Package ‘EDISON’

February 19, 2015

Type Package
Title Network Reconstruction and Changepoint Detection
Version 1.1
Date 2014-13-08
Author Frank Dondelinger, Sophie Lebre
Maintainer Frank Dondelinger <fdondelinger.work@gmail.com>
Description Package EDISON (Estimation of Directed Interactions from Sequences Of Non-homogeneous gene expression) runs an MCMC simulation to reconstruct networks from time series data, using a non-homogeneous, time-varying dynamic Bayesian network. Networks segments and changepoints are inferred concurrently, and information sharing priors provide a reduction of the inference uncertainty.
License GPL-2
LazyLoad yes
Suggests testthat
Depends corpcor, MASS
Repository CRAN
Date/Publication 2014-09-14 09:47:59
NeedsCompilation no

R topics documented:

EDISON-package .................................................. 3
AcceptableMove .................................................. 4
addProposalNetworkInfo ....................................... 5
BinoHyperMove .................................................. 5
BinoHyperRatio .................................................. 6
bp.computeAlpha ............................................... 7
buildXY .......................................................... 8
CalculateChanges .............................................. 9
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculateCPPGlobal</td>
<td>10</td>
</tr>
<tr>
<td>calculateCPPProbabilities</td>
<td>10</td>
</tr>
<tr>
<td>calculateEdgeProbabilities</td>
<td>11</td>
</tr>
<tr>
<td>calculateEdgeProbabilitiesSegs</td>
<td>12</td>
</tr>
<tr>
<td>calculateEdgeProbabilitiesTimePoints</td>
<td>13</td>
</tr>
<tr>
<td>CalculateLikelihoodRatio</td>
<td>14</td>
</tr>
<tr>
<td>CalculatePriorRatio</td>
<td>15</td>
</tr>
<tr>
<td>CollectNetworkInfo</td>
<td>16</td>
</tr>
<tr>
<td>computePx</td>
<td>17</td>
</tr>
<tr>
<td>computeRho4</td>
<td>18</td>
</tr>
<tr>
<td>convert_nets</td>
<td>19</td>
</tr>
<tr>
<td>cp.birth</td>
<td>19</td>
</tr>
<tr>
<td>cp.death</td>
<td>21</td>
</tr>
<tr>
<td>cp.shift</td>
<td>22</td>
</tr>
<tr>
<td>defaultOptions</td>
<td>23</td>
</tr>
<tr>
<td>dinvgamma</td>
<td>25</td>
</tr>
<tr>
<td>EDISON.run</td>
<td>25</td>
</tr>
<tr>
<td>ExpHyperMove</td>
<td>27</td>
</tr>
<tr>
<td>ExpHyperRatioTarget</td>
<td>28</td>
</tr>
<tr>
<td>fix_eigenvalues</td>
<td>29</td>
</tr>
<tr>
<td>generateNetwork</td>
<td>30</td>
</tr>
<tr>
<td>HyperparameterMove</td>
<td>31</td>
</tr>
<tr>
<td>HyperParms</td>
<td>32</td>
</tr>
<tr>
<td>init</td>
<td>33</td>
</tr>
<tr>
<td>main</td>
<td>34</td>
</tr>
<tr>
<td>make_structure_move</td>
<td>35</td>
</tr>
<tr>
<td>NetworkProbBino</td>
<td>36</td>
</tr>
<tr>
<td>NetworkProbExp</td>
<td>37</td>
</tr>
<tr>
<td>NetworkRatioBino</td>
<td>38</td>
</tr>
<tr>
<td>NetworkRatioExp</td>
<td>38</td>
</tr>
<tr>
<td>output</td>
<td>39</td>
</tr>
<tr>
<td>phase.update</td>
<td>40</td>
</tr>
<tr>
<td>PriorRatioPoisson</td>
<td>41</td>
</tr>
<tr>
<td>proposalTuning</td>
<td>42</td>
</tr>
<tr>
<td>proposeContinuous</td>
<td>43</td>
</tr>
<tr>
<td>ProposeDiscrete</td>
<td>44</td>
</tr>
<tr>
<td>psrf</td>
<td>45</td>
</tr>
<tr>
<td>psrf_check</td>
<td>46</td>
</tr>
<tr>
<td>psrf_check_hyper</td>
<td>47</td>
</tr>
<tr>
<td>readDataTS</td>
<td>47</td>
</tr>
<tr>
<td>rinvgamma</td>
<td>48</td>
</tr>
<tr>
<td>runDBN</td>
<td>49</td>
</tr>
<tr>
<td>sampleBinit</td>
<td>50</td>
</tr>
<tr>
<td>sampleBxy</td>
<td>51</td>
</tr>
<tr>
<td>sampleDelta2</td>
<td>52</td>
</tr>
<tr>
<td>sampleK</td>
<td>53</td>
</tr>
<tr>
<td>sampleParms</td>
<td>53</td>
</tr>
<tr>
<td>sampleSig2</td>
<td>54</td>
</tr>
</tbody>
</table>
EDISON-package

Allows for network reconstruction and changepoint detection.

Description

This package runs an MCMC simulation to reconstruct networks from time series data, using a non-homogeneous, time-varying dynamic Bayesian network. Networks segments and changepoints are inferred concurrently, and information sharing priors provide a reduction of the inference uncertainty.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>EDISON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.1</td>
</tr>
<tr>
<td>Date:</td>
<td>2014-09-13</td>
</tr>
<tr>
<td>License:</td>
<td>GPL-2</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>

Author(s)

Frank Dondelinger, Sophie Lebre

Maintainer: Frank Dondelinger <fdondelinger.work@gmail.com>

References


Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.

See Also

corpcor
Examples

# Generate random gene network and simulate data from it
dataset = simulateNetwork(l=25)

# Run MCMC simulation to infer networks and changepoint locations
result = EDISON.run(dataset$sim_data, num.iter=500)

# Calculate posterior probabilities of changepoints
cps = calculateCPProbabilities(result)

# Calculate marginal posterior probabilities of edges in the network
network = calculateEdgeProbabilities(result)

AcceptableMove  
Check if move is acceptable.

Description

This function takes as input a new network proposal and checks that the proposal does not exceed
the maximum number of parents for a node, and that there are no self loops (if self loops have been
disallowed).

Usage

AcceptableMove(proposal, qmax, self.loops, target, fixed.edges)

Arguments

proposal  
The proposed network (K-by-q matrix with K segments and q parent sets).
qmax  
Maximum number of parents allowed.
self.loops  
Flag indicating whether self loops are allowed.
target  
The current target node (only needed to find out which parent would be the self loop).
fixed.edges  
Which edges in the network should be fixed for all segments (q-by-q matrix with
entries 0 for fixed non-edge, 1 for fixed edge, -1 for non-fixed edge).

Value

Returns TRUE if the proposed move is acceptable, FALSE otherwise.

Author(s)

Frank Dondelinger

See Also

make_structure_move
addProposalNetworkInfo

Add the proposed new network to the new.nets list.

Description

Updates the network.info data structure so that it stays consistent.

Usage

addProposalNetworkInfo(network.info, newS, E)

Arguments

- network.info: Data structure containing the current network.
- newS: Proposed new network for this target, a num.local.segs by num.parents matrix.
- E: The current vector of local segments for this target (only used to check for consistency with the network.info change points).

Value

Updated network.info data structure, with new network added to new.nets.

Author(s)

Frank Dondelinger

BinoHyperMove

Makes a binomial hyperparameter move.

Description

This function proposes a move for one of the hyperparameters of the binomial prior, calculates the acceptance probability and accepts the move accordingly.

Usage

BinoHyperMove(network.info, node.sharing, GLOBvar)

Arguments

- network.info: The collected network information obtained using CollectNetworkInfo.
- node.sharing: Which type of node sharing is used, either 'soft' or 'hard' sharing.
- GLOBvar: Global variables of the MCMC.
Value

Returns a list with elements:

- move: The move type (in this case, 2).
- move.made: 1 if the move was proposed, 0 otherwise.
- network.info: The network information, including the new hyperparameters if the move was accepted.
- accept: Whether the move was accepted or not.

