantly associated with PC2
print(eqtl.genes_PC2[c(20,21)])

# Genes SERPINB8 and HMSD are in the eQTL-gene set #43 with snp rs55928920
data_GEU_Q43 <- data_GEUVADIS$Data_Q43$Data_EUR
colnames(data_GEU_Q43) <- c("rs55928920","SERPINB8","HMSD")

# Analyze the eQTL-gene set without PC2
n <- nrow (data_GEU_Q43)        # Number of rows
V <- colnames(data_GEU_Q43)     # Column names

# Calculate Pearson correlation
suffStat_C_Q43 <- list(C = cor(data_GEU_Q43, use = 'pairwise.complete.obs'),
                       n = n)

# Infer the graph by MRPC
MRPC.fit_withoutPC_GEU_Q43 <- MRPC(data_GEU_Q43,
                                   suffStat = suffStat_C_Q43,
```

```
                                    GV = 1,
                                    FDR = 0.05,
                                    indepTest = 'gaussCItest',
                                    labels = V,
                                    FDRcontrol = TRUE,
                                    verbose = FALSE)

# Analyze the eQTL-gene set with PC2
data_withPC_Q43 <- cbind(data_GEU_Q43,PCs_matrix[,2])
colnames(data_withPC_Q43)[4] <- "PC2"

n <- nrow (data_withPC_Q43)        # Number of rows
V <- colnames(data_withPC_Q43)     # Column names

# Calculate Pearson correlation
suffStat_C_withPC_Q43 <- list(C = cor(data_withPC_Q43, use = 'pairwise.complete.obs'),
                              n = n)

# Infer the graph by MRPC
MRPC.fit_withPC_GEU_Q43 <- MRPC(data_withPC_Q43,
                                suffStat = suffStat_C_withPC_Q43,
                                GV = 1,
                                FDR = 0.05,
                                indepTest = 'gaussCItest',
                                labels = V,
                                FDRcontrol = TRUE,
                                verbose = FALSE)

# Plot inferred graphs
par(mfrow=c(1,2))
plot(MRPC.fit_withoutPC_GEU_Q43,
     main = "Without PC" )
plot(MRPC.fit_withPC_GEU_Q43,
     main = "With PC")

#-------------
# Example 4
# Gene PLAC8 is significantly associated with PC2 and PC3
print(eqtl.genes_PC2[17])
print(eqtl.genes_PC3[12])

# Gene PLAC8 is in the eQTL-gene set  #34 with snp rs28718968 and gene COQ2
data_GEU_Q34 <- data_GEUVADIS$Data_Q34$Data_EUR
colnames(data_GEU_Q34) <- c("rs28718968","COQ2", "PLAC8")

# Analyze the eQTL-gene set without PC2 and PC3
n <- nrow (data_GEU_Q34)        # Number of rows
V <- colnames(data_GEU_Q34)     # Column names

# Calculate Pearson correlation
suffStat_C_Q34 <- list(C = cor(data_GEU_Q34, use = 'pairwise.complete.obs'),
                       n = n)
```

```
# Infer the graph by MRPC
MRPC.fit_withoutPC_GEU_Q34 <- MRPC(data_GEU_Q34,
                                   suffStat = suffStat_C_Q34,
                                   GV = 1,
                                   FDR = 0.05,
                                   indepTest = 'gaussCItest',
                                   labels = V,
                                   FDRcontrol = TRUE,
                                   verbose = FALSE)

# Analyze the eQTL-gene set with PC2 and PC3
data_withPC_Q34 <- cbind(data_GEU_Q34,PCs_matrix[,c(2,3)])
colnames(data_withPC_Q34)[4:5] <- c("PC2", "PC3")

n <- nrow (data_withPC_Q34)        # Number of rows
V <- colnames(data_withPC_Q34)     # Column names

# Calculate Pearson correlation
suffStat_C_withPC_Q34 <- list(C = cor(data_withPC_Q34, use = 'pairwise.complete.obs'),
                              n = n)

# Infer the graph by MRPC
MRPC.fit_withPC_GEU_Q34 <- MRPC(data_withPC_Q34,
                                suffStat = suffStat_C_withPC_Q34,
                                GV = 1,
                                FDR = 0.05,
                                indepTest = 'gaussCItest',
                                labels = V,
                                FDRcontrol = TRUE,
                                verbose = FALSE)

# Plot inferred graphs
par(mfrow=c(1,2))
plot(MRPC.fit_withoutPC_GEU_Q34,
     main = "Without PC" )
plot(MRPC.fit_withPC_GEU_Q34,
     main = "With PC")

#-------------
# Example 5
# Genes PIP4P1 and PNP are significantly associated with PC1 and PC3, respectively.
print(eqtl.genes_PC1[1])
print(eqtl.genes_PC3[7])

# Genes PIP4P1 and PNP are in the eQTL-gene set #8 with snp rs11305802 and gene  AL355075.3
data_GEU_Q8 <- data_GEUVADIS$Data_Q8$Data_EUR
colnames(data_GEU_Q8) <- c("rs11305802","PIP4P1", "AL355075.3", "PNP")

# Analyze the eQTL-gene set without PC1 and PC3
n <- nrow (data_GEU_Q8)        # Number of rows
V <- colnames(data_GEU_Q8)     # Column names

# Calculate Pearson correlation
```

```
suffStat_C_Q8 <- list(C = cor(data_GEU_Q8, use = 'pairwise.complete.obs'),
                      n = n)

# Infer the graph by MRPC
MRPC.fit_withoutPC_GEU_Q8 <- MRPC(data_GEU_Q8,
                                  suffStat = suffStat_C_Q8,
                                  GV = 1,
                                  FDR = 0.05,
                                  indepTest = 'gaussCItest',
                                  labels = V,
                                  FDRcontrol = TRUE,
                                  verbose = FALSE)

