

Package ‘mcmcabn’

June 2, 2021

Title Flexible Implementation of a Structural MCMC Sampler for DAGs

Version 0.4

Maintainer Gilles Kratzer <gilles.kratzer@gmail.com>

Description Flexible implementation of a structural MCMC sampler for Directed Acyclic Graphs (DAGs). It supports the new edge reversal move from Grzegorzczyk and Husmeier (2008) <[doi:10.1007/s10994-008-5057-7](https://doi.org/10.1007/s10994-008-5057-7)> and the Markov blanket resampling from Su and Borsuk (2016) <<https://jmlr.org/papers/v17/su16a.html>>. It supports three priors: a prior controlling for structure complexity from Koivisto and Sood (2004) <<https://dl.acm.org/doi/10.5555/1005332.1005352>>, an uninformative prior and a user-defined prior. The three main problems that can be addressed by this R package are selecting the most probable structure based on a cache of pre-computed scores, controlling for overfitting, and sampling the landscape of high scoring structures. It allows us to quantify the marginal impact of relationships of interest by marginalizing out over structures or nuisance dependencies. Structural MCMC seems an elegant and natural way to estimate the true marginal impact, so one can determine if its magnitude is big enough to consider as a worthwhile intervention.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports gRbase, abn, coda, ggplot2, cowplot, ggpubr

Suggests bnlearn, knitr, rmarkdown, ggdag, testthat

VignetteBuilder knitr

URL <https://www.math.uzh.ch/pages/mcmcabn/>

BugReports <https://git.math.uzh.ch/gkratz/mcmcabn/-/issues>

RoxygenNote 6.1.1

NeedsCompilation no

Author Gilles Kratzer [aut, cre] (<<https://orcid.org/0000-0002-5929-8935>>),
Reinhard Furrer [ctb] (<<https://orcid.org/0000-0002-6319-2332>>)

Repository CRAN

Date/Publication 2021-06-02 06:10:01 UTC

R topics documented:

| | |
|---------------------------------|----|
| . mcmcabn | 2 |
| bsc.compute.asia | 3 |
| CoupledHeatedmcmcabn | 4 |
| dist.asia | 8 |
| mcmc.out.asia | 9 |
| mcmcabn | 9 |
| mcmc_run_asia | 14 |
| plot.mcmcabn | 16 |
| print.mcmcabn | 17 |
| print.summary.mcmcabn | 18 |
| query | 19 |
| summary.mcmcabn | 20 |

| | |
|--------------|-----------|
| Index | 22 |
|--------------|-----------|

. mcmcabn . *mcmcabn Package*

Description

mcmcabn is a structural MCMC sampler for Directed Acyclic Graphs (DAGs). The three main problems that can be addressed by this R package are selecting the most probable structure based on a cache of pre-computed scores, controlling for overfitting and sampling the landscape of high scoring structures. It allows to quantify the marginal impact of relationships of interest by marginalising out over structures or nuisance dependencies.

General overview

What is **mcmcabn**:

Bayesian network modeling is a data analysis technique that is ideally suited to messy, highly correlated, and complex datasets. This methodology is somewhat distinct from other forms of statistical modeling in that its focus is on structure discovery - determining an optimal graphical model that describes the inter-relationships in the underlying processes which generated the data. It is a multivariate technique and can be used for one or many dependent variables. This is a data-driven approach, as opposed to, rely only on subjective expert opinion to determine how variables of interest are inter-related (for example, structural equation modeling).

The R package mcmcabn is a structural MCMC sampler for Directed Acyclic Graphs (DAGs). It contains routines to compute, analyze, and report MCMC samples. This structural sampler supports the new edge reversal move from Grzegorzczuk and Husmeier (2008) <doi:10.1007/s10994-008-5057-7> and the Markov blanket resampling from Su and Borsuk (2016) <<http://jmlr.org/papers/v17/su16a.html>>. It supports three priors: a prior controlling for structure complexity from Koivisto and Sood (2004) <<http://dl.acm.org/citation.cfm?id=1005332.1005352>>, an uninformative prior and a user-defined prior.

The web pages <https://www.math.uzh.ch/pages/mcmcabn/> provide further case studies. See also the files stored in the package directories `inst/FCV_code`.

Author(s)

Gilles Kratzer

References

Kratzer G, Lewis FI, Willi B, Meli ML, Boretti FS, Hofmann-Lehmann R, Torgerson P, Furrer R and Hartnack S (2020) Bayesian Network Modeling Applied to Feline Calicivirus Infection Among Cats in Switzerland. *Front. Vet. Sci.* 7:73. doi: 10.3389/fvets.2020.00073 (Preferred reference)

Kratzer, G. and Furrer, R. (2019). Is a single unique Bayesian network enough to accurately represent your data?. arXiv preprint arXiv:1902.06641

Kratzer, G. and Furrer, R. (2019). mcmcabn: a structural MCMC sampler for DAGs learned from observed systemic datasets. R package version 0.3. <https://CRAN.R-project.org/package=mcmcabn>

Examples

```
## Citations:  
citation('mcmcabn')
```

| | |
|------------------|---|
| bsc.compute.asia | <i>Cache of pre-computed scores related to the asia dataset</i> |
|------------------|---|

Description

This dataframe contains a cache of pre-computed scores with a maximum of two parents per node for the asia dataset.