Author(s)

Frank Dondelinger

References

For information about the binomial information sharing prior, see:

Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.


See Also

BinoHyperMove

BinoHyperRatio

Calculates the MH ratio of the binomial prior.

Description

This function calculates the ratio of the binomial information sharing prior with the proposed new hyperparameter values, and the binomial prior with the current hyperparameter values.

Usage

BinoHyperRatio(params.proposed, changed, node.sharing, network.info)

Arguments

- params.proposed: The new proposed hyperparameter values for the binomial prior.
- changed: Gives the index of the parameter that has changed.
- node.sharing: Type of information sharing among nodes: 'soft' or 'hard'.
- network.info: The network information as collected by CollectNetworkInfo.
bp.computeAlpha

Value
This function returns a number greater than zero which represents the ratio of binomial priors.

Author(s)
Frank Dondelinger

References
For information about the binomial information sharing prior, see:
Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.

See Also
BinoHyperMove

bp.computeAlpha Computes the acceptance ratio of two changepoint configurations.

Description
This function computes the acceptance ratio of two changepoint configurations with networks in a changepoint birth or death move.

Usage
bp.computeAlpha(birth, lNew, kminus, Ekl, Estar, Ekr, yL, PxL, yR, PxR, y2, Px2, D, delta2, q, smax, v0, gamma0, prior_ratio = 1)

Arguments
birth 1 for a changepoint birth move, -1 for a changepoint death move.
lNew Number of edges in the new segment.
kminus Minimal number of changepoints between the two compared models (equal to s for a birth move, s-1 for a death move.
Ekl Changepoint on the left of proposed changepoint.
Estar Changepoint being inserted or deleted.
Ekr Changepoint on the right of proposed changepoint.
yL Response data (left).
PxL Projection matrix (left).
yR  Response data (right).
P xR  Projection matrix (right).
y2  Response data (both).
P x2  Projection matrix (both).
D  Hyperparameters for the number of edges in each segment.
delta2  Hyperparameters for the empirical covariance (signal-to-noise ratio).
q  Total number of nodes in the network.
smax  Maximum number of changepoints.
v0  Hyperparameter.
gamma0  Hyperparameter.
prior_ratio  Ratio of network structure priors.

Author(s)
Sophie Lebre

References
For more information about the model, see:

See Also
cp.birth, cp.death

---

**buildXY**  
*Builds response Y and predictor X.*

**Description**
This function builds the response variables Y and predictor variables X from the input data.

**Usage**

```r
buildXY(targetData, predData, GLOBvar)
```

**Arguments**

- **targetData**  Target input data.
- **predData**  Predictor input data.
- **GLOBvar**  Global variables of the MCMC simulation.
CalculateChanges

Value

A list with elements:

X
Predictor variables.
Y
Response variables.

Author(s)

Sophie Lebre

---------------------------------------------------------------------

CalculateChanges

Function to calculate the number of differences between adjacent network segments.

---------------------------------------------------------------------

Description

This function takes the current network structure, compares each segment to the next one, and calculates the number of changes. If soft information sharing among nodes is active, then this procedure is only done for the current target node.

Usage

CalculateChanges(network.info, node.sharing)

Arguments

network.info The network information collected by function CollectNetworkInfo.
node.sharing Specifies the type of information sharing among nodes: 'soft' or 'hard'.

Value

Returns a vector with 4 elements: the number of coinciding edges, the number of edges in the previous segment that are absent in the next one, the number of edges in the next segment that are absent in the previous one and the number of coinciding non-edges.

Author(s)

Frank Dondelinger
calculateCPPGlobal   *Calculated the global changepoint probabilities.*

**Description**

This function calculates the global probability of a changepoint at each measured timepoint, using the node-specific probabilities.

**Usage**

`calculateCPPGlobal(prob.cps)`

**Arguments**

- `prob.cps`     Node-specific changepoint probabilities, a NumNodes by NumTimepoints matrix.

**Value**

A matrix of length 1 by NumTimepoints, containing the global changepoint probabilities.

**Author(s)**

Frank Dondelinger

**See Also**

- `calculateCPPProbabilities`

---

calculateCPPProbabilities  *Calculate the changepoint probabilities.*

**Description**

This function calculates the marginal changepoint probabilities from the changepoint samples taken during the MCMC simulation.

**Usage**

`calculateCPPProbabilities(network.samples)`

**Arguments**

- `network.samples`  List of network and changepoint samples collected during the MCMC simulation by `EDISON.run` and `runDBN`. 

**calculateEdgeProbabilities**

*Calculate the edge probabilities.*

**Description**

This function calculates the marginal posterior probabilities of the edges in the network segments, for each timepoint, and optionally calculates the same for specified changepoints.

**Usage**

```r
calculateEdgeProbabilities(network.samples, cps = NULL)
```

**Arguments**

- `network.samples`
  - Network samples obtained from the MCMC simulation using `EDISON.run` and `runDBN`.
- `cps`
  - Optionally specifies changepoints to allow for calculating the marginal posterior edge probabilities for specific segments.

**Value**

A list with elements:

- `probs.all`
  - A list containing marginal edge posterior probabilities for each timepoint.
- `probs.segs`
  - A list containing marginal edge posterior probabilities for each specified segment.
Author(s)
Frank Dondelinger

See Also

*calculateEdgeProbabilitiesTimePoints*,
*calculateEdgeProbabilitiesSegs*

Examples

```r
# Generate random gene network and simulate data from it
dataset = simulateNetwork(l=25)

# Run MCMC simulation to infer networks and changepoint locations
result = EDISON.run(dataset$sim_data, num.iter=500)

# Calculate marginal posterior probabilities of edges in the network
network = calculateEdgeProbabilities(result)

# Calculate marginal posterior probabilities of edges in the network,
# using the true changepoints
true.cps = c(2,dataset$epsilon)

network = calculateEdgeProbabilities(result, cps=true.cps)
```

calculateEdgeProbabilitiesSegs

*Calculate edge probabilities for fixed segments.*

Description

This function calculates the marginal posterior probabilities for the edges in each network for the specified segments.

Usage

```
calculateEdgeProbabilitiesSegs(prob.networks, cps, numNodes)
```

Arguments

- `prob.networks`: List containing the marginal posterior probabilities for the edges of each network at each timepoint, from `calculateEdgeProbabilitiesTimePoints`.
- `cps`: Changepoints defining the segments for which the edge probabilities should be calculated. Note that these are global changepoints that apply to the whole network.
- `numNodes`: Number of nodes in the network.
Value

Returns a list of length equal to the number of segments, with each entry containing a matrix of size NumNodes by NumNodes which contains the marginal edge probabilities for that segment.

Author(s)

Frank Dondelinger

See Also

calculateEdgeProbabilities, calculateEdgeProbabilitiesTimePoints

Description

This function calculates the marginal posterior edge probabilities of the network at each timepoint.

Usage

calculateEdgeProbabilitiesTimePoints(network.samples, cps, numNodes)

Arguments

network.samples Collection of network and changepoint samples of the MCMC simulation, as obtained by EDISON.run, runDBN.
cps Changepoint vector.
numNodes Number of nodes in the network.

Value

A list of length equal to the number of timepoints, where each entry contains a matrix of size NumNodes by NumNodes with the marginal posterior edge probabilities of the network at this timepoint.

Author(s)

Frank Dondelinger

See Also

calculateEdgeProbabilities, calculateEdgeProbabilitiesSegs
CalculateLikelihoodRatio

Calculates the ratio of two likelihoods in a structure move.

Description

This function calculates the ratio of the likelihoods in a network structure move. The returned value is the ratio for the modification of one edge in one segment.

Usage

CalculateLikelihoodRatio(gamma0, y, Pxl,m, Px1, v0, delta2, dir)

Arguments

- **gamma0**: Hyperparameter.
- **y**: Target data.
- **Pxl,m**: Projection matrix with modified edge.
- **Px1**: Original projection matrix.
- **v0**: Hyperparameter.
- **delta2**: Delta squared parameter (signal-to-noise).
- **dir**: Direction of the change: 1 = Added an edge. 2 = Removed an edge. 0 = No change.

