Package 'cia'

July 22, 2025

Title Learn and Apply Directed Acyclic Graphs for Causal Inference

Version 1.0.0

Description Causal Inference Assistance (CIA) for performing causal inference within the structural causal modelling framework. Structure learning is performed using partition Markov chain Monte Carlo (Kuipers & Moffa, 2017) and several additional functions have been added to help with causal inference. Kuipers and Moffa (2017) <doi:10.1080/01621459.2015.1133426>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Depends R (>= 4.4.0)

Imports bnlearn (>= 4.9), igraph, doParallel, parallel, foreach, arrangements, graphics, dplyr, rlang, fastmatch, methods, gRain, patchwork, tidyr

Suggests rmarkdown, knitr, testthat (>= 3.0.0), gtools, gRbase, ggplot2, qgraph, dagitty

Config/testthat/edition 3

URL https://spaceodyssey.github.io/cia/

BugReports https://github.com/SpaceOdyssey/cia/issues

NeedsCompilation no

Author Mathew Varidel [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1648-8317>), Victor An [ctb]

Maintainer Mathew Varidel <mathew.varidel@sydney.edu.au>

Repository CRAN

Date/Publication 2024-11-13 14:00:07 UTC

Contents

BNLearnScorer . . . 3
CalculateAcceptanceRates . . . 3
CalculateEdgeProbabilities . . . 4
CalculateFeatureMean . . . 5
CollectUniqueObjects . . . 6
CreateScorer . . . 8
DAGtoCPDAG . . . 9
DAGtoPartition . . . 9
DefaultProposal . . . 10
FlattenChains . . . 11
GetEmptyDAG . . . 12
GetIncrementalScoringEdges . . . 12
GetLowestPairwiseScoringEdges . . . 13
GetMAP . . . 14
MutilateGraph . . . 15
PartitionMCMC . . . 16
PartitiontoDAG . . . 17
PlotConcordance . . . 18
PlotCumulativeMeanTrace . . . 19
PlotScoreTrace . . . 20
PostProcessChains . . . 21
SampleChains . . . 22
SampleEdgeProbabilities . . . 23
SamplePosteriorPredictiveChains . . . 24
ScoreDAG . . . 25
ScoreLabelledPartition . . . 25
toBNLearn . . . 26
togRain . . . 27
toMatrix . . . 27
UniformlySampleDAG . . . 28
[.cia_chain . . . 28
[.cia_chains . . . 29
[.cia_post_chain . . . 30
[.cia_post_chains . . . 31
[[.cia_chains . . . 32
[[.cia_post_chains . . . 32

Index **34**

BNLearnScorer *BNLearnScorer*

Description

A thin wrapper on the bnlearn::score function.

Usage

```
BNLearnScorer(node, parents, ...)
```

Arguments

node	Name of node to score.
parents	The parents of node.
...	The ellipsis is used to pass other parameters to the scorer.

Value

A numeric value representing the log score of the node given the parents.

Examples

```
data <- bnlearn::learning.test
BNLearnScorer('A', c('B', 'C'), data = data)
BNLearnScorer('A', c(), data = data)
BNLearnScorer('A', vector(), data = data)
BNLearnScorer('A', NULL, data = data)
BNLearnScorer('A', c('B', 'C'), data = data, type = "bde", iss = 100)
BNLearnScorer('A', c('B', 'C'), data = data, type = "bde", iss = 1)
```

CalculateAcceptanceRates

Calculate acceptance rates

Description

This makes the assumption that the proposal has saved a variable "proposal_used" and mcmc has saved a variable 'accept'.

Usage

```
CalculateAcceptanceRates(chains, group_by = NULL)
```

Arguments

chains	MCMC chains.
group_by	Vector of strings that are in c("chain", "proposal_used"). Default is NULL which will return the acceptance rates marginalised over chains and the proposal used.

