Package ‘netchain’

October 13, 2022

Type Package
Title Inferring Causal Effects on Collective Outcomes under Interference
Version 0.2.0
Date 2020-02-15
Maintainer Youjin Lee <youjin.lee@pennmedicine.upenn.edu>
Description In networks, treatments may spill over from the treated individual to his or her social contacts and outcomes may be contagious over time. Under this setting, causal inference on the collective outcome observed over all network is often of interest. We use chain graph models approximating the projection of the full longitudinal data onto the observed data to identify the causal effect of the intervention on the whole outcome. Justification of such approximation is demonstrated in Ogburn et al. (2018) <arXiv:1812.04990>.
License GPL (>= 3) | file LICENSE
Imports Rcpp (>= 0.12.17), Matrix, gtools, stringr, stats, igraph
Suggests knitr, rmarkdown, testthat, R.rsp
LinkingTo Rcpp
VignetteBuilder R.rsp
RoxygenNote 7.0.0
Encoding UTF-8
NeedsCompilation yes
Author Elizabeth Ogburn [aut], Ilya Shpitser [aut], Youjin Lee [aut, cre]
Repository CRAN
Date/Publication 2020-02-16 22:10:06 UTC

R topics documented:

netchain-package .................................................. 2
causal.influence .................................................. 2
This package is for estimation of probability associated with collective counterfactual outcomes using approximation via causal graphical model. We apply a parsimonious parameterization for social network data with some specific kinds of interference and contagion, which corresponds to particular family of graphical models known as chain graphs.

Details

We provide functions to estimate the parameters in conditional log-linear model when the observations (outcomes, treatments, and confounders) and the structure of a causal graph is given. Based on the estimated parameters, we generate counterfactual outcomes using Gibbs sampling to infer the causal effect (or causal probability) of a certain treatment assignment on the collective outcomes. Moreover, we use this method to identify causally influential units on social network.

Author(s)

Youjin Lee
Maintainer: Youjin Lee <ylee160@jhu.edu>

See Also

https://github.com/youjin1207/netchain

This function calculates probability associated with counterfactual collective outcome(s) \( P(Y(a_j) = y) \) as a measure of influence of unit \( j \), where \( a_j \) indicates the sole intervention of unit \( j \).
Usage

causal.influence(
  targetoutcome = "mean",
  Avalues,
  inputY,
  inputA,
  listC,
  R.matrix,
  E.matrix,
  edgeinfo = NULL,
  n.obs = 1000,
  n.burn = 100,
  optim.method = "L-BFGS-B"
)

Arguments

- **targetoutcome** is a targeted counterfactual outcome of probability is in our interest, having different forms depending on the context of influence:
  - **a vector of length** $m$ a vector specifies every element of $y$.
  - **a $[q \times m]$ matrix** a collection of $y_1, y_2, ..., y_q$ of which we want to derive the probability.
  - **an integer** the number of 1’s in $y$ ($0 \leq \leq m$).
  - **‘mean’** when we want derive $E(Y(a))$ (default).

- **Avalues** distinct treatment values of which maximum indicates intervention. Defaults to $(0,1)$.

- **inputY** a $[n \times m]$ matrix of $n$ independent outcomes for $m$ units.

- **inputA** a $[n \times m]$ matrix of $n$ independent treatment assignments assigned to $m$ units.

- **listC** is either a matrix, list or NULL:
  - **a $[n \times m]$ matrix** a matrix of $n$ independent confounders for $m$ units under single confounder.
  - **a list of $[n \times m]$ matrices** a collection of $n$ independent confounders for $m$ units under multiple confounders.
  - **NULL** no confounders.

- **R.matrix** a $[m \times m]$ relational symmetric matrix where $R_{matrix, i, j} = 1$ indicates $Y_i$ and $Y_j$ are adjacent.

- **E.matrix** a $[m \times m]$ matrix where $E_{matrix, i, j} = 1$ indicates $A_i$ has a direct causal effect on $Y_j$. Defaults to diagonal matrix, which indicates no interference.

- **edgeinfo** a list of matrix specifying additional directed edges (from confounders or treatment to the outcomes) information. Defaults to NULL.

  - **first column**: "Y" indicates outcomes; "A" indicates treatment; "C" indicates confounders. Under multiple confounders, "C1", "C2", ... indicate each confounder.

  - **second column**: an index for unit corresponding to the variable in the first column, $i=1,2,...,m$. 
chain.causal.multi

Causal estimation on collective outcomes under multiple confounders and interference.

