Package 'causalDisco'

May 12, 2022

Title Tools for Causal Discovery on Observational Data

Version 0.9.1

Description Various tools for inferring causal models from observational data. The package includes an implementation of the temporal Peter-Clark (TPC) algorithm. Petersen, Osler and Ekstrøm (2021) <doi:10.1093/aje/kwab087>. It also includes general tools for evaluating differences in adjacency matrices, which can be used for evaluating performance of causal discovery procedures.

Depends R (>= 3.5.0)

License GPL-2

Encoding UTF-8

LazyData true

URL `https://github.com/annennenne/causalDisco`

BugReports `https://github.com/annennenne/causalDisco/issues`

RoxygenNote 7.1.1

Imports pcalg, igraph, RColorBrewer, gtools, clipr, methods, scales

NeedsCompilation no

Author Anne Helby Petersen [aut, cre]

Maintainer Anne Helby Petersen <ahpe@sund.ku.dk>

Repository CRAN

Date/Publication 2022-05-12 07:50:06 UTC

R topics documented:

`adj_confusion` *Compute confusion matrix for comparing two adjacency matrices*

Description

Two adjacency matrices are compared either in terms of adjacencies (`type = "adj"`) or orientations (`type = "dir"`).

Usage

```
adj_confusion(est_amat, true_amat)
```

Arguments

`est_amat`	The estimated adjacency matrix
`true_amat`	The true adjacency matrix

Details

In the former case, the confusion matrix is a cross-tabulation of adjacencies.

In the latter case, the orientation confusion matrix is conditional on agreement on adjacency. This means that only adjacencies that are shared in both input matrices are considered, and agreement wrt. orientation is then computed only among these edges that occur in both matrices. A true positive is a correctly placed arrowhead (1), a false positive marks placement of arrowhead (1) where there should have been a tail (0), a false negative marks placement of tail (0) where there should have been an arrowhead (1), and a true negative marks correct placement of a tail (0).

Value

A list with entries `$tp` (number of true positives), `$tn` (number of true negatives), `$fp` (number of false positives), and `$tp` (number of false negatives).

`as.graphNEL` *Convert adjacency matrix to graphNEL object*

Description

Convert adjacency matrix to graphNEL object

Usage

```
as.graphNEL(amat)
```

Arguments

`amat`	An adjacency matrix

Value

A `graphNEL` object, see `graphNEL-class`.

average_degree — *Compute average degree for adjacency matrix*

Description

Computes the average degree, i.e. the number of edges divided by the number of nodes.

Usage

```
average_degree(amat)
```

Arguments

amat	An adjacency matrix

Value

A numeric.

compare — *Compare two tpdag or tskeleton objects*

Description

Compare edges in two tpdag objects or two tskeleton objects. Note that they should be based on the same variables. Only edge absence/presence is compared, not edge orientation.

Usage

```
compare(x, y = NULL)
```

Arguments

x	First object
y	Second object (optional)

Value

A list with entries: `$nedges1` (the number of edges in the first object), `$nedges2` (the number of edges in the second object), `$psi1` (the test significance level of the first object), `$psi2` (the test significance level of the second object), `$nadded` (the number of additional edges in object 2, relative to object 1), and `nremoved` (the number of absent edges in object 2, relative to object 1).

confusion — *Compute confusion matrix for comparing two adjacency matrices*

Description

Two adjacency matrices are compared either in terms of adjacencies (`type = "adj"`) or orientations (`type = "dir"`).

Usage

```
confusion(est_amat, true_amat, type = "adj")
```

Arguments

`est_amat`	The estimated adjacency matrix
`true_amat`	The true adjacency matrix
`type`	String indicating whether the confusion matrix should be computed for adjacencies (`"adj"`, the default) or for (conditional) orientations (`dir`).

Details

In the former case, the confusion matrix is a cross-tabulation of adjacencies.

In the latter case, the orientation confusion matrix is conditional on agreement on adjacency. This means that only adjacencies that are shared in both input matrices are considered, and agreement wrt. orientation is then computed only among these edges that occur in both matrices. A true positive is a correctly placed arrowhead (1), a false positive marks placement of arrowhead (1) where there should have been a tail (0), a false negative marks placement of tail (0) where there should have been an arrowhead (1), and a true negative marks correct placement of a tail (0).

Value

A list with entries `$tp` (number of true positives), `$tn` (number of true negatives), `$fp` (number of false positives), and `$tp` (number of false negatives).

corTest — *Test for vanishing partial correlations*

Description

This function simply calls the `gaussCItest` function from the `pcalg` package.

