Package: ebdbNet (via r-universe)

October 24, 2026

Type Package

Title Empirical Bayes Estimation of Dynamic Bayesian Networks

Version 1.2.8

Date 2023-09-04

Author Andrea Rau <andrea.rau@inra.fr>

Maintainer Andrea Rau <andrea.rau@inra.fr>

Depends R (>= 4.1.0), igraph

Imports graphics, stats, methods

Suggests GeneNet

Description Infer the adjacency matrix of a network from time course data using an empirical Bayes estimation procedure based on Dynamic Bayesian Networks.

License GPL (>= 3)

URL https://github.com/andreamrau/ebdbNet

LazyLoad yes

NeedsCompilation yes

Repository https://cran.r-universe.dev

Date/Publication 2023-09-04 17:30:43 UTC

RemoteUrl https://github.com/cran/ebdbNet

RemoteRef HEAD

RemoteSha e8a9534550d82ea32c23682c929f58d29f5b0939

Contents

ebdbNet-package 2
calcAUC 3
calcSensSpec 4
dataFormat 6
ebdbn 7

ebdbn-internal 10
hankel 11
plot.ebdbNet 12
simulateVAR 14

Index **16**

---

ebdbNet-package *Empirical Bayes Dynamic Bayesian Network (EBDBN) Inference*

Description

This package is used to infer the adjacency matrix of a network from time course data using an empirical Bayes estimation procedure based on Dynamic Bayesian Networks.

Details

Posterior distributions (mean and variance) of network parameters are estimated using time-course data based on a linear feedback state space model that allows for a set of hidden states to be incorporated. The algorithm is composed of three principal parts: choice of hidden state dimension (see `hankel`), estimation of hidden states via the Kalman filter and smoother, and calculation of posterior distributions based on the empirical Bayes estimation of hyperparameters in a hierarchical Bayesian framework (see `ebdbn`).

Author(s)

Andrea Rau

Maintainer: Andrea Rau <andrea.rau AT inra.fr>

References

Andrea Rau, Florence Jaffrezic, Jean-Louis Foulley, and R. W. Doerge (2010). An Empirical Bayesian Method for Estimating Biological Networks from Temporal Microarray Data. *Statistical Applications in Genetics and Molecular Biology* 9. Article 9.

Examples

```
library(ebdbNet)
library(GeneNet)  ## Load GeneNet package to use T-cell activation data
tmp <- runif(1) ## Initialize random number generator
set.seed(4568818) ## Set seed

## Load T-cell activation data
data(tcell)
tc44 <- combine.longitudinal(tcell.10, tcell.34)

## Put data into correct format for algorithm
## (List, with one matrix per replicate (P rows and T columns)
tcell.dat <- dataFormat(tc44)
```

```
## Use only subset of T-cell data for faster example
R <- 20 ## 20 replicates
P <- 10 ## 10 genes
tcell.sub.dat <- vector("list", R)
rep.sample <- sample(1:44, R)
for(r in 1:R) {
 tcell.sub.dat[[r]] <- tcell.dat[[rep.sample[r]]][sample(1:58, P),]
}

##################################################
# Run EBDBN (no hidden states) with feedback loops
##################################################
## Choose alternative value of K using hankel if hidden states to be estimated
## K <- hankel(tcell.sub.dat, lag = 1)$dim

## Run algorithm (feedback network, no hidden states)
net <- ebdbn(y = tcell.sub.dat, K = 0, input = "feedback", conv.1 = 0.01,
 conv.2 = 0.01, conv.3 = 0.001, verbose = TRUE)

## Visualize results: in this example, mostly feedback loops
## plot(net, sig.level = 0.5)
```

`calcAUC` *Calculate the Approximate Area Under the Curve (AUC)*

Description

Returns the approximate Area Under the Curve (AUC) of a Receiver Operating Characteristic (ROC) curve.

Usage

```
calcAUC(sens, cspec)
```

Arguments

`sens`	Vector of sensitivity values, calculated for varying thresholds
`cspec`	Vector of complementary specificity values, calculated for the same varying thresholds as `sens`

Details

Let TP, FP, TN, and FN represent the number of true positives, false positives, true negatives and false negatives of inferred network edges, respectively. Sensitivity is defined as

$$\frac{TP}{TP + FN}$$

and complementary specificity is defined as

$$\frac{TN}{TN + FP}$$

Note that sens and cspc should be in the same order with respect to the threshold value so that their elements correspond. That is, if the first element of sens was calculated at a threshold value of 0.01 and the second at a threshold value of 0.02, then the first element of cpsec should be also be calculated at a threshold value of 0.01 and the second at a threshold value of 0.02, and so on. The AUC is approximated using the trapezoid method, and can take real values between 0 and 1. An AUC of 0.5 indicates a classifier with random performance, and an AUC of 1 indicates a classifer with perfect performance.

