Package 'ctbn'

May 1, 2014

Type Package

Title R Package for Continuous Time Bayesian Networks

Version 1.0

Date 2014-04-20

Author S. Villa, C. Shelton

Maintainer Riverside Lab for Artificial Intelligence Research, University of California, Riverside. <cshelton@cs.ucr.edu>

Depends R (>= 3.0.0), Rcpp

Description

The ctbn package provides an interface between R and the Continuous Time Bayesian Network Reasoning and Learning Engine (CTBN-RLE) C++ code. The main functionalities of the CTBN-RLE code are designed and implemented in the R environment.

License GPL (>= 3)

R topics documented:

`ctbn-package` *R Package for Continuous Time Bayesian Networks*

Description

The `ctbn` package provides an interface between `R` and the Continuous Time Bayesian Network Reasoning and Learning Engine (CTBN-RLE) C++ code. The main functionalities of the CTBN-RLE code are designed and implemented in the `R` environment.
This interface is built using the `Rcpp` package.

Details

Package: ctbn
Type: Package
Version: 1.0
Date: 2014-04-20
License: GPL (>= 3).

Note

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Author(s)

This project is written by S. Villa in collaboration with Christian Shelton.
The CTBN-RLE project is managed by Christian Shelton and development is done in R-LAIR, Riverside Lab for Artificial Intelligence Research at the University of California, Riverside.

References

Uri Nodelman, Christian R. Shelton, and Daphne Koller (2003). "Learning Continuous Time Bayesian Networks." Proceedings of the Nineteenth International Conference.

Uri Nodelman, Christian R. Shelton, and Daphne Koller (2005). "Expectation Maximization and Complex Duration Distributions for Continuous Time Bayesian Networks." Proceedings of the Twenty-First International Conference on Uncertainty in Artificial Intelligence (pp. 421-430).

Uri Nodelman, Daphne Koller, and Christian R. Shelton (2005). "Expectation Propagation for Continuous Time Bayesian Networks." Proceedings of the Twenty-First International Conference on Uncertainty in Artificial Intelligence (pp. 431-440).

Brenda Ng, Avi Pfeffer, and Richard Dearden (2005). "Continuous Time Particle Filtering." Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (pp. 1360-1365).

Nir Friedman and Rza Kupferman (2006). "Dimension Reduction in Singularly Perturbed Continuous-Time Bayesian Networks." Proceedings of the Twenty-Second International Conference on Uncertainty in Artificial Intelligence.

Suchi Saria, Uri Nodelman, and Daphne Koller (2007). "Reasoning at the Right Time Granularity." Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence" (pp. 421-430).

Yu Fan and Christian R. Shelton (2008). "Sampling for Approximate Inference in Continuous Time Bayesian Networks." Proceedings of the Tenth International Symposium on Artificial Intelligence and Mathematics.

Tal El-Hay, Nir Friedman, and Raz Kupferman (2008). "Gibbs Sampling in Factorized Continuous-Time Markov Processes." Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (pp. 169-178).

Ido Cohn, Tal El-Hay, Raz Kupferman, and Nir Friedman (2009). "Mean Field Variational Approximation for Continuous-Time Bayesian Networks." Proceedings of the Twenty-Fifth International Conference on Uncertainty in Artificial Intelligence.

Tal El-Hay, Ido Cohn, Nir Friedman, Raz Kupferman (2010). "Continuous-Time Belief Propagation." Proceedings of the Twenty-Seventh International Conference on Machine Learning.

Vinayak Rao, Yee Whye Teh (2011). "Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks." Proceedings of the Twenty-Seventh International Conference on Uncertainty in Artificial Intelligence.

E. Busra Celikkaya, Christian R. Shelton, and William Lam (2011). "Factored Filtering of Continuous-Time Systems." Proceedings of the Twenty-Seventh International Conference on Uncertainty in Artificial Intelligence.

`CloneCtbn` *CloneCtbn*

Description

Clone a continuous-time Bayesian network (CTBN) object.

Usage

```
CloneCtbn (xpCtbn)
```

Arguments

`xpCtbn`	external pointer to the CTBN C++ model.

Value

External pointer to the CTBN C++ model. `NULL` in the case of errors.

Note

You own the external pointer, so remember to delete it using the DeleteCtbn function only.

See Also

DeleteCtbn, NewCtbn

Examples

```
# create ctbn
vars    <- data.frame(Name=c("X1","X2"), Value=c(2,3), check.names=FALSE)
xpCtbn  <- NewCtbn(vars)

# clone ctbn
xpCtbn2 <- CloneCtbn(xpCtbn)

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
xpCtbn2 <- DeleteCtbn(xpCtbn2)
garbage <- gc()
```

DeleteCtbn *DeleteCtbn*

Description

Delete a continuous-time Bayesian network (CTBN) object.

Usage

```
DeleteCtbn (xpCtbn)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.

Value

NULL.

See Also

NewCtbn, CloneCtbn

Examples

```
# create ctbn
vars    <- data.frame(Name=c("X1","X2"), Value=c(2,3), check.names=FALSE)
xpCtbn  <- NewCtbn(vars)

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

DeleteInfEngine *DeleteInfEngine*

Description

Delete an inference engine object.

