Package 'SID'

January 20, 2025

Type Package

Title Structural Intervention Distance

Version 1.1

Date 2023-12-03

Encoding UTF-8

Imports pcalg, igraph, RBGL, Matrix, methods

Description The code computes the structural intervention distance (SID) between a true directed acyclic graph (DAG) and an estimated DAG. Definition and details about the implementation can be found in J. Peters and P. Bühlmann: ``Structural intervention distance (SID) for evaluating causal graphs", Neural Computation 27, pages 771-799, 2015 <doi:10.1162/NECO_a_00708>.

URL https://github.com/fkgruber/SID_cran

License FreeBSD

Repository CRAN

NeedsCompilation no

Author Fred Gruber [cre], Jonas Peters [aut]

Maintainer Fred Gruber <fgruber@gmail.com>

Date/Publication 2023-12-06 10:10:02 UTC

Contents

SID-package 2
computePathMatrix 2
computePathMatrix2 3
hammingDist 4
randomDAG 5
structIntervDist 6

Index **9**

SID-package *Structural Intervention Distance (SID) (package)*

Description

The package allows to compute the structural intervention distance (SID) between two graphs. This (pre-)metric evaluates graphs from a causal inference point of view. The package also contains a function for computing the Hamming distance and to generate random DAGs.

Details

Package:	SID
Type:	Package
Version:	1.0
Date:	2015-03-07
License:	FreeBSD

Computing the Structural Intervention Distance. The algorithm can be called with `structIntervDist`.

Author(s)

Jonas Peters <jonas.peters@tuebingen.mpg.de>

References

J. Peters, P. B\"uhlmann: Structural intervention distance (SID) for evaluating causal graphs, Neural Computation 27, pages 771-799, 2015

See Also

`randomDAG` `structIntervDist` `hammingDist`

---

computePathMatrix *auxiliary file for SID: computes a path matrix efficiently (can probably be made faster)*

Description

auxiliary file for SID: computes a path matrix efficiently (can probably be made faster) This function takes an adjacency matrix G from a DAG and computes a path matrix for which entry(i,j) being one means that there is a directed path from i to j the diagonal will also be one.

Usage

```
computePathMatrix(G, spars=FALSE)
```

Arguments

G	graph.
spars	boolean indicating whether G is a sparse matrix.

Value

pathMatrix

Author(s)

Jonas Peters <jonas.peters@tuebingen.mpg.de>

See Also

structIntervDist computePathMatrix2

---

computePathMatrix2 *auxiliary file for SID: computes a path matrix efficiently (can probably be made faster)*

Description

auxiliary file for SID: computes a path matrix efficiently (can probably be made faster) The only difference to the function computePathMatrix is that this function changes the graph by removing all edges that leave condSet. If condSet is empty, it just returns PathMatrix1.

Usage

```
computePathMatrix2(G,condSet,PathMatrix1, spars=FALSE)
```

Arguments

G	graph.
condSet	set of variables
PathMatrix1	PathMatrix1
spars	boolean indicating whether the matrices are sparse

Value

pathMatrix

Author(s)

Jonas Peters <jonas.peters@tuebingen.mpg.de>

See Also

structIntervDist computePathMatrix

`hammingDist` *Hamming Distance*

Description

Computes the Hamming distance between two graph objects.

Usage

```
hammingDist(G1, G2, allMistakesOne = TRUE)
```

Arguments

`G1`	p x p adjacency matrix of the first graph.
`G2`	p x p adjacency matrix of the second graph.
`allMistakesOne`	boolean variable (TRUE/FALSE) that indicates whether all mistakes should be counted as one. E.g., if it is set to FALSE the Hamming distance between X -> Y and X <- Y is two and one if it set to TRUE.

Details

The Hamming distance between two graphs counts the number of edges, in which the graphs do not coincide. allMistakesOne determines whether a reversed edge counts as two or as one mistake. The Hamming distance is symmetric, that is hammingDist(G1,G2) = hammingDist(G2,G1).

Value

The method outputs the computed Hamming distance.

Author(s)

Jonas Peters <jonas.peters@tuebingen.mpg.de>

See Also

`structIntervDist`

randomDAG — *outputs the adjacency matrix of a randomly generated directed acyclic graph (DAG).*

Description

After simulating a random topological order first (the order can also be prespecified), the algorithm connects two nodes with the porbability probConnect.

Usage

```
randomDAG(p, probConnect, causalOrder = sample(p, p, replace = FALSE))
```

Arguments

p	number of nodes.
probConnect	probability of connecting two nodes, determines the sparsity of the graph. Choosing probConnect = 2/(p-1), for example, leads to an expected number of p nodes.
causalOrder	OPTIONAL: causal or topological order of the nodes. If not provided, the topological order is chosen randomly.

Value

p x p adjacency matrix that describes a directed acyclic graph (DAG) with p nodes. The entry (i,j) equals one if and only if there is an edge from i to j.

