Package 'ddepn'

July 2, 2014

Type Package

Title Dynamic Deterministic Effects Propagation Networks: Infer signalling networks for timecourse RPPA data.

Version 2.2

Date 2011-06-27

Author Christian Bender

Maintainer Christian Bender <christian.bender@tron-mainz.de>

Depends R (>= 2.14), lattice, coda, igraph, graph

Suggests parallel, Rgraphviz, BoolNet

Imports genefilter, gam, gplots

Description DDEPN (Dynamic Deterministic Effects Propagation Networks): Infer signalling networks for timecourse data. Given a matrix of high-throughput genomic or proteomic timecourse data, generated after external perturbation of the biological system, DDEPN models the time-dependent propagation of active and passive states depending on a network structure. Optimal network structures given the experimental data are reconstructed. Two network inference algorithms can be used: inhibMCMC, a Markov Chain Monte Carlo sampling approach and GA, a Genetic Algorithm network optimisation. Inclusion of prior biological knowledge can be done using different network prior models.

License GPL (>= 2)

LazyLoad TRUE

Repository CRAN

Repository/R-Forge/Project ddepn

Repository/R-Forge/Revision 85

Repository/R-Forge/DateTimeStamp 2013-08-12 21:48:26

Date/Publication 2013-08-13 08:41:00

NeedsCompilation yes

R topics documented:

`ddepn-package` *Dynamic Deterministic Effects Propagation Networks: Infer signalling networks from high througput array data.*

Description

Uses high throughput array data (e.g. from Reverse Phase Protein arrays or mRNA microarrays) to infer regulatory or signalling relationships among the measured proteins after different stimuli/inhibitions. Each stimulus/inhibition is included as a separate node and edges to the measured proteins are inferred. The signal flow through the network is modelled in a boolean fashion and a set of reachable system states is created. Afterwards, the optimal path through the possible system states is searched for using an HMM. Estimation of Gaussian distributions for active and passive states for each node depending on the activity state of the protein is done and a likelihood based score is calculated for a network. Network structure search is performed in either a genetic algorithm that optimises a population of candidate networks or via a markov chain monte carlo sampling that summarises the distribution over the network structures via sampling.

Details

Package:	ddepn
Type:	Package
License:	GPL
LazyLoad:	yes

Author(s)

Christian Bender

Maintainer: Christian Bender <c.bender@dkfz-heidelberg.de>

References

Bender et. al. 2010: Dynamic deterministic effects propagation networks: learning signalling pathways from longitudinal protein array data; Bioinformatics, Vol. 26(18), pp. i596-i602

See Also

ddepn, netga
Visit also the project page on R-forge: http://ddepn.r-forge.r-project.org/.

Examples

```
## Not run:
## load package
library(ddepn)

## sample a network
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
phit <- signet$phi
stimuli <- signet$stimuli

## sample data
dataset <- makedata(phit, stimuli, mu.bg=1200, sd.bg=400, mu.signal.a=2000, sd.signal.a=1000)

## use original network as prior matrix
## reset all entries for inhibiting edges
## to -1
B <- phit
B[B==2] <- -1

## Genetic algorithm, using BIC score as optimisation criterion
ret1 <- ddepn(dataset$datx, phiorig=phit,inference="netga",
              maxiterations=30, p=15, q=0.3, m=0.8,
              usebics=TRUE)
x11()
```

```
plotdetailed(ret1$phi,stimuli=stimuli,fontsize=20)

## Genetic algorithm, using a uniform prior
ret2 <- ddepn(dataset$datx, phiorig=phit, inference="netga",
    maxiterations=15, p=15, q=0.3, m=0.8,
    usebics=FALSE, priortype="uniform")
x11()
plotdetailed(ret2$phi,stimuli=ret2$stimuli,fontsize=20)

## Genetic algorithm, using a laplaceinhib prior and the posterior
## probabilities as optimisation criterion
ret2 <- ddepn(dataset$datx, phiorig=phit, inference="netga",
              maxiterations=30, p=15, q=0.3, m=0.8,
              usebics=FALSE, lambda=0.01, B=B, priortype="laplaceinhib")
x11()
plotdetailed(ret2$phi,stimuli=ret2$stimuli,fontsize=20)

## MCMC sampling using a uniform prior
ret3 <- ddepn(dataset$datx,phiorig=phit, inference="mcmc",
    maxiterations=300, burnin=100,
    usebics=FALSE, priortype="uniform")

## MCMC sampling using a laplaceinhib prior
ret3 <- ddepn(dataset$datx,phiorig=phit, inference="mcmc",
              maxiterations=300, burnin=100,
              usebics=FALSE, lambda=0.01, B=B, priortype="laplaceinhib")

x11()
plotdetailed(ret3$samplings[[1]]$phi,stimuli=ret3$samplings[[1]]$stimuli)

## MCMC sampling using a scale free prior
ret4 <- ddepn(dataset$datx,phiorig=phit, inference="mcmc",
              maxiterations=300, burnin=100,
              usebics=FALSE, gam = 2.2, it = 500, K = 0.8, priortype="scalefree")
x11()
plotdetailed(ret4$samplings[[1]]$phi,stimuli=ret4$samplings[[1]]$stimuli)



## End(Not run)
```