# Analyze the eQTL-gene set with PC1 and PC3
data_withPC_Q8 <- cbind(data_GEU_Q8,PCs_matrix[,c(1,3)])
colnames(data_withPC_Q8)[5:6] <- c("PC1","PC3")


n <- nrow (data_withPC_Q8)        # Number of rows
V <- colnames(data_withPC_Q8)     # Column names

# Calculate Pearson correlation
suffStat_C_withPC_Q8 <- list(C = cor(data_withPC_Q8, use = 'pairwise.complete.obs'),
                             n = n)

# Infer the graph by MRPC
MRPC.fit_withPC_GEU_Q8 <- MRPC(data_withPC_Q8,
                               suffStat = suffStat_C_withPC_Q8,
                               GV = 1,
                               FDR = 0.05,
                               indepTest = 'gaussCItest',
                               labels = V,
                               FDRcontrol = TRUE,
                               verbose = FALSE)
# Plot inferred graphs
par(mfrow=c(1,2))
plot(MRPC.fit_withoutPC_GEU_Q8,
     main = "Without PC" )
plot(MRPC.fit_withPC_GEU_Q8,
     main = "With PC")
```

`data_without_outliers` *Example data without outliers (noises)*

Description

The data contain two genotype nodes, V1 and V2, and three phenotype nodes, T1, T2 and T3. The code below compares the performance of MRPC, pc pc.stable, mmpc, mmhc, and hc on this data set.

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

Examples

```
# Load packages

library(MRPC)     # MRPC
library(pcalg)    # pc
library(bnlearn)  # pc.stable, mmpc, mmhc, and hc

# Truth without outlier
tarmat <- matrix(0,
                 nrow = ncol(data_with_outliers),
                 ncol = ncol(data_with_outliers))

colnames(tarmat) <- colnames(data_with_outliers)
rownames(tarmat) <- colnames(data_with_outliers)

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

Truth <- as(tarmat,
            "graphNEL")

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

# Calculate Pearson correlation
suffStat_C1 <- list(C = cor(data_without_outliers),
                    n = n)

# Infer the graph by MRPC
MRPC.fit_withoutoutliers <- MRPC (data_without_outliers,
                                  suffStat = suffStat_C1,
                                  GV = 2,
                                  FDR = 0.05,
                                  indepTest ='gaussCItest',
                                  labels = V,
                                  FDRcontrol = TRUE,
                                  verbose = FALSE)

# Infer the graph by pc with Pearson correlation
pc.fit_withoutoutliers <- pc(suffStat = suffStat_C1,
```

```
                              indepTest = gaussCItest,
                              alpha = 0.05,
                              labels = V,
                              verbose = FALSE)

  # arcs not to be included from gene expression to genotype for blacklist argument
  # in pc.stable and mmpc

  GV <- 2
  to <- rep (colnames (data_without_outliers)[1:GV], each = (ncol (data_without_outliers) - GV))
  from <- rep (colnames (data_without_outliers)[(GV + 1):ncol (data_without_outliers)], GV)
  bl <- cbind (from, to)

  # Infer the graph by pc.stable
  pc.stable_withoutoutliers <- pc.stable (data.frame (data_without_outliers),
                                          blacklist = bl,
                                          alpha = 0.05,
                                          debug = FALSE,
                                          undirected = FALSE)
  # Infer the graph by mmpc
  mmpc_withoutoutliers <- mmpc (data.frame (data_without_outliers),
                                blacklist = bl,
                                alpha = 0.05,
                                debug = FALSE,
                                undirected = FALSE)
  # Infer the graph by mmhc
  mmhc_withoutoutliers <- mmhc (data.frame (data_without_outliers),
                                blacklist = bl,
                                debug = FALSE)
  # Infer the graph by hc
  hc_withoutoutliers <- hc (data.frame (data_without_outliers),
                            blacklist = bl,
                            debug = FALSE)
  # True graph
  plot (Truth, main = "Truth")

  #-------------
  # Plot inferred graphs
  par (mfrow = c (2,3))

  # Data without outliers
  # Inference with Pearson correlation
  plot (MRPC.fit_withoutoutliers, main = "MRPC")
  plot (pc.fit_withoutoutliers, main = "pc")
  graphviz.plot (pc.stable_withoutoutliers, main = "pc.stable")
  graphviz.plot (mmpc_withoutoutliers, main = "mmpc")
  graphviz.plot (mmhc_withoutoutliers, main = "mmhc")
  graphviz.plot (hc_withoutoutliers, main = "hc")
```

---

data_with_outliers *Example data with outliers (noises)*

Description

The data contain two genotype nodes, V1 and V2, and three phenotype nodes, T1, T2 and T3. The genotype nodes are discrete, whereas the phenotype nodes are continuous. The data matrix includes 10 outliers (noises) generated from a uniform distribution. The example code below compares the performance of MRPC, pc, pc.stable, mmpc, mmhc, and hc on this data set.

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

Examples

```
# Load packages

library(MRPC)     # MRPC
library(pcalg)    # pc
library(bnlearn)  # pc.stable, mmpc, mmhc, and hc


# Truth without outlier
tarmat <- matrix(0,
                 nrow = ncol(data_with_outliers),
                 ncol = ncol(data_with_outliers))
colnames(tarmat) <- colnames(data_with_outliers)
rownames(tarmat) <- colnames(data_with_outliers)

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

# Graph
Truth <- as(tarmat,
            "graphNEL")

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

# Calculate Pearson correlation
suffStat_C1 <- list(C = cor(data_without_outliers),
                    n = n)

# Infer the graph by MRPC
MRPC.fit_withoutoutliers <- MRPC (data_without_outliers,
                                  suffStat = suffStat_C1,
                                  GV = 2,
```

```
                                    FDR = 0.05,
                                    indepTest ='gaussCItest',
                                    labels = V,
                                    FDRcontrol = TRUE,
                                    verbose = FALSE)

# Infer the graph by pc
pc.fit_withoutoutliers <- pc(suffStat = suffStat_C1,
                             indepTest = gaussCItest,
                             alpha = 0.05,
                             labels = V,
                             verbose = FALSE)