Value

AUC of the ROC curve

Author(s)

Andrea Rau

Examples

```
library(ebdbNet)
tmp <- runif(1) ## Initialize random number generator

## Generate artificial values for sensitivity and complementary specificity.
fn <- function(x) {return(-1/(x^7)+1)}
set.seed(1459)
sens <- c(fn(seq(1, 2.7, length = 100)),1) ## Sensitivity
cspec <- seq(0, 1, by = 0.01) ## Complementary specificity

## Calculate the AUC of the ROC curve
AUC <- calcAUC(sens, cspec) ## AUC of this ROC curve is 0.9030868
```

---

calcSensSpec *Calculate Sensitivity and Specificity of a Network*

Description

Function to calculate the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) of an estimated network, given the structure of the true network.

Usage

```
calcSensSpec(trueMatrix, estMatrix)
```

Arguments

`trueMatrix`	Posterior mean or adjacency matrix of the true network
`estMatrix`	Posterior mean or adjacency matrix of the estimated network

Details

The matrices `trueMatrix` and `estMatrix` must be of the same dimension.

Value

`TP`	Number of true positives
`FP`	Number of false positives
`FN`	Number of false negatives
`TN`	Number of true negatives

Author(s)

Andrea Rau

See Also

`calcAUC`

Examples

```
library(ebdbNet)
tmp <- runif(1) ## Initialize random number generator
set.seed(16933) ## Set seed
P <- 10 ## 10 genes

## Create artificial true D matrix
Dtrue <- matrix(0, nrow = P, ncol = P)
index <- expand.grid(seq(1:P),seq(1:P))
selected.index <- sample(seq(1:(P*P)), ceiling(0.25 * P * P))
selected.edges <- index[selected.index,]
for(edge in 1:ceiling(0.25 * P * P)) {
 tmp <- runif(1)
 if(tmp > 0.5) {
  Dtrue[selected.edges[edge,1], selected.edges[edge,2]] <-
   runif(1, min = 0.2, max = 1)
 }
 else {
  Dtrue[selected.edges[edge,1], selected.edges[edge,2]] <-
   runif(1, min = -1, max = -0.2)
 }
}

## Create artificial estimated D matrix
Dest <- matrix(0, nrow = P, ncol = P)
index <- expand.grid(seq(1:P),seq(1:P))
```

```
selected.index <- sample(seq(1:(P*P)), ceiling(0.25 * P * P))
selected.edges <- index[selected.index,]
for(edge in 1:ceiling(0.25 * P * P)) {
 tmp <- runif(1)
 if(tmp > 0.5) {
  Dest[selected.edges[edge,1], selected.edges[edge,2]] <-
   runif(1, min = 0.2, max = 1)
 }
 else {
  Dest[selected.edges[edge,1], selected.edges[edge,2]] <-
   runif(1, min = -1, max = -0.2)
 }
}

check <- calcSensSpec(Dtrue, Dest)
check$TP ## 5 True Positives
check$FP ## 20 False Positives
check$TN ## 55 True Negatives
check$FN ## 20 False Negatives
```

`dataFormat` *Change the Format of Longitudinal Data to be Compatible with EBDBN*

Description

This function changes the format of longitudinal data to be compatible with the format required by the EBDBN, namely a list (of length R) of PxT matrices, where R, P, and T are the number of replicates, genes, and time points, respectively.

Usage

```
dataFormat(longitudinal.data)
```

Arguments

`longitudinal.data`
: Data in the longitudinal format

Details

The argument refers to the general data structure of the 'longitudinal' package.

Value

List of length R of PxT matrices, suitable to be used in the EBDBN algorithm.

Author(s)

Andrea Rau

Examples

```
library(ebdbNet)
library(GeneNet) ## Load GeneNet package to use T-cell activation data

data(tcell)  ## Load T-cell activation data
tc44 <- combine.longitudinal(tcell.10, tcell.34)

## Put data into correct format for algorithm
tcell.dat <- dataFormat(tc44)
```

---

ebdbn *Empirical Bayes Dynamic Bayesian Network (EBDBN) Estimation*

Description

A function to infer the posterior mean and variance of network parameters using an empirical Bayes estimation procedure for a Dynamic Bayesian Network (DBN).