Description

This function calculates probability associated with counterfactual collective outcome(s) \( P(Y(a) = y) \) when \( m \) units are subject to interference and contagion possibly with the presence of multiple confounders. To estimate the magnitude of main effects, two-way interaction effects, or any higher-order interaction effects we use hybrid graphical models combining features of both log-linear models on undirected graphs (\( R.matrix \)) and directed acyclic graphs (DAGs) models used to represent casual relationships.
Usage

chain.causal.multi(
  targetoutcome = "mean",
  treatment,
  inputY,
  inputA,
  listC,
  R.matrix,
  E.matrix,
  edgeinfo = NULL,
  n.obs = 1000,
  n.burn = 100,
  optim.method = "L-BFGS-B"
)

Arguments

targetoutcome is a targeted counterfactual outcome of probability is in our interest, having different forms:
  a vector of length \( m \) a vector specifies every element of \( y \).
  a \([q x m]\) matrix a collection of \( y_1, y_2, ..., y_q \) of which we want to derive the probability.
  an integer the number of 1’s in \( y \) (0 &le; \( m \)).
  ‘mean’ when we want derive \( E(Y(a)) \) (default).

treatment a vector of length \( m \) representing given treatment assignment \( a \).

inputY a \([n x m]\) matrix of \( n \) independent outcomes for \( m \) units.

inputA a \([n x m]\) matrix of \( n \) independent treatment assignments assigned to \( m \) units.

listC is either a matrix, list or NULL:
  a \([n x m]\) matrix a matrix of \( n \) independent confounders for \( m \) units under single confounder.
  a list of \([n x m]\) matrices a collection of \( n \) independent confounders for \( m \) units under multiple confounders.
  NULL no confounders.

R.matrix a \([m x m]\) relational symmetric matrix where \( R.matrix_{i,j} = 1 \) indicates \( Y_i \) and \( Y_j \) are adjacent.

E.matrix a \([m x m]\) matrix where \( E.matrix_{i,j} = 1 \) indicates \( A_i \) has a direct causal effect on \( Y_j \). Defaults to diagonal matrix, which indicates no interference.

edgeinfo a list of matrix specifying additional directed edges (from confounders or treatment to the outcomes) information. Defaults to NULL.

first column: "Y" indicates outcomes; "A" indicates treatment; "C" indicates confounders. Under multiple confounders, "C1", "C2", ... indicate each confounder.

second column: an index for unit corresponding to the variable in the first column, \( i=1,2,..m \).
n.obs the number of Gibbs samplers except for burn-in sample.
n.burn the number of burn-in sample in Gibbs sampling.
optim.method the method used in optim(). Defaults to "L-BFGS-B".

Value

returns "noconvergence" in case of failure to convergence or a list with components:

causalprob the estimated probability P(Y(a) = y).
n.par the number of parameters estimated in conditional log-linear model.
par.est the estimated parameters.

Author(s)

Youjin Lee

Examples

library(netchain)
set.seed(1234)
weight.matrix <- matrix(c(0.5, 1, 0, 1, 0.3, 0.5, 0, 0.3, -0.5), 3, 3)
simobs <- simGibbs(n.unit = 3, n.gibbs = 100, n.sample = 5, weight.matrix, treat.matrix = 0.5*diag(3), cov.matrix= (-0.3)*diag(3) )
inputY <- simobs$inputY
inputA <- simobs$inputA
inputC <- simobs$inputC
R.matrix <- ifelse(weight.matrix==0, 0, 1)
diag(R.matrix) <- 0
edgeinfo <- list(rbind(c("Y", 1), c("C", 1)), rbind(c("Y", 2), c("C", 2)),
                 rbind(c("Y", 3), c("C", 3)))
# implement a function (take > 10 seconds)
# result <- chain.causal.multi(targetoutcome = "mean",
# treatment <- c(1,0,0), inputY, inputA, listC = inputC, R.matrix, 
# E.matrix <- diag(3), edgeinfo = edgeinfo)

chaingibbs Generate Gibbs samplers for counterfactual collective outcomes.

Description

This function generates the outcomes using Gibbs sampling under the given treatment assignment and edge information.
Usage

```r
cachingibbs(
pars,  
n.obs,  
treatment,  
covariates,  
initprob = 0.5,  
yvalues = c(0, 1),  
Neighborind,  
Neighborpar,  
n.burn
)
```

Arguments

- `pars`: a set of parameters
- `n.obs`: the number of Gibbs samples.
- `treatment`: a set of given treatment assignment of length \( m \).
- `covariates`: given confounder(s):
  - NULL: no confounder.
  - a vector of length \( m \): under unique confounder.
  - a \([q \times m]\) matrix: a set of \( q \) different confounders.
- `initprob`: an initial probability generating outcomes. Defaults to `initprob = 0.5`
- `yvalues`: distinct binary values for outcomes. Defaults to `(0, 1)`.
- `Neighborind`: a list of matrix specifying edge information of which first column illustrates a type of variables (1:outcome, 2:treatment, 3~:confounders) and of which second column presents the index of those variables.
- `Neighborpar`: index for parameters in the order of Neighborind.
- `n.burn`: the number of burn-in sample in Gibbs sampling (\( \geq n.\text{obs} \)).

Value

- a \([n.\text{obs} \times m]\) matrix each row of which consists of outcomes.

---

`multiloglikechain`  
*Derive log-likelihood of conditional log-linear model given parameters.*

Description

Derive log-likelihood of conditional log-linear model given parameters.

