# Package 'CompareCausalNetworks'

May 18, 2018

**Type** Package

**Title** Interface to Diverse Estimation Methods of Causal Networks

**Version** 0.2.2

**Date** 2018-05-18

**Author** Christina Heinze-Deml <heinzedeml@stat.math.ethz.ch>,
Nicolai Meinshausen <meinshausen@stat.math.ethz.ch>

**Depends** R (>= 3.1.0)

**Imports** methods, Matrix, expm, data.table

**Maintainer** Christina Heinze-Deml <heinzedeml@stat.math.ethz.ch>

**Description** Unified interface for the estimation of causal networks, including the methods 'backShift' (from package 'backShift'), 'bivariateANM' (bivariate additive noise model), 'bivariateCAM' (bivariate causal additive model), 'CAM' (causal additive model) (from package 'CAM'), 'hiddenICP' (invariant causal prediction with hidden variables), 'ICP' (invariant causal prediction) (from package 'InvariantCausalPrediction'), 'GES' (greedy equivalence search), 'GIES' (greedy interventional equivalence search), 'LINGAM', 'PC' (PC Algorithm), 'FCI' (fast causal inference), 'RFCI' (really fast causal inference) (all from package 'pcalg') and regression.

**License** GPL

**LazyData** true

**Suggests** pcalg, InvariantCausalPrediction, glmnet, backShift, CAM, kernlab, mgcv, mboost, bnlearn, testthat, huge, flare

**URL** https://github.com/christinaheinze/CompareCausalNetworks

**BugReports** https://github.com/christinaheinze/CompareCausalNetworks/issues

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2018-05-18 13:42:55 UTC

# R topics documented:



---

`CompareCausalNetworks-package` *Compare estimates of causal graphs using a unified interface to various methods*

---

## Description

Provides a unified interface to various causal graph estimation methods.

## Details

| | |
|---|---|
| Package: | CompareCausalNetworks |
| Type: | Package |
| Version: | 0.2.2 |
| Date: | 2018-05-18 |
| License: | GPL |

The causal graphs can be estimated with function `getParents` and a stability-selection version is available at `getParentsStable`.

The supported methods are provided through the packages listed in `Suggests`. Thus, to use a particular method the corresponding package needs to be installed on your machine. To run the examples, most of these packages need to be installed.

## Author(s)

Christina Heinze-Deml <heinzedeml@stat.math.ethz.ch>, Nicolai Meinshausen <meinshausen@stat.math.ethz.ch>

---

`getParents` *Estimate the connectivity matrix of a causal graph*

---

## Description

Estimates the connectivity matrix of a directed causal graph, using various possible methods. Supported methods at the moment are ARGES, backShift, bivariateANM, bivariateCAM, CAM, FCI, FCI+, GES, GIES, hiddenICP, ICP, LINGAM, MMHC, rankARGES, rankFci, rankGES, rankGIES, rankPC, regression, RFCI and PC.