Value

Summary of acceptance rates per grouping.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
CalculateAcceptanceRates(results)
```

CalculateEdgeProbabilities

Calculate pairwise edge probabilities

Description

Calculate pairwise edge probabilities. The posterior probability of an edge E given the data D is given by marginalising out the graph structure g over the graph space G, such that

$$p(E|D) = \sum_{g \in G} p(E|g)p(g|D).$$

Usage

```
CalculateEdgeProbabilities(x, ...)
```

Arguments

x	A cia_chain(s) or collection object where states are DAGs.
...	Extra parameters sent to the methods. For a dag collection you can choose to use estimated p(g|D) in two ways which can be specified using the 'method' parameter.method='sampled' for MCMC sampled frequency (which is our recommended method) or method='score' which uses the normalised scores.

Details

The posterior probability for a given graph p(g|D) is estimated in two ways which can be specified using the 'method' parameter.

Value

Matrix of edge probabilities.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
dag_chains <- PartitiontoDAG(results, scorer)
CalculateEdgeProbabilities(dag_chains)
```

CalculateFeatureMean *Calculate arithmetic mean for a DAG feature*

Description

Calculate the posterior expected value for a feature ($f(g)$, e.g., existence of an edge in graph g) by marginalising out the graph structure q over the graph space G, thus

$$E(f|D) = \sum_{g \in G} f(g)p(g|D).$$

This can be useful for calculating point estimates of quantities of interests, such as the probability that an edge exists or the probability of one node being an ancestor of another.

Usage

```
CalculateFeatureMean(x, p_feature, ...)
```

Arguments

x	A chain(s) or collection object.
p_feature	A function that takes an adjacency matrix or collection object and returns a scalar corresponding to $f(g)$. The function must be of the form p_feature(dag).

... Extra parameters sent to the methods. For a dag collection you can choose to use estimated p(g|D) in two ways which can be specified using the 'method' parameter.method='sampled' for MCMC sampled frequency (which is our recommended method) or method='score' which uses the normalised scores.

Value

A numeric value representing the posterior probability of the feature.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
dag_chains <- PartitiontoDAG(results, scorer)

# Calculate the mean edge probability per chain.
CalculateFeatureMean(dag_chains, function(x) { return(x) })

# Calculate the mean edge probability across chains.
CalculateFeatureMean(FlattenChains(dag_chains), function(x) { return(x) })
```

CollectUniqueObjects *Collect unique objects*

Description

Get the unique set of states along with their log score.

Usage

```
CollectUniqueObjects(x)
```

Arguments

x A cia_chains or cia_chain object.

Details

This gets the unique set of states in cia_chain(s) referred to as objects (o). Then it estimates the probability for each state using two methods. The `log_sampling_prob` is the MCMC sampled frequency estimate for the posterior probability.

An alternative method to estimate the posterior probability for each state uses the state score. This is recorded in the `log_norm_state_score`. This approach estimates the log of the normalisation constant assuming $\tilde{Z}_O = \Sigma_{s=1}^{S} p(o_s)p(D|o_s)$ where $O = \{o_1, o_2, o_3, ..., o_S\}$ is the set of unique objects in the chain. This assumes that you have captured the most probable objects, such that $\tilde{Z}_O$ is approximately equal to the true evidence $Z = \Sigma_{g \in G} p(g)p(D|g)$ where the sum across all possible DAGs (G). This also makes the assumption that the exponential of the score is proportional to the posterior probability, such that

$$p(g|D) \propto p(g)p(D|g) = \prod_i \exp(\text{score}(X_i, \text{Pa}_g(X_i)|D))$$

where $\text{Pa}_g(X_i)$ is the parents set for node X_i given the graph g.

After the normalisation constant has been estimated we then estimate the log probability of each object as,

$$\log(p(o|D)) = \log(p(o)p(D|o)) - \log(\tilde{Z}_o).$$

Preliminary analysis suggests that the sampling frequency approach is more consistent across chains when estimating marginalised edge probabilities, and therefore is our preferred method. However, more work needs to be done here.

Value

A list with entries:

- state: List of unique states.
- log_evidence_state: Numeric value representing the evidence calculated from the states.
- log_state_score: Vector with the log scores for each state.
- log_sampling_prob: Vector with the log of the probability for each state estimated using the MCMC sampling frequency.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(100, partitioned_nodes, PartitionMCMC(), scorer)
collection <- CollectUniqueObjects(results)
```

`CreateScorer` *Scorer constructor*

Description

Scorer constructor

Usage

```
CreateScorer(
  scorer = BNLearnScorer,
  ...,
  max_parents = Inf,
  blacklist = NULL,
  whitelist = NULL,
  cache = FALSE,
  nthreads = 1
)
```

Arguments

`scorer`	A scorer function that takes (node, parents) as parameters. Default is BNLearnScorer.
`...`	Parameters to pass to scorer.
`max_parents`	The maximum number of allowed parents. Default is infinite.
`blacklist`	A boolean matrix of (parent, child) pairs where TRUE represents edges that cannot be in the DAG. Default is NULL which represents no blacklisting.
`whitelist`	A boolean matrix of (parent, child) pairs where TRUE represents edges that must be in the DAG. Default is NULL which represents no whitelisting.
`cache`	A boolean to indicate whether to build the cache. The cache only works for problems where the scorer only varies as a function of (node, parents). Default is FALSE.
`nthreads`	Number of threads used to construct cache.