`addstimuli` *Add dummy stimulus nodes to the data matrix.*

Description

Add the treatments (stimuli, inhibitions) defined in the column names of the data matrix to the data matrix as dummy nodes, if not there.

Usage

```
addstimuli(dat)
```

Arguments

dat	The data matrix. See ddepn for more information.

Details

Uses the information from the column names of the data matrix to derive the input treatments (either stimuli or inhibition) for the experiment. Each stimulus is added as node to the network to be inferred, and thus as row to the data matrix. For example, activating treatments could be stimulation of cells by growth factors, inhibiting treatments could be treatment with a specific drug.

Value

The modified data matrix.

Author(s)

Christian Bender

See Also

ddepn

Examples

```
## Not run:
TODO

## End(Not run)
```

---

adjacencyMatrix_to_logicalRules

Convert adjacency matrix to logical rules.

Description

Convert adjacency matrices of ddepn consensus nets to logical activation/inhibition rules of network component, i.e. target, wiring for further analysis with the *BoolNet-package*, e.g. for perturbation simulations, according to its *loadNetwork* function format.

Usage

```
adjacencyMatrix_to_logicalRules(adjMatrix, outfile)
```

Arguments

adjMatrix	Adjacency matrix as resulting from create_signetwork for example.
outfile	Path to txt file where to store the output.

Details

The input adjacency matrix must have the network components, e.g. the protein names, as row names which must be identical to the column names. As typical for the consensus net obtained from ddepn network reconstructions, the following number coding holds for the matrix:

- 1: target in row activates target in column,
- 2: target in row inhibits target in column,
- 0: no relation of net components.

Value

A text file containing row-wise activation rules per network component.

Note

The format of the output file is especially adapted to the *loadNetwork* function format of the *BoolNet-package*.

Author(s)

Silvia von der Heyde

References

DDEPN
Bender et. al. 2010: Dynamic deterministic effects propagation networks: learning signalling pathways from longitudinal protein array data; Bioinformatics, Vol. 26(18), pp. i596-i602

BoolNet
S. A. Kauffman (1969), Metabolic stability and epigenesis in randomly constructed nets. J. Theor. Biol. 22:437–467.

S. A. Kauffman (1993), The Origins of Order. Oxford University Press.

See Also

create_signetwork, *loadNetwork*

Examples

```
## Not run:
library(ddepn)
library(BoolNet)

# create example adjacency matrix
example.mat <- matrix(sample(0:2, size=16, replace=TRUE), nrow=4,
 ncol=4, dimnames=list(x=paste("protein",letters[1:4],sep="_"),
  y=paste("protein",letters[1:4],sep="_")))

# define output file
example.out <- "exampleOutput.txt"

# convert adjacency matrix to logical rules
adjacencyMatrix_to_logicalRules(adjMatrix=example.mat,
 outfile=example.out)

## End(Not run)
```

`bestgam` *Fit an optimal smoothing spline to a data vector y, indexed by time vector tp.*

Description

For a data vector `y`, where `tp` assigns a time point to each value in `y`, a smoothing spline is fitted through the data points. To find an optimal spline, a series of fits is performed with increasing number of degrees of freedom and the optimal fit is selected via ANOVA testing and selection using the `selection.criterion`.