Usage

```
DeleteInfEngine (engine)
```

Arguments

engine	external pointer to the inference engine C++ object.

Value

NULL.

See Also

NewInfEngine, GetInfEngineType

Examples

```
# create engine
engine  <- NewInfEngine(inf.type="importance", num.samples=1000)

# delete engine
engine  <- DeleteInfEngine(engine)
garbage <- gc()
```

GetBnCPTs *GetBnCPTs*

Description

Get the definition of the initial Bayesian network (conditional probability tables) of the CTBN object.

Usage

```
GetBnCPTs (xpCtbn)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.

Value

List. List (I) of lists (II) of dataframes (III), in which the I level contains the nodes Xn, the second level contains the parents of Xn (Xn|U), and the III level contains the values of Xn given a particular instantiation of its parents (Xn|u) specified as a dataframe. This dataframe is 1 x m, where m is number of values that Xn can assume (specified in the column header that must be from 0 to value-1), and the row contains the respective probabilities (that must sum up to 1). The names contained in the I level list must be Xn and the names of the II level list must be in the form Xn$U=u. `NULL` in the case of errors.

See Also

`SetBnCPTs`

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

bn.cpts   <- GetBnCPTs(xpCtbn)

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

`GetBnStruct` *GetBnStruct*

Description

Get the structure of the initial Bayesian network of the CTBN object.

Usage

```
GetBnStruct (xpCtbn)
```

Arguments

`xpCtbn`	external pointer to the CTBN C++ model.

Value

Dataframe k x 2. It is used to define the structure of the initial Bayesian network (specified as a directed acyclic graph). Each row consists of the name of the parent of Xn (from) and the name of Xn (to). Both names are contained in the variables definition of the CTBN model (case sensitive). `NULL` in the case of errors.

See Also

`SetBnStruct`

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

bn.str    <- GetBnStruct(xpCtbn)

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

GetCtbnJointDyn *GetCtbnJointDyn*

Description

Get the joint intensity matrix for the entire CTBN object.

Usage

```
GetCtbnJointDyn (xpCtbn)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.

Value

Matrix. Joint intensity matrix for the entire CTBN model (not recommended if your CTBN is large).

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

jointIntsMat <- GetCtbnJointDyn(xpCtbn)

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

GetCtbnVars *GetCtbnVars*

Description

Get the variables definition of the CTBN object.

Usage

```
GetCtbnVars (xpCtbn)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.

Value

Dataframe n x 2. It is used to define the variables of the CTBN model. Each row consists of the name (string) of n variables Xn and a value (integer) corresponding of the number of values that the variable can assume (defined from 0 to value-1, -1 if unknown). NULL in the case of errors.

See Also

NewCtbn

Examples

```
# create ctbn
vars   <- data.frame(Name=c("X1","X2"), Value=c(2,3), check.names=FALSE)
xpCtbn <- NewCtbn(vars)

vars2  <- GetCtbnVars(xpCtbn)

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

GetDynIntMats *GetDynIntMats*

Description

Get the definition of the dynamic component (conditional intensity matrices) of the CTBN object.

Usage

```
GetDynIntMats (xpCtbn)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.

Value

List. List (I) of lists (II) of dataframes (III), in which the I level contains the nodes Xn, the second level contains the parents of Xn (Xn|U), and the III level contains the values of Xn given a particular instantiation of its parents (Xn|u) specified as a dataframe. This dataframe is m x m, where m is number of values that Xn can assume (specified in the column header that must be from 0 to value-1) and it contains the conditional intensity probabilities (each row must sum up to 0). The names contained in the I level list must be Xn and the names of the II level list must be in the form Xn$U=u. NULL in the case of errors.

See Also

SetDynIntMats

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

dyn.cims  <- GetDynIntMats(xpCtbn)

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

GetDynStruct *GetDynStruct*

Description

Get the structure of the dynamic component of the CTBN object.

Usage

```
GetDynStruct (xpCtbn)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.

Value

Dataframe k x 2. It is used to define the structure of the dynamic structure (possibly cycle). Each row consists of the name of the parent of Xn (from) and the name of Xn (to). Both names are contained in the variables definition of the CTBN model (case sensitive). NULL in the case of errors.

See Also

SetDynStruct

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

dyn.str   <- GetDynStruct(xpCtbn)

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

GetInfEngineType *GetInfEngineType*

Description

Get the type of the inference engine C++ object.

Usage

```
GetInfEngineType (engine)
```

Arguments

engine	external pointer to the inference engine C++ object.

Value

Character. Inference engine type.

See Also

NewInfEngine, DeleteInfEngine

Examples

```
# create engine
engine <- NewInfEngine(inf.type="importance", num.samples=1000)

print(GetInfEngineType(engine))

# delete engine
engine  <- DeleteInfEngine(engine)
garbage <- gc()
```

`LearnCtbnParams` *LearnCtbnParams*

Description

Learn the parameters of the CTBN object (both static and dynamic components) given fully or partially observed trajectories. It is possible to specify an inference engine used to perform learning in the missing data case.