Usage

```
bsc.compute.asia
```

Format

The data contains a cache of pre-computed scores with a maximum of two parents per node using [buildScoreCache](#) from the [abn](#) R package.

- `bsc.compute.asia`: cache of score with a maximum of two parents per node.

Examples

```
## This data set was generated using the following code:  
data(asia, package='bnlearn') # for the dataset  
library("abn") # for the cache of score function  
  
# Renaming columns of the dataset  
colnames(asia) <- c("Asia",  
  "Smoking",  
  "Tuberculosis",  
  "LungCancer",
```

```

    "Bronchitis",
    "Either",
    "XRay",
    "Dyspnea")

bsc.compute.asia <- buildScoreCache(data.df = asia,
                                   data.dists = dist.asia,
                                   max.parents = 2)

```

CoupledHeatedmcmcabcn *Coupled Heated Structural MCMC sampler for DAGs*

Description

This function is a coupled heated structural Monte Carlo Markov Chain sampler that is equipped with two large scale MCMC moves and parallel tempering that are purposed to synergistically accelerate chain mixing.

Usage

```

CoupledHeatedmcmcabcn(score.cache = NULL,
                      score = "mlik",
                      data.dists = NULL,
                      max.parents = 1,
                      mcmc.scheme = c(100,1000,1000),
                      seed = 42,
                      verbose = FALSE,
                      start.dag = NULL,
                      prior.dag = NULL,
                      prior.lambda = NULL,
                      prob.rev = 0.05,
                      prob.mbr = 0.05,
                      heating = 1,
                      n.chains = 4,
                      prior.choice = 2)

```

Arguments

| | |
|-------------|---|
| score.cache | output from buildScoreCache from the abn R package. |
| score | character giving which network score should be used to sample the DAGs landscape. |
| data.dists | a named list giving the distribution for each node in the network, see details. |
| max.parents | a constant giving the maximum number of parents allowed. |
| mcmc.scheme | a sampling scheme. It is vector giving in that order: the number of returned DAGS, the number of thinned steps and length of the burn-in phase. |

| | |
|--------------|---|
| seed | a non-negative integer which sets the seed. |
| verbose | extra output, see output for details. |
| start.dag | a DAG given as a matrix, see details for format, which can be used to provide a starting point for the structural search explicitly. Alternatively, the character "random" will select a random DAG as a starting point. Character "hc" will call a hill-climber to select a DAG as a starting point. |
| prior.dag | user defined prior. It should be given as a matrix where entries range from zero to one. 0.5 is non-informative for the given arc. |
| prior.lambda | hyper parameter representing the strength of belief in the user-defined prior. |
| prob.rev | probability of selecting a new edge reversal. |
| prob.mbr | probability of selecting a Markov blanket resampling move. |
| heating | a real positive number that heats the chains. The default is one. See details |
| n.chains | Number of chains to be run, see details. |
| prior.choice | an integer, 1 or 2, where 1 is a uniform structural prior and 2 uses a weighted prior, see details. |

Details

The procedure runs a coupled heated structural Monte Carlo Markov Chain to find the most probable posterior network (DAG). The default algorithm is based on three MCMC moves in a parallel tempering scheme: edge addition, edge deletion, and edge reversal. This algorithm is known as the (MC)³. It is known to mix slowly and getting stuck in low probability regions. Indeed, changing of Markov equivalence region often requires multiple MCMC moves. Then large scale MCMC moves are implemented. The user can set the relative frequencies. The new edge reversal move (REV) from Grzegorzcyk and Husmeier (2008) and the Markov blanket resampling (MBR) from Su and Borsuk (2016). The classical reversal move depends on the global configuration of the parents and children and fails to propose MCMC jumps that produce valid but very different DAGs in a unique move. The REV move sample globally a new set of parents. The MBR workaround applies the same idea but to the entire Markov blanket of a randomly chosen node.

The classical (MC)³ is unbiased but inefficient in mixing. The two radical MCMC alternative moves are known to accelerate mixing without introducing biases. Those MCMC moves are computationally expensive. Then low frequencies are advised. The REV move is not necessarily ergodic. Then it should not be used alone.