Usage

```
bestgam(y, tp, xn, selection.criterion = "aic")
```

Arguments

`y`	Numeric vector. The data points.
`tp`	Numeric vector. The corresponding time points.
`xn`	Numeric vector. The time values for the prediction.
`selection.criterion`	String. Either `"aic"` for Akaike's Information Criterion, `"BIC"` for Bayesian Information Criterion or `"pvalue"` for model selection according to the p-value.

Details

Several smoothing splines with increasing number of degrees of freedom are fit to the data (maximal number of df is one less than time points). Additionally, a constant and linear model fit is produced. The linear and smoothing spline fits are each compared to the constant fit via analysis of variance/deviance (ANOVA) to decide whether significant improvement of the complex model fits compared to the simple constant fit were got. This can also be seen as identification of a significant effect over time. To select a specific curve, the resulting p-values from the ANOVA are used. Either directly by selecting the minimal p-value of all fit comparisons, or by plugging in the p-value into AIC or BIC and maximising the respective statistic.

After the optimal fit is selected, a prediciton of the time profile is calculated, using the time vector `xn` as predictor variable in the fit.

`esub` is a helper for the gam fitting, usually not called directly.

Value

Returns a vector with the predicted values for the time vector `xn`.

Author(s)

Christian Bender
Tim Beissbarth
Stephan Gade

Examples

```
## Not run:
## load package
library(ddepn)
library(multicore)
library(gam)

## sample a network and data
set.seed(1234)
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
phit <- signet$phi
stimuli <- signet$stimuli
dataset <- makedata(phit, stimuli, mu.bg=1200, sd.bg=400, mu.signal.a=2000, sd.signal.a=1000)

## select an experiment

prepare_bestgam <- function(dataset, experiment, protein) {
  stimuli <- dataset$stimuli
  dat <- dataset$datx
  stim <- names(stimuli[[experiment]])
  sel <- grep(paste("^",stim,"_.*$",sep=""), colnames(dat))
  y <- dat[protein,sel]
```

```
  tp <- as.numeric(sub("^.*_","",colnames(dat)[sel]))
  #ord <- order(tp)
  #y <- y[ord]
  #tp <- tp[ord]
  xn <- seq(0,max(tp),0.1)
  return(list(y=y, tp=tp, xn=xn))
}

## no effect observed in experiment 1, protein 2
lst <- prepare_bestgam(dataset, 1, 2)
yhat <- bestgam(lst$y, lst$tp, lst$xn, "aic")
plot(lst$y~lst$tp)
lines(lst$xn,yhat)

## effect observed in experiment 1, protein 5
lst <- prepare_bestgam(dataset, 1, 5)
yhat <- bestgam(lst$y, lst$tp, lst$xn, "aic")
plot(lst$y~lst$tp)
lines(lst$xn,yhat)


## End(Not run)
```

`center_ddepn` *Perform centering of single replicate time courses.*

Description

For each experiment in a data matrix (i.e. stimulation/inhibition) subtract the median of the curve and shift up to the median of the whole experiment, i.e. median of all biological and technical replicates.

Usage

```
center_ddepn(dat)
```

Arguments

`dat`	The data matrix obtained in `hcc1954raw`, after fcf normalisation. See examples for how to fcf normalise and center the data.

Details

Used to center the replicate time courses. The time curves should lie as close together as possible, so that the estimated active/passive Gaussians estimated in `ddepn` are not obfuscated by the variance between the replicates. FCF normalising and centering will yield the data matrix obtained in `hcc1954`. The function is provided for convenience. It should be traceable, how the matrices were generated.

Value

The centered data matrix.

Author(s)

Christian Bender

See Also

ddepn hcc1954 hcc1954raw

Examples

```
## Not run:
library(ddepn)
data(hcc1954raw)

## perform FCF normalisation
datfcf <- hcc1954raw / hcc1954fcf * hcc1954fcfmedian

## center the data
datc <- center_ddepn(datfcf)

## End(Not run)
```

---

compareGraphs *compareGraphs*

Description

Given an original and inferred network, count number of true/false positives and negatives. Calculate sensitivity and specificity measures.

Usage

```
compareGraphs(phiorig,phi,ignore.type=FALSE)
```

Arguments

phiorig	Original graph. Stored as adjacency matrix.
phi	Inferred graph. Stored as adjaceny matrix.
ignore.type	Boolean. If TRUE, only compare edge abundance, if FALSE, also take into account the type of the edge for counting numbers of true and false edges.