Usage

```
corTest(x, y, S, suffStat)
```

Arguments

x	Index of x variable
y	Index of y variable
S	Index of S variable(s), possibly NULL
suffStat	Sufficient statistic; list with data, binary variables and order.

Value

A numeric, which is the p-value of the test.

---

dir_confusion *Compute confusion matrix for comparing two adjacency matrices*

Description

Two adjacency matrices are compared either in terms of adjacencies (`type = "adj"`) or orientations (`type = "dir"`).

Usage

```
dir_confusion(est_amat, true_amat)
```

Arguments

est_amat	The estimated adjacency matrix
true_amat	The true adjacency matrix

Details

In the former case, the confusion matrix is a cross-tabulation of adjacencies.

In the latter case, the orientation confusion matrix is conditional on agreement on adjacency. This means that only adjacencies that are shared in both input matrices are considered, and agreement wrt. orientation is then computed only among these edges that occur in both matrices. A true positive is a correctly placed arrowhead (1), a false positive marks placement of arrowhead (1) where there should have been a tail (0), a false negative marks placement of tail (0) where there should have been an arrowhead (1), and a true negative marks correct placement of a tail (0).

Value

A list with entries `$tp` (number of true positives), `$tn` (number of true negatives), `$fp` (number of false positives), and `$tp` (number of false negatives).

edges — *List of edges in adjacency matrix*

Description

Produces a list of edges from an adjacency matrix.

Usage

```
edges(amat)
```

Arguments

`amat`	An adjacency matrix.

Value

A list consisting of two lists: One for oriented edges (`$dir`), and one for unoriented edges (`$undir`).

essgraph2amat — *Convert essential graph to adjacency matrix*

Description

Extracts the adjacency matrix from an `EssGraph-class` object. This object is returned by score-based causal discovery algorithms in the pcalg package.

Usage

```
essgraph2amat(essgraph, p = length(essgraph$field(".nodes")))
```

Arguments

`essgraph`	An `EssGraph` object
`p`	The number of nodes in the graph

Value

An adjacency matrix (square matrix with 0/1 entries).

evaluate *Evaluate adjacency matrix estimation*

Description

Applies several different metrics to evaluate difference between estimated and true adjacency matrices. Intended to be used to evaluate performance of causal discovery algorithms.

Usage

```
evaluate(est, true, metrics, ...)
```

Arguments

`est`	Estimated adjacency matrix/matrices.
`true`	True adjacency matrix/matrices.
`metrics`	List of metrics, see details.
`...`	Further arguments that depend on input type. Currently only `list.out` is allowed, and only if the first argument is a matrix (see details under Value).

Details

Two options for input are available: Either `est` and `true` can be two adjacency matrices, or they can be two arrays of adjacency matrices. The arrays should have shape $n * p * p$ where n is the number of of matrices, and p is the number of nodes/variables.

The metrics should be given as a list with slots `$adj`, `$dir` and `$other`. Metrics under `$adj` are applied to the adjacency confusion matrix, while metrics under `$dir` are applied to the conditional orientation confusion matrix (see confusion). Metrics under `$other` are applied without computing confusion matrices first.

Available metrics to be used with confusion matrices are precision, recall, specificity, FOR, FDR, NPV, F1 and G1. The user can supply custom metrics as well: They need to have the confusion matrix as their first argument and should return a numeric.

Available metrics to be used as "other" is: shd. The user can supply custom metrics as well: They need to have arguments `est_amat` and `true_amat`, where the former is the estimated adjacency matrix and the latter is the true adjacency matrix. The metrics should return a numeric.

Value

A data.frame with one column for each computed metric and one row per evaluated matrix pair. Adjacency metrics are prefixed with "adj_", orientation metrics are prefixed with "dir_", other metrics do not get a prefix. If the first argument is a matrix, `list.out = TRUE` can be used to change the return object to a list instead. This list will contain three lists, where adjacency, orientation and other metrics are reported, respectively.

`evaluate.array` *Evaluate adjacency matrix estimation*

Description

Applies several different metrics to evaluate difference between estimated and true adjacency matrices. Intended to be used to evaluate performance of causal discovery algorithms.

Usage

```
## S3 method for class 'array'
evaluate(est, true, metrics, ...)
```

Arguments

`est`	Estimated adjacency matrix/matrices.
`true`	True adjacency matrix/matrices.
`metrics`	List of metrics, see details.
`...`	Further arguments that depend on input type. Currently only `list.out` is allowed, and only if the first argument is a matrix (see details under Value).