## Usage

```
getParents(X, environment = NULL, interventions = NULL,
  parentsOf = 1:ncol(X), method = c("arges", "backShift", "bivariateANM",
  "bivariateCAM", "CAM", "fci", "fciplus", "ges", "gies", "hiddenICP", "ICP",
  "LINGAM", "mmhc", "rankArges", "rankFci", "rankGes", "rankGies", "rankPc",
  "rfci", "pc", "regression")[12], alpha = 0.1, mode = c("raw", "parental",
  "ancestral")[1], variableSelMat = NULL, excludeTargetInterventions = TRUE,
  onlyObservationalData = FALSE, indexObservationalData = 1,
  returnAsList = FALSE, sparse = FALSE, directed = FALSE,
  pointConf = FALSE, setOptions = list(), assumeNoSelectionVars = TRUE,
  verbose = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `X` | A $(n$ x $p)$-data matrix with n observations of $p$ variables. |
| `environment` | An optional vector of length $n$, where the entry for observation $i$ is an index for the environment in which observation $i$ took place (Simplest case: entries `1` for observational data and entries `2` for interventional data of unspecified type. Encoding for observational data can be changed with `indexObservationalData`). Is required for methods `ICP`, `hiddenICP` and `backShift`. |
| `interventions` | A optional list of length $n$. The entry for observation $i$ is a numeric vector that specifies the variables on which interventions happened for observation $i$ (a scalar if an intervention happened on just one variable and `integer(0)` if no intervention occured for this observation). Is used for methods `gies`, `rankGies` and `CAM` and will generate the vector `environment` if the latter is set to `NULL`. (However, this might generate too many different environments for some data sets, so a hand-picked vector `environment` is preferable). Is also used for `ICP` and `hiddenICP` to exclude interventions on the target variable of interest. |
| `parentsOf` | The variables for which we would like to estimate the parents. Default are all variables. Currently only used with `mode = "raw"`. Speeds up computation for methods `bivariateANM`, `bivariateCAM`, `ICP`, `hiddenICP` and `regression`; for other methods only affects output. Also see `variableSelMat` for possibly speeding up computational time by restricting the set of potential parents for a variable. |
| `method` | A string that specfies the method to use. The methods `pc` (PC-algorithm), `LINGAM` (LINGAM), `arges` (Adaptively restricted greedy equivalence search), `ges` (Greedy equivalence search), `gies` (Greedy interventional equivalence search), `fci` (Fast causal inference), `fciplus` and `rfci` (Really fast causal inference) are imported from the package "pcalg" and are documented there in more detail, including the additional options that can be supplied via `setOptions`. The "rank |

versions" of arges, fci, ges, gies and pc are based on [1]. The method CAM (Causal additive models) is documented in the package "CAM" and the methods ICP (Invariant causal prediction), hiddenICP (Invariant causal prediction with hidden variables) are from the package "InvariantCausalPrediction". The method backShift comes from the package "backShift". The method mmhc comes from the package "bnlearn". Finally, the methods bivariateANM and bivariateCAM are for now implemented internally but will hopefully be part of another package at some point in the near future.

alpha — The level at which tests are done. This leads to confidence intervals for ICP and hiddenICP and is used internally for pc, rankPc, mmhc, fci, rankFci, fciplus and rfci. For all other methods alpha is not used.