Details

Counts the number of tp as: activations in phi and $phiorig$ + inhibitions in phi and $phiorig$.
Counts the number of fp as: activations or inhibitions in phi and no edge in $phiorig$.
Counts the number of tn as: no edge in phi and no edge in $phiorig$.
Counts the number of fn as: no edge in phi and activation or inhibition in $phiorig$.

Value

Vector of comparison measures:

`tp`	True positives
`fp`	False positives
`tn`	True negatives
`fn`	False negatives
`sn`	Sensitivity: $sn = tp/(tp + fn)$
`sp`	Specificity: $sp = tn/(tn + fp)$
`prec`	Precision: $prec = tp/(tp + fp)$
`f1`	F1: $2 * prec * sn/(prec + sn)$

Author(s)

Christian Bender

Examples

```
## Not run:
  library(ddepn)
  phi <- matrix(sample(c(0,1,2),9,replace=TRUE),nrow=3,ncol=3)
  phiorig <- matrix(sample(c(0,1,2),9,replace=TRUE),nrow=3,ncol=3)
  compareGraphs(phi,phiorig)

## End(Not run)
```

---

create_signetwork *inhibMCMC Analysis: find significant edges from independent MCMC chains*

Description

Uses statistical tests (Wilcox Rank-Sum or T-test) to find significant edges from a number of independent MCMC runs (performed by passing `inference="mcmc"` to the ddepn function call).

Usage

```
create_signetwork(ret, alpha = 0.05, adj.method = "BY", plot = FALSE,
type = "wilcox", alternative = "one.sided",
paired = FALSE, ord = NULL)

create_signetwork_cv(ret, alpha = 0.05, adj.method = "BY", plot = FALSE,
type = "wilcox", alternative = "one.sided",
paired = FALSE, ord = NULL, sel_policy = "strict")

perform_network_tests(ret, alpha=0.05, plot=FALSE, type="wilcox",
alternative="one.sided", paired=FALSE, ord=NULL)

perform_edge_test(ret, from, to, alpha=0.05, plot=TRUE, type="wilcox",
alternative="one.sided", paired=FALSE)

draw_segments(tx, rng, alpha)
```

Arguments

ret	List. Either the returned object of the ddepn call using the inhibMCMC inference (argument inference="mcmc"), or the sublist samplings from this object.
alpha	Numeric in $[0; 1]$. Significance level for the tests.
adj.method	String. Adjustment for multiple testing. See function p.adjust from the *stats* package for p-value adjustment methods. Most common settings are "none", "BH" or "none".
plot	Boolean. Should boxplots of the activation/inhibition edge counts (respectively confidences) be plotted?
type	String. Which test to use, either "wilcox" or "ttest".
alternative	String. Alternative to test. One out of "one.sided", "two.sided".
paired	Boolean. Paired test? Should normally be unpaired, i.e. FALSE.
ord	Vector of strings. Defines a node ordering.
from	String. Defines the source node, from which an edge originates. Used in perform_edge_test and must be one of the node names of the components in the network.
to	String. Defines the target node, to which an edge points. Used in perform_edge_test and must be one of the node names of the components in the network.
tx	Numeric Vector. Holds the pvalue for alternative greater and alternative less.
rng	The range of the values that were tested against each other.
sel_policy	String. If "strict", an edge is included in the final network if it occurred in all CV runs. If "medium", an edge is included if it occurs in more than half of the CV runs. If "lenient", an edge is included if it occurs in any of the CV runs.

Details

The main function is create_signetwork.
Using multiple inhibMCMC chains (a number L), a significance testing procedure is used to calculate signfificantly occurring edges. For each edge, the number of sampled activations and inhibitions

is counted in each run, divided by the total number of sampled edges, resulting in a 'confidence' for each edge for each run. To find significant edges, a statistical test is performed (usually Wilcoxon Rank Sum test) to test the NULL that the means of the *L* activation confidences (c_a) equal the mean of the *L* inhibition confidences (c_i). The alternatives $c_a > c_i$ and $c_a < c_i$ are used to determine the type of interaction.

In `create_signetwork_cv` a leave-one-out crossvalidation is performed to test for significant edges, i.e. every MCMC chain is left out once and the `create_signetwork` is called for the remaining runs. A final network is assembled from the CV networks according to the edge selection policy given in argument `sel_policy`. Note that at least 4 independent inhibMCMC chains have to be present to use this function.