Usage

```
LearnCtbnParams (xpCtbn, trjs, inf.type="exact", num.samples=100,
                 burn.iters=100, eps=1e-5, inf.engine=NULL)
```

Arguments

`xpCtbn`	external pointer to the CTBN C++ model.
`trjs`	list. List of dataframes in which each dataframe represents a trajectory.
`inf.type`	character. Inference type: `exact`, `importance`, `gibbs`, `gibbsaux`, `meanfield`.
`num.samples`	integer. Number of samples for `importance`, `gibbs`, `gibbsaux` inference engines.
`burn.iters`	integer. Number of burn-in iterations for `gibbs`, `gibbsaux` inference engines.
`eps`	numeric. Epsilon value for `meanfield` inference engine.
`inf.engine`	external pointer to the inference engine C++ model.

Value

Character. Status of the learning process. `NULL` in the case of errors.

See Also

`LearnCtbnStruct`, `SampleFullTrjs`, `SamplePartialTrjs`, `GetBnCPTs`, `GetDynIntMats`.

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

# parameters learning from fully observed data
samples   <- SampleFullTrjs(xpCtbn, num=100)
xpCtbn2   <- CloneCtbn(xpCtbn)
LearnCtbnParams(xpCtbn2, samples)
cptsOut2  <- GetBnCPTs(xpCtbn2)
cimsOut2  <- GetDynIntMats(xpCtbn2)
xpCtbn2   <- DeleteCtbn(xpCtbn2)
garbage   <- gc()

# parameters learning from partially observed data
samples   <- SamplePartialTrjs(xpCtbn, num=10, rem.frac=0.01)
```

```
xpCtbn2   <- CloneCtbn(xpCtbn)
inf       <- "importance";
LearnCtbnParams(xpCtbn2, samples, inf.type=inf, num.samples=100)
cptsOut2  <- GetBnCPTs(xpCtbn2)
cimsOut2  <- GetDynIntMats(xpCtbn2)
xpCtbn2   <- DeleteCtbn(xpCtbn2)
garbage   <- gc()

# delete ctbn
xpCtbn    <- DeleteCtbn(xpCtbn)
garbage   <- gc()
```

`LearnCtbnStruct` *LearnCtbnStruct*

Description

Learn the structure (and the relative parameters) of the CTBN object (both static and dynamic components) given fully or partially observed trajectories. It is possible to specify an inference engine used to perform learning in the missing data case.

Usage

```
LearnCtbnStruct (xpCtbn, trjs, alpha=1.0, tau=0.1, inf.type="exact",
                 num.samples=100, burn.iters=100, eps=1e-5, inf.engine=NULL)
```

Arguments

`xpCtbn`	external pointer to the CTBN C++ model.
`trjs`	list. List of dataframes in which each dataframe represents a trajectory.
`alpha`	numeric. Hyperparameter: pseudo-counts of the number of transitions from one state to another.
`tau`	numeric. Hyperparameter: imaginary amount of time spent in each state.
`inf.type`	character. Inference type: `exact`, `importance`, `gibbs`, `gibbsaux`, `meanfield`.
`num.samples`	integer. Number of samples for `importance`, `gibbs`, `gibbsaux` inference engines.
`burn.iters`	integer. Number of burn-in iterations for `gibbs`, `gibbsaux` inference engines.
`eps`	numeric. Epsilon value for `meanfield` inference engine.
`inf.engine`	external pointer to the inference engine C++ model.

Value

Character. Status of the learning process. `NULL` in the case of errors.

See Also

`LearnCtbnParams`, `SampleFullTrjs`, `SamplePartialTrjs`, `GetBnStruct`, `GetDynStruct`, `GetBnCPTs`, `GetDynIntMats`.

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

# structural learning from fully observed data
samples   <- SampleFullTrjs(xpCtbn, num=100)
xpCtbn2   <- NewCtbn(GetCtbnVars(xpCtbn))
LearnCtbnStruct(xpCtbn2, samples)
dynStr2   <- GetDynStruct(xpCtbn2)
stcStr2   <- GetBnStruct(xpCtbn2)
cimsOut2  <- GetDynIntMats(xpCtbn2)
cptsOut2  <- GetBnCPTs(xpCtbn2)
xpCtbn2   <- DeleteCtbn(xpCtbn2)
garbage   <- gc()

# parameters learning from partially observed data
samples   <- SamplePartialTrjs(xpCtbn, num=10, rem.frac=0.01)
xpCtbn2   <- CloneCtbn(xpCtbn)
inf       <- "importance";
LearnCtbnStruct(xpCtbn2, samples, inf.type=inf, num.samples=100)
dynStr2   <- GetDynStruct(xpCtbn2)
stcStr2   <- GetBnStruct(xpCtbn2)
cimsOut2  <- GetDynIntMats(xpCtbn2)
cptsOut2  <- GetBnCPTs(xpCtbn2)
xpCtbn2   <- DeleteCtbn(xpCtbn2)
garbage   <- gc()

# delete ctbn
xpCtbn    <- DeleteCtbn(xpCtbn)
garbage   <- gc()
```

`LoadRCtbn` *LoadRCtbn*

Description

Load a new CTBN object saved in a rctbn format file.