# arcs not to be included from gene expression to genotype for blacklist argument
# in pc.stable and mmpc

GV <- 2
to <- rep (colnames (data_without_outliers)[1:GV], each = (ncol (data_without_outliers) - GV))
from <- rep (colnames (data_without_outliers)[(GV + 1):ncol (data_without_outliers)], GV)
bl <- cbind (from, to)

# Infer the graph by pc.stable
pc.stable_withoutoutliers <- pc.stable (data.frame (data_without_outliers),
                                        blacklist = bl,
                                        alpha = 0.05,
                                        debug = FALSE,
                                        undirected = FALSE)
# Infer the graph by mmpc
mmpc_withoutoutliers <- mmpc (data.frame (data_without_outliers),
                              blacklist = bl,
                              alpha = 0.05,
                              debug = FALSE,
                              undirected = FALSE)
# Infer the graph by mmhc
mmhc_withoutoutliers <- mmhc (data.frame (data_without_outliers),
                              blacklist = bl,
                              debug = FALSE)
# Infer the graph by hc
hc_withoutoutliers <- hc (data.frame (data_without_outliers),
                          blacklist = bl,
                          debug = FALSE)


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

# Calculate Pearson correlation
suffStat_C2 <- list (C = cor (data_with_outliers),
                     n = n)

# Infer the graph by MRPC
MRPC.fit_withoutliers_C2 <- MRPC (data_with_outliers,
```

```
                                    suffStat = suffStat_C2,
                                    GV = 2,
                                    FDR = 0.05,
                                    indepTest ='gaussCItest',
                                    labels = V,
                                    FDRcontrol = TRUE,
                                    verbose = FALSE)

# Infer the graph by pc
pc.fit_withoutliers_C2 <- pc (suffStat = suffStat_C2,
                              indepTest = gaussCItest,
                              alpha = 0.05,
                              labels = V,
                              verbose = FALSE)

# arcs not to be included from gene expression to genotype for blacklist argument
# in pc.stable and mmpc

GV <- 2
to <- rep (colnames (data_with_outliers)[1:GV], each = (ncol (data_with_outliers) - GV))
from <- rep (colnames (data_with_outliers)[(GV + 1):ncol (data_with_outliers)], GV)
bl <- cbind (from, to)

# Infer the graph by pc.stable
pc.stable_withoutliers_C2 <- pc.stable (data.frame (data_with_outliers),
                                        blacklist = bl,
                                        alpha = 0.05,
                                        B = NULL,
                                        max.sx = NULL,
                                        debug = FALSE,
                                        undirected = FALSE)
# Infer the graph by mmpc
mmpc_withoutliers_C2 <- mmpc (data.frame (data_with_outliers),
                              blacklist = bl,
                              alpha = 0.05,
                              B = NULL,
                              max.sx = NULL,
                              debug = FALSE,
                              undirected = FALSE)
# Infer the graph by mmhc
mmhc_withoutliers_C2 <- mmhc (data.frame (data_with_outliers),
                              blacklist = bl,
                              debug = FALSE)


# Infer the graph by hc
hc_withoutliers_C2 <- hc (data.frame (data_with_outliers),
                          blacklist = bl,
                          debug = FALSE)

# Calculate robust correlation (Beta = 0.005)
Rcor_R1 <- RobustCor (data_with_outliers, 0.005)
suffStat_R1 <- list (C = Rcor_R1$RR,
```

```
                    n = n)

# Infer the graph by MRPC with robust correlation
MRPC.fit_withoutliers_R1 <- MRPC (data_with_outliers,
                                  suffStat = suffStat_R1,
                                  GV = 2,
                                  FDR = 0.05,
                                  indepTest = 'gaussCItest',
                                  labels = V,
                                  FDRcontrol = TRUE,
                                  verbose = FALSE)

# Infer the graph by pc with robust correlation
pc.fit_withoutliers_R1 <- pc (suffStat = suffStat_R1,
                              indepTest = gaussCItest,
                              alpha = 0.05,
                              labels = V,
                              verbose = FALSE)


# True graph
plot (Truth, main = "Truth")

#-------------
# Plot inferred graphs
par (mfrow = c (2,6))

# Data without outliers
# Inference with Pearson correlation
plot (MRPC.fit_withoutoutliers, main = "MRPC")
plot (pc.fit_withoutoutliers, main = "pc")
graphviz.plot (pc.stable_withoutoutliers, main = "pc.stable")
graphviz.plot (mmpc_withoutoutliers, main = "mmpc")
graphviz.plot (mmhc_withoutoutliers, main = "mmhc")
graphviz.plot (hc_withoutoutliers, main = "hc")


# Data with outliers
# Inference with Pearson correlation
plot (MRPC.fit_withoutliers_C2, main = " ")
plot (pc.fit_withoutliers_C2, main = " ")
graphviz.plot (pc.stable_withoutliers_C2, main = " ")
graphviz.plot (mmpc_withoutliers_C2, main = " ")
graphviz.plot (mmhc_withoutliers_C2, main = " ")
graphviz.plot (hc_withoutliers_C2, main = " ")