Usage

```
ebdbn(y, K, input = "feedback", conv.1 = .15, conv.2 = .05, conv.3 = .01, verbose = TRUE,
max.iter = 100, max.subiter = 200)
```

Arguments

y	A list of R (PxT) matrices of observed time course profiles (P genes, T time points)
K	Number of hidden states
input	"feedback" for feedback loop networks, or a list of R (MxT) matrices of input profiles
conv.1	Value of convergence criterion 1
conv.2	Value of convergence criterion 2
conv.3	Value of convergence criterion 3
verbose	Verbose output
max.iter	Maximum overall iterations (default value is 100)
max.subiter	Maximum iterations for hyperparameter updates (default value is 200)

Details

An object of class `ebdbNet`.

This function infers the parameters of a network, based on the state space model

$$x_t = Ax_{t-1} + Bu_t + w_t$$

$$y_t = Cx_t + Du_t + z_t$$

where x_t represents the expression of K hidden states at time t, y_t represents the expression of P observed states (e.g., genes) at time t, u_t represents the values of M inputs at time t, $w_t \sim MVN(0, I)$, and $z_t \sim MVN(0, V^{-1})$, with $V = diag(v_1, \ldots, v_P)$. Note that the dimensions of the matrices A, B, C, and D are (KxK), (KxM), (PxK), and (PxM), respectively. When a network is estimated with feedback rather than inputs (`input` = "feedback"), the state space model is

$$x_t = Ax_{t-1} + By_{t-1} + w_t$$

$$y_t = Cx_t + Dy_{t-1} + z_t$$

The parameters of greatest interest are typically contained in the matrix D, which encodes the direct interactions among observed variables from one time to the next (in the case of feedback loops), or the direct interactions between inputs and observed variables at each time point (in the case of inputs).

The value of K is chosen prior to running the algorithm by using `hankel`. The hidden states are estimated using the classic Kalman filter. Posterior distributions of A, B, C, and D are estimated using an empirical Bayes procedure based on a hierarchical Bayesian structure defined over the parameter set. Namely, if $a_{(j)}$, $b_{(j)}$, $c_{(j)}$, $d_{(j)}$, denote vectors made up of the rows of matrices A, B, C, and D respectively, then

$$a_{(j)}|\alpha \sim N(0, diag(\alpha)^{-1})$$

$$b_{(j)}|\beta \sim N(0, diag(\beta)^{-1})$$

$$c_{(j)}|\gamma \sim N(0, diag(\gamma)^{-1})$$

$$d_{(j)}|\delta \sim N(0, diag(\delta)^{-1})$$

where $\alpha = (\alpha_1, ..., \alpha_K)$, $\beta = (\beta_1, ..., \beta_M)$, $\gamma = (\gamma_1, ..., \gamma_K)$, and $\delta = (\delta_1, ..., \delta_M)$. An EM-like algorithm is used to estimate the hyperparameters in an iterative procedure conditioned on current estimates of the hidden states.

`conv.1`, `conv.2`, and `conv.3` correspond to convergence criteria Δ_1, Δ_2, and Δ_3 in the reference below, respectively. After terminating the algorithm, the z-scores of the D matrix is calculated, which in turn determines the presence or absence of edges in the network.

See the reference below for additional details about the implementation of the algorithm.

Value

`APost`	Posterior mean of matrix A
`BPost`	Posterior mean of matrix B
`CPost`	Posterior mean of matrix C

DPost	Posterior mean of matrix D
CvarPost	Posterior variance of matrix C
DvarPost	Posterior variance of matrix D
xPost	Posterior mean of hidden states x
alphaEst	Estimated value of α
betaEst	Estimated value of β
gammaEst	Estimated value of γ
deltaEst	Estimated value of δ
vEst	Estimated value of precisions v
muEst	Estimated value of μ
sigmaEst	Estimated value of Σ
alliterations	Total number of iterations run
z	Z-statistics calculated from the posterior distribution of matrix D
type	Either "input" or "feedback", as specified by the user

Author(s)

Andrea Rau

References

Andrea Rau, Florence Jaffrezic, Jean-Louis Foulley, and R. W. Doerge (2010). An Empirical Bayesian Method for Estimating Biological Networks from Temporal Microarray Data. *Statistical Applications in Genetics and Molecular Biology* 9. Article 9.