Usage

```
LoadRCtbn (pathfile)
```

Arguments

`pathfile`	character. File to be read.

Value

External pointer to the CTBN C++ model. `NULL` in the case of errors.

Note

You own the external pointer, so remember to delete it using the DeleteCtbn function only.

See Also

DeleteCtbn, SaveRCtbn.

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

# delete ctbn
xpCtbn    <- DeleteCtbn(xpCtbn)
garbage   <- gc()
```

NewCtbn *NewCtbn*

Description

Creation of a new continuous-time Bayesian network (CTBN) object calling the respective C++ function.

Usage

```
NewCtbn (vars, bn.str=NULL, bn.cpts=NULL, dyn.str=NULL, dyn.cims=NULL)
```

Arguments

vars	dataframe n x 2. It is used to define the variables of the CTBN model. Each row consists of the name (string) of n variables Xn and a value (integer) corresponding of the number of values that the variable can assume (defined from 0 to value-1, -1 if unknown).
bn.str	dataframe k x 2. It is used to define the structure of the initial Bayesian network (specified as a directed acyclic graph). Each row consists of the name of the parent of Xn (from) and the name of Xn (to). Both names must be contained in the vars dataframe (case sensitive).
bn.cpts	list. List (I) of lists (II) of dataframes (III), in which the I level contains the nodes Xn, the second level contains the parents of Xn (Xn|U), and the III level contains the values of Xn given a particular instantiation of its parents (Xn|u) specified as a dataframe. This dataframe is 1 x m, where m is number of values that Xn can assume (specified in the column header that must be from 0 to value-1), and the row contains the respective probabilities (that must sum up to 1). The names contained in the I level list must be Xn and the names of the II level list must be in the form Xn$U=u.
dyn.str	dataframe k x 2. It is used to define the structure of the dynamic structure (possibly cycle). Each row consists of the name of the parent of Xn (from) and the name of Xn (to). Both names must be contained in the vars dataframe (case sensitive).

dyn.cims — list. List (I) of lists (II) of dataframes (III), in which the I level contains the nodes Xn, the second level contains the parents of Xn (Xn|U), and the III level contains the values of Xn given a particular instantiation of its parents (Xn|u) specified as a dataframe. This dataframe is m x m, where m is number of values that Xn can assume (specified in the column header that must be from 0 to value-1) and it contains the conditional intensity probabilities (each row must sum up to 0). The names contained in the I level list must be Xn and the names of the II level list must be in the form Xn$U=u.

Value

External pointer to the CTBN C++ model. NULL in the case of errors.

Note

You own the external pointer, so remember to delete it using the DeleteCtbn function only.

See Also

DeleteCtbn, CloneCtbn

Examples

```
# files
vars.file     <- system.file("DrugNetwork", "DrugNetwork_Vars.txt",       package="ctbn")
bn.str.file   <- system.file("DrugNetwork", "DrugNetwork_BN_Struct.txt",  package="ctbn")
bn.cpts.file  <- system.file("DrugNetwork", "DrugNetwork_BN_CPTs.txt",    package="ctbn")
dyn.str.file  <- system.file("DrugNetwork", "DrugNetwork_Dyn_Struct.txt", package="ctbn")
dyn.cims.file <- system.file("DrugNetwork", "DrugNetwork_Dyn_CIMs.txt",   package="ctbn")

# variables
vars     <- read.table(    vars.file ,   sep=",", header=TRUE)
# static part
bn.str   <- read.table(    bn.str.file,  sep=",", header=TRUE)
bn.cpts  <- ReadProbStruct(bn.cpts.file)
# dynamic part
dyn.str  <- read.table(    dyn.str.file, sep=",", header=TRUE)
dyn.cims <- ReadProbStruct(dyn.cims.file)

# default constructor
xpCtbn   <- NewCtbn(vars,bn.str=bn.str,bn.cpts=bn.cpts,dyn.str=dyn.str,dyn.cims=dyn.cims)

# delete ctbn
xpCtbn    <- DeleteCtbn(xpCtbn)
garbage   <- gc()
```

NewInfEngine *NewInfEngine*

Description

Creation of a new inference engine object.

Usage

```
NewInfEngine (inf.type="exact", hold=FALSE,
              num.samples=100, burn.iters=100, eps=1e-5)
```

Arguments

inf.type	character. Inference type: exact, importance, gibbs, gibbsaux, meanfield.
hold	logical. If you are using the same model and the same data, you can preserve the past inference computations using hold=TRUE.
num.samples	integer. Number of samples for importance, gibbs, gibbsaux inference engines.
burn.iters	integer. Number of burn-in iterations for gibbs, gibbsaux inference engines.
eps	numeric. Epsilon value for meanfield inference engine.

Value

External pointer to the inference engine C++ model. NULL in the case of errors.

Note

You own the external pointer, so remember to delete it using the DeleteInfEngine function only.

See Also

DeleteInfEngine, GetInfEngineType

Examples

```
# load ctbn and trajectory
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
trj.file  <- system.file("DrugNetwork", "Trajectory.csv",    package="ctbn")
trj       <- read.table(trj.file, sep=",", header=TRUE, check.names=FALSE)
xpCtbn    <- LoadRCtbn(ctbn.file)