#-------------
# Data with outliers
# Inference with robust correlation
par (mfrow = c (1,2))
plot (MRPC.fit_withoutliers_R1, main = "MRPC")
plot (pc.fit_withoutliers_R1, main = "pc")
```

`EdgeOrientation` *Perform edge orientation under the MRPC algorithm*

Description

This function performs the second step of the MRPC algorithm where it determines the edge direction in the graph skeleton inferred by the function ModiSkeleton. If the data contain genetic variants, this function first determines the edges between genetic variants and phenotype nodes based on the principle of Mendelian randomization. Next it identifies potential v-structures and orients the edges in them. For the remaining edges, it examines triplets in turn to see whether a triplet is compatible with one of the basic models. See the references for details.

Usage

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

Arguments

`gInput`	Object containing the skeleton and 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`	False discovery rate (number between 0 and 1). If FDR = 0.05, this ensures that the FDR and mFDR remains below 0.05.
`alpha`	Significance level (number in (0,1) for the individual tests.
`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. There are three options for different data types, for example, Gaussian data = gaussCItest, discrete data = disCItest and Binary data = binCItest. See help(gaussCItest)
`FDRcontrol`	(optional) The default is TRUE which implements a sequential FDR control method, otherwise used fixed significance level for the individual tests.
`verbose`	(optional) 1: detailed output is provided; 0: No output is provided

Details

The orientation of the edge directions based on the principle of Mendelian randomization involves four cases, which are four of the five basic models in Badsha and Fu, 2019 and Badsha et al., 2018. For example, we consider x to be a genetic variant, y and z the phenotype nodes.

The four 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 there is an edge between two genetic variants, 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 and x and y are conditionally dependent given z and x and z are conditionally dependent given y, then the edge direction will be y <–> z.

Value

An object 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 (mbbadshar@gmail.com)

References

1. Badsha MB and Fu AQ (2019). Learning causal biological networks with the principle of Mendelian randomization. Frontiers in Genetics, 10(460).

2. Badsha MB, Martin EA and Fu AQ (2018). MRPC: An R package for accurate inference of causal graphs. arXiv arXiv:1806.01899.

See Also

MRPC; ModiSkeleton; SimulateData.

Examples

```
# 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

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

# Infer a graph skeleton
Skel.fit <- ModiSkeleton(data,
                         suffStat = suffStat_C,
                         FDR = 0.05,
                         indepTest = 'gaussCItest',
                         labels = V,
                         FDRcontrol = TRUE,
                         verbose = FALSE)

# Edge Orientation
Edge_orientation <- EdgeOrientation(Skel.fit,
                                    suffStat = suffStat_C,
                                    GV = 1,
                                    FDR = 0.05,
                                    indepTest = 'gaussCItest',
                                    FDRcontrol = TRUE,
                                    verbose = FALSE)
# 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 (mbbadshar@gmail.com)

`IdentifyAssociatedPCs` *Identify principal components (PCs) that are significantly associated with eQTLs and genes*

Description

This function identifies PCs that are significantly associated the eQTLs or genes, and merge the associated PCs with the data on the eQTL and genes. PCs may be derived from Principal Component Analysis (PCA) of the entire gene expression matrix, and may be viewed as potential confounders in the sequent causal network analysis on the eQTLs and genes. See details in Badsha and Fu (2019) and Badsha et al. (2018).

Usage

```
IdentifyAssociatedPCs(PCs.matrix,no.PCs,data,fdr.level,corr.threshold,corr.value)
```

Arguments

`PCs.matrix`	A matrix of PCs.
`no.PCs`	Number of top PCs to test for association. The default is 10.
`data`	Data of the eQTLs and genes, containing the genotypes of the eQTLs and the expression of the genes.
`fdr.level`	(optional) The false discover rate (FDR) for association tests. Must be in (0,1]. The default is "0.05".
`corr.threshold`	(optional). The default is "FALSE". If "TRUE" then a constraint on the correlation between a PC and an eQTL or a gene is applied in addition to the FDR control.
`corr.value`	The threshold for the Pearson correlation between a PC and an eQTL or a gene when `corr.threshold` is "TRUE". The default is 0.3.

Value

A list of object that containing the following:

- `AssociatedPCs`: All the PCs that are significantly associated with the eQTLs and genes.
- `data.withPC`: The data matrix that contains eQTLs, gene expression, and associated PCs.
- `corr.PCs`: The matrix of correlations between PCs and eQTLs/genes.
- `PCs.asso.list`: List of all associated PCs for each of the eQTLs and genes.
- `qobj`: The output from applying the qvalue function.

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

References

1. Badsha MB and Fu AQ (2019). Learning causal biological networks with the principle of Mendelian randomization. Frontiers in Genetics, 10(460).

2. Badsha MB, Martin EA and Fu AQ (2018). MRPC: An R package for accurate inference of causal graphs. arXiv:1806.01899.

See Also

data_GEUVADIS_combined

Examples

```
# Load genomewide gene expression data in GEUVADIS
# 373 individuals
# 23722 genes
data_githubURL <- "https://github.com/audreyqyfu/mrpc_data/raw/master/data_GEUVADIS_allgenes.RData"
load(url(data_githubURL))
PCs <- prcomp(data_GEUVADIS_allgenes,scale=TRUE)
# Extract the PCs matrix
PCs.matrix <- PCs$x

# The eQTL-gene set contains eQTL rs7124238 and genes SBF1-AS1 and SWAP70

data_GEU_Q50 <- data_GEUVADIS$Data_Q50$Data_EUR
colnames(data_GEU_Q50) <- c("rs7124238","SBF2-AS1","SWAP70")
data <- data_GEU_Q50

# Identify associated PCs for this eQTL-gene set

Output <- IdentifyAssociatedPCs(PCs.matrix,no.PCs=10,data,fdr.level=0.05,corr.threshold=TRUE
,corr.value = 0.3)

# Gene SBF2-AS1 is significantly associated with PC2
# Data with PC2 as a potential confounder

data_withPC <- Output$data.withPC

n <- nrow(data_withPC)         # Number of rows
V <- colnames(data_withPC)     # Column names

# Calculate Pearson correlation for MRPC analysis

suffStat <- list(C = cor(data_withPC,use = "complete.obs"),
                 n = n)

# Infer the graph by MRPC

MRPC.fit_FDR<- MRPC(data_withPC,
```

```
                    suffStat,
                    GV = 1,
                    FDR = 0.05,
                    indepTest = 'gaussCItest',
                    labels = V,
                    FDRcontrol = TRUE,
                    verbose = TRUE)

plot(MRPC.fit_FDR, main="MRPC with PCs (potential confounders)")
```

ModiSkeleton *Infer a graph skeleton (undirected graph)*

Description

This function infers a graph skeleton (i.e., an undirected graph). It is based on the function skeleton from the pcalg package. Both functions perform marginal and conditional indpenendence tests. However, ModiSkeleton implements an online false discovery rate (FDR) control method in order to control the overall FDR, whereas skeleton controls only the type I error rate for each individual test. See details below.