The parallel tempering scheme has been proposed by (Geyer, 1991). The idea is to run multiple MCMC chains at different temperatures. This will flatten the posterior probability distribution and then making jumps across low probability regions more probable. At each iteration, a swap between two randomly chosen chains is evaluated through a Metropolis like probability. The temperature is sequentially increased in the chains following

The parameter `start.dag` can be: "random", "hc" or user defined. If user select "random" then a random valid DAG is selected. The routine used favourise low density structure. If "hc" (for Hill-climber: [searchHeuristic](#)) then a DAG is selected using 100 different searches with 500 optimization steps. A user defined DAG can be provided. It should be a named square matrix containing only zeros and ones. The DAG should be valid (i.e. acyclic).

The parameter `prior.choice` determines the prior used within each node for a given choice of parent combination. In Koivisto and Sood (2004) p.554, a form of prior is used, which assumes

that the prior probability for parent combinations comprising of the same number of parents are all equal. Specifically, that the prior probability for parent set G with cardinality $|G|$ is proportional to $1/[n-1 \text{ choose } |G|]$ where there are n total nodes. Note that this favors parent combinations with either very low or very high cardinality, which may not be appropriate. This prior is used when `prior.choice=2`. When `prior.choice=1` an uninformative prior is used where parent combinations of all cardinalities are equally likely. When `prior.choice=3` a user-defined prior is used, defined by `prior.dag`. It is given by an adjacency matrix (squared and same size as number of nodes) where entries ranging from zero to one give the user prior belief. An hyperparameter defining the global user belief in the prior is given by `prior.lambda`.

MCMC sampler comes with asymptotic statistical guarantees. Therefore it is highly advised to run multiple long enough chains. The burn-in phase length (i.e., throwing away first MCMC iterations) should be adequately chosen.

The argument `data.dists` must be a list with named arguments, one for each of the variables in `data.df`, where each entry is either "poisson", "binomial", or "gaussian".

The parameter heating could improve convergence. It should be a real positive number. One is neutral. The larger, the more probable to accept any move.

Value

A list with an entry for the list of sampled DAGs, the list of scores, the acceptance probability, the method used for each MCMC jump, the rejection status for each MCMC jump, the total number of iterations the thinning, the length of burn-in phase, the named list of distribution per node, the heating parameter for each chain and a data.frame with the score of all chains. The returned object is of class `mcmcabn`.

Author(s)

Gilles Kratzer

References

For the general methodology:

Kratzer G, Lewis FI, Willi B, Meli ML, Boretti FS, Hofmann-Lehmann R, Torgerson P, Furrer R and Hartnack S (2020) Bayesian Network Modeling Applied to Feline Calicivirus Infection Among Cats in Switzerland. *Front. Vet. Sci.* 7:73. doi: 10.3389/fvets.2020.00073.

For the new edge reversal:

Grzegorzczuk, M., Husmeier, D. (2008). "Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move", *Machine Learning*, vol. 71(2-3), 265.

For the Markov Blanket resampling move:

Su, C., Borsuk, M. E. (2016). "Improving structure MCMC for Bayesian networks through Markov blanket resampling", *The Journal of Machine Learning Research*, vol. 17(1), 4042-4061.

For the Coupled Heated MCMC algorithm:

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood.

For the Koivisto prior:

Koivisto, M. V. (2004). Exact Structure Discovery in Bayesian Networks, *Journal of Machine Learning Research*, vol 5, 549-573.

For the user-defined prior:

Werhli, A. V., Husmeier, D. (2007). "Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge". *Statistical Applications in Genetics and Molecular Biology*, 6 (Article 15).

Imoto, S., Higuchi, T., Goto, T., Tashiro, K., Kuhara, S., Miyano, S. (2003). Using Bayesian networks for estimating gene networks from microarrays and biological knowledge. In *Proceedings of the European Conference on Computational Biology*.

For the asia dataset:

Scutari, M. (2010). Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 35(3), 1-22. doi:<http://dx.doi.org/10.18637/jss.v035.i03>.

Examples

```
## Example from the asia dataset from Lauritzen and Spiegelhalter (1988)
## provided by Scutari (2010)

# The number of MCMC run is deliberately chosen too short (computing time)
# no thinning (usually not recommended)
# no burn-in (usually not recommended,
# even if not supported by any theoretical arguments)

data("mcmc_run_asia")

# run: 0.03 REV, 0.03 MBR, 0.94 MC3 MCMC jumps and 4 chains
# with a random DAG as starting point

mcmc.out.asia.small1 <- CoupledHeatedmcmcabcn(score.cache = bsc.compute.asia,
      score = "mlik",
      data.dists = dist.asia,
      max.parents = 2,
      mcmc.scheme = c(150,0,0),
      seed = 5416,
      verbose = FALSE,
      start.dag = "random",
      prob.rev = 0.03,
      prob.mbr = 0.03,
      prior.choice = 2,heating = 0.,n.chains = 4)

summary(mcmc.out.asia.small1)

# compared to the mcmcabcn() function

mcmc.out.asia.small2 <- mcmcabcn(score.cache = bsc.compute.asia,
      score = "mlik",
      data.dists = dist.asia,
      max.parents = 2,
      mcmc.scheme = c(150,0,0),
      seed = 5416,
      verbose = FALSE,
      start.dag = "random",
      prob.rev = 0.03,
```

```
      prob.mbr = 0.03,  
      prior.choice = 2,heating = 0.25)  
  
summary(mcmc.out.asia.small2)
```

| | |
|-----------|--|
| dist.asia | <i>Named list of distributions to analyze asia dataset</i> |
|-----------|--|