Value

Returns the likelihood ratio.

Author(s)

Frank Dondelinger

References

For more information about the hyperparameters and the functional form of the likelihood, see: Dondelinger et al. (2012), "Non-homogeneous dynamic Bayesian networks with Bayesian regularization for inferring gene regulatory networks with gradually time-varying structure", Machine Learning.

See Also

CalculatePriorRatio
CalculatePriorRatio

Calculates the network prior ratio.

Description
This function calculates the ratio of the network structure priors for a structure move.

Usage
CalculatePriorRatio(method, q, lambda, network.info)

Arguments

- **method**: Indicates which prior to use: 'poisson' for the standard Poisson prior (no information sharing), 'exp_soft' or 'exp_hard' for the exponential information sharing prior with soft or hard sharing among nodes and 'bino_soft' or 'bino_hard' for the binomial information sharing prior with soft or hard sharing among nodes.
- **q**: Number of nodes in the network.
- **lambda**: Vector of lambda hyperparameter values for each network (needed for the Poisson prior).
- **network.info**: The network information collected using CollectNetworkInfo.

Value
Returns the ratio of the network structure priors for the proposed structure move.

Author(s)
Frank Dondelinger

References
For more information on the network structure priors, see:

Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.


See Also
CalculateLikelihoodRatio
CollectNetworkInfo

Collect all the network information in one list.

Description

This function collects information about the current network segments and hyperparameters for the information sharing priors.

Usage

CollectNetworkInfo(Sall, Eall, prior.params, posPhase, target, q, self.loops, k)

Arguments

- **Sall**: Structure of all segments. A list of length q, where each element is a \( K_i \) by \( q \) matrix containing the parents for the current node in each of the \( K_i \) segments.
- **Eall**: Positions of segment boundaries. A list of length q, where each element is a vector containing the segment boundaries for the current parent node.
- **prior.params**: The hyperparameters of the information sharing prior (if applicable).
- **posPhase**: The segment being changed.
- **target**: The target parent node whose edge is being changed.
- **q**: The total number of nodes in the network.
- **self.loops**: Whether self-loops are allowed in the network.
- **k**: The level-2 hyperparameter for the exponential prior.

Value

The function returns a list with the following elements:

- **nets**: The structure of all segments, a list of length \( K \) where \( K \) is the total number of segments over all nodes.
- **segment**: Identical to posPhase.
- **target.nets**: Identical to Sall.
- **prior.params**: Identical to prior.params.
- **self.loops**: Identical to self.loops.
- **k**: Identical to k.
- **new.nets**: Dummy variable for holding the proposed network in a network structure move. Originally identical to variable nets.

Author(s)

Frank Dondelinger
computePx

Compute projection matrix.

Description

This function computes the projection matrix that is needed for calculation of the likelihood.

Usage

computePx(len, x, delta2)

Arguments

len Delimiting breakpoints.

x The observations of x in the corresponding state.

delta2 Signal-to-noise ratio hyperparameter.

Value

The projection matrix.

Author(s)

Sophie Lebre

References

For more information about the hyperparameters and the functional form of the likelihood, see:


See Also

CalculateLikelihoodRatio
computeRho4  
*Calculate proposal frequencies for changepoint moves.*

**Description**

This function calculates the frequency at which each of the different changepoint moves is proposed. For the poisson network structure prior, this ensures that the proposal frequency is equal to the prior probability.

**Usage**

```r
computeRho4(k, kmin, kmax, c, lambda)
```

**Arguments**

- `k`  
  The number of hidden states.

- `kmin`  
  Minimum number of hidden states.

- `kmax`  
  Maximum number of hidden states

- `c`  
  Parameter.

- `lambda`  
  Hyperparameter controlling the number of hidden states.

**Value**

Vector containing the proposal frequencies for the different changepoint moves.

**Author(s)**

Sophie Lebre

**References**

For more information about the hyperparameters and the functional form of the likelihood, see:

**convert_nets**

*Convert internal representation of networks.*

**Description**

Converts from representing the network as a list of target nodes to representing it as a list of segments.

**Usage**

`convert_nets(Ball, Eall)`

**Arguments**

- **Ball**: Input network: List of target nodes, where each element is a NumSegs by NumNodes matrix giving the parents for the target node in each segment.
- **Eall**: Changepoints: List of target nodes, where each element contains a vector of changepoints.

**Value**

List with elements:

- **B_nets**: List of segments, where each element contains a matrix of size NumNodes by NumNodes, representing the network for that segment.
- **segs**: Vector containing the global segment boundaries.

**Author(s)**

Frank Dondelinger

**cp.birth**

*Make changepoint birth move.*

**Description**

This function makes a changepoint birth move, possibly adding a changepoint.

**Usage**

`cp.birth(Eall, Sall, Ball, Sig2all, X, Y, D, GLOBvar, HYPERvar, target)`
Arguments

Eall  Changepoints: List of target nodes, where each element contains a vector of changepoints.

Sa11  Network structure: List of target nodes, where each element is a NumSegs by NumNodes matrix giving the parents for the target node in each segment. A binary matrix.

Ba11  Network parameters: Similar to network structure, but with regression parameters included.

Sig2a11  Sigma squared parameters.

X  Response data.

Y  Target data.

D  Hyperparameter.

GL0Bvar  Global variables of the MCMC simulation.

HYPERvar  Hyperparameter variables.

target  Which target node the move is being proposed for.

Value

A list with elements:

e  New changepoint vector for target node.

Sa11  Updated network structure.

Ba11  Updated network structure with regression parameters.

Sig2a11  Updated sigma squared.

prior.params  Updated vector of structure prior hyperparameters.

accept  Whether the move was accepted or not.

move  What type of move was made. In this case move=1 for a changepoint birth move.

alpha  The acceptance ratio of the move.

estar  The location of the new changepoint.

k  Hyperparameter.

Author(s)

Sophie Lebre
Frank Dondelinger

References

For more information about the different changepoint moves, see:
See Also

*cp.death*, *cp.shift*

---

**Description**

This function makes a changepoint death move, possibly removing a changepoint.

**Usage**

```r
cp.death(Eall, Sall, Ball, Sig2all, X, Y, D, GLOBvar, HYPERvar, target)
```

**Arguments**

- `Eall`: Changepoints: List of target nodes, where each element contains a vector of changepoints.
- `Sall`: Network structure: List of target nodes, where each element is a NumSegs by NumNodes matrix giving the parents for the target node in each segment. A binary matrix.
- `Ball`: Network parameters: Similar to network structure, but with regression parameters included.
- `Sig2all`: Sigma squared parameters.
- `X`: Response data.
- `Y`: Target data.
- `D`: Hyperparameter.
- `GLOBvar`: Global variables of the MCMC simulation.
- `HYPERvar`: Hyperparameter variables.
- `target`: Which target node the move is being proposed for.

**Value**

A list with elements:

- `E`: New changepoint vector for target node.
- `Sall`: Updated network structure.
- `Ball`: Updated network structure with regression parameters.
- `Sig2all`: Updated sigma squared.
- `prior.params`: Updated vector of structure prior hyperparameters.
- `accept`: Whether the move was accepted or not.
- `move`: What type of move was made. In this case move=2 for a changepoint death move.
- `alpha`: The acceptance ratio of the move.
- `estar`: The location of the removed changepoint.
- `k`: Hyperparameter.
Author(s)

Sophie Lebre
Frank Dondelinger

References

For more information about the different changepoint moves, see:

See Also

cp.birth, cp.shift

---

**cp.shift**

*Makes a changepoint shift move.*

**Description**

This function makes a changepoint shift move, possibly moving one of the changepoints.