# advanced option: use an inference engine object for queries
var1 <- "Concentration"
var2 <- "Hungry"
sample.part        <- trj
sample.part[,var1] <- -1
sample.part[,var2] <- -1

engine  <- NewInfEngine(inf.type="importance", num.samples=1000)
print(GetInfEngineType(engine))
QueryCtbnStats(xpCtbn, sample.part, c("Hungry"), inf.engine=engine)
engine  <- DeleteInfEngine(engine)
garbage <- gc()

# if you are using the same model and the same data,
# you can preserve the past inference computations using the hold flag
engine  <- NewInfEngine(inf.type="importance", num.samples=1000, hold=TRUE)
print(GetInfEngineType(engine))
QueryCtbnStats(xpCtbn, sample.part, c("Hungry"), inf.engine=engine)
QueryCtbnStats(xpCtbn, sample.part, c("Concentration"), inf.engine=engine)
```

```
print(GetInfEngineType(engine))
engine  <- DeleteInfEngine(engine)
garbage <- gc()

# this reset the past computations (the model has been changed)
xpCtbn2 <- CloneCtbn(xpCtbn)
QueryCtbnStats(xpCtbn2, sample.part, c("Hungry"), inf.engine=engine)
engine  <- DeleteInfEngine(engine)
xpCtbn2 <- DeleteCtbn(xpCtbn2)
garbage <- gc()

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

`QueryCtbnFilter` *QueryCtbnFilter*

Description

Perform the filtering query (i.e. the probability of the state x at time t given the trajectory up to time t).

Usage

```
QueryCtbnFilter (xpCtbn, evid, insts, inf.type="exact",
                 num.samples=100, burn.iters=100, eps=1e-5, inf.engine=NULL)
```

Arguments

`xpCtbn`	external pointer to the CTBN C++ model.
`evid`	dataframe. Evidence specified as a trajectory.
`insts`	dataframe k x n+1. Dataframe of k instantiations in which the first column represents the time t.
`inf.type`	character. Inference type: `exact`, `importance`, `gibbs`, `gibbsaux`, `meanfield`.
`num.samples`	integer. Number of samples for `importance`, `gibbs`, `gibbsaux` inference engines.
`burn.iters`	integer. Number of burn-in iterations for `gibbs`, `gibbsaux` inference engines.
`eps`	numeric. Epsilon value for `meanfield` inference engine.
`inf.engine`	external pointer to the inference engine C++ model.

Value

Dataframe k x 2. Each row contains the time and the probability according to the `insts` dataframe. `NULL` in the case of errors.

See Also

`QueryCtbnStats`, `QueryCtbnTime`, `QueryCtbnSmooth`.

Examples

```
# load ctbn and trajectory
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
trj.file  <- system.file("DrugNetwork", "Trajectory.csv",    package="ctbn")
trj       <- read.table(trj.file, sep=",", header=TRUE, check.names=FALSE)
xpCtbn    <- LoadRCtbn(ctbn.file)

# sample.part is a partial trajectory (Concentration and Hungry are not observed at some time)
# insts dataframe of times and instantiations (Concentration=1, Hungry=0)
var1 <- "Concentration"
var2 <- "Hungry"
val1 <- 1
val2 <- 0
sample.part <- trj
sample.part[,var1] <- -1
sample.part[,var2] <- -1
time  <- seq(0,10,0.01)
vals1 <- rep(val1,length(time))
vals2 <- rep(val2,length(time))
insts <- data.frame(time,vals1,vals2)
names(insts) <- c("Time",var1,var2)

# output: dataframe (time, probability)
x <- QueryCtbnFilter(xpCtbn,sample.part,insts)

# plot output
tlt <- paste (paste(var1,val1,sep="="),paste(var2,val2,sep="="),sep=", ")
plot(x,main=tlt,xlab="Time",ylab="Probability",type="l")

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

`QueryCtbnSmooth` *QueryCtbnSmooth*

Description

Perform the smoothing query (i.e. the probability of the state x at time t given the whole trajectory).

Usage

```
QueryCtbnSmooth (xpCtbn, evid, insts, inf.type="exact",
                 num.samples=100, burn.iters=100, eps=1e-5, inf.engine=NULL)
```

Arguments

`xpCtbn`	external pointer to the CTBN C++ model.
`evid`	dataframe. Evidence specified as a trajectory.
`insts`	dataframe k x n+1. Dataframe of k instantiations in which the first column represents the time t.

inf.type	character. Inference type: exact, importance, gibbs, gibbsaux, meanfield.
num.samples	integer. Number of samples for importance, gibbs, gibbsaux inference engines.
burn.iters	integer. Number of burn-in iterations for gibbs, gibbsaux inference engines.
eps	numeric. Epsilon value for meanfield inference engine.
inf.engine	external pointer to the inference engine C++ model.

Value

Dataframe k x 2. Each row contains the time and the probability according to the insts dataframe. NULL in the case of errors.

See Also

QueryCtbnStats, QueryCtbnTime, QueryCtbnFilter.

Examples