Usage

```r
multiloglikechain(pars, listobservations, permutetab, edgeY, edgeAY, edgeExtra)
```
Arguments

pars \(\) a set of parameters

listobservations \(\) a collection of \([(2+\text{nc}) \times m]\) matrices comprised of outcomes (first row), treatments (second row), and confounders (from the third row), where \(\text{nc}\) is the number of confounders.

permutetab \(\) a matrix comprised of every possible values for outcome in each row.

dgeY \(\) a matrix of which each row indicates a pair of index for adjacent outcomes.

dgeAY \(\) a matrix of which each row indicates an index for treatment (first column) and for outcome (second column) on which the treatment has a direct effect.

dgeExtra \(\) a list of edges of which a list of matrix specifying additional directed edges (from confounders or treatment to the outcomes) information.

Value

log-likelihood of conditional log-linear model given parameters, observations, and edge information.

multimainfunction \(\)

Extracting factors for conditional log-linear model

Description

This is an auxiliary function to print out the factors for conditional log-linear model given edge information.

Usage

multimainfunction(pars, newcombined, dgeY, dgeAY, dgeExtra)

Arguments

pars \(\) a set of parameters

newcombined \(\) a \([(2+\text{nc}) \times m]\) matrix comprised of outcomes (first row), treatments (second row), and confounders (from the third row), where \(\text{nc}\) is the number of confounders.

dgeY \(\) a matrix of which each row indicates a pair of index for adjacent outcomes.

dgeAY \(\) a matrix of which each row indicates an index for treatment (first column) and for outcome (second column) on which the treatment has a direct effect.

dgeExtra \(\) a list of edges of which a list of matrix specifying additional directed edges (from confounders or treatment to the outcomes) information.

Value

a sum of factors.
### multipartition

**Calculating normalizing constant in conditional log-linear model.**

**Description**

Calculating normalizing constant in conditional log-linear model.

**Usage**

```
multipartition(pars, combined, permutetab, edgeY, edgeAY, edgeExtra)
```

**Arguments**

- **pars**: a set of parameters
- **combined**: a \([(2+nc) \times m]\) matrix comprised of outcomes (first row), treatments (second row), and confounders (from the third row), where \(nc\) is the number of confounders.
- **permutetab**: a matrix comprised of every possible values for outcome in each row.
- **edgeY**: a matrix of which each row indicates a pair of index for adjacent outcomes.
- **edgeAY**: a matrix of which each row indicates an index for treatment (first column) and for outcome (second column) on which the treatment has a direct effect.
- **edgeExtra**: a list of edges of which a list of matrix specifying additional directed edges (from confounders or treatment to the outcomes) information.

**Value**

A normalizing constant

### simGibbs

**Generate binary (Y, A, C) from chain graph model under simplest scenario.**

**Description**

Generate binary (Y, A, C) from chain graph model under simplest scenario.

**Usage**

```
simGibbs(
    n.unit,
    n.gibbs,
    n.sample,
    weight.matrix,
    treat.matrix,
);```
cov.matrix,
init.prob = 0.5,
treat.prob = 0.5,
cov.prob = 0.5,
n.burn = 100,
yvalues = c(1, 0)
)

Arguments

n.unit the number of units (m).
n.gibbs the number of independent Gibbs sampler.
n.sample the number of samples from each Gibbs sampling (n = n.gibbs x n.sample).
weight.matrix a $[m \times m]$ symmetric relational matrix where $W_{ij} = 1$ indicates the existence of undirected edges between $Y_i$ and $Y_j$ and its magnitude. Here $W_{ii}$ represents the main effect of $Y_i$.
treat.matrix a $[m \times m]$ matrix where $treat.matrix_{ij}$ indicates the magnitude of direct effect from $A_i$ to $Y_j$.
cov.matrix a $[m \times m]$ matrix where $treat.matrix_{ij}$ indicates the magnitude of direct effect from $C_i$ to $Y_j$.
init.prob an initial probability generating ($Y, A, C$) from Bernoulli distribution.
treat.prob a probability updating $A$ in Gibbs sampling procedure.
cov.prob a probability updating $C$ in Gibbs sampling procedure.
n.burn the number of burn-in sample in Gibbs sampling ($\geq$ n.obs).
yvalues a vector of distinct binary outcome values. Defaults to c(0, 1).

Value

a list of $[n.gibbs] \times [n.sample]$ independent observations:

inputY a $([[n.gibbs] \times [n.sample]]) \times m$ matrix for outcomes.
inputA a $([[n.gibbs] \times [n.sample]]) \times m$ matrix for treatments.
inputC a $([[n.gibbs] \times [n.sample]]) \times m$ matrix for confounders.

Examples

library(netchain)
weight.matrix <- matrix(c(0.5, 1, 0, 1, 0.3, 0.5, 0, 0.5, -0.5), 3, 3)
simobs <- simGibbs(n.unit = 3, n.gibbs = 200, n.sample = 10,
weight.matrix,
treat.matrix = 0.5*diag(3), cov.matrix= (-0.3)*diag(3) )
inputY <- simobs$inputY
inputA <- simobs$inputA
inputC <- simobs$inputC
Index

* package
  - netchain-package, 2

causal.influence, 2
chain.causal.multi, 4
chaingibbs, 6

multiloglikechain, 7
multimainfunction, 8
multipartition, 9

netchain (netchain-package), 2
netchain-package, 2
simGibbs, 9