Details

Two options for input are available: Either `est` and `true` can be two adjacency matrices, or they can be two arrays of adjacency matrices. The arrays should have shape $n * p * p$ where n is the number of of matrices, and p is the number of nodes/variables.

The metrics should be given as a list with slots `$adj`, `$dir` and `$other`. Metrics under `$adj` are applied to the adjacency confusion matrix, while metrics under `$dir` are applied to the conditional orientation confusion matrix (see confusion). Metrics under `$other` are applied without computing confusion matrices first.

Available metrics to be used with confusion matrices are precision, recall, specificity, FOR, FDR, NPV, F1 and G1. The user can supply custom metrics as well: They need to have the confusion matrix as their first argument and should return a numeric.

Available metrics to be used as "other" is: shd. The user can supply custom metrics as well: They need to have arguments `est_amat` and `true_amat`, where the former is the estimated adjacency matrix and the latter is the true adjacency matrix. The metrics should return a numeric.

Value

A data.frame with one column for each computed metric and one row per evaluated matrix pair. Adjacency metrics are prefixed with "adj_", orientation metrics are prefixed with "dir_", other metrics do not get a prefix. If the first argument is a matrix, `list.out = TRUE` can be used to change the return object to a list instead. This list will contain three lists, where adjacency, orientation and other metrics are reported, respectively.

`evaluate.matrix` *Evaluate adjacency matrix estimation*

Description

Applies several different metrics to evaluate difference between estimated and true adjacency matrices. Intended to be used to evaluate performance of causal discovery algorithms.

Usage

```
## S3 method for class 'matrix'
evaluate(est, true, metrics, list.out = FALSE, ...)
```

Arguments

`est`	Estimated adjacency matrix/matrices.
`true`	True adjacency matrix/matrices.
`metrics`	List of metrics, see details.
`list.out`	If `FALSE` (default), output is returned as a data.frame, otherwise it will be a list.
`...`	Further arguments that depend on input type. Currently only `list.out` is allowed, and only if the first argument is a matrix (see details under Value).

Details

Two options for input are available: Either `est` and `true` can be two adjacency matrices, or they can be two arrays of adjacency matrices. The arrays should have shape $n * p * p$ where n is the number of of matrices, and p is the number of nodes/variables.

The metrics should be given as a list with slots `$adj`, `$dir` and `$other`. Metrics under `$adj` are applied to the adjacency confusion matrix, while metrics under `$dir` are applied to the conditional orientation confusion matrix (see confusion). Metrics under `$other` are applied without computing confusion matrices first.

Available metrics to be used with confusion matrices are precision, recall, specificity, FOR, FDR, NPV, F1 and G1. The user can supply custom metrics as well: They need to have the confusion matrix as their first argument and should return a numeric.

Available metrics to be used as "other" is: shd. The user can supply custom metrics as well: They need to have arguments `est_amat` and `true_amat`, where the former is the estimated adjacency matrix and the latter is the true adjacency matrix. The metrics should return a numeric.

Value

A data.frame with one column for each computed metric and one row per evaluated matrix pair. Adjacency metrics are prefixed with "adj_", orientation metrics are prefixed with "dir_", other metrics do not get a prefix. If the first argument is a matrix, `list.out = TRUE` can be used to change the return object to a list instead. This list will contain three lists, where adjacency, orientation and other metrics are reported, respectively.

`evaluate.tamat` *Evaluate adjacency matrix estimation*

Description

Applies several different metrics to evaluate difference between estimated and true adjacency matrices. Intended to be used to evaluate performance of causal discovery algorithms.

Usage

```
## S3 method for class 'tamat'
evaluate(est, true, metrics, ...)
```

Arguments

`est`	Estimated adjacency matrix/matrices.
`true`	True adjacency matrix/matrices.
`metrics`	List of metrics, see details.
`...`	Further arguments that depend on input type. Currently only `list.out` is allowed, and only if the first argument is a matrix (see details under Value).

Details

Two options for input are available: Either `est` and `true` can be two adjacency matrices, or they can be two arrays of adjacency matrices. The arrays should have shape $n * p * p$ where n is the number of of matrices, and p is the number of nodes/variables.

The metrics should be given as a list with slots `$adj`, `$dir` and `$other`. Metrics under `$adj` are applied to the adjacency confusion matrix, while metrics under `$dir` are applied to the conditional orientation confusion matrix (see confusion). Metrics under `$other` are applied without computing confusion matrices first.