```
# load ctbn and trajectory
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
trj.file  <- system.file("DrugNetwork", "Trajectory.csv",    package="ctbn")
trj       <- read.table(trj.file, sep=",", header=TRUE, check.names=FALSE)
xpCtbn    <- LoadRCtbn(ctbn.file)

# sample.part is a partial trajectory (Concentration and Hungry are not observed at some time)
# insts dataframe of times and instantiations (Concentration=1, Hungry=0)
var1 <- "Concentration"
var2 <- "Hungry"
val1 <- 1
val2 <- 0
sample.part <- trj
sample.part[,var1] <- -1
sample.part[,var2] <- -1
time  <- seq(0,10,0.01)
vals1 <- rep(val1,length(time))
vals2 <- rep(val2,length(time))
insts <- data.frame(time,vals1,vals2)
names(insts) <- c("Time",var1,var2)

# output: dataframe (time, probability)
x <- QueryCtbnSmooth(xpCtbn,sample.part,insts)

# plot output
tlt <- paste (paste(var1,val1,sep="="),paste(var2,val2,sep="="),sep=", ")
plot(x,main=tlt,xlab="Time",ylab="Probability",type="l")

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

QueryCtbnStats *QueryCtbnStats*

Description

Perform the query to the CTBN model about the expected time spent in a state and the expected number of transitions from one state to another state of the given variables.

Usage

```
QueryCtbnStats (xpCtbn, evid, vars, inf.type="exact",
                num.samples=100, burn.iters=100, eps=1e-5, inf.engine=NULL)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.
evid	dataframe. Evidence specified as a trajectory.
vars	vector of characters. This vector contains the n variables names to perform the query.
inf.type	character. Inference type: exact, importance, gibbs, gibbsaux, meanfield.
num.samples	integer. Number of samples for importance, gibbs, gibbsaux inference engines.
burn.iters	integer. Number of burn-in iterations for gibbs, gibbsaux inference engines.
eps	numeric. Epsilon value for meanfield inference engine.
inf.engine	external pointer to the inference engine C++ model.

Value

List of n dataframes. Each dataframe contains the expected time spent in a state (diagonal elements) and the expected number of transitions from one state to another state (off-diagonal elements). Each state is specified in the row (from) and column (to) headers. NULL in the case of errors.

See Also

QueryCtbnTime, QueryCtbnFilter, QueryCtbnSmooth.

Examples

```
# load ctbn and trajectory
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
trj.file  <- system.file("DrugNetwork", "Trajectory.csv",    package="ctbn")
trj       <- read.table(trj.file, sep=",", header=TRUE, check.names=FALSE)
xpCtbn    <- LoadRCtbn(ctbn.file)

# sample.part is a partial trajectory (Eating and Drowsy are not observed at some time)
# v1 vector of variable  Eating
# v2 vector of variables Eating and Drowsy
var1 <- "Eating"
var2 <- "Drowsy"
```

```
sample.part <- trj
sample.part[5,var1]=-1
sample.part[6,var2]=-1
v1 <- c(var1)
v2 <- c(var1,var2)

print(QueryCtbnStats(xpCtbn, sample.part, v1))
print(QueryCtbnStats(xpCtbn, sample.part, v2))

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

QueryCtbnTime *QueryCtbnTime*

Description

Perform the query to the CTBN model about the expected time that the process stays in some states.

Usage

```
QueryCtbnTime (xpCtbn, evid, insts, inf.type="exact",
               num.samples=100, burn.iters=100, eps=1e-5, inf.engine=NULL)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.
evid	dataframe. Evidence specified as a trajectory.
insts	dataframe k x n. Dataframe of k instantiations.
inf.type	character. Inference type: exact, importance, gibbs, gibbsaux, meanfield.
num.samples	integer. Number of samples for importance, gibbs, gibbsaux inference engines.
burn.iters	integer. Number of burn-in iterations for gibbs, gibbsaux inference engines.
eps	numeric. Epsilon value for meanfield inference engine.
inf.engine	external pointer to the inference engine C++ model.

Value

Dataframe k x 1. Each row contains the expected time that the process stays in the specified state. NULL in the case of errors.

See Also

QueryCtbnStats, QueryCtbnFilter, QueryCtbnSmooth.

Examples

```
# load ctbn and trajectory
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
trj.file  <- system.file("DrugNetwork", "Trajectory.csv",    package="ctbn")
trj       <- read.table(trj.file, sep=",", header=TRUE, check.names=FALSE)
xpCtbn    <- LoadRCtbn(ctbn.file)

# sample.part is a partial trajectory (Eating and Drowsy are not observed at some time)
# insts dataframe of instantiations
var1 <- "Eating"
var2 <- "Drowsy"
sample.part <- trj
sample.part[5,var1]=-1
sample.part[6,var2]=-1
insts <- trj[7:8,c(3:4)]
print(insts)

print(QueryCtbnTime(xpCtbn,sample.part,insts))

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

`ReadProbStruct` *ReadProbStruct*

Description

Read the definition of both static and dynamic parts of the CTBN object from a textual file.

Usage

```
ReadProbStruct (pathfile, s.tag=NULL, e.tag=NULL)
```

Arguments

`pathfile`	character. File to be read.
`s.tag`	character. Initial tag, `NULL` from the start of the file.
`e.tag`	character. End tag, `NULL` to the end of the file.