Value

A list with entries:

- scorer: Function that takes (node, parents) as parameters and returns the score.
- parameters: List of extra parameters passed to the scorer.
- max_parents: Integer representing the maximum number of possible possible parents that any child can have.
- blacklist: Matrix where each cell represents the (parent, child) pairs that must not be present when equal to 1.
- whitelist: Matrix where each cell represents the (parent, child) pairs that must be present when equal to 1. state estimated using the MCMC sampling frequency.

Examples

```
scorer <- CreateScorer(data = bnlearn::asia)
```

DAGtoCPDAG *Convert DAG to CPDAG*

Description

Converts a directed acyclic graph (DAG) into it's equivalence class corresponding to a completed partially directed acyclic graph (CPDAG).

Usage

```
DAGtoCPDAG(x)
```

Arguments

x	A matrix, cia_chain, or cia_chains object. When it is a chain(s) object the state must be an adjacency matrix.

Value

x Returns same object type converted to a CPDAG.

Examples

```
dag <- UniformlySampleDAG(LETTERS[1:3])
DAGtoCPDAG(dag)
```

DAGtoPartition *Convert DAG to partition*

Description

This converts a DAG to it's partition by iteratively constructing sets of outpoints. This is further explained in section 4.1 of Kuipers & Moffa (2017).

Usage

```
DAGtoPartition(dag)
```

Arguments

dag	A directed acyclic graph represented as an adjacency matrix, igraph, or bnlearn object.

Value

Labelled partition for the given adjacency matrix.

References

Kuipers, J., & Moffa, G. (2017). Partition MCMC for inference on acyclic digraphs. Journal of the American Statistical Association, 112(517), 282-299.

Examples

```
dag <- UniformlySampleDAG(LETTERS[1:3])
partitioned_nodes <- DAGtoPartition(dag)
```

DefaultProposal *Default proposal constructor*

Description

This constructs a proposal function for PartitionMCMC.

Usage

```
DefaultProposal(p = c(0.33, 0.33, 0.165, 0.165, 0.01), verbose = TRUE)
```

Arguments

p	Probability for each proposal in the order (split_join, node_move, swap_node, swap_adjacent, stay_still).
verbose	Boolean flag to record proposal used.

Value

A function corresponding to the default proposal.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes,
```

```
PartitionMCMC(
  proposal = DefaultProposal(p = c(0.0, 1.0, 0.0, 0.0, 0.0))
  ),
scorer)
```

FlattenChains *Flatten chains*

Description

Flatten a cia_chains object into a single cia_chain object. This is helpful for when you want to calculate a feature across using all samples across the cia_chains.

Usage

```
FlattenChains(chains)
```

Arguments

chains	A cia_chains object.

Value

A cia_chain object of flattened samples.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
FlattenChains(results)[1:3]
```

GetEmptyDAG *Get an empty DAG given a set of nodes.*

Description

Get an empty DAG given a set of nodes.

Usage

```
GetEmptyDAG(nodes)
```

Arguments

nodes	A vector of node names.

Value

An adjacency matrix with elements designated as (parent, child).

Examples

```
GetEmptyDAG(LETTERS[1:3])
```

GetIncrementalScoringEdges *Get incremental edges*

Description

Get edges that do not incrementally improve the score over an empty DAG greater than a cutoff. In detail, this returns the edges where a graph with the edge E given by g_E such that Score(g_E) - Score(g_empty) < cutoff. Assuming that the scorer returns the log of the marginalised posterior, then the cutoff corresponds to the log of the Bayes Factor. The output can be used as a blacklist.

Usage

```
GetIncrementalScoringEdges(scorer, cutoff = 0)
```

Arguments

scorer	A scorer object.
cutoff	A score cutoff. The score cutoff is equal to the log of the Bayes Factor between the two models.

Value

A Boolean matrix of (parent, child) pairs for blacklisting.