Usage

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

Arguments

	Many arguments are similar to those in skeleton and pc in the pcalg package. Several arguments here are also arguments for the function MRPC.
data	Data matrix, where the rows are samples and the columns are features (e.g., genetic variants (GVs) and phenotypes). Columns are for GVs, if available, appear before other columns for phenotypes (e.g., gene expression). For example, if there is one GV, then the first column of the data matrix is the GV and the remaining columns are the gene expression data.
suffStat	A list consisting of the correlation matrix of the data and the sample size.
FDR	Desired overall FDR level.
alpha	significance level (number in (0,1) for the individual tests.
indepTest	Name of the statistical test. It is used to test the independence of x and y given S, where x and y are variables and S is a vector, possibly empty, of variables. The return value of indepTest is the p-value of the test for conditional independence. Different tests may used for different data types. For example, indepTest='gaussCItest' for Gaussian data, indepTest='disCItest' for

discrete data, and `indepTest='binCItest'` for binary data. See additional details in help(gaussCItest).

ci.test in the `bnlearn` package (Marco Scutari, 2010) may also be used for testing conditional independence and return a p-value. 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).

`labels` A character vector of names of variables (nodes). These are typically the column names of the data matrix.

`p` (optional) The number of variables (nodes). Need to 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.

`FDRcontrol` (optional) The default is TRUE which is used sequential FDR control method, otherwise used fixed significance level for the individual tests.

`verbose` (optional) If TRUE, detailed output is provided. Default is FALSE for no output details

Details

The ModiSkeleton function incorporates sequential hypothesis testing to infer the graph skeleton. This function starts with a complete graph (all nodes are connected with undirected edges) and performs a series of marginal and conditional independence tests, removing the corresponding edge if the test is not rejected.

First, all pairs of nodes are tested for marginal independence. If two nodes x and y are judged to be marginally independent at a type I error rate alpha, 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, some edges may be removed.

Second, nodes (x, y) with an edge are tested for conditional independence given all subsets of the neighboring 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.

Unlike existing algorithms, which control only the type I error rate for each individual test, MRPC implements the LOND (Level On the Number of Discoveries) method (Javanmard and Montanari, 2015), which is a sequential hypothesis testing procedure and sets value of alpha for each test based on the number of discoveries (i.e., rejections), to control the overall false discovery rate.

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 (mbbadshar@gmail.com)

References

1. Badsha MB, Mollah MN, 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 Hochberg Y (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 A and Montanari A (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. Scutari M (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

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

7.Tsamardinos I, Brown LE and Aliferis CF (2006). The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. Machine Learning, 65 (1), 31-78.

See Also

MRPC; EdgeOrientation; SimulateData.

Examples

```
# 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

data <- simu_data_M1 # load data for model 1
n <- nrow(data)      # Number of row
V <- colnames(data)  # Column names

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

# Infer a graph skeleton
Skel.fit <- ModiSkeleton(data,
                         suffStat = suffStat_C,
                         FDR = 0.05,
                         alpha = 0.05,
                         indepTest = 'gaussCItest',
                         labels = V,
                         FDRcontrol = TRUE,
                         verbose = FALSE)

# Plot the results

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` *Calculate the inverse matrix*

Description

This function calculates the inverse of the non-square matrix as part of the calculation of the robust correlation matrix.

Usage

```
mpinv(X)
```

Arguments

`X`	Data Matrix

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

Examples

```
Inversematrix <- mpinv(simu_data_M0)
```

MRPC *Infer a causal network using the MRPC algorithm*

Description

This function is used to infer a causal network (or a causal graph) with directed and undirected edges from observational data. It implements the MRPC (PC with the principle of Mendelian randomization) algorithm described in Badsha and Fu, 2019 and Badsha et al., 2018, and the implementation is based on the pc algorithm in the pcalg package. The MRPC algorithm contains four major updates over the pc algorithm: (i) incorporating a sequential testing method to control the False Discovery Rate (FDR), (ii) improved v-structure identification; (iii) allowing for calculation of robust correlation to reduce the impact of outliers, and (iv) a new procedure for edge orientation based on the principle of Mendelian randomization (PMR) (Badsha and Fu, 2019 and Badsha et al., 2018). See details below.

Usage