Value

List. List (I) of lists (II) of dataframes (III), in which the I level contains the nodes Xn, the second level contains the parents of Xn (Xn|U), and the III level contains the values of Xn given a particular instantiation of its parents (Xn|u) specified as a dataframe. This dataframe can be 1 x m (bn.cpts) or m x m (dyn.cims), where m is number of values that Xn can assume (specified in the column header that must be from 0 to value-1). The names contained in the I level list must be Xn and the names of the II level list must be in the form Xn$U=u.

Note

The file format should be defined as: a row that identifies the variable Xn given a particular instantiation of its parents (Xn|u) and the definition of the respective probabilities in terms of a vector (`Vec`) or a matrix (`Mat`). See the following two examples.

An excerpt of the definition file of the static component:

```
 Eating$Hungry=0
Vec,0,1
"0",0.1,0.9
Concentration$Full stomach=0,Uptake=0
Vec,0,1,2
"0",0.7,0.2,0.1
```

An excerpt of the definition file of the dynamic component:

```
 Eating$Hungry=0
Mat,0,1
"0",-0.01,0.01
"1",10.0,-10.0
Concentration$Full stomach=0,Uptake=0
Mat,0,1,2
"0",-0.02,0.01,0.01
"1",0.25,-0.26,0.01
"2",0.01,0.5,-0.51
```

See Also

`WriteProbStruct`

Examples

```
# cpts and cims files
bn.cpts.file  <- system.file("DrugNetwork", "DrugNetwork_BN_CPTs.txt",  package="ctbn")
dyn.cims.file <- system.file("DrugNetwork", "DrugNetwork_Dyn_CIMs.txt", package="ctbn")

bn.cpts  <- ReadProbStruct(bn.cpts.file)

dyn.cims <- ReadProbStruct(dyn.cims.file)
```

`SampleFullTrjs` *SampleFullTrjs*

Description

Sample fully observed trajectories from a given CTBN object.

Usage

```
SampleFullTrjs (xpCtbn, t.begin=0.0, t.end=10.0, num=10)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.
t.begin	numeric. Begin time of the trajectory.
t.end	numeric. End time of the trajectory.
num	integer. Number of trajectories.

Value

List. List of length num of dataframes k x n+1. Each dataframe represents a trajectory in which the columns names are the time and the n variables of the CTBN model and the k rows (that can be different from each trajectory) contain the the temporal evolution of the variables. Recall that in the CTBN model, only one single variable can change at any single instant, so each row of the dataframe contains at most one state transition of a variable. The last row contains all -1 (unknown state) because the variables are observed in the [t1;t2),[t2;t3), ..., [tk-1,tk) time intervals. NULL in the case of errors.

See Also

SamplePartialTrjs

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

samplesFull <- SampleFullTrjs(xpCtbn, num=100)

sampleFull  <- samplesFull[[1]]

plot(sampleFull[1:nrow(sampleFull)-1,c(1,4)],type="s",main="Full Trajectory")

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

SamplePartialTrjs *SamplePartialTrjs*

Description

Sample partially observed trajectories from a given CTBN object.

Usage

```
SamplePartialTrjs (xpCtbn, t.begin=0.0, t.end=10.0, num=10,
                   rem.frac=0.1, rem.ite=1)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.
t.begin	numeric. Begin time of the trajectory.
t.end	numeric. End time of the trajectory.
num	integer. Number of trajectories.
rem.frac	numeric. Fraction of information removed in each iteration.
rem.ite	integer. Number of iterations.

Value

List. List of length num of dataframes k x n+1. Each dataframe represents a trajectory in which the columns names are the time and the n variables of the CTBN model and the k rows (that can be different from each trajectory) contain the the temporal evolution of the variables. Recall that in the CTBN model, only one single variable can change at any single instant, so each row of the dataframe contains at most one state transition of a variable. Some information is randomly removed (-1 values) from each trajectory according to the given parameters. The last row contains all -1 (unknown state) because the variables are observed in the [t1;t2),[t2;t3), ..., [tk-1,tk) time intervals. NULL in the case of errors.

See Also

SampleFullTrjs

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

samplesPart <- SamplePartialTrjs(xpCtbn, num=100, rem.frac=0.01)

samplePart  <- samplesPart[[1]]

plot(samplePart[1:nrow(samplePart)-1,c(1,4)],type="s",main="Partial Trajectory")

# delete ctbn
xpCtbn  <- DeleteCtbn(xpCtbn)
garbage <- gc()
```

SaveRCtbn *SaveRCtbn*

Description

Save the CTBN object in a rctbn format file.

Usage

```
SaveRCtbn (xpCtbn, pathfile)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.
pathfile	character. File to save the CTBN C++ model.