Available metrics to be used with confusion matrices are precision, recall, specificity, FOR, FDR, NPV, F1 and G1. The user can supply custom metrics as well: They need to have the confusion matrix as their first argument and should return a numeric.

Available metrics to be used as "other" is: shd. The user can supply custom metrics as well: They need to have arguments `est_amat` and `true_amat`, where the former is the estimated adjacency matrix and the latter is the true adjacency matrix. The metrics should return a numeric.

Value

A data.frame with one column for each computed metric and one row per evaluated matrix pair. Adjacency metrics are prefixed with "adj_", orientation metrics are prefixed with "dir_", other metrics do not get a prefix. If the first argument is a matrix, `list.out = TRUE` can be used to change the return object to a list instead. This list will contain three lists, where adjacency, orientation and other metrics are reported, respectively.

F1 *F1 score*

Description

Computes F1 score from a confusion matrix, see confusion. The F1 score is defined as $2 * TP/(2 * TP + FP + FN)$, where TP are true positives, FP are false positives, and FN are false negatives. If TP + FP + FN = 0, 1 is returned.

Usage

```
F1(confusion)
```

Arguments

`confusion`	Confusion matrix as obtained from confusion

Value

A numeric in [0,1].

FDR *False Discovery Rate*

Description

Computes false discovery rate from a confusion matrix, see confusion. False discovery rate is defined as FP/(FP + TP), where FP are false positives and TP are true positives. If FP + TP = 0, 0 is returned.

Usage

```
FDR(confusion)
```

Arguments

`confusion`	Confusion matrix as obtained from confusion

Value

A numeric in [0,1].

FOR — *False Omission Rate*

Description

Computes false omission rate from a confusion matrix, see confusion. False omission rate is defined as FN/(FN + TN), where FN are false negatives and TN are true negatives. If FN + TN = 0, 0 is returned.

Usage

```
FOR(confusion)
```

Arguments

`confusion`	Confusion matrix as obtained from confusion

Value

A numeric in [0,1].

G1 — *G1 score*

Description

Computes G1 score from a confusion matrix, see confusion. G1 score is F1 score with reversed roles of 0/1 classifications, see Petersen et al. 2022. The G1 score is defined as $2 * TN/(2 * TN + FN + FP)$, where TN are true negatives, FP are false positives, and FN are false negatives. If TN + FN + FP = 0, 1 is returned.

Usage

```
G1(confusion)
```

Arguments

`confusion`	Confusion matrix as obtained from confusion

Value

A numeric in [0,1].

References

Petersen, Anne Helby, et al. "Causal discovery for observational sciences using supervised machine learning." arXiv preprint arXiv:2202.12813 (2022).

gausCorScore — *Gaussian L0 score computed on correlation matrix*

Description

The score is intended to be used with score-based causal discovery algorithms from the pcalg package. It is identical to the `GaussL0penObsScore-class`, except that it takes in a correlation matrix instead of the full data set. `GaussL0penObsScore-class`.

Usage

```
gausCorScore(cormat, n, p = NULL, lambda = NULL, ...)
```

Arguments

`cormat`	A correlation matrix. Needs to be symmetric.
`n`	The number of observations in the dataset that the correlation matrix was computed from.
`p`	The number of variables. This is inferred from the cormat if not supplied.
`lambda`	Penalty to use for the score. If `NULL` (default), the BIC score penalty is used. See `GaussL0penObsScore-class` for further details.
`...`	Other arguments passed along to `GaussL0penObsScore-class`.

Value

A `Score` object (S4), see `Score-class`.

Examples

```
# Simulate data and compute correlation matrix
x1 <- rnorm(100)
x2 <- rnorm(100)
x3 <- x1 + x2 + rnorm(100)
d <- data.frame(x1, x2, x3)
cmat <- cor(d)

# Use gausCorScore with pcalg::ges()
pcalg::ges(gausCorScore(cmat, n = 100))
```

graph2amat — *Convert graphNEL object to adjacency matrix*

Description

Convert graphNEL object to adjacency matrix

Usage

```
graph2amat(graph)
```

Arguments

graph	A graphNEL object.

is_cpdag — *Check for CPDAG*

Description

Check for CPDAG

Usage

```
is_cpdag(amat)
```

Arguments

amat	An adjacency matrix

Details

Check: Is adjacency matrix proper CPDAG? See `isValidGraph` for definition.

Value

A logical.

`is_pdag` *Check for PDAG*

Description

Check for PDAG

Usage

```
is_pdag(amat)
```

Arguments

amat	An adjacency matrix

Details

Check: Is adjacency matrix proper PDAG? See isValidGraph for definition.