**Usage**

cp.shift(Eall, Sall, Ball, Sig2all, X, Y, GLOBvar, HYPERvar, target)

**Arguments**

- **Eall**: Changepoints: List of target nodes, where each element contains a vector of changepoints.
- **Sall**: Network structure: List of target nodes, where each element is a NumSegs by NumNodes matrix giving the parents for the target node in each segment. A binary matrix.
- **Ball**: Network parameters: Similar to network structure, but with regression parameters included.
- **Sig2all**: Sigma squared parameters.
- **X**: Response data.
- **Y**: Target data.
- **GLOBvar**: Global variables of the MCMC simulation.
- **HYPERvar**: Hyperparameter variables.
- **target**: Which target node the move is being proposed for.
Value

A list with elements:

- `E` New changepoint vector for target node.
- `Sall` Updated network structure.
- `Ball` Updated network structure with regression parameters.
- `Sig2all` Updated sigma squared.
- `prior.params` Updated vector of structure prior hyperparameters.
- `accept` Whether the move was accepted or not.
- `move` What type of move was made. In this case `move=2` for a changepoint death move.
- `alpha` The acceptance ratio of the move.
- `estar` The location of the removed changepoint.
- `k` Hyperparameter.

Author(s)

Sophie Lebre

References

For more information about the different changepoint moves, see:


See Also

`cp.birth`, `cp.death`

defaultOptions

Set the default options for the MCMC simulation.

Description

This function creates a list with the default options of the MCMC simulation.

Usage

defaultOptions()
Value

A list of default options with elements:

- lmax: Maximum number of parent nodes. Default=5.
- m: Number of repeated measurements. Default=1 (no repeats).
- dynt: Lag for the DBN model. Default = 1 when X(t) depends on the previous measurement X(t-1), but dynt can be chosen equal to 2, 3, ...
- minPhase: Minimal length of a segment. Default=2.
- maxCP: Maximal number of changepoints. Default=10.
- maxTF: Maximal number of incoming edges for each node. Default=5.
- alphaCP: Hyperparameter for the number of changepoints. Default=1.
- betaCP: Hyperparameter for the number of changepoints. Default=0.5.
- alphaTF: Hyperparameter for the number of incoming edges. Default=1.
- betaTF: Hyperparameter for the number of incoming edges. Default=0.5.
- burnin: Whether to include a burnin period. Default=F.
- psrf.check: Whether to calculate the potential scale reduction factor (PSRF). Default=F.
- pp.11: Proposal frequency for level-1 hyperparameter moves. Default=0.2.
- pp.12: Proposal frequency for level-2 hyperparameter moves. Default=0.01.
- save.by.node: Whether to save results separately for each target node. Default=F.
- save.file: Whether to save the results to a file. Default=F.
- hyper.fixed: Whether to keep the network structure prior hyperparameters fixed. Default=F.
- cp.fixed: Whether to keep the changepoints fixed. Default=F.
- hyper.init: Initial values for the network structure prior hyperparameters. Default=NULL.
- cp.init: Initial values for the changepoint locations. Default=NULL.

Author(s)

Frank Dondelinger

Examples

```r
# Set options to allow saving network and changepoint samples to file
options = defaultOptions()
options$save.file = TRUE

# NOT EXECUTED
# result.bino2 = EDISON.run(dataset$sim_data,
#     information.sharing="bino_hard",
#     num.iter=5000, output.file='bino2.results',
#     options=options)
```
**dinvgamma**  
*Calculate inverse gamma distribution.*

**Description**

This function calculates the density of the inverse gamma distribution.

**Usage**

```
dinvgamma(x, shape, scale = 1, log = FALSE)
```

**Arguments**

- `x`  
  Input.
- `shape`  
  Shape parameter.
- `scale`  
  Scale parameter (1/rate).
- `log`  
  Whether to return the log density.

**Value**

Returns the density (or log density).

**Author(s)**

Frank Dondelinger

**Examples**

```r
# Draw samples from inverse gamma distribution with shape parameter 1
# and scale parameter 1
samples = dinvgamma(100, shape=1, scale=1)

densities = dinvgamma(samples, shape=1, scale=1)
```

---

**EDISON.run**  
*Wrapper function for starting an MCMC simulation*

**Description**

This function provides a wrapper for starting an MCMC simulation, using only the data and some basic options as input.
Usage

EDISON.run(input, output.file = "EDISON.output",
  information.sharing = "poisson", num.iter = 10000, prior.params = NULL,
  options = NULL, fixed.edges = NULL)

Arguments

input       Input data. Either a filename pointing to an R data file containing the results of 
            simulateNetwork, or a NumTimePoints by NumNodes matrix.
output.file Where to save the output of the MCMC simulation.
information.sharing Which information sharing prior to use: 'poisson' for the Poisson prior (no 
                         information sharing), 'exp_hard' or 'exp_soft' for the exponential prior with 
                         hard or soft coupling among nodes, respectively, and 'bino_hard' or 'bino_soft' 
                         for the binomial prior with hard or soft coupling among nodes.
num.iter    Number of iterations of the MCMC simulation.
prior.params Initial values of the hyperparameters of the information sharing priors.
options     Settings for the MCMC simulation, as generated by defaultOptions.
fixed.edges Matrix of size NumNodes by NumNodes, with fixed.edges[i,j]==1 if the 
              edge between nodes i and j is fixed, and -1 otherwise. Defaults to NULL (no edges 
              fixed).

Value

Returns the results of the MCMC simulation, similar to runDBN.

Author(s)

Sophie Lebre Frank Dondelinger

References

For details on the model and MCMC simulation, see:
Dondelinger et al. (2012), "Non-homogeneous dynamic Bayesian networks with Bayesian regu-
larization for inferring gene regulatory networks with gradually time-varying structure", Machine 
Learning.

See Also

runDBN

Examples

# Generate random gene network and simulate data from it
dataset = simulateNetwork(l=25)

# Run MCMC simulation to infer networks and changepoint locations
# Uses default settings: Poisson prior and 1500 iterations
ExpHyperMove

result.poisson = EDISON.run(dataset$sim_data, num.iter=500)

# Use the binomial information sharing prior with hard node coupling, and
# run for 5000 iterations

# NOT EXECUTED
result.bino = EDISON.run(dataset$sim_data,
    information.sharing='bino_hard', num.iter=5000)

# Set options to allow saving network and changepoint samples to file
options = defaultOptions()
options$save.file = TRUE

# NOT EXECUTED
# result.bino2 = EDISON.run(dataset$sim_data,
#    information.sharing='bino_hard',
#    num.iter=5000, output.file='bino2.results',
#    options=options)

ExpHyperMove  Makes an exponential hyperparameter move.

Description

This function tries to make a level-1 or level-2 hyperparameter move for the exponential prior

Usage

ExpHyperMove(network.info, node.sharing, GLOBvar, hyper.proposals)

Arguments

network.info  The network information collected by CollectNetworkInfo.
node.sharing  The type of information sharing among nodes: 'soft' or 'hard'.
GLOBvar       Collection of global variables of the MCMC.
hyper.proposals Proposal width of the hyperparameter move.

Value

Returns a list with elements:

move.made  1 if a level-1 hyperparameter move has been made, 0 otherwise.
network.info Network information with updated hyperparameters if the move was accepted.
accept  Whether a level-1 hyperparameter move has been accepted or not.
move.made.k  1 if a level-2 hyperparameter move has been made, 0 otherwise.
accept.k  Whether a level-2 hyperparameter move has been accepted or not.
move  Type of move: 2 for a level-1 hyperparameter move, 3 for a level-2 hyperparameter move.
Author(s)
Frank Dondelinger

References
For information about the exponential information sharing prior, see:
Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.

See Also
ExpHyperRatioTarget

---

**ExpHyperRatioTarget**  
*Calculates the ratio of an exponential hyperparameter move.*

Description
This function calculates the acceptance ratio of a level-1 hyperparameter move for a given target node.

Usage
ExpHyperRatioTarget(beta.proposed, beta.old, target.net, self.loops)

Arguments
- **beta.proposed**: Proposed new hyperparameter value.
- **beta.old**: Previous value of hyperparameter beta.
- **target.net**: Network segments for the target node associated with this hyperparameter value.
- **self.loops**: 'TRUE' if self-loops are acceptable, 'FALSE' otherwise.