mode — Determines output type - can be "raw" or one of the queries "isParent", "isMaybeParent", "isNoParent", "isAncestor","isMaybeAncestor", "isNoAncestor". If "raw", getParents() returns the connectivity matrix computed by the specified method in sparse matrix format if sparse is set to TRUE; else in dense matrix format (or as list if returnAsList = TRUE). The options directed and pointConf will be ignored for all modes except for "raw" if set to TRUE. The different mode types are explained in the help for getRanking.

variableSelMat — An optional logical matrix of dimension $(p \text{ x } p)$. An entry TRUE for entry $(i, j)$ says that variable $i$ should be considered as a potential parent for variable $j$ and vice versa for FALSE. If the default value of NULL is used, all variables will be considered, but this can be very slow, especially for methods pc, ges, gies, rfci and CAM. Ignored for methods backShift, fciplus, LINGAM and CAM.

excludeTargetInterventions — When looking for parents of variable $k$ in $1, ..., p$, set to TRUE if observations where an intervention on variable $k$ occured should be excluded. Default is TRUE. Used in ICP and hiddenICP.

onlyObservationalData — If set to TRUE, only observational data is used. It will take the index in environment specified by indexObservationalData. If environment is NULL, all observations are used. Default is FALSE.

indexObservationalData — Index in environment that encodes observational data. Default is 1.

returnAsList — If set to TRUE, will return a list, where entry $k$ is a list containing the estimated parents of variable $k$. Default is FALSE.

sparse — If set to TRUE and returnAsList is FALSE, output matrix will be in sparse matrix format.

directed — If TRUE, an edge will be returned if and only if an edge has been detected to be directed. I.e. entry will be set to 0 for entry $(j, k)$ if both $j- > k$ and $k- > j$ are estimated (ICP, hiddenICP, regression), if $j - -k$ is undirected (in case of CPDAGs) or if the edge type is not of type $i - - > j$ in case of PAGs. If assumeNoSelectionVars = TRUE the edge type $i - -oj$ is also considered 'directed' for methods returning PAGs. Default is FALSE. Only supported in mode "raw".

pointConf — If TRUE, numerical estimates will be returned if possible. For methods ICP and hiddenICP, these are the values in the individual confidence intervals (at chosen

level alpha) that are closest to 0; for other methods these are point estimates. Some methods do not return numerical point estimates; for these the output will remain binary 0/1 (no-edge/edge). Default is FALSE. Only supported in mode "raw".

setOptions — A list that can take method-specific options; see the individual documentations of the methods for more options and their possible values.

assumeNoSelectionVars — Set to TRUE is you want to assume the absence of selection variables.

verbose — If TRUE, detailed output is provided.

... — Parameters to be passed to underlying method's function.

## Value

If option returnAsList is FALSE, a sparse matrix, where a 0 entry in position (j,k) corresponds to an estimate of "no edge" j -> k, while an entry 1 corresponds to an estimated egde. If option pointConf is TRUE, the 1 entries will be replaced by numerical values that are either point estimates of the causal coefficients or confidence bounds (see above). If option returnAsList is TRUE, a list will be returned. The k-th entry in the list is the numeric vector with the indices of the estimated parents of node k.

## Author(s)

Christina Heinze-Deml <heinzedeml@stat.math.ethz.ch>, Nicolai Meinshausen <meinshausen@stat.math.ethz.ch>

## References

1. Naftali Harris and Mathias Drton: PC Algorithm for Nonparanormal Graphical Models. J. Mach. Learn. Res. 14(1) 2013.

## See Also

getParentsStable for stability selection-based estimation of the causal graph.

## Examples