Value

A logical.

`maketikz` *Generate Latex tikz code for plotting a temporal DAG or PDAG.*

Description

Generate Latex tikz code for plotting a temporal DAG or PDAG.

Usage

```
maketikz(
  model,
  xjit = 2,
  yjit = 2,
  markperiods = TRUE,
  xpgap = 4,
  annotateEdges = NULL,
  addAxis = TRUE,
  varLabels = NULL,
  periodLabels = NULL,
  annotationLabels = NULL,
  clipboard = TRUE,
  colorAnnotate = NULL
)
```

Arguments

- `model` — `tpdag` or `tamat` object to plot.
- `xjit` — How much should nodes within a period be jittered horizontally.
- `yjit` — Vertical distance between nodes within a period.
- `markperiods` — If `TRUE`, gray boxes are drawn behind each period.
- `xpgap` — Horizontal gap between different periods.
- `annotateEdges` — If `TRUE`, add a text annotation to edges. If `annotationlabels` are supplied, these labels will be used. Otherwise, the value in the inputted adjacency matrix corresponding to the edge will be used.
- `addAxis` — If `TRUE`, a horizontal axis with period labels are added.
- `varLabels` — Optional labels for nodes (variables). Should be given as a named list, where the name is the variable name, and the entry is the label, e.g. `list(vname = "Label for vname")`.
- `periodLabels` — Optional labels for periods. Should be given as a named list, where the name is the period name (as stored in the `tamat`), and the entry is the label, e.g. `list(periodname = "Label for period")`.
- `annotationLabels` — Optional labels for edge annotations. Only used if `annotateEdges = TRUE`. Should be given as a named list, where the name is the edge annotation (as stored in the `tamat`), and the entry is the label, e.g. `list(h = "High")`.
- `clipboard` — If `TRUE`, the tikz code is not printed, but instead copied to the clipboard, so it can easily be pasted into a Latex document.
- `colorAnnotate` — Named list of colors to use to mark edge annotations instead of labels. This overrules `annotateEdges` and both are not available at the same time. The list should be given with annotations as names and colors as entries, e.g. `list(h = "blue")`.

Details

Note that it is necessary to read in relevant tikz libraries in the Latex preamble. The relevant lines of code are (depending a bit on parameter settings):

```
\usepackage{tikz}
\usetikzlibrary{arrows,shapes,snakes,automata,backgrounds,petri}
\usepackage{pgfplots}
```

Value

Silently returns a character vector with lines of tikz code. The function furthermore has a side-effect. If `clipboard = TRUE`, the side-effect is that the tikz code is also copied to the clipboard. If `clipboard = FALSE`, the tikz code is instead printed in the console.

`maxnedges` *Compute maximal number of edges for graph*

Description

Computes the number of edges a graph with `p` nodes will have if its fully connected.

Usage

```
maxnedges(p)
```

Arguments

`p`	The number of nodes in the graph

Value

A numeric.

`nDAGs` *Number of different DAGs*

Description

Computes the number of different possible DAGs that can be constructed over a given number of nodes.

Usage

```
nDAGs(p)
```

Arguments

`p`	The number of nodes.

Value

A numeric.

nedges — *Number of edges in adjacency matrix*

Description

Counts the number of edges in an adjacency matrix.

Usage

```
nedges(amat)
```

Arguments

`amat`	An adjacency matrix

Value

A numeric (non-negative integer).

NPV — *Negative predictive value*

Description

Computes negative predictive value recall from a confusion matrix, see confusion. Negative predictive value is defined as TN/(TN + FN), where TN are true negatives and FN are false negatives. If TP + FN = 0, 0 is returned.

Usage

```
NPV(confusion)
```

Arguments

`confusion`	Confusion matrix as obtained from confusion

Value

A numeric in [0,1].

`plot.tamat` *Plot adjacency matrix with order information*

Description

Plot adjacency matrix with order information

Usage

```
## S3 method for class 'tamat'
plot(x, ...)
```

Arguments

`x`	tamat (temporal adjacency matrix) object to be plotted (as outputted from `tamat`).
`...`	Further plotting arguments passed along to `plotTempoMech`.

Value

No return value, the function is called for its side-effects (plotting).

`plot.tpdag` *Plot temporal partially directed acyclic graph (TPDAG)*

Description

Plot temporal partially directed acyclic graph (TPDAG)

Usage

```
## S3 method for class 'tpdag'
plot(x, ...)
```

Arguments

`x`	tpdag (temporal partially directed acyclic graph) object to be plotted (as outputted from `tpc`).
`...`	Further plotting arguments passed along to `plotTempoMech`.