Examples

```
data <- bnlearn::learning.test

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

blacklist <- GetIncrementalScoringEdges(scorer, cutoff = -10.0)

blacklist_scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data,
  cache = TRUE
  )

# Randomly sample a starting DAG consistent with the blacklist. Then
# convert to a partition.
dag <- UniformlySampleDAG(colnames(data)) * !blacklist
partitioned_nodes <- DAGtoPartition(dag)

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), blacklist_scorer)
```

GetLowestPairwiseScoringEdges

Preprocessing for blacklisting Get the lowest pairwise scoring edges.

Description

Get the lowest pairwise scoring edges represented as a blacklist matrix. This blacklisting procedure is motivated by Koller & Friedman (2003). This is rarely used now as we found that it blacklists edges that have significant dependencies but are not in the top n edges. We prefer the GetIncrementalScoringEdges method.

Usage

```
GetLowestPairwiseScoringEdges(scorer, n_retain)
```

Arguments

scorer	A scorer object.
n_retain	An integer representing the number of edges to retain.

Value

A boolean matrix of (parent, child) pairs for blacklisting.

References

1. Koller D, Friedman N. Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. Mach Learn. 2003;50(1):95–125.

Examples

```
data <- bnlearn::learning.test

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

blacklist <- GetLowestPairwiseScoringEdges(scorer, 3)

blacklist_scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data,
  blacklist = blacklist,
  cache = TRUE
  )

# Randomly sample a starting DAG consistent with the blacklist. Then
# convert to a partition.
dag <- UniformlySampleDAG(colnames(data)) * !blacklist
partitioned_nodes <- DAGtoPartition(dag)

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), blacklist_scorer)
```

GetMAP *Get the maximum a posteriori state*

Description

Get the maximum a posteriori state

Usage

```
GetMAP(x)
```

Arguments

x	A collection of unique objects or chains object.

Value

A list with the adjacency matrix for the map and it's posterior probability. It is possible for it to return multiple DAGs. The list has elements;

- state: List of MAP DAGs.
- log_p: Numeric vector with the log posterior probability for each state.
- log_state_score: Numeric vector representing the log score for each state.
- log_norm_state_score: Numeric vector representing the log of the normalised score for each state.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)

# Get the MAP per chain. Can be helpful to compare chains.
GetMAP(results)

# Get MAP across all chains.
results |>
  FlattenChains() |>
  GetMAP()
```

MutilateGraph *Mutilate graph*

Description

Mutilate a graph in accordance with an intervention. This is typically used to perform a do-operation on a given graph. Please note that any evidence set within the original grain object will not be passed to the new object.

Usage

```
MutilateGraph(grain_object, intervention)
```

Arguments

grain_object	A grain object.
intervention	A list of nodes and their corresponding intervention distribution represented as a vector of unconditional probabilities.

Value

A grain object.

Examples

```
# This creates a mutilated graph in accordance with turning the sprinkler
# on in the wet grass example (i.e, do(S = 'yes')).
yn <- c("yes", "no")
p.R <- gRain::cptable(~R, values=c(.2, .8), levels=yn)
p.S_R <- gRain::cptable(~S:R, values=c(.01, .99, .4, .6), levels=yn)
p.G_SR <- gRain::cptable(~G:S:R, values=c(.99, .01, .8, .2, .9, .1, 0, 1), levels=yn)
wet.cpt <- gRain::grain(gRain::compileCPT(p.R, p.S_R, p.G_SR))

mut_graph <- MutilateGraph(wet.cpt, list(S = c(1.0, 0.0)))

# You can then use querygrain to perform an intervention query. For example,
# p(G | do(S = 'yes')) is given by,
gRain::querygrain(mut_graph, 'G')

# You can also perform an observational query for a node not affected
# by the intervention. For example, p(R | do(S = 'yes')) is given by,
gRain::querygrain(mut_graph, 'R')
```

PartitionMCMC *Transition objects. Partition MCMC*

Description

This is a constructor for a single Tempered Partition MCMC step. The function constructs an environment with the proposal, inverse temperature, and verbose flag. It then returns a function that takes the current_state and a scorer object. This only allows the scores to be raised to a constant temperature for every step.

Usage