Value
Returns the ratio of the exponential prior with the previous hyperparameter value and the proposed new hyperparameter value.

Author(s)
Frank Dondelinger
References
For information about the exponential information sharing prior, see:
Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.

See Also
ExpHyperMove

generateNetwork
generateNetwork  

Generate a random network.

Description

This function generates a random network with changepoints for structure changes, for simulating synthetic data.

Usage

generateNetwork(lambda_2 = 0.45, q = 10, min_phase_length = 1, k_bar = 5, l = 10, lambda_3 = 2, spacing = 1, gauss_weights = TRUE, same = FALSE, change_method = "sequential", fixed = FALSE, cps = NULL)

Arguments

- lambda_2: Average number of parents for each node in the network (parameter for a Poisson distribution).
- q: Number of nodes.
- min_phase_length: Minimum segment length.
- k_bar: Maximum number of changepoints. If fixed=TRUE, this is equal to the number of changepoints.
- l: Length of the time series.
- lambda_3: Average number of structure changes between two segments (parameter for a Poisson distribution).
- spacing: 1 if segments are equally spaced, 0 if they are spaced randomly (subject to the constraints of min_phase_length).
- gauss_weights: 1 if edge weights in the network are drawn from N(0, 1), 0 if they are fixed to be 1.
- same: 1 if all segments have the same network structure (no changes), 0 otherwise.
- change_method: 'sequential' if the changes happen sequentially (i.e. changes at segment i are applied to segment i-1), 'hierarchical' if the changes happen with respect to a hypernetwork (i.e. changes at segment i are applied to segment 0).
- fixed: T if the changepoint locations are fixed, F if they should be sampled.
- cps: Changepoint locations (if they are fixed).

Value

A list with the following elements:

- network: The network, a list of length NumSegs, where each element is a NumNodes by NumNodes matrix.
- epsilon: The vector of changepoint locations.
- k: The number of changepoint.
- changes: The number of changes among segments.
HyperparameterMove

Author(s)

Frank Dondelinger

See Also

simulatenetwork

Examples

# Generate random network with default parameters
network = generateNetwork()

# Simulate data using generated network
dataset = simulatenetwork(net=network)

# Generate random network with 4 changepoints and 15 nodes,
# with changepoints distributed over a timeseries of length 50
network = generateNetwork(l=50, q=15, fixed=TRUE, k_bar=4)

# Simulate data of length 50 using generated network
dataset = simulatenetwork(net=network)

HyperparameterMove  Make a hyperparameter move.

Description

This function makes a hyperparameter move for the information sharing prior selected (or no move if no information sharing prior is selected).

Usage

HyperparameterMove(method, network.info, GLOBvar, hyper.proposals)

Arguments

method  The information sharing method used: 'poisson' for the Poisson prior (no information sharing), 'exp_soft' and 'exp_hard' for the exponential information sharing prior with soft or hard information sharing among nodes, respectively, 'bino_soft' and 'bino_hard' for the binomial information sharing prior with soft or hard information sharing among nodes, respectively.

network.info  Network information collected using CollectNetworkInfo.

GLOBvar  Global variables used during the MCMC.

hyper.proposals  Proposal width for hyperparameters.
HyperParms

Value
List summing up the result of the hypermove. Contains at least:

- `move.made` Whether a hyperparameter move has been made.
- `network.info` The network information, possibly updated if the hyperparameter move was made and accepted.

May contain further elements depending on the type of information sharing prior used. See the prior-specific functions `ExpHyperMove` and `BinoHyperMove` for details.

Author(s)
Frank Dondelinger

References
For information about the information sharing priors, see:
Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.

HyperParms

Sets up initial values of hyperparameters.

Description
This function initialises the variable HYPERvar with values for the various hyperparameters in the model.

Usage

HyperParms(options)

Arguments

- `options` MCMC settings, possibly from `defaultOptions`.

Value
Settings for the HYPERvar variable:

- `cD` Proportion of changepoint moves proposed.
- `alphaD` Prior settings for the number of changepoints.
- `betaD` Prior settings for the number of changepoints.
init

Initialise the MCMC simulation.

Description

This function initialises the parameters and variables needed for the MCMC simulation.

Usage

init(X, Y, sinit, GLOBvar, HYPERvar, options)

Arguments

X Input response data.
Y Input target data.
sinit Initial changepoints.
GLOBvar Global variables used during the MCMC simulation.
HYPERvar Hyperparameter variables.
options MCMC simulation options as obtained e.g. by defaultOptions.

Value

List with elements:
counters Matrices for counting the number of moves made and accepted.
initState Initial state of the variables of the MCMC simulation.
listStock Variables for recording the network, changepoint and hyperparameter samples.

Author(s)

Sophie Lebre
Frank Dondelinger
See Also

sampleParms

main

Main function of the MCMC simulation.

Description

This function executes the main loop of the MCMC simulation, making the different moves and recording samples.

Usage

main(X, Y, initiation, GLOBvar, HYPERvar)

Arguments

X  Input response data.
Y  Input target data.
initiation  Initialisation of the MCMC simulation, as obtained by function init.
GLOBvar  Global variables of the MCMC simulation.
HYPERvar  Hyperparameter variables.

Value

Returns a list with the following elements:
counters  List containing the different move counters for the number of times moves have been proposed and accepted.
listStock  List containing the recorded samples for the networks, changepoints and hyperparameters.

Author(s)

Sophie Lebre
Frank Dondelinger

References

For more information about the MCMC simulations, see:

See Also

runDBN
**make_structure_move**  

Makes a structure move.

**Description**

This function makes a network structure move.

**Usage**

```
make_structure_move(x, y, S, B, Sig2, q, qmax, network.info, method, Mphase, E, fixed.edges, HYPERvar)
```

**Arguments**

- `x`  
  Response data.
- `y`  
  Target data.
- `S`  
  Network structure for the current target node, a NumSegs by NumNodes matrix.
- `B`  
  Same as `S`, but including the regression parameters.
- `Sig2`  
  Sigma squared parameters.
- `q`  
  Number of nodes.
- `qmax`  
  Maximum number of parents.
- `network.info`  
  Network information, as collected by `CollectNetworkInfo`.
- `method`  
- `Mphase`  
  Segment boundary positions.
- `E`  
  Changepoint vector.
- `fixed.edges`  
  Matrix of size NumNodes by NumNodes, with `fixed.edges[i,j]=1` if the edge between nodes i and j is fixed, and -1 otherwise. Defaults to NULL (no edges fixed).
- `HYPERvar`  
  Hyperparameter variables.

**Value**

Returns a list containing the following elements:

- `newS`  
  Updated network structure.
- `newB`  
  Updated network structure with regression parameters.
- `move`  
  Type of move being made: 1 for network structure moves.
- `accept`  
  1 if the move has been accepted, 0 otherwise.

**Author(s)**

Frank Dondelinger
NetworkProbBino

Calculates the prior probability of the network segments under the binomial prior.

Description

This function calculates the (log) probability of the network segments using the binomial information sharing prior.

Usage

```
NetworkProbBino(network.info, node.sharing = "soft")
```

Arguments

- `network.info`: Network information collected by function `CollectNetworkInfo`.
- `node.sharing`: Coupling of hyperparameters among nodes: 'hard' or 'soft'.

Value

Returns the log prior probability of the network segments under the binomial prior.

Author(s)

Frank Dondelinger

References

For more information about the MCMC moves, see:

For information about the binomial information sharing prior, see:
Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.

See Also

- `NetworkRatioBino`
- `CalculatePriorRatio`
NetworkProbExp

**NetworkProbExp**

*Calculates the prior probability of the network using the exponential prior.*

---

**Description**

This function calculates the log prior probability of the network structure. It uses the exponential information sharing prior.

**Usage**

```r
NetworkProbExp(network.info)
```

**Arguments**

- `network.info`   Network information collected using the function `CollectNetworkInfo`

**Value**

Returns the log prior probability of the network segments.

**Author(s)**

Frank Dondelinger

**References**

For information about the exponential information sharing prior, see:

Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.