Author(s)

Jonas Peters <jonas.peters@tuebingen.mpg.de>

See Also

structIntervDist hammingDist

Examples

```
randomDAG(p = 5, probConnect = 0.6)
```

`structIntervDist` *Structural Intervention Distance (SID)*

Description

computes the structural intervention distance (SID) between two graphs; it evaluates graphs from a causal inference point of view.

Usage

```
structIntervDist(trueGraph, estGraph, output = FALSE, spars = FALSE)
```

Arguments

`trueGraph`	p x p adjacecncy matrix; it must be a directed acyclic graph (DAG)
`estGraph`	again a p x p adjacecncy matrix; it can be either a DAG or a completed partially directed graph (CPDAG) that represents a Markov equivalence class.
`output`	boolean (TRUE/FALSE) that indicates whether output shall be shown
`spars`	boolean (TRUE/FALSE) that indicates whether the input matrices trueGraph and estGraph are sparse. If matrices are indeed large and sparse this increases the speed of the algorithm.

Details

The structural intervention distance (SID) considers each pair (i,j) and checks whether the parents of i in estGraph is a valid adjustment set in trueGraph for the causal effect from i to j. If it is, estimating the causal effect from i to j using parent adjustment in estGraph is a correct procedure, independently of the distribution. If it is not, the pair (i,j) contributes a mistake of one to the overall structural intervention distance.

The SID satisfies the properties of a pre-metric; we have SID(G,G) = 0 and 0 <= SID(G1,G2) <= p(p-1), where p is the number of nodes. The SID is not symmetric: in general, we have SID(G1, G2) != SID(G2,G1). Furthermore, the SID can be zero although the two graphs are not the same.

If estGraph is a completed partially directed acyclic graph (CPDAG), that is it describes a Markov equivalence class, the SID outputs a lower and an upper bound of the SID, that correspond to the best and worst DAG in the equivalence class, respectively.

Value

The main function is structIntervDist. It takes two DAGs (adjacency matrices) of the same dimension as input and provides a list with

`sid`	the SID
`sidUpperBound`	relevant if estGraph is a CPDAG: it is the highest SID that can be achieved by a graph within the Markov equivalence class ("worst DAG")
`sidLowerBound`	relevant if estGraph is a CPDAG: it is the lowest SID that can be achieved by a graph within the Markov equivalence class ("best DAG")
`incorrectMat`	the matrix of incorrect interventional distributions; its sum equals $sid

Author(s)

Jonas Peters <jonas.peters@tuebingen.mpg.de>

References

J. Peters, P. B\"uhlmann: Structural intervention distance (SID) for evaluating causal graphs, Neural Computation 27, pages 771-799, 2015

See Also

hammingDist randomDAG

Examples

```
G <- rbind(c(0,1,1,1,1),
           c(0,0,1,1,1),
           c(0,0,0,0,0),
           c(0,0,0,0,0),
           c(0,0,0,0,0))

H1 <- rbind(c(0,1,1,1,1),
            c(0,0,1,1,1),
            c(0,0,0,1,0),
            c(0,0,0,0,0),
            c(0,0,0,0,0))

H2 <- rbind(c(0,0,1,1,1),
            c(1,0,1,1,1),
            c(0,0,0,0,0),
            c(0,0,0,0,0),
            c(0,0,0,0,0))

cat("true DAG G:\n")
show(G)
cat("\n")
cat("==============","\n")
cat("\n")
cat("estimated DAG H1:\n")
show(H1)
sid <- structIntervDist(G,H1)
shd <- hammingDist(G,H1)
cat("SHD between G and H1: ",shd,"\n")
cat("SID between G and H1: ",sid$sid,"\n")
#cat("The matrix of incorrect interventional distributions is: ","\n")
#show(sid$incorrectMat)
cat("\n")
cat("==============","\n")
cat("\n")
cat("estimated DAG H2:\n")
show(H2)
sid <- structIntervDist(G,H2)
shd <- hammingDist(G,H2)
```

```
cat("SHD between G and H2: ",shd,"\n")
cat("SID between G and H2: ",sid$sid,"\n")
cat("The matrix of incorrect interventional distributions is: ","\n")
show(sid$incorrectMat)

readline("The SID can also be applied to CPDAGs. Please press enter...")
cat("\n")
cat("==============","\n")
cat("\n")
cat("estimated CPDAG H1c:\n")
H1c <- rbind(c(0,1,1,1,1),c(1,0,1,1,1),c(1,1,0,1,0),c(1,1,1,0,0),c(1,1,0,0,0))
show(H1c)
sid <- structIntervDist(G,H1c)
cat("SID between G and H1: \n")
cat("lower bound: ",sid$sidLowerBound," upper bound: ", sid$sidUpperBound, "\n")
cat("\n")
```

Index

∗ **Adjacency graphs**
structIntervDist, 6
∗ **Causality**
structIntervDist, 6
∗ **Hamming distance**
hammingDist, 4
∗ **causal (topological) order**
randomDAG, 5
∗ **randomDAG**
randomDAG, 5

computePathMatrix, 2, *3*
computePathMatrix2, *3*, 3

hammingDist, *2*, 4, *5*, *7*

randomDAG, *2*, 5, *7*

SID-package, 2
structIntervDist, *2–5*, 6