```
MRPC(data, suffStat, GV, FDR = 0.05, alpha = 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, FDRcontrol = TRUE,
    verbose = FALSE)
```

Arguments

	This function is based on the pc function in the pcalg package. Therefore, many arguments are similar to those in pc.
	Data matrix, where the rows are observations and the columns are features (i.e., variables, or nodes). If genetic variants (GVs) are included, then the columns start from GVs (e.g., single-nucleotide polymorphisms, or SNPs; insertions and deletions, or indels; copy number variation, or CNVs; and expression quantitative trait loci, or eQTLs to genes), and followed by phenotypes (e.g., gene expression). For example, if the data contains one GV, then the first column of the input matrix is the GV and the remaining columns are the gene expression data.
~~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 the desired level of the overall false discovery rate.

`alpha` significance level (number in (0,1) for the individual tests.

`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.

`FDRcontrol` (optional) The default is TRUE which is used sequential FDR control method, otherwise used fixed significance level for the individual tests.

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

Details

The PC algorithm is computationally efficient for learning a directed acyclic graph (Spirtes et al., 2000). Several variants of the original PC algorithms are available (Kalisch and Buhlmann, 2007; Kalisch et al., 2012). Similar to these PC-like algorithms, our MRPC algorithm also contains two main steps:

Step-1: Inference of the graph skeleton. A graph skeleton is an undirected graph with edges that are supported by the data. Similar to existing PC-like algorithms, we perform statistical tests for marginal and conditional independence tests. If the null hypothesis of independence is not rejected, then the corresponding edge is removed and never tested again.

However, unlike existing algorithms, which control only the type I error rate for each individual test, MRPC implements the LOND (Level On the Number of Discoveries) method (Javanmard and Montanari, 2015), which is a sequential hypothesis testing procedure and sets the significance level for each test based on the number of discoveries (i.e., rejections), to control the overall false discovery rate (FDR). See ModiSkeleton.

Genome data may have outliers that drastically alter the topology of the inferred graph. MRPC allows for the estimate of robust correlation, which may be the substitute of the Pearson correlation as the input to graph inference (Badsha et al., 2013).

Step-2: Edge orientation. With the graph skeleton inferred from Step 1, we orient each edge that is present in the graph. MRPC is fundamentally different from algorithms in the `pcalg` (Kalisch and Buhlmann, 2007; Kalisch et al., 2012) and `bnlearn` (Scutari, 2010) packages in the following ways (see EdgeOrientation):

(i) When analyzing genomic data, genetic variants provide additional information that helps distinguish the casual direction between two genes. Our MRPC algorithm incorporates the principle of Mendelian randomization in graph inference, which greatly reduces the space of possible graphs and increases the inference efficiency.

(ii) Next or if the input is not genomic data, we search for possible triplets that may form a v-structure (e.g.,X–>Y<–Z). We check conditional test results from step I to see whether X and Z are independent given Y. If they are, then this is not a v-structure; alternative models for the triplet may be any of the following three Markov equivalent graphs: X–>Y–>Z, X<–Y<–Z, and X<–Y–>Z. If this test is not performed in the first step, we conduct it in this step. This step improves the accuracy of the v-structure identification over existing methods.

(iii) If there are undirected edges after steps (i) and (ii), we examine iteratively triplets of nodes with at least one directed edge and no more than one undirected edge. We check the marginal and conditional test results from Step I to determine which of the basic models is consistent with the test results. It is plausible that some undirected edges cannot be oriented, and we leave them as undirected.

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 (mbbadshar@gmail.com)

References

1. Badsha MB and Fu AQ (2019). Learning causal biological networks with the principle of Mendelian randomization. Frontiers in Genetics, 10(460).

2. Badsha MB, Martin EA and Fu AQ (2018). MRPC: An R package for accurate inference of causal graphs. arXiv:1806.01899.

3. Badsha MB, Mollah MN, 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.

4. Javanmard A and Montanari A (2015). On Online Control of False Discovery Rate. arXiv:150206197 [statME].

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

6. Kalisch M, Machler M, Colombo D, Maathuis MH and Buhlmann P (2012). Causal Inference Using Graphical Models with the R Package pcalg. Journal of Statistical Software, 47, 26.

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

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

See Also

ModiSkeleton for inferring a graph skeleton (i.e., an undirected graph); EdgeOrientation for edge orientation in the inferred graph skeleton; SimulateData for generating data under a topology.

Examples

```
# Load packages
# We compare different simulated data across six methods: MRPC,
# PC in pcalg (Kalisch et al., 2012), and pc.stable, mmpc mmhc and hc in
# bnlearn (Marco Scutari, 2010)

library(MRPC)     # MRPC
library(pcalg)    # pc
library(bnlearn)  # pc.stable, mmpc, mmhc and hc

# 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.

# Load predefined simulated data
# 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

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

# Infer the graph by MRPC
MRPC.fit <- MRPC(data,
                 suffStat = suffStat_C,
                 GV = 1,
                 FDR = 0.05,
                 indepTest = 'gaussCItest',
                 labels = V,
                 FDRcontrol = TRUE,
                 verbose = FALSE)

# Infer the graph by PC
pc.fit <- pc(suffStat = suffStat_C,
             indepTest = gaussCItest,
             alpha = 0.05,
             labels = V,
             verbose = FALSE)

# arcs not to be included from gene expression to genotype used in pc.stable, mmpc
bl <- data.frame (from = colnames (data)[-1],
                  to = 'V1')

# Infer the graph by pc.stable
pc.stable.fit <- pc.stable(data.frame(data),
```

```
                          blacklist = bl,
                          undirected = FALSE)

# Infer the graph by mmpc
mmpc.fit <- mmpc(data.frame(data),
                 blacklist = bl,
                 undirected = FALSE,
                 debug = FALSE)

# Infer the graph by mmhc
mmhc.fit <- mmhc(data.frame(data),
                 blacklist = bl,
                 debug = FALSE)

# Infer the graph by hc
hc.fit <- hc (data.frame (data),
                          blacklist = bl,
                          debug = FALSE)

# Plot the inferred graphs
par(mfrow = c(1, 7))
plot(Truth,
     main = "(A) Truth")
plot(MRPC.fit,
     main = "(B) MRPC")
plot(pc.fit,
    main ="(C) pc")
graphviz.plot(pc.stable.fit,
    main = "(D) pc.stable")
graphviz.plot(mmpc.fit,
    main = "(E) mmpc")
graphviz.plot(mmhc.fit,
    main = "(F) mmhc")
graphviz.plot(hc.fit,
    main = "(G) hc")

# 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 dendrogram with modules colors of nodes
PlotDendrogramObj <- PlotDendrogram(Adj_directed,
                                    minModuleSize = 2)

# Visualization of inferred graph with modules colors
PlotGraphWithModulesObj <- PlotGraphWithModules(Adj_directed,
                                                PlotDendrogramObj,
                                                GV = 1,
                                                node.size = 8,
```

```
                                                arrow.size = 5,
                                                label.size = 3,
                                                alpha = 1)

# Plot
plot(PlotGraphWithModulesObj)


# 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 (mbbadshar@gmail.com)

See Also

MRPC, ModiSkeleton

Examples

```
showClass("MRPCclass")

# Generate a MRPCclass object
data <- simu_data_M1 # load data for model 1
n <- nrow(data)      # Number of rows
V <- colnames(data)  # Column names

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

# Infer the graph by MRPC
MRPC.fit <- MRPC(data,
                 suffStat_C,
                 GV = 1,
                 FDR = 0.05,
                 indepTest ='gaussCItest',
                 labels = V,
                 FDRcontrol = TRUE,
                 verbose = FALSE)


# 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 *Graphs used as truth in simulation*

Description

Topologies of the five basic models and three common graphs in biology: namely the multi-parent graph, the star graph and the layered graph. See details in Badsha and Fu, 2019.

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

References

1. Badsha MB and Fu AQ (2019). Learning causal biological networks with the principle of Mendelian randomization. Frontiers in Genetics, 10(460).

Examples

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

# Plots
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")
```

PlotDendrogram *Plot a dendrogram and display node groups in colored modules*

Description

Generate a dendrogram of nodes with dissimilarity based on topological overlap, and group nodes into modules indicated by colors.