```
PartitionMCMC(
  proposal = DefaultProposal(),
  temperature = 1,
  prerejection = TRUE,
  verbose = TRUE
)
```

Arguments

proposal	Proposal function. Default is the DefaultProposal.
temperature	Numeric value representing the temperature to raise the score to. Default is 1.
prerejection	Boolean flag to reject due to the proposal disobeying the black or white lists. Only set to FALSE if you want to understand how often you are proposing states that disobey the black or white lists. Can be useful for debugging or understanding the efficiency of specific proposal distributions.
verbose	Flag to pass MCMC information.

Details

One step implementation of the tempered partition MCMC.

Value

Function that takes the current state and scorer that outputs a new state.

Examples

```
dag <- UniformlySampleDAG(c('A', 'B', 'C', 'D', 'E', 'F'))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = bnlearn::learning.test
  )

current_state <- list(
  state = partitioned_nodes,
  log_score = ScoreLabelledPartition(partitioned_nodes, scorer)
  )

pmcmc <- PartitionMCMC(proposal = DefaultProposal(), temperature = 1.0)
pmcmc(current_state, scorer)
```

PartitiontoDAG *Sample DAG from partition*

Description

Samples a DAG in accordance with it's posterior probability conditional on it being consistent with a partition.

Usage

```
PartitiontoDAG(partitions, scorer)
```

Arguments

partitions	A cia_chain(s) object or data.frame representing the partition.
scorer	A scorer object.

Value

A cia_chain(s) object or adjacency matrix. For a cia_chain(s) object each state will be an adjacency matrix.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partition <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

# Used to sample from a single partition.
PartitiontoDAG(partition, scorer)

# Used to convert a chain of partitions to DAGs.
results <- SampleChains(3, partition, PartitionMCMC(), scorer)
PartitiontoDAG(results, scorer)
```

PlotConcordance *Concordance plot*

Description

Plot a concordance plot to compare point-estimates for quantities of interest between chains.

Usage

```
PlotConcordance(x, ...)
```

Arguments

x	A list of adjacency matrices representing edge probabilities, a chains object, or a collections object with states as DAGs.
...	Additional parameter to send to the appropriate method. This includes 'highlight' (defauled to 0.3) which sets the cutoff difference that is used to highlight the points, and the probability edge estimation 'method' for a cia_collections object.

Value

A ggplot object or patchwork of ggplot objects.

Examples

```
data <- bnlearn::learning.test
dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)
scorer <- CreateScorer(scorer = BNLearnScorer, data = data)

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer, n_parallel_chains = 2)
dags <- PartitiontoDAG(results, scorer)

p_edge <- CalculateEdgeProbabilities(dags)
PlotConcordance(p_edge)
```

PlotCumulativeMeanTrace

Plot cumulative mean trace plot.

Description

Plot cumulative mean trace plot.

Usage

```
PlotCumulativeMeanTrace(
  x,
  ncol = NULL,
  nrow = NULL,
  scales = "fixed",
  dir = "v"
)
```

Arguments

x	A posterior predictive sample object.
ncol	Number of columns.
nrow	Number of rows.
scales	Whether the scales should the fixed ('fixed', the default), free ('free') or free in one dimension ('free_x', 'free_y')?
dir	Direction to fill facets. Either 'h' for horizontal or 'v' for vertical.

Value

A ggplot object.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
dag_chains <- PartitiontoDAG(results, scorer)

# Sample the edge probability.
p_edge <- function(dag) { return(as.vector(dag)) }
pedge_sample <- SamplePosteriorPredictiveChains(dag_chains, p_edge)

PlotCumulativeMeanTrace(pedge_sample,
                        nrow = length(data),
                        ncol = length(data))
```

PlotScoreTrace *Plot the score trace*

Description

Plot the score trace

Usage

```
PlotScoreTrace(
  chains,
  attribute = "log_score",
  n_burnin = 0,
  same_plot = TRUE,
  col = NULL,
  ...
)
```

Arguments

chains	MCMC chains.
attribute	Name of attribute to plot. Default is "log_score".
n_burnin	Number of steps to remove as burnin.
same_plot	Whether to plot on the same figure or on multiple figures.

col	A string representing a color for a single chain or a vector of strings to cycle through for multiple chains.
...	Extra parameters to pass to the plot and graphics::line functions.

Value

No return value. Called to produce a base R trace plot.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)

# Plot partition score trace.
PlotScoreTrace(results, type = 'l')

# Plot DAG score trace.
dag_chains <- PartitiontoDAG(results, scorer)
PlotScoreTrace(dag_chains, type = 'l')
```

PostProcessChains *Index chains for further analysis*

Description

This allows you to remove a burnin and thin the chains after processing. This is mostly redundant as you can now index the cia_chain(s) objects directly.