Description

Named list of distribution to analyze asia dataset.

Usage

```
dist.asia
```

Format

The dataset contains a list of named distribution used to analyse the asia dataset.

- `dist.asia`: a named list giving the distribution for each node in the network.

Examples

```
## This data set was generated using the following code:  
data(asia, package='bnlearn') # for the dataset  
  
# Renaming columns of the dataset  
colnames(asia) <- c("Asia",  
  "Smoking",  
  "Tuberculosis",  
  "LungCancer",  
  "Bronchitis",  
  "Either",  
  "XRay",  
  "Dyspnea")  
  
# Defining the distribution list  
dist.asia <- list(Asia = "binomial",  
  Smoking = "binomial",  
  Tuberculosis = "binomial",  
  LungCancer = "binomial",  
  Bronchitis = "binomial",  
  Either = "binomial",  
  XRay = "binomial",  
  Dyspnea = "binomial")
```

| | |
|---------------|--|
| mcmc.out.asia | <i>MCMC searches from the synthetic asia dataset for use with mcmcabn library examples</i> |
|---------------|--|

Description

This dataframe contains a cache of pre-computed scores with a maximum of two parents per node for the asia dataset.

Usage

```
mcmc.out.asia
```

Format

The data contains an object of class mcmcabn.

- mcmc.out.asia: an object of class mcmcabn.

Examples

```
## Not run:
## This data set was generated using the following code:
data(asia, package='bnlearn') #for the dataset
library(abn) #for the cache of scores computing function

mcmc.out.asia <- mcmcabn(score.cache = bsc.compute.asia,
  score = "mlik",
  data.dists = dist.asia,
  max.parents = 2,
  mcmc.scheme = c(1000,99,1000),
  seed = 42,
  verbose = FALSE,
  start.dag = "random",
  prob.rev = 0.03,
  prob.mbr = 0.03,
  prior.choice = 2)

## End(Not run)
```

| | |
|---------|---|
| mcmcabn | <i>Structural MCMC sampler for DAGs</i> |
|---------|---|

Description

This function is a structural Monte Carlo Markov Chain Model Choice (MC)³ sampler that is equipped with two large scale MCMC moves that are purposed to accelerate chain mixing.

Usage

```
mcmcabn(score.cache = NULL,
        score = "mlik",
        data.dists = NULL,
        max.parents = 1,
        mcmc.scheme = c(100,1000,1000),
        seed = 42,
        verbose = FALSE,
        start.dag = NULL,
        prior.dag = NULL,
        prior.lambda = NULL,
        prob.rev = 0.05,
        prob.mbr = 0.05,
        heating = 1,
        prior.choice = 2)
```

Arguments

| | |
|--------------|---|
| score.cache | output from buildScoreCache from the abn R package. |
| score | character giving which network score should be used to sample the DAGs landscape. |
| data.dists | a named list giving the distribution for each node in the network, see details. |
| max.parents | a constant giving the maximum number of parents allowed. |
| mcmc.scheme | a sampling scheme. It is vector giving in that order: the number of returned DAGS, the number of thinned steps and length of the burn-in phase. |
| seed | a non-negative integer which sets the seed. |
| verbose | extra output, see output for details. |
| start.dag | a DAG given as a matrix, see details for format, which can be used to provide a starting point for the structural search explicitly. Alternatively, character "random" will select a random DAG as a starting point. Character "hc" will call a hill-climber to select a DAG as a starting point. |
| prior.dag | user defined prior. It should be given as a matrix where entries range from zero to one. 0.5 is non-informative for the given arc. |
| prior.lambda | hyper parameter representing the strength of belief in the user-defined prior. |
| prob.rev | probability of selecting a new edge reversal. |
| prob.mbr | probability of selecting a Markov blanket resampling move. |
| heating | a real positive number that heats up the chain if between zero and one and apply an exponential decrease scheme if larger than one. The default is one. See details |
| prior.choice | an integer, 1 or 2, where 1 is a uniform structural prior and 2 uses a weighted prior, see details. |