See Also

LoadRCtbn

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)

SaveRCtbn(xpCtbn, "myDrugNetwork.rctbn")

# delete ctbn
xpCtbn    <- DeleteCtbn(xpCtbn)
garbage   <- gc()
```

SetBnCPTs *SetBnCPTs*

Description

Set the definition of the initial Bayesian network (conditional probability tables) of the CTBN object.

Usage

```
SetBnCPTs (xpCtbn, bn.cpts)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.
bn.cpts	list. List (I) of lists (II) of dataframes (III), in which the I level contains the nodes Xn, the second level contains the parents of Xn (Xn|U), and the III level contains the values of Xn given a particular instantiation of its parents (Xn|u) specified as a dataframe. This dataframe is 1 x m, where m is number of values that Xn can assume (specified in the column header that must be from 0 to value-1), and the row contains the respective probabilities (that must sum up to 1). The names contained in the I level list must be Xn and the names of the II level list must be in the form Xn$U=u.

Value

0 if it is set correctly, NULL in the case of errors.

See Also

SetBnCPTs, ReadProbStruct

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)
bn.cpts   <- GetBnCPTs(xpCtbn)

SetBnCPTs(xpCtbn, bn.cpts)

# delete ctbn
xpCtbn    <- DeleteCtbn(xpCtbn)
garbage   <- gc()
```

`SetBnStruct` *SetBnStruct*

Description

Set the structure of the initial Bayesian network of the CTBN object.

Usage

```
SetBnStruct (xpCtbn, bn.str)
```

Arguments

`xpCtbn`	external pointer to the CTBN C++ model.
`bn.str`	dataframe k x 2. It is used to define the structure of the initial Bayesian network (specified as a directed acyclic graph). Each row consists of the name of the parent of Xn (from) and the name of Xn (to). Both names must be contained in the variables definition of the CTBN model (case sensitive).

Value

`0` if it is set correctly, `NULL` in the case of errors.

See Also

`GetBnStruct`

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)
bn.str    <- GetBnStruct(xpCtbn)

SetBnStruct(xpCtbn, bn.str)

# delete ctbn
xpCtbn    <- DeleteCtbn(xpCtbn)
garbage   <- gc()
```

SetDynIntMats *SetDynIntMats*

Description

Set the definition of the dynamic component (conditional intensity matrices) of the CTBN object.

Usage

```
SetDynIntMats (xpCtbn, dyn.cims)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.
dyn.cims	list. List (I) of lists (II) of dataframes (III), in which the I level contains the nodes Xn, the second level contains the parents of Xn (Xn|U), and the III level contains the values of Xn given a particular instantiation of its parents (Xn|u) specified as a dataframe. This dataframe is m x m, where m is number of values that Xn can assume (specified in the column header that must be from 0 to value-1) and it contains the conditional intensity probabilities (each row must sum up to 0). The names contained in the I level list must be Xn and the names of the II level list must be in the form Xn$U=u.

Value

0 if it is set correctly, NULL in the case of errors.

See Also

GetDynIntMats, ReadProbStruct

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)
dyn.cims  <- GetDynIntMats(xpCtbn)

SetDynIntMats (xpCtbn, dyn.cims)

# delete ctbn
xpCtbn    <- DeleteCtbn(xpCtbn)
garbage   <- gc()
```

SetDynStruct *SetDynStruct*

Description

Set the structure of the dynamic component of the CTBN object.

Usage

```
SetDynStruct (xpCtbn, dyn.str)
```

Arguments

xpCtbn	external pointer to the CTBN C++ model.
dyn.str	dataframe k x 2. It is used to define the structure of the dynamic structure (possibly cycle). Each row consists of the name of the parent of Xn (from) and the name of Xn (to). Both names must be contained in the variables definition of the CTBN model (case sensitive).

Value

0 if it is set correctly, NULL in the case of errors.

See Also

GetDynStruct

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)
dyn.str   <- GetDynStruct(xpCtbn)

SetDynStruct(xpCtbn, dyn.str)

# delete ctbn
xpCtbn    <- DeleteCtbn(xpCtbn)
garbage   <- gc()
```

WriteProbStruct *WriteProbStruct*

Description

Write the definition of both static and dynamic parts of the CTBN object to a textual file.

Usage

```
WriteProbStruct (prob.struct, pathfile, append=FALSE)
```

Arguments

prob.struct	list. List (I) of lists (II) of dataframes (III), in which the I level contains the nodes Xn, the second level contains the parents of Xn (Xn|U), and the III level contains the values of Xn given a particular instantiation of its parents (Xn|u) specified as a dataframe. This dataframe can be 1 x m (bn.cpts) or m x m (dyn.cims), where m is number of values that Xn can assume (specified in the column header that must be from 0 to value-1). The names contained in the I level list must be Xn and the names of the II level list must be in the form Xn$U=u.
pathfile	character. File to write.
append	boolean. Append to pathfile.

See Also

ReadProbStruct, GetBnCPTs, GetDynIntMats

Examples

```
# load ctbn
ctbn.file <- system.file("DrugNetwork", "DrugNetwork.rctbn", package="ctbn")
xpCtbn    <- LoadRCtbn(ctbn.file)
bn.cpts   <- GetBnCPTs(xpCtbn)

WriteProbStruct(bn.cpts, "myDrugNetwork_BN_CPTs.txt")

# delete ctbn
xpCtbn    <- DeleteCtbn(xpCtbn)
garbage   <- gc()
```

Index