Value

No return value, the function is called for its side-effects (plotting).

`plot.tskeleton` *Plot temporal skeleton*

Description

Plot temporal skeleton

Usage

```
## S3 method for class 'tskeleton'
plot(x, ...)
```

Arguments

`x`	tskeleton (temporal skeleton) object to be plotted (as outputted from `tpc`).
`...`	Further plotting arguments passed along to `plotTempoMech`.

Value

No return value, the function is called for its side-effects (plotting).

`plotTempoMech` *Plot temporal data generating mechanism*

Description

Plots tpdag, tskeleton and tamat objects.

Usage

```
plotTempoMech(
  x,
  addTimeAxis = TRUE,
  addPsi = TRUE,
  varLabels = NULL,
  periodLabels = NULL,
  colors = NULL,
  ...
)
```

Arguments

`x`	The tpdag/tskeleton or tamat to plot.
`addTimeAxis`	Logical indicating whether a time axis should be added to the plot.
`addPsi`	Logical indicating whether the sparsity level should be added to the plot.
`varLabels`	A named list of variable labels.
`periodLabels`	A character vector with labels for periods.
`colors`	A character vector with colors to use for marking periods. Should have at least as many elements as the numbers of periods.
`...`	Additional arguments passed to `plot.igraph`.

Value

No return value, the function is called for its side-effects (plotting).

`precision` *Precision*

Description

Computes precision (aka positive predictive value) from a confusion matrix, see confusion. Precision is defined as TP/(TP + FP), where TP are true positives and FP are false positives. If TP + FP = 0, 0 is returned.

Usage

```
precision(confusion)
```

Arguments

`confusion`	Confusion matrix as obtained from confusion

Value

A numeric in [0,1].

probmat2amat *Convert a matrix of probabilities into an adjacency matrix*

Description

Convert a matrix of probabilities into an adjacency matrix

Usage

```
probmat2amat(
  probmat,
  threshold,
  method = "cutoff",
  keep_vnames = TRUE,
  graph_criterion = "pdag",
  deletesym = FALSE
)
```

Arguments

`probmat` Square matrix of probabilities.

`threshold` Value between 0 and 1. Any probabilities lower than this value will be set to 0 (no arrowhead).

`method` Either `"cutoff"` or `"bpco"`, see details.

`keep_vnames` If `TRUE`, variable names (provided as rownames in the input probmat) will be preserved in the output.

`graph_criterion` Which criterion to check if the output graph fulfills for the bpco method. Should be one of `"dag"`, `"pdag"` or `"cpdag"` or `NULL`. Choosing `NULL` (the default) puts no further restrictions on the output. See `isValidGraph` for definitions.

`deletesym` If `TRUE`, edges are deleted symmetrically in the bcpo method. This means that instead of removing arrowheads (setting singular elements to 0), the procedure removes full edges (setting both potential arrowheads for the given edge to zero). This only makes a difference if the graph may include undirected edges, which should be encoded as bidirected edges.

Details

Two methods for converting the probability matrix into an adjacency matrix are implemented. First, the cutoff-method (`method = "cutoff"`) simply uses a threshold value and sets all values below that to zero in the outputted adjacency matrix. No checks are performed to ensure that the resulting matrix is a proper dag/pdag/cpdag adjacency matrix. Second, the backwards PC orientation method (`method = "bpco"`) first uses a cutoff, and then sets further elements to zero until the resulting matrix can be converted into a proper adjacency matrix (using the graph criterion specified in the `graph_criterion` argument) by applying the PC algorithm orientation rules. See Petersen et al. 2022 for further details.

Value

A square matrix of probabilities (all entries in [0,1]).

References

Petersen, Anne Helby, et al. "Causal discovery for observational sciences using supervised machine learning." arXiv preprint arXiv:2202.12813 (2022).

Examples

```
#Make random probability matrix that can be
#converted into adjancency matrix
pmat <- matrix(runif(25, 0, 1), 5, 5)
diag(pmat) <- 0

#Convert to adjacency matrix using cutoff-method (threshold = 0.5)
probmat2amat(pmat, threshold = 0.5)

#Convert to adjacency matrix using BPCO-method (threshold = 0.5)
probmat2amat(pmat, threshold = 0.5, method = "bpco")
```

`recall` *Recall*

Description

Computes recall from a confusion matrix, see confusion. Recall is defined as TP/(TP + FN), where TP are true positives and FN are false negatives. If TP + FN = 0, 0 is returned.