Details

The procedure runs a structural Monte Carlo Markov Chain Model Choice (MC)³ to find the most probable posterior network (DAG). The default algorithm is based on three MCMC moves: edge addition, edge deletion, and edge reversal. This algorithm is known as the (MC)³. It is known to mix slowly and getting stuck in low probability regions. Indeed, changing of Markov equivalence region often requires multiple MCMC moves. Then large scale MCMC moves are implemented. The user can set the relative frequencies. The new edge reversal move (REV) from Grzegorzczuk and Husmeier (2008) and the Markov blanket resampling (MBR) from Su and Borsuk (2016). The classical reversal move depends on the global configuration of the parents and children and fails to propose MCMC jumps that produce valid but very different DAGs in a unique move. The REV move sample globally a new set of parents. The MBR workaround applies the same idea but to the entire Markov blanket of a randomly chosen node.

The classical (MC)³ is unbiased but inefficient in mixing. The two radical MCMC alternative moves are known to accelerate mixing without introducing biases. Those MCMC moves are computationally expensive. Then low frequencies are advised. The REV move is not necessarily ergodic, then it should not be used alone.

The parameter `start.dag` can be: "random", "hc" or user defined. If user select "random" then a random valid DAG is selected. The routine used favourise low density structure. If "hc" (for Hill-climber: [searchHeuristic](#)) then a DAG is selected using 100 different searches with 500 optimization steps. A user defined DAG can be provided. It should be a named square matrix containing only zeros and ones. The DAG should be valid (i.e. acyclic).

The parameter `prior.choice` determines the prior used within each node for a given choice of parent combination. In Koivisto and Sood (2004) p.554, a form of prior is used, which assumes that the prior probability for parent combinations comprising of the same number of parents are all equal. Specifically, that the prior probability for parent set G with cardinality $|G|$ is proportional to $1/[n-1 \text{ choose } |G|]$ where there are n total nodes. Note that this favors parent combinations with either very low or very high cardinality, which may not be appropriate. This prior is used when `prior.choice=2`. When `prior.choice=1` an uninformative prior is used where parent combinations of all cardinalities are equally likely. When `prior.choice=3` a user-defined prior is used, defined by `prior.dag`. It is given by an adjacency matrix (squared and same size as number of nodes) where entries ranging from zero to one give the user prior belief. An hyperparameter defining the global user belief in the prior is given by `prior.lambda`.

MCMC sampler comes with asymptotic statistical guarantees. Therefore it is highly advised to run multiple long enough chains. The burn-in phase length (i.e. throwing away first MCMC iterations) should be adequately chosen.

The argument `data.dists` must be a list with named arguments, one for each of the variables in `data.df`, where each entry is either "poisson", "binomial", or "gaussian".

The parameter `heating` could improve convergence. It should be a real positive number. If smaller than one, it is a tuning parameter which transforms the score by raising it to this power. One is neutral. The smaller, the more probable to accept any move. If larger than one, it indicates the number of returned steps where an exponentially decrease heating scheme is applied. After this number of steps, the heating parameter is set to one.

Value

A list with an entry for the list of sampled DAGs, the list of scores, the acceptance probability, the method used for each MCMC jump, the rejection status for each MCMC jump, the total number of

iterations the thinning, the length of burn-in phase, the named list of distribution per node and the heating parameter. The returned object is of class `mcmcabn`.

Author(s)

Gilles Kratzer

References

For the implementation of the function:

Kratzer G, Lewis FI, Willi B, Meli ML, Boretti FS, Hofmann-Lehmann R, Torgerson P, Furrer R and Hartnack S (2020) Bayesian Network Modeling Applied to Feline Calicivirus Infection Among Cats in Switzerland. *Front. Vet. Sci.* 7:73. doi: 10.3389/fvets.2020.00073.

For the new edge reversal:

Grzegorzczuk, M., Husmeier, D. (2008). "Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move", *Machine Learning*, vol. 71(2-3), 265.

For the Markov Blanket resampling move:

Su, C., Borsuk, M. E. (2016). "Improving structure MCMC for Bayesian networks through Markov blanket resampling", *The Journal of Machine Learning Research*, vol. 17(1), 4042-4061.

For the Koivisto prior:

Koivisto, M. V. (2004). Exact Structure Discovery in Bayesian Networks, *Journal of Machine Learning Research*, vol 5, 549-573.

For the user defined prior:

Werhli, A. V., Husmeier, D. (2007). "Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge". *Statistical Applications in Genetics and Molecular Biology*, 6 (Article 15).

Imoto, S., Higuchi, T., Goto, T., Tashiro, K., Kuhara, S., Miyano, S. (2003). Using Bayesian networks for estimating gene networks from microarrays and biological knowledge. In *Proceedings of the European Conference on Computational Biology*.