See Also

hankel, calcSensSpec, plot.ebdbNet

Examples

```
library(ebdbNet)
tmp <- runif(1) ## Initialize random number generator
set.seed(125214) ## Save seed

## Simulate data
R <- 5
T <- 10
P <- 10
simData <- simulateVAR(R, T, P, v = rep(10, P), perc = 0.10)
Dtrue <- simData$Dtrue
y <- simData$y

## Simulate 8 inputs
u <- vector("list", R)
M <- 8
for(r in 1:R) {
```

```
 u[[r]] <- matrix(rnorm(M*T), nrow = M, ncol = T)
}

#####################################################
## Run EB-DBN without hidden states
#####################################################
## Choose alternative value of K using hankel if hidden states are to be estimated
## K <- hankel(y)$dim

## Run algorithm
net <- ebdbn(y = y, K = 0, input = u, conv.1 = 0.15, conv.2 = 0.10, conv.3 = 0.10,
 verbose = TRUE)

## Visualize results
## Note: no edges here, which is unsurprising as inputs were randomly simulated
## (in input networks, edges only go from inputs to genes)
## plot(net, sig.level = 0.95)
```

`ebdbn-internal` *Internal functions for Empirical Bayes Dynamic Bayesian Network (EBDBN) Estimation*

Description

Internal functions for the ebdbNet package.

Usage

```
sumFunc(x, cutoff)
fdbkFunc(y)
```

Arguments

`x`	Vector of singular values from singular value decomposition of block-Hankel matrix
`cutoff`	Value to determine cutoff to be considered for singular values (e.g., 0.90)
`y`	A list of R (PxT) matrices of observed time course profiles

Author(s)

Andrea Rau

See Also

`hankel`, `ebdbn`

hankel *Perform Singular Value Decomposition of Block-Hankel Matrix*

Description

This function constructs a block-Hankel matrix based on time-course data, performs the subsequent singular value decomposition (SVD) on this matrix, and returns the number of large singular values as defined by a user-supplied cutoff criterion.

Usage

```
hankel(y, lag, cutoff, type)
```

Arguments

y	A list of R (PxT) matrices of observed time course profiles
lag	Maximum relevant time lag to be used in constructing the block-Hankel matrix
cutoff	Cutoff to be used, determined by desired percent of total variance explained
type	Method to combine results across replicates ("median" or "mean")

Details

Constructs the block-Hankel matrix H of autocovariances of time series observations is constructed (see references for additional information), where the maximum relevant time lag must be specified as lag. In the context of gene networks, this corresponds to the maximum relevant biological time lag between a gene and its regulators. This quantity is experiment-specific, but will generally be small for gene expression studies (on the order of 1, 2, or 3).

The singular value decomposition of H is performed, and the singular values are ordered by size and scaled by the largest singular value. Note that if there are T time points in the data, only the first (T - 1) singular values will be non-zero.

To choose the number of large singular values, we wish to find the point at which the inclusion of an additional singular value does not increase the amount of explained variation enough to justify its inclusion (similar to choosing the number of components in a Principal Components Analysis). The user-supplied value of cutoff gives the desired percent of variance explained by the first set of K principal components. The algorithm returns the value of K, which may subsequently be used as the dimension of the hidden state in ebdbn.

The argument 'type' takes the value of "median" or "mean", and is used to determine how results from replicated experiments are combined (i.e., median or mean of the per-replicate final hidden state dimension).

Value

svs	Vector of singular values of the block-Hankel matrix H
dim	Number of large singular values, as determined by the user-supplied cutoff

Author(s)

Andrea Rau

References

Masanao Aoki and Arthur Havenner (1991). State space modeling of multiple time series. *Econometric Reviews* 10(1), 1-59.

Martina Bremer (2006). *Identifying regulated genes through the correlation structure of time dependent microarray data*. Ph. D. thesis, Purdue University.

Andrea Rau, Florence Jaffrezic, Jean-Louis Foulley, and R. W. Doerge (2010). An Empirical Bayesian Method for Estimating Biological Networks from Temporal Microarray Data. *Statistical Applications in Genetics and Molecular Biology* 9. Article 9.