Usage

```
PostProcessChains(chains, n_burnin = 0, n_thin = 1)
```

Arguments

chains	cia_chain(s) object.
n_burnin	Number of steps to remove at the start as a burnin. Default is 0.
n_thin	Number of steps between retained states. Default is 1.

Value

A cia_chain(s) object.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(100, partitioned_nodes, PartitionMCMC(), scorer)
thinned_results <- PostProcessChains(results, n_thin = 2)
```

SampleChains *Sample chains*

Description

Sample chains

Usage

```
SampleChains(
  n_results,
  init_state,
  transition,
  scorer,
  n_thin = 1,
  n_parallel_chains = 2
)
```

Arguments

n_results	Number of saved states per chain.
init_state	An initial state that can be passed to transition. This can be a single state or a list of states for each parallel chain.
transition	A transition function.
scorer	A scorer object.
n_thin	Number of steps between saved states.
n_parallel_chains	Number of chains to run in parallel. Default is 2.

Value

A cia_chains object.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
```

SampleEdgeProbabilities

Sample edge probabilities

Description

Sample edge probabilities

Usage

```
SampleEdgeProbabilities(x)
```

Arguments

x	A chain(s) or collection object where states are DAGs.

Value

p_edge A posterior sample for the marginalised edge probabilities.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )
```

```
results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
dag_chains <- PartitiontoDAG(results, scorer)

pedge_sample <- SampleEdgeProbabilities(dag_chains)
```

SamplePosteriorPredictiveChains
Draw from a posterior predictive distribution

Description

Simulate samples from a posterior predictive distribution for a feature $f(g)$ a graph g.

Usage

```
SamplePosteriorPredictiveChains(x, p_predict, ...)
```

Arguments

x	A cia_chain(s) object.
p_predict	A function that draws from the posterior predictive distribution of interest given an adjacency matrix representing a DAG. The function must be of the form p_predict(dag, ...) and return either a vector of numeric values.
...	Parameters to be passed to p_predict.

Value

A cia_post_chain(s) object.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
dag_chains <- PartitiontoDAG(results, scorer)

# Sample the edge probability.
SamplePosteriorPredictiveChains(dag_chains, function(dag) { return(dag) })
```

ScoreDAG *Score DAG.*

Description

Score DAG.

Usage

```
ScoreDAG(dag, scorer)
```

Arguments

dag	Adjacency matrix of (parent, child) entries with 1 denoting an edge and 0 otherwise.
scorer	Scorer object.

Value

Log of DAG score.

Examples

```
dag <- UniformlySampleDAG(names(bnlearn::asia))
scorer <- CreateScorer(data = bnlearn::asia)
ScoreDAG(dag, scorer)
```

ScoreLabelledPartition *Score labelled partition*

Description

Score labelled partition

Usage

```
ScoreLabelledPartition(partitioned_nodes, scorer)
```

Arguments

partitioned_nodes	Labelled partition.
scorer	Scorer object.

Value

Log of the node score.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(names(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

ScoreLabelledPartition(partitioned_nodes, scorer)
```

---

toBNLearn *Convert to bnlearn object.*

Description

Convert to bnlearn object.

Usage

```
toBNLearn(x)
```

Arguments

x	An object that represents a DAG.

Value

bn_obj A bn object.

Examples

```
adj <- UniformlySampleDAG(c('A', 'B', 'C'))
toBNLearn(adj)
```

togRain *Convert to a gRain object.*

Description

Convert to a gRain object.

Usage

```
togRain(x, ...)
```

Arguments

x	An adjacency matrix or igraph object.
...	extra parameters to gRain compile.

Value

A gRain object.

Examples

```
dag <- bnlearn::model2network("[A][C][F][B|A][D|A:C][E|B:F]")
gRain_obj <- togRain(x = dag |> toMatrix(), data = bnlearn::learning.test)
```

toMatrix *Convert to adjacency matrix.*

Description

Convert a DAG object from other libraries to an adjacency matrix.

Usage

```
toMatrix(network)
```

Arguments

network	A bnlearn or igraph object.

Value

An adjacency matrix representation of network.

Examples

```
toMatrix(bnlearn::empty.graph(LETTERS[1:6]))
toMatrix(igraph::sample_k_regular(10, 2))
```

UniformlySampleDAG *Uniformly sample DAG*

Description

Uniformly sample DAG

Usage