Usage

```
recall(confusion)
```

Arguments

`confusion`	Confusion matrix as obtained from confusion

Value

A numeric in [0,1].

regTest — *Regression-based information loss test*

Description

We test whether `x` and `y` are associated, given `S` using a generalized linear model.

Usage

```
regTest(x, y, S, suffStat)
```

Arguments

`x`	Index of x variable
`y`	Index of y variable
`S`	Index of S variable(s), possibly NULL
`suffStat`	Sufficient statistic; list with data, binary variables and order.

Details

All included variables should be either numeric or binary. If `y` is binary, a logistic regression model is fitted. If `y` is numeric, a linear regression model is fitted. `x` and `S` are included as explanatory variables. Any numeric variables among `x` and `S` are modeled with spline expansions (natural splines, 3 df). This model is tested against a numeric where `x` (including a possible spline expansion) has been left out using a likelihood ratio test. The model is fitted in both directions (interchanging the roles of `x` and `y`). The final p-value is the maximum of the two obtained p-values.

Value

A numeric, which is the p-value of the test.

shd — *Structural hamming distance between adjacency matrices*

Description

Computes the structural hamming distance between two adjacency matrices. This implementation is a modification of the `shd` function from the pcalg package, but here we avoid working on the heavy `graphNEL` objects for representing graphs that are used in the pcalg package.

Usage

```
shd(est_amat, true_amat)
```

Arguments

`est_amat`	Estimated adjacency matrix
`true_amat`	True adjacency matrix

Details

Note that the function is symmetric in the two inputted adjacency matrices.

Value

A numeric (a non-negative integer).

`simDAG` *Simulate a random DAG*

Description

Simulates a random directed acyclic graph adjacency (DAG) matrix with the provided edge sparsity. The edge sparsity is the percentage of edges that are absent, relative to a fully connected DAG.

Usage

```
simDAG(p, sparsity = NULL, sparsityLim = c(0, 0.8), permute = TRUE)
```

Arguments

`p`	The number of nodes.
`sparsity`	If `NULL` (the default), a random edge sparsity is sampled from the interval provided in `sparsityLim`. Otherwise, the sparsity should be provided as a numeric in [0,1].
`sparsityLim`	A vector of two numerics, both must be in [0,1].
`permute`	If `FALSE`, the adjacency matrix will include nodes in their causal ordering. This is avoided by setting `permute = TRUE`, in which case the node order is permuted randomly.

Value

An adjacency matrix.

Examples

```
# Simulate a DAG adjacency matrix with 5 nodes
simDAG(5)
```

simGausFromDAG *Simulate Gaussian data according to DAG*

Description

Simulates a jointly Gaussian dataset given a DAG adjacency matrix. The data is simulated using linear structural equations and the parameters (residual standard deviations and regression coefficients) are sampled from chosen intervals.

Usage

```
simGausFromDAG(
  amat,
  n,
  regparLim = c(0.5, 2),
  resSDLim = c(0.1, 1),
  pnegRegpar = 0.4,
  standardize = FALSE
)
```

Arguments

amat	An adjacency matrix.
n	The number of observations that should be simulated.
regparLim	The interval from which regression parameters are sampled.
resSDLim	The interval from which residual standard deviations are sampled.
pnegRegpar	The probability of sampling a negative regression parameter.
standardize	If FALSE (the default), the raw data are returned. If TRUE, the data are first standardized, i.e., each variable will have its mean subtracted and be divided by its standard deviation.

Details

A variable X_i is simulated as
$X_i := \sum_{Z \in pa(X_i)} \beta_Z * Z + e_i$
where $pa(X_i)$ are the parents of X_i in the DAG. The residual, e_i, is drawn from a normal distribution.

Value

A data.frame of identically distributed simulated observations.

`specificity` *Specificity*

Description

Computes specificity from a confusion matrix, see confusion. Specificity is defined as TN/(TN + FP), where TN are true negatives and FP are false positives. If TN + FP = 0, 0 is returned.

Usage

```
specificity(confusion)
```

Arguments

`confusion`	Confusion matrix as obtained from confusion

Value

A numeric in [0,1].

`tamat` *Make a temporal adjacency matrix*

Description

Make a temporal adjacency matrix

Usage

```
tamat(amat, order)
```

Arguments

`amat`	Adjacency matrix. A square matrix of 0 or 1s. A 1 in the (i,j)th entry means that there is an edge from j to i. Row names and column names should be identical and be the names of the variables/nodes. Variable names should be prefixed with their period, e.g. "child_x" for variable "x" at period "child"
`order`	A character vector with the periods in their order.