Usage

```
PlotDendrogram(Adj_directed, minModuleSize, groupLabels = " ",
               dendroLabels = FALSE, hclustHang = 0.03,
               dendroAddGuide = FALSE, dendroGuideHang = 0.05,
               dendroMain = "Dendrogram with modules of nodes in colors", ...)
```

Arguments

Adj_directed	Adjacency matrix from directed graph
minModuleSize	Minimum module size.
groupLabels	Argument for plotDendroAndColors. Labels for the colorings given in colors. The labels will be printed to the left of the color rows in the plot.
dendroLabels	Argument for plotDendroAndColors. Dendrogram labels.
hclustHang	Argument hang for plot.hclust. The fraction of the plot height by which labels should hang below the rest of the plot.
dendroAddGuide	Argument addGuide for plotDendroAndColors. Logical: should vertical "guide lines" be added to the dendrogram plot? The lines make it easier to identify color codes with individual samples.
dendroGuideHang	Argument guideHang for plotDendroAndColors. The fraction of the dendrogram height to leave between the top end of the guide line and the dendrogram merge height.
dendroMain	Argument main for plot.hclust. Title of the plot.
...	Additional plotting arguments for plotDendroAndColors and plot.hclust.

Value

A list containing the graph objects as follows:

- PlotDendrogramObj: An object of class "graph" of the estimated graph.
- dynamicColors: A list of colors with corresponding nodes.
- GroupMods: Dynamic tree cut to identify modules whose phenotype profiles are very similar.
- GroupModsColors: A table for number of nodes with corresponding colors.
- Adj_symmetric_matrix: A symmetric matrix from ddjacency matrix of directed graph.

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

MRPC.

Examples

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

# Plot of dendrogram with modules colors of nodes
PlotDendrogramObj <- PlotDendrogram(Adj_directed,
                                    minModuleSize = 5)
```

`PlotGraphWithModules` *Plot a graph with nodes in modules indicated by colors*

Description

Visualization of a graph with nodes in modules inferred from the clustering dendrogram by PlotDendrogram.

Usage

```
PlotGraphWithModules(Adj_directed, PlotDendrogramObj,
                     GV, node.size = 8, arrow.size = 5,
                     label.size = 3,alpha = 1,...)
```

Arguments

`Adj_directed`	Adjacency matrix of a graph.
`PlotDendrogramObj`	The graphical objects from PlotDendrogram.
`GV`	The number of genetic variants (SNPs/indels/CNVs/eQTL) in the input data matrix. For example, if the data has one SNPs/indels/CNV/eQTL in the first column, then GV = 1, if 2 SNPs/indels/CNVs/eQTL in the 1st and 2nd Column, then GV = 2, and so on. If no GV then GV = 0.
`node.size`	The size of the nodes in the graph. Defaults to 8.
`arrow.size`	The size of the arrows for directed network edges, in points. Defaults to 5.
`label.size`	The size of the node labels in points, as a numeric value, a vector of numeric values, or as a vertex attribute containing numeric values. Defaults to 3.
`alpha`	The level of transparency of the edges and nodes. Defaults to 1 (no transparency).
`...`	Other arguments passed to ggnet2.

Value

- `PlotGraphWithModulesObj`: An object of class "graph" of the graph.

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

PlotDendrogram

Examples

```
# Adjacency matrix from a graph in the example
Adj_directed <- as(data_examples$complex$cont$withGV$graph,
                   "matrix")

# A clustering dendrogram with nodes grouped in colored modules
PlotDendrogramObj <- PlotDendrogram(Adj_directed,
                                    minModuleSize = 5)

# A graph object with nodes in modules
PlotGraphWithModulesObj <- PlotGraphWithModules(Adj_directed,
                                                PlotDendrogramObj,
                                                GV = 14,
                                                node.size = 8,
                                                arrow.size = 5,
                                                label.size = 3,
                                                alpha = 1)

# Plot the graph with nodes in different colors
plot(PlotGraphWithModulesObj)
```

RecallPrecision *Calculate recall and precision for two graphs*

Description

This function counts the number of true and false positives, and calculates recall and precision, which are defined as follows:

Recall = (# edges correctly identified in inferred graph) / (# edges in true graph).

Precision = (# edges correctly identified in inferred graph) / (# edges in inferred graph).

Usage

```
RecallPrecision(g1, g2, GV, includeGV, 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.
includeGV	If TRUE, include edges involving genetic variants (GV) when calculating recall and precision. If FALSE, excluded edges involving genetic variants (GV) when calculating recall and precision.
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 as the default to an edge with the correct direction and 0.5 to an edge with the wrong direction or no direction (Badsha and Fu, 2019; 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.
- NTE: Total number of edges in the true graph.
- NIE: Total number of edges in the inferred 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 (mbbadshar@gmail.com)

References

1. Badsha MB and Fu AQ (2019). Learning causal biological networks with the principle of Mendelian randomization. Frontiers in Genetics, 10(460).