```
UniformlySampleDAG(nodes)
```

Arguments

nodes	A vector of node names.

Value

Adjacency matrix with elements designated as (parent, child).

Examples

```
UniformlySampleDAG(LETTERS[1:3])
```

[.cia_chain *Index a cia_chain object*

Description

Index a cia_chain object

Usage

```
## S3 method for class 'cia_chain'
x = list()[i, ...]
```

Arguments

x	A cia_chain object.
i	An index.
...	ellipsis for extra indexing parameters.

Value

A cia_chain.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
results[[1]][5]
```

[.cia_chains *Index a cia_chains object*

Description

Index a cia_chains object

Usage

```
## S3 method for class 'cia_chains'
x = list()[i, ...]
```

Arguments

x	A cia_chain object.
i	An index to get the cia_chain iterations.
...	ellipsis for extra indexing parameters.

Value

A cia_chains object.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)
```

```
scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
results[5]
```

`[.cia_post_chain` *Indexing with respect to iterations.*

Description

Indexing with respect to iterations.

Usage

```
## S3 method for class 'cia_post_chain'
x = list()[i, ...]
```

Arguments

x	A cia_post_chain object.
i	An index.
...	ellipsis for extra indexing parameters.

Value

chain A cia_post_chain.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
dag_chains <- PartitiontoDAG(results, scorer)

pedge_sample <- SampleEdgeProbabilities(dag_chains)
pedge_sample[5, ]
```

`[.cia_post_chains` *Index a cia_post_chains object with respect to iterations.*

Description

Index a cia_post_chains object with respect to iterations.

Usage

```
## S3 method for class 'cia_post_chains'
x = list()[i, ...]
```

Arguments

`x`	A cia_post_chain object.
`i`	An index to get the cia_post_chain iterations.
`...`	ellipsis for extra indexing parameters.

Value

chain A cia_post_chains object.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
dag_chains <- PartitiontoDAG(results, scorer)

pedge_sample <- SampleEdgeProbabilities(dag_chains)
pedge_sample[5, ]
```

[[.cia_chains *Index a cia_chains object*

Description

Index a cia_chains object

Usage

```
## S3 method for class 'cia_chains'
x[[i, ...]]
```

Arguments

x	A cia_chains object.
i	An index to get the cia_chain.
...	ellipsis for extra indexing parameters.

Value

A cia_chains object.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
results[[1]][1:3]
```

[[.cia_post_chains *Index a cia_post_chains object.*

Description

Index a cia_post_chains object.

Usage

```
## S3 method for class 'cia_post_chains'
x[[i, ...]]
```

Arguments

x	A cia_post_chains object.
i	An index to get the cia_post_chain.
...	ellipsis for extra indexing parameters.

Value

chain A cia_post_chains object.

Examples

```
data <- bnlearn::learning.test

dag <- UniformlySampleDAG(colnames(data))
partitioned_nodes <- DAGtoPartition(dag)

scorer <- CreateScorer(
  scorer = BNLearnScorer,
  data = data
  )

results <- SampleChains(10, partitioned_nodes, PartitionMCMC(), scorer)
dag_chains <- PartitiontoDAG(results, scorer)

pedge_sample <- SampleEdgeProbabilities(dag_chains)
head(pedge_sample[[1]])
```

Index

[.cia_chain, 28
[.cia_chains, 29
[.cia_post_chain, 30
[.cia_post_chains, 31
[[.cia_chains, 32
[[.cia_post_chains, 32

BNLearnScorer, 3

CalculateAcceptanceRates, 3
CalculateEdgeProbabilities, 4
CalculateFeatureMean, 5
CollectUniqueObjects, 6
CreateScorer, 8

DAGtoCPDAG, 9
DAGtoPartition, 9
DefaultProposal, 10

FlattenChains, 11

GetEmptyDAG, 12
GetIncrementalScoringEdges, 12
GetLowestPairwiseScoringEdges, 13
GetMAP, 14

MutilateGraph, 15

PartitionMCMC, 16
PartitiontoDAG, 17
PlotConcordance, 18
PlotCumulativeMeanTrace, 19
PlotScoreTrace, 20
PostProcessChains, 21

SampleChains, 22
SampleEdgeProbabilities, 23
SamplePosteriorPredictiveChains, 24
ScoreDAG, 25
ScoreLabelledPartition, 25

toBNLearn, 26
togRain, 27
toMatrix, 27

UniformlySampleDAG, 28