For the asia dataset:

Scutari, M. (2010). Learning Bayesian Networks with the `bnlearn` R Package. *Journal of Statistical Software*, 35(3), 1-22. doi:<http://dx.doi.org/10.18637/jss.v035.i03>.

Examples

```
## Example from the asia dataset from Lauritzen and Spiegelhalter (1988)
## provided by Scutari (2010)

# The number of MCMC run is deliberately chosen too small (computing time)
# no thinning (usually not recommended)
# no burn-in (usually not recommended,
# even if not supported by any theoretical arguments)

data("mcmc_run_asia")

# Let us run: 0.03 REV, 0.03 MBR, 0.94 MC3 MCMC jumps
```

```

# with a random DAG as starting point

mcmc.out.asia.small <- mcmcabn(score.cache = bsc.compute.asia,
  score = "mlik",
  data.dists = dist.asia,
  max.parents = 2,
  mcmc.scheme = c(100,0,0),
  seed = 321,
  verbose = FALSE,
  start.dag = "random",
  prob.rev = 0.03,
  prob.mbr = 0.03,
  prior.choice = 2)

summary(mcmc.out.asia.small)

# Soly with MC3 moves

mcmc.out.asia.small <- mcmcabn(score.cache = bsc.compute.asia,
  score = "mlik",
  data.dists = dist.asia,
  max.parents = 2,
  mcmc.scheme = c(50,0,0),
  seed = 42,
  verbose = FALSE,
  start.dag = "random",
  prob.rev = 0,
  prob.mbr = 0,
  prior.choice = 2)

summary(mcmc.out.asia.small)

# Defining a starting DAG
startDag <- matrix(data = c(0, 0, 0, 1, 0, 0, 0, 0,
  0, 0, 1, 0, 0, 0, 0, 0,
  1, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 1, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0),nrow = 8,ncol = 8, byrow = TRUE)

colnames(startDag) <- rownames(startDag) <- names(dist.asia)

# Additionally, let us use the non informative prior
mcmc.out.asia.small <- mcmcabn(score.cache = bsc.compute.asia,
  score = "mlik",
  data.dists = dist.asia,
  max.parents = 2,
  mcmc.scheme = c(100,0,0),
  seed = 42,
  verbose = FALSE,
  start.dag = startDag,

```

```

        prob.rev = 0,
        prob.mbr = 0,
        prior.choice = 1)

summary(mcmc.out.asia.small)

# let us define our very own prior
# we know that there should be a link between Smoking and LungCancer nodes

# uninformative prior matrix
priorDag <- matrix(data = 0.5, nrow = 8, ncol = 8)
# name it
colnames(priorDag) <- rownames(priorDag) <- names(dist.asia)
# parent = smoking; child = LungCancer
priorDag["LungCancer", "Smoking"] <- 1

mcmc.out.asia.small <- mcmcabn(score.cache = bsc.compute.asia,
                             score = "mlik",
                             data.dists = dist.asia,
                             max.parents = 2,
                             mcmc.scheme = c(100, 0, 0),
                             seed = 42,
                             verbose = FALSE,
                             start.dag = startDag,
                             prob.rev = 0,
                             prob.mbr = 0,
                             prior.choice = 3,
                             prior.dag = priorDag)

summary(mcmc.out.asia.small)

# let us improve the convergence rate. The 20 first MCMC moves are performed with an
# heating parameter different than one, afterward, heating is set up to one.

mcmc.out.asia.small <- mcmcabn(score.cache = bsc.compute.asia,
                             score = "mlik",
                             data.dists = dist.asia,
                             max.parents = 2,
                             mcmc.scheme = c(150, 0, 0),
                             seed = 41242,
                             verbose = FALSE,
                             start.dag = "random",
                             prob.rev = 0.03,
                             prob.mbr = 0.03,
                             prior.choice = 2, heating = 20)

summary(mcmc.out.asia.small)

```

Description

10⁵ MCMC runs with 1000 burn-in runs from the asia synthetic dataset from Lauritzen and Spiegelhalter (1988) provided by Scutari (2010). A named list of distributions and pre-computed scores.

Usage

```
data("mcmc_run_asia")
```

Format

The data contains an object of class `mcmcabn` and a cache of score computed using `buildScoreCache` from the `abn` R package.

- `bsc.compute.asia`: cache of score with a maximum of two parents per node computed using `buildScoreCache` from the `abn` R package.
- `dist.asia`: a named list giving the distribution for each node in the network.
- `mcmc.out.asia`: an object of class `mcmcabn`.

