Package ‘OrdCD’

May 7, 2026

Type Package

Title Ordinal Causal Discovery

Version 1.1.2

Date 2023-05-14

Description Algorithms for ordinal causal discovery. This package aims to enable users to discover causality for observational ordinal categorical data with greedy and exhaustive search. See Ni, Y., & Mallick, B. (2022) <https://proceedings.mlr.press/v180/ni22a/ni22a.pdf> ``Ordinal Causal Discovery. Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence, (UAI 2022), PMLR 180:1530–1540".

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports gRbase, MASS, bnlearn, igraph, stats, Matrix

URL https://github.com/nySTAT/OrdCD

BugReports https://github.com/nySTAT/OrdCD/issues

NeedsCompilation no

Author Yang Ni [aut, cre] (ORCID: <https://orcid.org/0000-0003-0636-2363>)

Maintainer Yang Ni <yni@stat.tamu.edu>

Repository CRAN

Date/Publication 2023-05-17 21:40:11 UTC

Contents

OrdCD 2

Index **4**

OrdCD *Causal Discovery for Ordinal Categorical Data*

Description

Estimate a causal directed acyclic graph (DAG) for ordinal categorical data with greedy or exhaustive search.

Usage

```
OrdCD(
  y,
  search = "greedy",
  ic = "bic",
  edge_list = NULL,
  link = "probit",
  G = NULL,
  nstart = 1,
  verbose = FALSE,
  maxit = 50,
  boot = NULL
)
```

Arguments

y	a data frame with each column being an ordinal categorical variable, which must be a factor.
search	the search method used to find the best-scored DAG. The default search method is "greedy". When the number of nodes is less than 4, "exhaust" search is available.
ic	the information criterion (AIC or BIC) used to score DAGs. The default is "bic".
edge_list	an edge list of a CPDAG, which may contain both directed and undirected edges. This option can significantly speed up the algorithm by restricting the search space to DAGs that are Markov equivalent to the input CPDAG. Such CPDAG may be obtained by e.g., the PC algorithm; see the example below.
link	the link function for ordinal regression. The default is "probit". Other choices are "logistic", "loglog", "cloglog", and "cauchit".
G	a list of DAG adjacency matrices that users want to restrict their search on for the "exhaust" search. The default is "NULL" meaning no restriction imposed on the search.
nstart	number of random graph initializations for the "greedy" search.
verbose	if TRUE, messages are printed during the run of the greedy search algorithm.
maxit	the maximum number of iterations for the greedy search algorithm. The default is 50. When the maximum number of iteration is achieved, a warning message will be generated to caution the user that the algorithm has not converged.
boot	the number of bootstrap samples. Default is no bootstrapping.

Value

A list with "boot" elements. Each element is a list with two elements, gam and ic_best. gam is an estimated DAG adjacency matrix whose (i,j)th entry is 1 if j->i is present in the graph and 0 otherwise. ic_best is the corresponding information criterion value.

Examples

```
set.seed(2020)
n=1000 #sample size
q=3 #number of nodes
y = u = matrix(0,n,q)
u[,1] = 4*rnorm(n)
y[,1] = (u[,1]>1) + (u[,1]>2)
for (j in 2:q){
  u[,j] = 2*y[,j-1] + rnorm(n)
  y[,j]=(u[,j]>1) + (u[,j]>2)
}
A=matrix(0,q,q) #true DAG adjacency matrix
A[2,1]=A[3,2]=1
y=as.data.frame(y)
for (j in 1:q){
  y[,j]=as.factor(y[,j])
}

time=proc.time()
G=OrdCD(y) #estimated DAG adjacency matrix
time=proc.time() - time
print(A) #display the true adjacency
print(G) #display the estimated adjacency
print(time[3]) #elapsed time

time2=proc.time()
colnames(y)=1:ncol(y)
PC=bnlearn::pc.stable(y,test="mi-sh",alpha=0.01)
edge_list=matrix(as.numeric(PC$arcs),ncol=2)
G2=OrdCD(y, edge_list=edge_list) #estimated DAG with an input CPDAG from PC algorithm
time2=proc.time()-time2
print(G2) #display the estimated adjacency
print(time2[3]) #elapsed time

## Not run:
time3=proc.time()
G3=OrdCD(y,boot=10) #estimated DAG adjacency matrix with bootstrapping
time3=proc.time() - time3
#average over bootstrap samples
G_avg = matrix(colMeans(matrix(unlist(G3),nrow=q^2+1,byrow=TRUE))[1:(q^2)],q,q)
print(G_avg) #display the estimated adjacency averaged over 10 bootstrap samples
print(time3[3]) #elapsed time

## End(Not run)
```

Index

OrdCD, 2