2. Badsha MB, Martin EA and Fu AQ (2018). MRPC: An R package for accurate inference of causal graphs. arXiv:1806.01899.

See Also

aSHD: adjusted Structural Hamming Distance (aSHD)

Examples

```
# True model
# True graph (V1 --> T1 --> T2 --> T3)
# Where V1 is a genetic variant (GV) and T1, T2, and T3 are phenotypes
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)
# Where V1 is a genetic variant (GV) and T1, T2, and T3 are phenotypes
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 <- RecallPrecision(Truth,
                                    Inferred,
                                    GV = 1,
                                    includeGV = TRUE,
                                    edge.presence = 1.0,
                                    edge.direction = 0.5)
```

RobustCor *Calculate robust correlation matrix*

Description

Calculate 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 (mbbadshar@gmail.com)

References

1. Badsha MB, Mollah MN, 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 and 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

```
RobustCor_objects <- RobustCor(simu_data_M0,
                               Beta = 0.005,
                               plot = FALSE)

Rcorr <- RobustCor_objects $RR  # Correlation matrix
```

`seqDiff` *Deviation between two graphs represented by two sequences*

Description

This function evaluates whether two graphs are identical. Each graph is represented first by a binary vector, which is the vectorized adjacency matrix, and then converted to a decimal number. The difference in the decimal numberes is the deviation between the two graphs.

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 (mbbadshar@gmail.com)

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.
Results <- 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 (mbbadshar@gmail.com)

References

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

2. Foster DP and Stine RA (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.

SimulateData *Simulate data under certain graphs*

Description

This function simulates data using linear models for several graphs: the five basic topologies and three topologies that are common in biology, namely the multi-parent graph, the star graph and the layered graph. See references for details.

Usage

```
SimulateData(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 node.
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 node.
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 node.

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 node.
sd.1	Standard deviation for corresponding data generated nodes.

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 node expression data.

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

References

1. Badsha MB and Fu AQ (2019). Learning causal biological networks with the principle of Mendelian randomization. Frontiers in Genetics, 10(460).

2. Badsha MB, Martin EA and Fu AQ (2018). MRPC: An R package for accurate inference of causal graphs. arXiv:1806.01899.

See Also

MRPC; SimulateDataNP, which simulates data for a node with no parent; SimulateData1P for a node with one parent; SimulateData2P for a node with 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 <- SimulateData(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 <- SimulateData(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 <- SimulateData(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 <- SimulateData(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 <- SimulateData(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 <- SimulateData(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 <- SimulateData(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 <- SimulateData(N = 10^3,
                                  p = 0.45,
                                  'layered',
                                  b0.1 = 0,
                                  b1.1 = 1,
                                  b1.2 = 1,
                                  b1.3 = 1,
                                  sd.1 = 1)
```

`SimulateData1P` *Simulate data for a node with one parent*

Description

Simulate data for a node with one parent

Usage

```
SimulateData1P(N, P1, b0.1, b1.1, sd.1)
```

Arguments

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

Value

Vector

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData for simulated data generating function.

Examples

```
Data1P <- SimulateData1P(N = 10^3,
                         P1 = 1,
                         b0.1 = 0,
                         b1.1 = 1,
                         sd.1 = 1)
```

`SimulateData2P` *Simulate data for a node with two parents*

Description

Simulate data for a node with two parents

Usage

```
SimulateData2P(N, P1, P2, b0.1, b1.1, b1.2, sd.1)
```

Arguments

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

Value

Vector

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData for simulated data generating function.

Examples

```
Data2P <- SimulateData2P(N = 10^3,
                         P1 = 1,
                         P2 = 1,
                         b0.1 = 0,
                         b1.1 = 1,
                         b1.2 = 1,
                         sd.1 = 1)
```

`SimulateData3P` *Simulate data for a node with three parents*

Description

Simulate data for a node with three parents

Usage

```
SimulateData3P(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 node, P1.
`P2`	Data vector of the parent node, P2.
`P3`	Data vector of the parent node, 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 node.
`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 node.
`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 node.
`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 node.
`sd.1`	Standard deviation for corresponding data generated node.

Value

Vector

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData for simulated data generating function.

Examples

```
Data3P <- SimulateData3P(N = 10^3,
                         P1 = 1,
                         P2 = 1,
                         P3 = 1,
                         b0.1 = 0,
                         b1.1 = 1,
                         b1.2 = 1,
                         b1.3 = 1,
                         sd.1 = 1)
```

SimulateDataNP *Simulate data for a node with no parent*

Description

Simulate data for a node with no parent

Usage

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

Arguments

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

Value

Vector

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData for simulated data generating function.

Examples

```
DataNP <- SimulateDataNP(N = 10^3,
                         b0.1 = 0,
                         sd.1 = 1)
```

simu_data_layered *Data for the layered model*

Description

Data simulated under the layered Model.

Details

The columns of the data matrix are the genetic variant (V node) and phenotype nodes (T nodes).

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData.

simu_data_M0 *Data for Model 0*

Description

Data simulated under Model 0.

Details

The columns of the data matrix are the genetic variant (V node) and phenotype nodes (T nodes).

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData.

simu_data_M1 *Data for Model 1*

Description

Data simulated under Model 1.

Details

The columns of the data matrix are the genetic variant (V node) and phenotype nodes (T nodes).

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData.

simu_data_M2 *Data for Model 2*

Description

Data simulated under Model 2.

Details

The columns of the data matrix are the genetic variant (V node) and phenotype nodes (T nodes).

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData.

simu_data_M3 *Data for Model 3*

Description

Data simulated under Model 3.

Details

The columns of the data matrix are the genetic variant (V node) and phenotype nodes (T nodes).

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData.

simu_data_M4 *Data for Model 4*

Description

Data simulated under Model 4.

Details

The columns of the data matrix are the genetic variant (V node) and phenotype nodes (T nodes).

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData.

simu_data_multiparent *Data for the multiple-parent model*

Description

Data simulated under the multiple-parent model, where a phenotype node has multiple parent nodes.

Details

The columns of the data matrix are the genetic variant (V node) and phenotype nodes (T nodes).

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData.

simu_data_starshaped *Data for the star model*

Description

Data simulated under the star model, where one gene has more than two children.

Details

The columns of the data matrix are the genetic variant (V node) and phenotype nodes (T nodes).

Value

Matrix

Author(s)

Md Bahadur Badsha (mbbadshar@gmail.com)

See Also

SimulateData.

Index