```
## load the backShift package for data generation and plotting functionality
if(require(backShift) & require(pcalg)){
  # Simulate data with connectivity matrix A with assumptions
  # 1) hidden variables present
  # 2) precise location of interventions is assumed unknown
  # 3) different environments can be distinguished

  ## simulate data
  myseed <- 1

  # sample size n
  n <- 10000

  # p=3 predictor variables and connectivity matrix A
  p <- 3
```

```
labels <- c("1", "2", "3")
A <- diag(p)*0
A[1,2] <- 0.8
A[2,3] <- 0.8
A[3,1] <- -0.4

# divide data in 10 different environments
G <- 10

# simulate
simResult <- backShift::simulateInterventions(n, p, A, G, intervMultiplier = 3,
             noiseMult = 1, nonGauss = TRUE, hiddenVars = TRUE,
             knownInterventions = FALSE, fracVarInt = NULL, simulateObs = TRUE,
             seed = myseed)
X <- simResult$X
environment <- simResult$environment

## apply all  methods given in vector 'methods'
## (using all data pooled for pc/LINGAM/rfci/ges -- can be changed with option
## 'onlyObservationalData=TRUE')

methods <- c("backShift", "LINGAM") #c("pc", "rfci", "ges")

# select whether you want to run stability selection
stability <- FALSE

# arrange graphical output into a rectangular grid
sq <- ceiling(sqrt(length(methods)+1))
par(mfrow=c(ceiling((length(methods)+1)/sq),sq))

## plot and print true graph
cat("\n true graph is  ------  \n" )
print(A)
plotGraphEdgeAttr(A, plotStabSelec = FALSE, labels = labels, thres.point = 0,
 main = "TRUE GRAPH")

## loop over all methods and compute and print/plot estimate
for (method in methods){
  cat("\n result for method", method,"  ------  \n" )

  if(!stability){
    # Option 1): use this estimator as a point estimate
    Ahat <- getParents(X, environment, method=method, alpha=0.1, pointConf = TRUE)
  }else{
    # Option 2): use a stability selection based estimator
    # with expected number of false positives bounded by EV=2
    Ahat <- getParentsStable(X, environment, EV=2, method=method, alpha=0.1)
  }

 # print and plot estimate (point estimate thresholded if numerical estimates
 # are returned)
  print(Ahat)
  if(!stability)
```

```
      plotGraphEdgeAttr(Ahat, plotStabSelec = FALSE, labels = labels,
       thres.point = 0.05,
       main=paste("POINT ESTIMATE FOR METHOD\n", toupper(method)))
    else
      plotGraphEdgeAttr(Ahat, plotStabSelec = TRUE, labels = labels,
       thres.point = 0, main = paste("STABILITY SELECTION
       ESTIMATE\n FOR METHOD", toupper(method)))
   }
}else{
    cat("\nThe packages 'backShift' and 'pcalg' are needed for the examples to
work. Please install them.")
}
```

---

getParentsStable *Estimate the connectivity matrix of a causal graph using stability selection.*

---

## Description

Estimates the connectivity matrix of a directed causal graph, using various possible methods. Supported methods at the moment are ARGES, backShift, bivariateANM, bivariateCAM, CAM, FCI, FCI+, GES, GIES, hiddenICP, ICP, LINGAM, MMHC, rankARGES, rankFci, rankGES, rankGIES, rankPC, regression, RFCI and PC. Uses stability selection to select an appropriate sparseness.

## Usage