Examples

```
## Not run:
## This data set was generated using the following code:
data(asia, package='bnlearn') #for the dataset
library(abn) #for the cache of score function

#renaming columns of the dataset
colnames(asia) <- c("Asia",
  "Smoking",
  "Tuberculosis",
  "LungCancer",
  "Bronchitis",
  "Either",
  "XRay",
  "Dyspnea")

#lets define the distribution list
dist.asia <- list(Asia = "binomial",
  Smoking = "binomial",
  Tuberculosis = "binomial",
  LungCancer = "binomial",
  Bronchitis = "binomial",
  Either = "binomial",
  XRay = "binomial",
  Dyspnea = "binomial")

bsc.compute.asia <- buildScoreCache(data.df = asia,
  data.dists = dist.asia,
  max.parents = 2)

mcmc.out.asia <- mcmcabn(score.cache = bsc.compute.asia,
```

```

score = "mlik",
data.dists = dist.asia,
max.parents = 2,
mcmc.scheme = c(1000,99,1000),
seed = 42,
verbose = FALSE,
start.dag = "random",
prob.rev = 0.03,
prob.mbr = 0.03,
prior.choice = 2)

## End(Not run)

```

plot.mcmcabn

Function to plot mcmcabn class objects

Description

Generic function to plot mcmcabn objects.

Usage

```

## S3 method for class 'mcmcabn'
plot(x,
     max.score = FALSE,
     ...)

```

Arguments

| | |
|-----------|---|
| x | object of class mcmcabn. |
| max.score | logical to plot the cumulative maximum network score. |
| ... | arguments to be passed to methods. |

Details

The plot function for mcmcabn objects is based on **ggplot2**, **ggpubr** and **cowplot** packages. By default, it returns a trace plot with coloured points when MBR and REV methods have been used. It displays histograms on the right of the densities of $(MC)^3$, MBR and REV MCMC jumps respectively.

Author(s)

Gilles Kratzer

References

Plotting ability: H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Alboukadel Kassambara (2018). ggpubr: 'ggplot2' Based Publication Ready Plots. R package version 0.2. <https://CRAN.R-project.org/package=ggpubr>

Claus O. Wilke (2019). cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'. R package version 0.9.4. <https://CRAN.R-project.org/package=cowplot>

Data: Scutari, M. (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22. doi:<http://dx.doi.org/10.18637/jss.v035.i03>.

Examples

```
## Example from the asia dataset from Lauritzen and Spiegelhalter (1988)
## provided by Scutari (2010)
data("mcmc_run_asia")

# plot the mcmc run
plot(mcmc.out.asia)

# plot cumulative max score
plot(mcmc.out.asia, max.score = TRUE)
```

print.mcmcabn *Methods for mcmcabn objects*

Description

Method for computing on mcmcabn objects.

Usage

```
## S3 method for class 'mcmcabn'
print(x, ...)
```

Arguments

x an object of class mcmcabn.
... additional arguments passed to print.

Details

There exists a [summary](#) S3 function that displays more details.

Author(s)

Gilles Kratzer

Examples

```
## Example from the asia dataset from Lauritzen and Spiegelhalter (1988)
## provided by Scutari (2010)

print(mcmc.out.asia)
```

`print.summary.mcmcabn` *Methods for printing the summary of mcmcabn objects*

Description

Method for printing the summary of mcmcabn objects.

Usage

```
## S3 method for class 'summary.mcmcabn'
print(x, ...)
```

Arguments

| | |
|------------------|---|
| <code>x</code> | an object of class <code>summary.mcmcabn</code> . |
| <code>...</code> | additional arguments passed to <code>print</code> . |

Details

There exists a [summary](#) S3 function that displays more details.

Author(s)

Gilles Kratzer

Examples

```
## Example from the asia dataset from Lauritzen and Spiegelhalter (1988)
## provided by Scutari (2010)

summary(mcmc.out.asia)
```

| | |
|-------|--|
| query | <i>Function to query MCMC samples generated by mcmcabn</i> |
|-------|--|

Description

The function allows users to perform structural queries over MCMC samples produced by mcmcabn.

Usage

```
query(mcmcabn = NULL,  
      formula = NULL)
```

Arguments

| | |
|---------|--|
| mcmcabn | object of class mcmcabn. |
| formula | formula statement or adjacency matrix to query the MCMC samples, see details. If this argument is NULL, then the average arc-wise frequencies is reported. |

Details

The query can be formulated using an adjacency matrix or a formula-wise expression.

The adjacency matrix should be squared of dimension equal to the number of nodes in the networks. Their entries should be either 1, 0, or -1. The 1 indicates the requested arcs, the -1 the excluded, and the 0 all other entries that are not subject to query. The rows indicated the set of parents of the index nodes. The order of rows and column should be the same as the one used in the mcmcabn() function in the data.dist argument.