**See Also**

`NetworkRatioExp`, `CalculatePriorRatio`
NetworkRatioExp

Calculates the ratio of exponential network prior probabilities.

Description

This function calculates the ratio of exponential network information sharing prior probabilities.

Usage

NetworkRatioExp(network.info)

Arguments

network.info Network information collected by function CollectNetworkInfo. Note that network.info$new.nets has to be set.

Value

Returns the ratio of [prior of new network]/[prior of old network].

Author(s)

For information about the exponential information sharing prior, see:

Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.


See Also

NetworkProbBino, CalculatePriorRatio

NetworkRatioBino

Calculates the ratio of binomial prior probabilities.

Description

This function calculates the ratio of binomial prior probabilities of two networks.

Usage

NetworkRatioBino(network.info, node.sharing)

Arguments

network.info Network information collected by function CollectNetworkInfo. Note that network.info$new.nets has to be set.

node.sharing Type of coupling of hyperparameters among nodes: 'hard' or 'soft'.

Value

Returns the ratio of [prior of new network]/[prior of old network].

Author(s)

For information about the binomial information sharing prior, see:

Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.


See Also

NetworkProbBino, CalculatePriorRatio
output

Arguments

network.info Network information collected using the function `CollectNetworkInfo`. Note that `network.info$new.nets` has to be set.

Value

Returns the ratio [prior of new network]/[prior of old network].

Author(s)

Frank Dondelinger

References

For information about the exponential information sharing prior, see:
Husmeier et al. (2010), "Inter-time segment information sharing for non-homogeneous dynamic Bayesian networks", NIPS.

See Also

NetworkProbExp, CalculatePriorRatio

output Collects and saves output.

Description

This function collects the network, changepoint and hyperparameter samples taken from the MCMC simulation, and saves them to a file if appropriate.

Usage

`output(counters, listStock, GLOBvar, HYPERvar, OUTvar)`

Arguments

counters List of counters for the number of moves that have been proposed and accepted.
listStock Network, changepoint and hyperparameter samples.
GLOBvar Global variables of the MCMC simulation.
HYPERvar Hyperparameter variables.
OUTvar Output variables, including the output file.
Value

Returns a list with an element for each target node which is also a list. Each sublist contains the elements:

- **cp_samples**: Changepoint samples, a NumSamples by MaxNumChangePoints matrix.
- **edge_samples**: Network samples (with regression parameters), a NumSamples by (NumSegs * NumNodes) matrix.
- **target**: The target node for this subnetwork.
- **hyper_samples**: Information sharing prior hyperparameter samples, a NumSamples by NumHyperParams matrix.
- **sampled**: Sampled iterations.
- **counters**: Counters for the number of proposed and accepted moves.

Author(s)

Frank Dondelinger

phase.update

Make a network structure or hyperparameter move.

Description

This function makes a network structure or information sharing hyperparameter move.

Usage

phase.update(Eall, Sall, Ball, Sig2all, X, Y, GLOBvar, HYPERvar, target)

Arguments

- **Eall**: List of changepoints with one entry for each target node. Each entry has length equal to the number of changepoints for that target node.
- **Sall**: Network structure: List of length equal to the number of target nodes, where each list entry is a NumSegs by NumNodes matrix.
- **Ball**: Network structure with regression coefficients: Same as Sall, but with regression coefficients as matrix entries.
- **Sig2all**: Sigma squared.
- **X**: Input response data.
- **Y**: Input target data.
- **GLOBvar**: Global variables used during the MCMC simulation.
- **HYPERvar**: Hyperparameter variables.
- **target**: Current target node.
PriorRatioPoisson

**Value**

Returns a list with the following elements:

- **E** Changepoints for the current target node.
- **Sall** Network structure (possibly updated).
- **Ball** Network structure regression coefficients (possibly updated).
- **Sig2all** Sigma squared.
- **prior.params** Information sharing prior hyperparameters (possibly updated).
- **k** Level-2 exponential prior hyperparameter (possibly updated).
- **move** Move type: 4 for a network structure move, 5 hyperparameter move.
- **move** Structure Move type: 1 for a network structure move, 2 for a level-1 hyperparameter move, 3 for a level-2 hyperparameter move.
- **accept** 1 if the move has been accepted, 0 otherwise.

**Author(s)**

Sophie Lebre
Frank Dondelinger

**References**

For more information on network structure moves and information sharing priors, see:


**See Also**

`make_structure_move`

---

**PriorRatioPoisson**

*Calculate network prior ratio with Poisson prior.*

**Description**

This function calculates the ratio of the Poisson prior for two networks.

**Usage**

`PriorRatioPoisson(network.info, q, lambda)`
**proposalTuning**

**Arguments**

- `network.info` Network information collected using `CollectNetworkInfo`. Note that one needs to set `network.info$new.nets`.
- `q` Number of nodes in the network.
- `lambda` Vector of lambda hyperparameters for each network.

**Value**

Returns the ratio \([\text{prior of new network}] / [\text{prior of old network}]\).

**Author(s)**

Frank Dondelinger

**References**

For more information on the network structure priors, see:

**See Also**

- `CalculatePriorRatio`

**Description**

This function adjusts the proposal width for the beta hyperparameter(s) of the exponential information sharing prior, so that the acceptance rate is close to 0.25.

**Usage**

```
proposalTuning(acceptRate, hyper.proposals)
```

**Arguments**

- `acceptRate` Current acceptance rate.
- `hyper.proposals` Current proposal width.

**Value**

Returns the new proposal width.
proposeContinuous

Author(s)
Frank Dondelinger

proposeContinuous  Propose a new real hyperparameter value.

Description
This function proposes a new real values hyperparameter for the information sharing prior.

Usage
proposeContinuous(orig_beta, proposal_range, limit = 30)

Arguments
- orig_beta: Current value of the hyperparameter.
- proposal_range: Range for the new value.
- limit: Hard limit on the range.

Value
Returns a new uniformly random value within proposal_range of orig_beta and limited by limit.

Author(s)
Frank Dondelinger

See Also
ProposeDiscrete

Examples
# Previous parameter value
param = runif(1, 0, 1)

# Propose new value within range [0, 1], with proposal width 0.1
new.param = proposeContinuous(param, 0.1, 1)
ProposeDiscrete

Propose a new discrete value.

Description

This function proposes a new discrete parameter, based on the previous value, within the given proposal range, making sure that the maximum range is not exceeded.

Usage

ProposeDiscrete(params.old, proposal.range, max.range)

Arguments

- params.old: Old parameter value (an integer).
- proposal.range: Range for new proposal (an integer).
- max.range: Maximum value for new proposal (an integer).

Value

Returns the new proposed parameter, which will be an integer in the range $[0, \text{max.range}]$, and within at most proposal.range of params.old.

Author(s)

Frank Dondelinger

See Also

proposeContinuous

Examples

```r
# Previous parameter value
param = rpois(1, 5)

# Propose new value within range [0, 10], with proposal width 2
new.param = ProposeDiscrete(param, 2, 10)
```
psrf

Calculates the potential scale reduction factor.

Description
This function calculates the potential scale reduction factor of parameters or hyperparameters over several MCMC simulations (or one simulation split up). This can serve as a convergence diagnostic.

Usage
psrf(parameters)

Arguments
parameters A list of MCMC trajectories, where each trajectory is a matrix with NumParams rows and NumIterations columns, where NumParams is the number of parameters and NumIterations is the number of samples.

Value
A vector of length NumParams, containing the PSRF values for each parameter.

Author(s)
Sophie Lebre
Frank Dondelinger

References

See Also
psrf_check, psrf_check_hyper

Examples
# Generate 5 'runs' of random samples from Gaussian N(0,1)
samples = list()

for (run in 1:5) {
  samples[[run]] = matrix(rnorm(1000), 1, 1000)
}

# Check potential scale reduction factor
# (Will be very close to 1 due to the samples being from
# the same distribution)
psrf.val = psrf(samples)
# Now use slightly different Gaussian distributions for each 'run'.
for(run in 1:5) {
    mean = runif(1, 0, 2)
    samples[[run]] = matrix(rnorm(1000, mean, 1), 1, 1000)
}

# Check potential scale reduction factor
# (Should be > 1.1)
psrf.val = psrf(samples)

---

**psrf_check**

*Check the potential scale reduction factors for all parameters (edges).*

**Description**

This function treats the edges of the network as parameters, calculates their potential scale reduction factors and returns the highest value.