```
getParentsStable(X, environment, interventions = NULL, EV = 1,
  nodewise = TRUE, threshold = 0.75, nsim = 100,
  sampleSettings = 1/sqrt(2), sampleObservations = 1/sqrt(2),
  parentsOf = 1:ncol(X), method = c("ICP", "hiddenICP", "backShift", "pc",
  "LINGAM", "ges", "gies", "CAM", "fci", "rfci", "regression", "bivariateANM",
  "bivariateCAM")[1], alpha = 0.1, mode = c("raw", "parental",
  "ancestral")[1], variableSelMat = NULL, excludeTargetInterventions = TRUE,
  onlyObservationalData = FALSE, indexObservationalData = NULL,
  setOptions = list(), verbose = FALSE)
```

## Arguments

| | |
|---|---|
| X | A (nxp)-data matrix with n observations of p variables. |
| environment | A vector of length n, where the entry for observation i is an index for the environment in which observation i took place (simplest case entries 1 for observational data and entries 2 for interventional data of unspecified type). Is required for methods ICP, hiddenICP, backShift. |

`interventions` A optional list of length n. The entry for observation i is a numeric vector that specifies the variables on which interventions happened for observation i (a scalar if an intervention happened on just one variable and `numeric(0)` if no intervention occured for this observation). Is used for method `gies` but will generate the vector `environment` if this is set to `NULL` (even though it might generate too many different environments for some data so a hand-picked vector `environment` is preferable). Is also used for `ICP` and `hiddenICP` to exclude interventions on the target variable of interest.

`EV` A bound on the expected number of falsely selected edges.

`nodewise` If `FALSE`, stability selection retains for each subsample the largest overall entries in the connectivity matrix. If `TRUE`, values are ordered row- and node-wise first and then the largest entries in each row and column are retained. Error control is valid (under exchangeability assumption) in both cases. The latter setting `TRUE` is perhaps more robust and is the default.

`threshold` The empirical selection frequency in (0.5,1) under subsampling that needs to be surpassed for an edge to be selected.

`nsim` The number of resamples for stability selection.

`sampleSettings` The fraction of different environments to resample in each resampling (at least two different environments will be selected so the argument is without effect if there are just two different environments in total).

`sampleObservations`
The fraction of samples to resample in each environment.

`parentsOf` The variables for which we would like to estimate the parents. Default are all variables.

`method` A string that specfies the method to use. The methods `pc` (PC-algorithm), `LINGAM` (LINGAM), `arges` (Adaptively restricted greedy equivalence search), `ges` (Greedy equivalence search), `gies` (Greedy interventional equivalence search), `fci` (Fast causal inference) and `rfci` (Really fast causal inference) are imported from the package "pcalg" and are documented there in more detail, including the additional options that can be supplied via `setOptions`. The method `CAM` (Causal additive models) is documented in the package "CAM" and the methods `ICP` (Invariant causal prediction), `hiddenICP` (Invariant causal prediction with hidden variables) are from the package "InvariantCausalPrediction". The method `backShift` comes from the package "backShift". The method `mmhc` comes from the package "bnlearn". Finally, the methods `bivariateANM` and `bivariateCAM` are for now implemented internally but will hopefully be part of another package at some point in the near future.

`alpha` The level at which tests are done. This leads to confidence intervals for `ICP` and `hiddenICP` and is used internally for `pc` and `rfci`.

`mode` Output type - can be "raw", "parental" or "ancestral". If "raw" output is the output of the underlying method, without modifications. If "parental" output described parental relations; if "ancestral" output is casted to ancestral relations. #TODO explain further

`variableSelMat` An optional logical matrix of dimension (pxp). An entry `TRUE` for entry (i,j) says that variable i should be considered as a potential parent for variable j and

vice versa for FALSE. If the default value of NULL is used, all variables will be considered, but this can be very slow, especially for methods pc, ges, gies, rfci and CAM.

excludeTargetInterventions
: When looking for parents of variable k in 1,...,p, set to TRUE if observations where an intervention on variable k occured should be excluded. Default is TRUE.

onlyObservationalData
: If set to TRUE, only observational data is used. It will take the index in environment specified by indexObservationalData. If environment is NULL, all observations are used. Default is FALSE.

indexObservationalData
: Index in environment that encodes observational data. Default is 1.

setOptions
: A list that can take method-specific options; see the individual documentations of the methods for more options and their possible values.

verbose
: If TRUE, detailed output is provided.

...
: Parameters to be passed to underlying method's function.

## Value

A sparse matrix, where a 0 entry in (j,k) corresponds to an estimate of 'no edge' j -> parentsOf[k]. Entries between 0 and 100 give the selection percentage of this edge over all resamples (set to 0 if below critical threshold) and all non-zero values are considered as selected edges.

## Author(s)

Nicolai Meinshausen <meinshausen@stat.math.ethz.ch>, Christina Heinze-Deml <heinzedeml@stat.math.ethz.ch>

## References

Stability selection (2010): N. Meinshausen and P. Buhlmann, Journal of the Royal Statistical Society: Series B, 72, 417-473

## See Also

getParents for the underlying point-estimate of the causal graph.

---

# getRanking *Estimate a ranking of edges for causal relations in the underlying graph structure using stability ranking.*

---

## Description

Estimates a ranking of edges for a given query, e.g. for parental relations in the underlying causal graph structure, using various possible methods.

Supported methods at the moment are ARGES, backShift, bivariateANM, bivariateCAM, CAM, FCI, FCI+, GES, GIES, hiddenICP, ICP, LINGAM, MMHC, rankARGES, rankFci, rankGES, rankGIES, rankPC, regression, RFCI and PC.

## Usage