The functions `perform_network_tests`, `perform_edge_test` and `draw_segments` are usually not called directly.

Value

An adjacency matrix containing the significant edges.

Note

Note that create_signetwork can only be performed for the results of the MCMC inference types and also only if multiple chains were run.

Author(s)

Christian Bender

References

Bender 2011, Systematic analysis of time resolved high-throughput data using stochastic network inference methods, Dissertation, University of Heidelberg, Faculty of Biology

See Also

ddepn, mcmc_ddepn

Examples

```
## Not run:
## load package
library(ddepn)
library(multicore)

## sample a network and data
set.seed(1234)
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
```

```
phit <- signet$phi
stimuli <- signet$stimuli
dataset <- makedata(phit, stimuli, mu.bg=1200, sd.bg=400,
mu.signal.a=2000, sd.signal.a=1000)

## use original network as prior matrix
## reset all entries for inhibiting edges
## to -1
B <- phit
B[B==2] <- -1

## perform inhibMCMC inference, using 4 CPU cores to get 4 MCMC chains
ret <- ddepn(dataset$datx, phiorig=phit, maxiterations=300, burnin=50,
            plotresults=FALSE, inference="mcmc",
            usebics=FALSE, priortype="laplaceinhib", lambda=0.01, B=B,
            multicores=TRUE, cores=4)

## get the signal network containing the significant edges
net <- create_signetwork(ret, alpha=0.05, adj.method="none", plot=FALSE,
 type="wilcox", alternative="one.sided", paired=FALSE, ord=NULL)

## compare inferred net and prior
par(mfrow=c(1,2))
plotdetailed(net, main="inferred", stimuli=ret$samplings[[1]]$stimuli )
plotdetailed(ret$samplings[[1]]$phi.orig, main="prior",
stimuli=ret$samplings[[1]]$stimuli )

## use the LOO crossvalidation for finding edges
net2 <- create_signetwork_cv(ret, alpha=0.05, adj.method="none",
plot=FALSE, type="wilcox", alternative="one.sided",
paired=FALSE, ord=NULL, sel_policy="strict")


## End(Not run)
```

---

ddepn *ddepn*

Description

Main function for DDEPN modelling. Takes a data matrix containing longitudinal measurements as argument and infers a network structure underlying the data using either a genetic algorithm or MCMC sampling.

Usage

```
ddepn(dat, phiorig=NULL, phi=NULL, th=0.8, inference="netga",
      outfile=NULL, multicores=FALSE, maxiterations=1000,
      p=500, q=0.3, m=0.8, P=NULL,
      usebics=TRUE, cores=1, priortype="laplaceinhib",
```

```
      lambda=NULL, B=NULL, samplelambda=NULL,
      hmmiterations=100, fanin=4,
      gam=NULL,it=NULL,K=NULL,quantL=.5,quantBIC=.5,
      debug=0, burnin=500, thin=FALSE, plotresults=TRUE,
      always_sample_sf=FALSE, scale_lik=FALSE, allow.stim.off=FALSE,
      implementation="C")

resume_ddepn(ret,maxiterations=10000,outfile=NULL,th=0.8,plotresults=TRUE,
debug=0,cores=NULL, implementation="C", thin=FALSE)
```

Arguments

dat	Matrix of double values. The data matrix to be used. Contains antibody measurements in the rows and experiments (T timepoints in each R replicates) in the columns. Each experiment is labeled by the respective perturbation in the column name. See section Details for an example.
phiorig	Adjacency matrix. Reference network used for comparison to the inferred net. Entries can be either 0, 1 or 2, for *no edge*, *activation* or *inhibition*, respectively. NULL if no reference network is given.
phi	Adjacency matrix. Seed network to start the search. Entries can be either 0, 1 or 2, for *no edge*, *activation* or *inhibition*, respectively. NULL if no start network should be given, but initialised automatically.
th	Threshold for inclusion of an edge in the final network (for netga). If an edge occurs more than $th * p$ times in all individuals, it is included in the resulting network.
inference	String. Giving the type of network search. netga Uses a genetic algorithm for network inference. mcmc MCMC sampling for network inference.
outfile	String. Output path for plotting. NULL if plotting should be done to the display.
multicores	Boolean. TRUE for using multiple cores and parallelise the network reconstruction. In case of netga the HMMs for each individual in the population are distributed on multiple cores. In case of mcmc, several independent MCMC runs are started, each on a separate core. FALSE for standard calculation on only one core (needs R-package multicore).
maxiterations	Integer, Maximum number of generations in netga or maximum number of iterations in mcmc_ddepn.
p	Integer, number of individuals in the population in netga.
q	Double $\in [0; 1]$, selection (1-q) and crossover (q) rate in netga.
m	Double $\in [0; 1]$, mutation rate in netga.
P	List containing an initial population of networks for netga. Set to NULL if start population should be generated automatically.
usebics	Use BIC statistic for model selection (only for netga).