**Usage**

```
psrf_check(params, q, k_max, num_it)
```

**Arguments**

- **params**: Matrix of parameters.
- **q**: Number of nodes.
- **k_max**: Number of segments.
- **num_it**: Number of iterations/samples.

**Value**

Returns the highest PSRF value.

**Author(s)**

Frank Dondelinger

**References**


**See Also**

`psrf`, `psrf_check_hyper`
**psrf_check_hyper**

Checks the potential scale reduction factor for the hyperparameters.

**Description**

This function checks the potential scale reduction factors for the hyperparameters of the information sharing priors.

**Usage**

```r
psrf_check_hyper(params, num_it)
```

**Arguments**

- `params` : Matrix of hyperparameters.
- `num_it` : Number of iterations/samples.

**Value**

Returns the maximum PSRF value.

**Author(s)**

Frank Dondelinger

**References**


**See Also**

`psrf, psrf_check`

---

**readDataTS**

*Read target data.*

**Description**

This function reads in the target data.

**Usage**

```r
readDataTS(data, posI, t0, tf, m, n)
```
Arguments

- **data**: Input data matrix to read.
- **posI**: Position of interest.
- **t0**: First timepoint.
- **tf**: Last timepoint.
- **m**: Number of repetitions.
- **n**: Number of timepoints.

Value

Returns the target data.

Author(s)

Sophie Lebre

See Also

- `buildXY`

---

**rinvgamma**

*Samples from the inverse gamma distribution.*

Description

This function samples from the inverse gamma distribution.

Usage

`rinvgamma(n, shape, scale)`

Arguments

- **n**: Number of values to sample.
- **shape**: Shape parameter.
- **scale**: Scale parameter (1/rate).

Value

Random sample from the inverse gamma distribution.

Author(s)

Frank Dondelinger
See Also
dinvgamma

Examples

# Draw samples from inverse gamma distribution with shape parameter 1
# and scale parameter 1
samples = rinvgamma(100, shape=1, scale=1)

# Calculate density of samples
densities = dinvgamma(samples, shape=1, scale=1)

Description

This function initialises the variables for the MCMC simulation, runs the simulation and returns the output.

Usage

rundbn(targetdata, preddata = NULL, q, n, multipleVar = TRUE,
minphase = 2, niter = 20000, scaling = TRUE, method = "poisson",
prior.params = NULL, self.loops = TRUE, k = 15, options = NULL,
outputfile = ".", fixed.edges = NULL)

Arguments

targetdata Target input data: A matrix of dimensions NumNodes by NumTimePoints.
preddata Optional: Input response data, if different from the target data.
q Number of nodes.
n Number of timepoints.
multipleVar TRUE when a specific variance is estimated for each segment, FALSE otherwise.
minphase Minimal segment length.
niter Number of MCMC iterations.
scaling If TRUE, scale the input data to mean 0 and standard deviation 1, else leave it unchanged.
method Network structure prior to use: 'poisson' for a sparse Poisson prior (no information sharing), 'exp_hard' or 'exp_soft' for the exponential information sharing prior with hard or soft node coupling, 'bino_hard' or 'bino_soft' with hard or soft node coupling.
prior.params Initial hyperparameters for the information sharing prior.
self.loops If TRUE, allow self-loops in the network, if FALSE, disallow self-loops.
k 
Initial value for the level-2 hyperparameter of the exponential information sharing prior.

options 
MCMC options as obtained e.g. by the function `defaultOptions`.

outputFile 
File where the output of the MCMC simulation should be saved.

fixed.edges 
Matrix of size `NumNodes` by `NumNodes`, with `fixed.edges[i,j]=1` if the edge between nodes `i` and `j` is fixed, and `-1` otherwise. Defaults to NULL (no edges fixed).

Value
A list containing the results of the MCMC simulation: network samples, changepoint samples and hyperparameter samples. For details, see `output`.

Author(s)
Sophie Lebre
Frank Dondelinger

References
For more information about the MCMC simulations, see:

See Also

output

---

`sampleBinit`  
*Sample initial regression coefficients.*

Description
This function samples the initial regression coefficients for the networks.

Usage

```r
sampleBinit(Si, sig2, delta2, X, q)
```

Arguments

- **Si** : Network structure.
- **sig2** : Sigma squared.
- **delta2** : Signal-to-noise ratio hyperparameter.
- **X** : Input data.
- **q** : Number of nodes.
**sampleBxy**

*Value*

Returns a vector of regression coefficients.

*Author(s)*

Sophie Lebre

*References*

For details of the regression model, see:

---

| sampleBxy | Sample regression coefficients. |

*Description*

This function samples the regression coefficients given the current state of the MCMC simulation.

*Usage*

```r
sampleBxy(xi, y, Sig2, delta2)
```

*Arguments*

- `xi` Response data.
- `y` Target data.
- `Sig2` Sigma squared.
- `delta2` Signal-to-noise hyperparameter.

*Value*

The regression parameters.

*Author(s)*

Sophie Lebre

*References*

For details of the regression model, see:
**sampleDelta2**

**Sample delta squared.**

---

**Description**

This function samples the signal-to-noise hyperparameter delta squared.

**Usage**

```
sampleDelta2(pos, x, q, B, S, sig2, alphad2, betad2)
```

**Arguments**

- **pos**
  - The current segment.
- **x**
  - Data,
- **q**
  - Number of nodes.
- **B**
  - Regression coefficients.
- **S**
  - Network structure.
- **sig2**
  - Sigma squared.
- **alphad2**
  - Gamma prior hyperparameter.
- **betad2**
  - Gamma prior hyperparameter.

**Value**

New sample of delta squared.

**Author(s)**

Sophie Lebre

**References**

For details of the sampling, see:

**sampleK**

Sample initial number of changepoints.

**Description**

This function samples the initial number of changepoints from a sparse Poisson prior.

**Usage**

```r
sampleK(mini, maxi, lambda, nb)
```

**Arguments**

- `mini`: Minimum value.
- `maxi`: Maximum value.
- `lambda`: Parameter of the Poisson distribution.
- `nb`: Number of values to sample.

**Value**

The sampled number of changepoints.

**Author(s)**

Sophie Lebre

**References**

For more information on the prior choice and sampling, see:


---

**sampleParms**

Sample initial parameters for the MCMC simulation.

**Description**

This function samples the initial hyperparameters and parameters that are needed for the MCMC simulation.

**Usage**

```r
sampleParms(X, GLOBvar, HYPERvar, s_init = NULL, options)
```
sampleSig2

Arguments

- `X`: Input data.
- `GLOBvar`: Global variables of the MCMC simulation.
- `HYPERvar`: Hyperparameter variables.
- `s_init`: Initial number of changepoints.
- `options`: MCMC options, as given by e.g. `defaultOptions`.

Value

Returns a list with elements:

- `E`: The initial changepoint vector.
- `S`: The initial networks structure.
- `B`: The initial regression parameters.
- `Sig2`: The initial sigma squared variances.
- `betas`: The initial hyperparameters for the exponential information sharing prior.
- `hyper_params`: The initial hyperparameters for the binomial information sharing prior.

Author(s)

- Sophie Lebre
- Frank Dondelinger

References

For more information about the parameters and hyperparameters, see:


See Also

- `init`

---

**sampleSig2**  
*Sample initial sigma squared.*

**Description**

This function samples the initial values for the sigma squared variance from the inverse gamma prior.

**Usage**

```r
sampleSig2(y, Px, v0, gamma0)
```
simulateNetwork

Arguments

- \( y \) (Input data).
- \( P \) (Projection matrix).
- \( v \) (Inverse gamma prior hyperparameter).
- \( \gamma \) (Inverse gamma prior hyperparameter).