```
getRanking(X, environment, interventions = NULL, queries = c("isParent",
  "isMaybeParent", "isNoParent", "isAncestor", "isMaybeAncestor",
  "isNoAncestor"), method = c("ICP", "hiddenICP", "backShift", "pc", "LINGAM",
  "ges", "gies", "CAM", "fci", "rfci", "regression", "bivariateANM",
  "bivariateCAM")[1], alpha = 0.1, variableSelMat = NULL,
  excludeTargetInterventions = TRUE, onlyObservationalData = FALSE,
  indexObservationalData = NULL, setOptions = list(),
  assumeNoSelectionVars = TRUE, nsim = 100, sampleSettings = 1/sqrt(2),
  sampleObservations = 1/sqrt(2), verbose = FALSE, ...)
```

## Arguments

| | |
|---|---|
| X | A $(nxp)$-data matrix with $n$ observations of $p$ variables. |
| environment | A vector of length $n$, where the entry for observation $i$ is an index for the environment in which observation $i$ took place (simplest case entries 1 for observational data and entries 2 for interventional data of unspecified type). Is required for methods ICP, hiddenICP, backShift. |
| interventions | A optional list of length n. The entry for observation i is a numeric vector that specifies the variables on which interventions happened for observation i (a scalar if an intervention happened on just one variable and numeric(0) if no intervention occured for this observation). Is used for method gies but will generate the vector environment if this is set to NULL (even though it might generate too many different environments for some data so a hand-picked vector environment is preferable). Is also used for ICP and hiddenICP to exclude interventions on the target variable of interest. |
| queries | One (or more of) "isParent", "isMaybeParent", "isNoParent", "isAncestor","isMaybeAncestor", "isNoAncestor" |
| method | A string that specfies the method to use. The methods pc (PC-algorithm), LINGAM (LINGAM), arges (Adaptively restricted greedy equivalence search), ges (Greedy equivalence search), gies (Greedy interventional equivalence search), fci (Fast causal inference) and rfci (Really fast causal inference) are imported from the package "pcalg" and are documented there in more detail, including the additional options that can be supplied via setOptions. The method CAM (Causal additive models) is documented in the package "CAM" and the methods ICP (Invariant causal prediction), hiddenICP (Invariant causal prediction with hidden variables) are from the package "InvariantCausalPrediction". The method backShift comes from the package "backShift". The method mmhc comes from the package "bnlearn". Finally, the methods bivariateANM and bivariateCAM are for now implemented internally but will hopefully be part of another package at some point in the near future. |
| alpha | The level at which tests are done. This leads to confidence intervals for ICP and hiddenICP and is used internally for pc and rfci. |
| variableSelMat | An optional logical matrix of dimension (pxp). An entry TRUE for entry (i,j) says that variable i should be considered as a potential parent for variable j and vice versa for FALSE. If the default value of NULL is used, all variables will be |

considered, but this can be very slow, especially for methods pc, ges, gies, rfci and CAM.

excludeTargetInterventions
: When looking for parents of variable k in 1,...,p, set to TRUE if observations where an intervention on variable k occured should be excluded. Default is TRUE.

onlyObservationalData
: If set to TRUE, only observational data is used. It will take the index in environment specified by indexObservationalData. If environment is NULL, all observations are used. Default is FALSE.

indexObservationalData
: Index in environment that encodes observational data. Default is 1.

setOptions
: A list that can take method-specific options; see the individual documentations of the methods for more options and their possible values.

assumeNoSelectionVars
: Set to TRUE is you want to assume the absence of selection variables.

nsim
: The number of resamples for stability selection.

sampleSettings
: The fraction of different environments to resample in each resampling (at least two different environments will be selected so the argument is without effect if there are just two different environments in total).

sampleObservations
: The fraction of samples to resample in each environment.

verbose
: If TRUE, detailed output is provided.

...
: Parameters to be passed to underlying method's function.

## Details

For both parental and ancestral relations, three queries are supported. The existence of a relation is assessed by the queries isParent and isAncestor; the absence of a relation is assessed by the queries isNoParent and isNoAncestor; the potential existence of a relation is addressed by the queries isMaybeParent and isMaybeAncestor.

All queries return a connectivity matrix which we denote by $A$. The interpretation of the entries of $A$ differs according to the considered query:

**Parental relations:** Queries concerning parental relations can only be answered by those methods under consideration that return a DAG, a CPDAG or a directed cyclic graph. When we say that a particular method cannot answer a given query, then the method's output with respect to this query will be the zero matrix. However, the eventual ranking for such a query will not necessarily be random due to the tie breaking scheme that is applied when ranking pairs of variables (see below).

1. isParent In the connectivity matrix $A$ returned by this query, the entry $A_{i,j} = 1$ means that there is *a directed edge* from node $i$ to node $j$ in the graph structure estimated by the method under consideration. Otherwise, $A_{i,j} = 0$.
2. isMaybeParent $A_{i,j} = 1$ means that there is *a directed or an undirected edge* from node $i$ to node $j$ in the estimated graph structure. Otherwise, $A_{i,j} = 0$.
3. isNoParent $A_{i,j} = 1$ means that there is neither a directed nor an undirected edge from node $i$ to node $j$ in the estimated graph structure. Otherwise, $A_{i,j} = 0$.

**Ancestral relations:** Queries concerning ancestral relations can be answered by all methods under consideration.

1. `isAncestor` $A_{i,j} = 1$ means that there is a *directed path* from node $i$ to node $j$ in the estimated graph structure. Otherwise, $A_{i,j} = 0$. In case of PAGs, directed paths can contain the edge types $i - - > j$ and $i - -oj$. Including the latter edge type in this category implies that we exclude the existence of selection variables.
2. `isMaybeAncestor` $A_{i,j} = 1$ then means that there is a path from node $i$ to node $j$ that contains directed and/or undirected edges. Otherwise, $A_{i,j} = 0$. For PAGs, such paths can contain the edge types $i - - > j$, $i - -oj$, $io - oj$ and/or $io - > j$. Otherwise, $A_{i,j} = 0$.
3. `isNoAncestor` $A_{i,j} = 1$ means that there is neither a directed path nor a partially directed path from node $i$ to node $j$ in the estimated graph structure. Otherwise, $A_{i,j} = 0$.

**Stability ranking:** To obtain a ranking of edges for a given set of queries, we run the method under consideration on `nsims` random subsamples of the data. In each round, we draw samples from a fraction of settings, where the size of the fraction is specified by `sampleSettings`. In each chosen setting, we sample a fraction of observations uniformly at random without replacement, where the size of the fraction is specified by `sampleObservations`.

For each subsample we randomly permute the order of the variables in the input. Methods that are order-dependent can therefore not exploit any potential advantage stemming from a data matrix with columns ordered according to the causal ordering or a similar one. We then run the method on each subsample.

For each subsample and a particular query, we obtain the corresponding connectivity matrix $A$. We can then rank all pairs of nodes $i, j$ according to the frequency of the occurrence of $A_{i,j} = 1$ across subsamples. Ties between pairs of variables can be broken with the results of the other queries if they are also computed as specified by `queries`; otherwise ties are broken at random:

- If the query is `isParent`, ties are broken with counts for `isMaybeParent`.
- For the query `isMaybeParent` ties are broken with counts for `isParent`, i.e. in case of equal counts we give a preference to the edge that was considered more often to be a 'certain' parent. For methods returning DAGs this scheme makes the ranking for `isMaybeParent` equal to the result for `isParent`, up to the random tie breaking that is applied for `isParent`.
- If the query is `isNoParent`, ties are broken according to which edge was selected less often in the query `isMaybeParent`.
- If the query is `isAncestor`, ties are broken with counts for `isMaybeAncestor`.
- For the query `isMaybeAncestor` ties are broken with counts for `isAncestor`, i.e. in case of equal counts we give a preference to the edge that was considered more often to be a 'certain' ancestor. For methods returning DAGs this scheme makes the ranking for `isMaybeAncestor` equal to the result for `isAncestor`, up to the random tie breaking that is applied for `isAncestor`.
- If the query is `isNoAncestor`, ties are broken according to which one was selected less often in the query `isMaybeAncestor`.

If the tie breaking matrix defined according to these rules is 0, a matrix with standard normal random entries is used to break ties. Similarly, if there are remaining ties after applying the tie breaking rules described above, ties are broken randomly.

## Value

A list with the following entries:

- `ranking` A list of length `length(queries)`. For each query, the corresponding list entry contains a matrix of dimension $(pxp)x2$ with the ranking of edges. E.g. the first row indicates that the edge from ranking$isParent[1,1] to ranking$isParent[1,2] is the most likely edge according to the method under consideration.
- `resList` A list of length `length(queries)`. For each query, the corresponding list entry contains a matrix of dimension $(pxp)$ with the counts for $A_{i,j} = 1$ across the `nsim` subsamples.
- `simEstimates` A list of length `nsim` with the method's output for each of the `nsim` subsamples.

## Author(s)

Christina Heinze-Deml <heinzedeml@stat.math.ethz.ch>

## See Also

`getParents` for the underlying point-estimate of the causal graph.

## Examples