Examples

```
library(ebdbNet)
tmp <- runif(1) ## Initialize random number generator
set.seed(125214) ## Save seed

## Simulate data
y <- simulateVAR(R = 5, T = 10, P = 10, v = rep(10, 10), perc = 0.10)$y

## Determine the number of hidden states to be estimated (with lag = 1)
K <- hankel(y, lag = 1, cutoff = 0.90, type = "median")$dim
## K = 5
```

---

plot.ebdbNet *Visualize EBDBN network*

Description

A function to visualize graph estimated using the Empirical Bayes Dynamic Bayesian Network (EBDBN) algorithm.

Usage

```
## S3 method for class 'ebdbNet'
plot(x, sig.level, interactive = FALSE, clarify = "TRUE",
layout = layout.fruchterman.reingold,  ...)
```

Arguments

x	An object of class "ebdbNet"
sig.level	Desired significance level (between 0 and 1) for edges in network
interactive	If TRUE, interactive plotting through tkplot

clarify	If TRUE, unconnected nodes should be removed from the plot
layout	Layout parameter for graphing network using igraph0
...	Additional arguments (mainly useful for plotting)

Details

For input networks, the default colors for nodes representing inputs and genes are green and blue, respectively. For feedback networks, the default color for all nodes is blue.

The interactive plotting option should only be used for relatively small networks (less than about 100 nodes).

Author(s)

Andrea Rau

References

Andrea Rau, Florence Jaffrezic, Jean-Louis Foulley, and R. W. Doerge (2010). An Empirical Bayesian Method for Estimating Biological Networks from Temporal Microarray Data. *Statistical Applications in Genetics and Molecular Biology* 9. Article 9.

See Also

ebdbn

Examples

```
library(ebdbNet)
tmp <- runif(1) ## Initialize random number generator
set.seed(125214) ## Save seed

## Simulate data
R <- 5
T <- 10
P <- 10
simData <- simulateVAR(R, T, P, v = rep(10, P), perc = 0.10)
Dtrue <- simData$Dtrue
y <- simData$y

## Simulate 8 inputs
u <- vector("list", R)
M <- 8
for(r in 1:R) {
 u[[r]] <- matrix(rnorm(M*T), nrow = M, ncol = T)
}

##################################################
## Run EB-DBN without hidden states
##################################################
## Choose alternative value of K using hankel if hidden states are to be estimated
## K <- hankel(y)$dim
```

```
## Run algorithm
## net <- ebdbn(y = y, K = 0, input = u, conv.1 = 0.15, conv.2 = 0.10, conv.3 = 0.10,
## verbose = TRUE)

## Visualize results
## plot(net, sig.level = 0.95)
```

`simulateVAR` *Simulate Simple Autoregressive Process*

Description

Function to simulate a simple autoregressive process based on a network adjacency matrix with a given percentage of non-zero elements.

Usage

```
simulateVAR(R, T, P, v, perc)
```

Arguments

`R`	Number of replicates
`T`	Number of time points
`P`	Number of observations (genes)
`v`	(Px1) vector of gene precisions
`perc`	Percent of non-zero edges in adjacency matrix

Details

Data are simulated with R replicates, T time points, and P genes, based on a first-order autoregressive process with Gaussian noise. The user can specify the percentage of non-zero edges to be randomly selected in the adjacency matrix.

Value

`Dtrue`	Adjacency matrix used to generate data (i.e., the true network)
`y`	Simulated data

Author(s)

Andrea Rau

See Also

ebdbn

Examples

```
library(ebdbNet)
tmp <- runif(1) ## Initialize random number generator
set.seed(125214) ## Save seed

## Simulate data
simData <- simulateVAR(R = 5, T = 10, P = 10, v = rep(10, 10), perc = 0.10)
Dtrue <- simData$Dtrue
y <- simData$y
```

Index

∗ **methods**
    calcAUC, 3
    calcSensSpec, 4
    dataFormat, 6
    ebdbn, 7
    ebdbn-internal, 10
    hankel, 11
    plot.ebdbNet, 12
    simulateVAR, 14
∗ **models**
    ebdbn, 7
    plot.ebdbNet, 12
∗ **package**
    ebdbNet-package, 2

calcAUC, 3, *5*
calcSensSpec, 4, *9*

dataFormat, 6

ebdbn, *2*, 7, *10, 11*, *13, 14*
ebdbn-internal, 10
ebdbNet *(*ebdbNet-package*)*, 2
ebdbNet-package, 2

fdbkFunc *(*ebdbn-internal*)*, 10

hankel, *2*, *8–10*, 11

plot.ebdbNet, *9*, 12

simulateVAR, 14
sumFunc *(*ebdbn-internal*)*, 10