Value

The sampled sigma squared values.

Author(s)

Sophie Lebre

References

For more information about the model, see:

simulateNetwork

Generate network and simulate data.

Description

This function generates a random network with structure changepoints (or takes one as input) and simulated data from it using a regression model.

Usage

```
simulateNetwork(l = 100, min_phase_length = 10, k_bar = 10, q = 10, 
lambda_2 = 0.45, noise = 0.25, net = NULL, lambda_3 = 2, 
spacing = 0, gauss_weights = FALSE, same = FALSE, 
changes = "sequential", fixed = FALSE, cps = NULL, saveFile = NULL)
```

Arguments

- \( l \) (Length of the time series).
- \( \text{min\_phase\_length} \) (Minimum segment length).
- \( k\_\text{bar} \) (Maximum number of changepoints).
- \( q \) (Number of nodes).
- \( \lambda_2 \) (Average number of parents for each node in the network (parameter for a Poisson distribution)).
noise Standard deviation of the Gaussian observation noise. Can be constant, or segment specific (in which case the number of changepoints needs to be fixed and the noise needs to be a vector of the same length).

net Input network, can be NULL if a new network should be generated.

lambda_S Average number of structure changes between two segments (parameter for a Poisson distribution).

spacing 1 if segments are equally spaced, 0 if they are spaced randomly (subject to the constraints of min_phase_length).

gauss_weights 1 if edge weights in the network are drawn from N(0, 1), 0 if they are fixed to be 1.

same 1 if the networks should all be the same (no changes), 0 otherwise.

changes 'sequential' if the changes happen sequentially (i.e. changes at segment i are applied to segment i-1), 'hierarchical' if the changes happen with respect to a hypernetwork (i.e. changes at segment i are applied to segment 0).

fixed T if the changepoint locations are fixed, F if they should be sampled.

cps Changepoint locations (if they are fixed).

saveFile If not NULL, indicates the filename for saving the output in R data format.

Value
A list with elements:

sim_data A matrix of length NumNodes by NumTimepoints containing the simulated data from the regression model.

epsilon Changepoint vector.

k Number of changepoints.

network The network, a list of length NumSegs, where each element is a NumNodes by NumNodes matrix.

noise The standard deviation of the applied Gaussian noise.

Author(s)
Frank Dondelinger

See Also
generateNetwork

Examples
# Generate random network and simulate data with default parameters
dataset = simulateNetwork()

# Generate random network and simulate data with an average of 1 change per node among network segments
dataset = simulateNetwork(lambda_S=1)
# Generate random network and simulate data with an average of 1 change per node among network segments and standard deviation of the Gaussian observation noise 0.5
dataset = simulateNetwork(lambda=1, noise=0.5)

# Generate random network with default parameters
network = generateNetwork()

# Simulate data using generated network
dataset = simulateNetwork(net=network)

# Generate random network with 4 changepoints and 15 nodes, with changepoints distributed over a timeseries of length 50
network = generateNetwork(l=50, q=15, fixed=TRUE, k_bar=4)

# Simulate data of length 50 using generated network
dataset = simulateNetwork(net=network)

updateSigMulti  

Update sigma squared variances.

Description
This function samples new values for the sigma squared variances, given the current network structure. A multivariate distribution is assumed.

Usage
updateSigMulti(phase, X, Y, E, Sall, Ball, Sig2, Mphase, alphad2, betad2, v0, gamma0)

Arguments
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phase</td>
<td>Current segment.</td>
</tr>
<tr>
<td>X</td>
<td>Input response data.</td>
</tr>
<tr>
<td>Y</td>
<td>Input target data.</td>
</tr>
<tr>
<td>E</td>
<td>Changepoints.</td>
</tr>
<tr>
<td>Sall</td>
<td>Network structure.</td>
</tr>
<tr>
<td>Ball</td>
<td>Regression coefficients.</td>
</tr>
<tr>
<td>Sig2</td>
<td>Current sigma squared values.</td>
</tr>
<tr>
<td>Mphase</td>
<td>Segment positions.</td>
</tr>
<tr>
<td>alphad2</td>
<td>Hyperparameter for gamma prior.</td>
</tr>
<tr>
<td>betad2</td>
<td>Hyperparameter for gamma prior.</td>
</tr>
<tr>
<td>v0</td>
<td>Hyperparameter for inverse gamma prior.</td>
</tr>
<tr>
<td>gamma0</td>
<td>Hyperparameter for inverse gamma prior.</td>
</tr>
</tbody>
</table>
**Value**

The new samples sigma squared values.

**Author(s)**

Sophie Lebre

**References**

For more information about the model, see:


**See Also**

updateSigSolo

---

**updateSigSolo**

*Sample new values for sigma squared.*

**Description**

This function samples new values for the sigma squared variances, given the current network structure. A univariate distribution is assumed.

**Usage**

```
updateSigSolo(X, Y, E, Sall, Ball, Sig2, Mphase, alphad2, betad2, v0, gamma0)
```

**Arguments**

- **X**  
  Input response data.

- **Y**  
  Input target data.

- **E**  
  Changepoints.

- **Sall**  
  Network structure.

- **Ball**  
  Regression coefficients.

- **Sig2**  
  Current sigma squared.

- **Mphase**  
  Segment position.

- **alphad2**  
  Gamma prior hyperparameter.

- **betad2**  
  Gamma prior hyperparameter.

- **v0**  
  Inverse gamma prior hyperparameter.

- **gamma0**  
  Inverse gamma prior hyperparameter.
**updateSigSolo**

**Value**

Returns the new samples sigma squared values.

**Author(s)**

Sophie Lebre

**References**

For more information about the model, see:


**See Also**

*updateSigMulti*
Index

*Topic package
   EDISON-package, 3

AcceptableMove, 4
addProposalNetworkInfo, 5
BinoHyperMove, 5, 6, 7, 32
BinoHyperRatio, 6
bp.computeAlpha, 7
buildXY, 8, 48
CalculateChanges, 9
calculateCPPGlobal, 10
calculateCPPProbabilities, 10, 10
calculateEdgeProbabilities, 11, 13
calculateEdgeProbabilitiesSegs, 12, 12, 13
calculateEdgeProbabilitiesTimePoints, 12, 13, 13
CalculateLikelihoodRatio, 14, 15, 17
CalculatePriorRatio, 14, 15, 36–39, 42
CollectNetworkInfo, 5, 6, 9, 15, 16, 27, 31, 35–39, 42
computePx, 17
computeRho4, 18
convert_nets, 19
corpcor, 3
cp.birth, 8, 19, 22, 23
cp.death, 8, 21, 21, 23
cp.shift, 21, 22, 22
defaultOptions, 23, 26, 32, 33, 50, 54
dinvgamma, 25, 49
EDISON (EDISON-package), 3
EDISON-package, 3
EDISON.run, 10, 11, 13, 25
ExHyperMove, 27, 29, 32
ExHyperRatioTarget, 28, 28
fix_eigenvalues, 29
generateNetwork, 29, 30, 56
HyperparameterMove, 31
HyperParms, 32
init, 33, 34, 54
main, 34
make_structure_move, 4, 35, 41
NetworkProbBino, 36, 38
NetworkProbExp, 37, 39
NetworkRatioBino, 36, 38
NetworkRatioExp, 37, 38
output, 39, 50
phase.update, 40
PriorRatioPoisson, 41
proposalTuning, 42
proposeContinuous, 43, 44
ProposeDiscrete, 43, 44
psrf, 45, 46, 47
psrf_check, 45, 46, 47
psrf_check_hyper, 45, 46, 47
readDataTS, 47
dinvgamma, 48
runDBN, 10, 11, 13, 26, 34, 49
sampleBinit, 50
sampleBxy, 51
sampleDelta2, 52
sampleK, 53
sampleParms, 34, 53
sampleSig2, 54
simulateNetwork, 26, 31, 55
updateSigMulti, 57, 59
updateSigSolo, 58, 58