```
data("simDataInv")
X <- simDataInv$X
set.seed(1)
if(require(pcalg)){
  rank <- getRanking(X,
                environment = simDataInv$environment,
                queries = c("isParent","isMaybeParent"),
                method = c("LINGAM"),
                verbose = FALSE)
  # estimated ranking
  print(rank$ranking$isParent)

  # true adjacency matrix
  print(simDataInv$configs$trueA)
}else{
  cat("\nThe packages 'pcalg' is needed for the example to
work. Please install it.")
}
```

---

# simDataInv *Data from a causal model with interventions*

---

## Description

A dataset to run the tests.

## Usage

```
simDataInv
```

## Format

A list created by `simulateInterventions`. All inputs are contained in the list element `configs`. For details see the help page of `simulateInterventions`.

---

`simulateInterventions` *Simulate data of a causal (possibly cyclic model) under interventions.*

---

## Description

Simulate data of a causal (possibly cyclic model) under interventions.

## Usage

```
simulateInterventions(n, p, df, rhoNoise, snrPar, sparse, doInterv, numberInt,
  strengthInt, cyclic, strengthCycle, modelMis = FALSE, modelMisPar = 1,
  seed = 1)
```

## Arguments

| | |
|---|---|
| `n` | Number of observations. |
| `p` | Number of variables. |
| `df` | Degrees of freedom in t-distribution of noise and interventions. |
| `rhoNoise` | Correlation between noise terms to model hidden variabkes. Set to 0 for independent noise. |
| `snrPar` | Signal-to-noise parameter: steers what proportion of the variance stems from the signal resp.\ from the noise: The SNR is given by $SNR = (1-`snrPar`)/`snrPar`$), see details. Only holds when `cyclic = FALSE`. |
| `sparse` | Probability that an entry $i, j$ in adjacency matrix is 1. |
| `doInterv` | Set to TRUE if interventions should be do-interventions; otherwise noise interventions (also called shift interventions) are generated. |
| `numberInt` | Total number of settings. |
| `strengthInt` | Regulates the strength of the interventions, see details. |
| `cyclic` | Set to TRUE is resulting graph should contain a cycle. |
| `strengthCycle` | Steers strength of feedback, see details. |
| `modelMis` | Add a model misspecification that applies `tanh(modelMisPar*x)/modelMisPar)` morginally to each variable after having generated X from the causal DAG. |
| `modelMisPar` | Parameter steering the strength of the model misspecification. |
| `seed` | Random seed. |

## Details

The adjacency matrix $A$ is generated as follows. Assume the variables with indices $1, \ldots, p$ are causally ordered. For each edge from node $i$ to node $j$ where $i$ precedes $j$ in the causal ordering, we draw a sample from Bin(sparse) to determine whether to add an edge from node $i$ to node $j$. After having sampled the non-zero entries of $A$ in this fashion, we sample the coefficients from Unif(-1,1). As described below, the edge weights are later rescaled to achieve a specified signal-to-noise ratio. We exclude the possibility of $A = 0$, i.e. we resample until $A$ contains at least one non-zero entry.

Second, the interventions are generated as follows. `numberInt` denotes the total number of (interventional and observational) settings that are generated. For each variable, we sample uniformly at random with replacement one setting in which this variable is intervened on. In other words, each variable is intervened on in exactly one setting. Hence it is possible that there are settings where no interventions take place which then correspond to the observational case. Similarly, there may be settings where interventions are performed on multiple variables at once. After defining the settings, we sample (uniformly at random with replacement) what setting each data point belongs to. So for each setting we generate approximately the same number of samples. In one generated data set, the interventions are all of the same type, i.e. they are either all shift interventions (when `doInterv = FALSE`) or do-interventions (when `doInterv = TRUE`). In both cases, an intervention on $X_j$ is modelled by generating $Z_j$ as $Z_j$ `strengthInt` $*t($`dfNoise`$)$. If `strengthInt` = 0, all interventional settings correspond to purely observational data.

Third, the noise terms $\epsilon$ are generated by first sampling from $N(0, \Sigma)$ where $\Sigma_{i,i} = 1$ and $\Sigma_{i,j} =$ `rhoNoise`. To steer the signal-to-noise ratio, we set the variance of the noise terms of all nodes except source nodes to `snrPar` where $0 <$`snrPar`$\leq 1$. Stepping through the variables in causal order, for each variable $X_j$ that has parents, we uniformly rescale the edge weights $\beta_{j,k}$ for $k = 1, \ldots, p$ in the structural equation of variable $X_j$ such that the variance of the sum $\sum_{k=1}^{p} \beta_{j,k} X_k + \epsilon_j$ is approximately 1 in the observational setting. In other words, the parameter `snrPar` steers what proportion of the variance stems from the signal given by $\sum_{k=1}^{p} \beta_{j,k} X_k$ and what proportion stems from the noise $\epsilon_j$. The signal-to-noise ratio can then be computed as SNR = (1-`snrPar`)/`snrPar`.

Forth, a cycle is added to the causal graph if `cyclic = TRUE`. If the causal graph shall contain a cycle, we sample two nodes $i$ and $j$ such that adding an edge between them creates a cycle in the causal graph. We then compute the largest possible coefficient for this edge such that the cycle product is smaller than 1. Subsequently, we sample the sign of the coefficient and set the magnitude by scaling the largest possible coefficient by `strengthCycle` where $0 <$`strengthCycle`$< 1$.

Fifth, we rescale the noise variables to obtain a $t$-distribution with `dfNoise` degrees of freedom. $X$ is then generated as $X = (I - A)^{-1}\epsilon$ in the observational case; under a shift interventions $X$ can be generated as $X = (I - A)^{-1}(\epsilon + Z)$ where the coordinates of $Z$ are only non-zero for the variables that are intervened on. Under a do-intervention on $X_j$, $\beta_{j,k}$ for $k = 1, \ldots, p$ are set to 0 to yield $A'$ and $\epsilon_j$ is set to $Z_j$ to yield $\epsilon'_j$. We then obtain $X$ as $X = (I - A')^{-1}\epsilon'$.

Lastly, if `modelMis = TRUE` a model misspecification is added to the data by marginally transforming all variables as `tanh(modelMisPar*x)/modelMisPar)`.

## Value

A list with the following elements:

- `X` $nxp$-dimensional data matrix

- `environment` Indicator of the experiment or the intervention type an observation belongs to. A numeric vector of length $n$.
- `interventions` A list of length $n$. Indicates location of interventions for each data point.
- `whereInt` A list of length `numberInt`. Indicates location of interventions in each setting.
- `noise`
- `configs` A list with the generated adjacency matrix (`trueA`) as well as all input arguments.

# Index