Value

A `tamat` object, which is a matrix with a "order" attribute(a character vector listing the temporal order of the variables in the adjacency matrix).

tpc *Perform causal discovery using the temporal PC algorithm (TPC)*

Description

Perform causal discovery using the temporal PC algorithm (TPC)

Usage

```
tpc(
  data,
  order,
  sparsity = 10^(-1),
  test = regTest,
  suffStat = NULL,
  output = "tpdag",
  ...
)
```

Arguments

`data`	A data.frame with data. All variables should be assigned to exactly one period by prefixing them with the period name (see example below).
`order`	A character vector with period-prefixes in their temporal order (see example below).
`sparsity`	The sparsity level to be used for independence testing (i.e. significance level threshold to use for each test).
`test`	A procedure for testing conditional independence. The default, `regTest` uses a regression-based information loss test. Another available option is `corTest` which tests for vanishing partial correlations. User supplied functions may also be used, see details below about the required syntax.
`suffStat`	Sufficient statistic. If this argument is supplied, the sufficient statistic is not computed from the inputted data. The format and contents of the sufficient statistic depends on which test is being used.
`output`	One of `"tpdag"` or `"tskeleton"`. If `"skeleton"`, a temporal skeleton is constructed and outputted, but the edges are not directed. If `"tpdag"` (the default), a the edges are directed, resulting in a temporal partially directed acyclic graph.
`...`	Further optional arguments which are currently not in use.

Details

Note that all independence test procedures implemented in the `pcalg` package may be used, see `pc`.

Value

A tpdag or tskeleton object. Both return types are S3 objects, i.e., lists with entries: $amat (the estimated adjacency matrix), $order (character vector with the order, as inputted to this function), $psi (the significance level used for testing), and $ntests (the number of tests conducted).

Examples

```
#TPC on included example data, use sparsity psi = 0.01, default test (regression-based
#information loss):
data(tpcExample)
tpc(tpcExample, order = c("child", "youth", "oldage"), sparsity = 0.01)


#TPC on included example data, use sparsity psi = 0.01, use test for vanishing partial
# correlations:
data(tpcExample)
tpc(tpcExample, order = c("child", "youth", "oldage"), sparsity = 0.01,
test = corTest)


#TPC on another simulated data set

#Simulate data
set.seed(123)
n <- 500
child_x <- rnorm(n)^2
child_y <- 0.5*child_x + rnorm(n)
child_z <- sample(c(0,1), n, replace = TRUE,
                  prob = c(0.3, 0.7))

adult_x <- child_x + rnorm(n)
adult_z <- as.numeric(child_z + rnorm(n) > 0)
adult_w <- 2*adult_z + rnorm(n)
adult_y <- 2*sqrt(child_x) + adult_w^2 + rnorm(n)

simdata <- data.frame(child_x, child_y, child_z,
                      adult_x, adult_z, adult_w,
                      adult_y)

#Define order
simorder <- c("child", "adult")

#Perform TPC with sparsity psi = 0.001
results <- tpc(simdata, order = simorder, sparsity = 10^(-3))
```

tpcExample *Simulated data example*

Description

A small simulated data example intended to showcase the TPC algorithm. Note that the variable name prefixes defines with period they are related to ("child", "youth" or "oldage").

Usage

```
tpcExample
```

Format

A data.frame with 200 rows and 6 variables.

child_x1 Structural equation: $X1 := \epsilon 1$ with $\epsilon 1$ $Unif 0, 1$

child_x2 Structural equation: $X2 := 2 * X1 + \epsilon 2$ with $\epsilon 2$ $N(0, 1)$

youth_x3 Structural equation: $X3 := \epsilon 3$ with $\epsilon 3$ $Unif 0, 1$

youth_x4 Structural equation: $X4 := X2 + \epsilon 4$ with $\epsilon 4$ $N(0, 1)$

oldage_x5 Structural equation: $X5 := X3^2 + X3 - 3 * X2 + \epsilon 5$ with $\epsilon 5$ $N(0, 1)$

oldage_x6 Structural equation: $X6 := X4^3 + X4^2 + 2 * X5 + \epsilon 6$ with $\epsilon 6$ $N(0, 1)$

References

Petersen, AH; Osler, M \& Ekstrøm, CT (2021): Data-Driven Model Building for Life-Course Epidemiology, American Journal of Epidemiology.

Examples

```
data(tpcExample)
```

Index