The formula statement has been designed to ease querying over the MCMC sample. It allows user to make complex queries without explicitly writing an adjacency matrix (which can be painful when the number of variables is large). The formula argument can be provided using a formula alike: ~ node1|parent1:parent2 + node2:node3|parent3. The formula statement has to start with '~'. In this example, node1 has two parents (parent1 and parent2). node2 and node3 have the same parent3. The parents' names have to match those given in name exactly. ':' is the separator between either children or parents, '|' separates children (left side) and parents (right side), '+' separates terms, '.' replaces all the variables in name. Additional, when one wants to exclude an arc simply put '-' in front of that statement. Then a formula alike: ~ -node1|parent1 exclude all DAGs that have an arc between parent1 and node1.

If the formula argument is not provided, the function returns the average support of all individual arcs using a named matrix.

Value

A frequency for the requested query. Alternatively a matrix with arc-wise frequencies.

Author(s)

Gilles Kratzer

References

Kratzer, G., Furrer, R. "Is a single unique Bayesian network enough to accurately represent your data?". arXiv preprint arXiv:1902.06641.

Lauritzen, S., Spiegelhalter, D. (1988). "Local Computation with Probabilities on Graphical Structures and their Application to Expert Systems (with discussion)". Journal of the Royal Statistical Society: Series B, 50(2):157–224.

Scutari, M. (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1–22. doi:<http://dx.doi.org/10.18637/jss.v035.i03>.

Examples

```
## Example from the asia dataset from Lauritzen and Spiegelhalter (1988)
## provided by Scutari (2010)
data("mcmc_run_asia")

## Return a named matrix with individual arc support
query(mcmcabn = mcmc.out.asia)

## What is the probability of LungCancer node being children of the Smoking node?
query(mcmcabn = mcmc.out.asia, formula = ~LungCancer|Smoking)

## What is the probability of Smoking node being parent of
## both LungCancer and Bronchitis node?
query(mcmcabn = mcmc.out.asia,
      formula = ~ LungCancer|Smoking+Bronchitis|Smoking)

## What is the probability of previous statement, when there
## is no arc from Smoking to Tuberculosis and from Bronchitis to XRay?
query(mcmcabn = mcmc.out.asia,
      formula = ~LungCancer|Smoking + Bronchitis|Smoking -
              Tuberculosis|Smoking - XRay|Bronchitis)
```

summary.mcmcabn

Function to summarize MCMC run generated by mcmcabn

Description

Summary method for mcmcabn objects.

Usage

```
## S3 method for class 'mcmcabn'
summary(object,
        quantiles = c(0.025, 0.25, 0.5, 0.75, 0.975),
        lag.max = 10,
        ...)
```

Arguments

| | |
|-----------|--|
| object | object of class mcmcabn. |
| quantiles | numeric vector of probabilities with values in [0,1]. (Values up to 2e-14 outside that range are accepted and moved to the nearby endpoint.) |
| lag.max | maximum lag at which to calculate the acf . Default is set to 10. |
| ... | arguments to be passed to methods. |

Details

The summary function for mcmcabn objects returns multiple summary metrics for assessing the quality of the MCMC run. Thinning is the number of thinned MCMC steps for one MCMC returned.

Value

This method prints: the number of burn-in steps, the number of MCMC steps, the thinning, the maximum achieved score, the empirical mean of the MCMC samples, the empirical standard deviation of the MCMC samples, the user defined quantiles of the posterior network score, the global acceptance rate, a table of the accepted and rejected moves in function of the methods used, the sample size adjusted for autocorrelation and the autocorrelations by lag.

Author(s)

Gilles Kratzer

References

Scutari, M. (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1 - 22. doi:<http://dx.doi.org/10.18637/jss.v035.i03>.

Examples

```
## Example from the asia dataset from Lauritzen and Spiegelhalter (1988)
## provided by Scutari (2010)
#summary the MCMC run
summary(mcmc.out.asia)
```

Index

- * **datasets**
 - bsc.compute.asia, [3](#)
 - dist.asia, [8](#)
 - mcmc.out.asia, [9](#)
 - mcmc_run_asia, [14](#)
- * **documentation**
 - . mcmcabn ., [2](#)
- * **package**
 - . mcmcabn ., [2](#)
 - . mcmcabn ., [2](#)

[acf](#), [21](#)

[bsc.compute.asia](#), [3](#)

[buildScoreCache](#), [3](#), [4](#), [10](#), [15](#)

[CoupledHeatedmcmcabn](#), [4](#)

[dist.asia](#), [8](#)

[mcmc.out.asia](#), [9](#)

[mcmc_run_asia](#), [14](#)

[mcmcabn](#), [9](#)

[mcmcabn-package](#) (. mcmcabn .), [2](#)

[overview](#) (. mcmcabn .), [2](#)

[plot.mcmcabn](#), [16](#)

[print.mcmcabn](#), [17](#)

[print.summary.mcmcabn](#), [18](#)

[query](#), [19](#)

[searchHeuristic](#), [5](#), [11](#)

[summary](#), [17](#), [18](#)

[summary.mcmcabn](#), [20](#)