cores	Number of cores to use in case of multicores=TRUE. For netga, the parallel calculations of the HMMs are distributed on cores cores, for mcmc cores independent MCMC runs are started. In resume_ddepn, cores is used for resuming a netga run, while for resuming an mcmc run, the argument is omitted and derived from the mcmc return object.
hmmiterations	Integer. Maximum number of iterations in the HMM search.
lambda	NULL, Numeric or NA. The Prior influence hyperparameter for the laplace prior. If numeric, used as fixed prior strength or starting value for prior strength sampling (when samplelambda is numeric, too). If NA, lambda is integrated out in the calculation of the prior. If NULL, no laplace prior is used.
B	The Prior information matrix. See prior for details.
fanin	Integer: maximal indegree for each node.
gam	Prior influence strength for scalefree prior. Also used as exponent in laplaceinhib prior: see prior for details.
it	Number of iterations to generate the background distribution for scalefree prior.
K	Proportionality factor for scalefree prior.
quantL	Quantile of Population Likelihood/Posterior, used as selection threshold in netga. Note that the Likelihood or Posterior have to be maximised, so all networks with a likelihood/posterior *greater* than this threshold are selected.
quantBIC	Quantile of Population BIC, used as selection threshold in netga. Note that the BIC is minimised, so all networks with BIC *less* than the threshold are selected.
samplelambda	Numeric or NULL. If NULL, the Laplace hyperparameter lambda is kept fix during the MCMC inference. If numeric, lambda is sampled uniformly around the initial value of lambda, with an interval size defined by samplelambda.
debug	Numeric. If 0, a status bar indicates the progress of the algorithm. If 1 or 2, extra information is printed to the console (for debug=2 more information than for debug=1).
burnin	Integer. Specifies the number of iterations used as burnin phase for mcmc_ddepn.
priortype	Character. One of none, uniform, laplaceinhib, laplace or scalefree for use of the respective prior type. Ignored if usebics=TRUE for netga. For netga, usebics=FALSE, priortype="none" means optimising the likelihood directly. This is equivalent to setting usebics=FALSE, priortype="uniform". For mcmc_ddepn, priortype="none" is not allowed. Use priortype="uniform" instead. laplaceinhib uses prior information for edges with two types (activation/inhibition), laplace ignores the edge type. Useful if only knowledge about the presence of an edge is available, but not about its type. scalefree assumes scale-free network architectures.
thin	Boolean. If TRUE, makes sure that the MCMC return objects are shortened to at most 10000 iterations. Defaults to FALSE.
plotresults	Boolean. If TRUE, the resulting network(s) and in case of MCMC sampling, the score traces are plotted.
always_sample_sf	Boolean. Update scaling factor in inhibMCMC sampling through the whole sampling if TRUE. Keep scaling factor fixed after burn-in if FALSE.

`scale_lik` Boolean. Perform scaling of the likelihood according to how many data points were used to calculate the overall likelihood.

`allow.stim.off` Boolean. If TRUE, the stimulus can become passive at some time. This will generate additional reachable system states, in particular all states from the normal state matrix, generated by the propagation, but with the stimulus node set to 0.

`ret` List. The output generated during an `netga` or `mcmc_ddepn` run. Used in function `resume_ddepn` to resume the inference.

`implementation` String. One of `"C","R","R_globalest","C_globalest"`. Different implementations of the HMM in `perform.hmmsearch`. If `"R"`, the original pure R-implementation is used, if `"C"`, a ported C-implementation is used. If `"R_globalest"`, an experimental version of the parameter estimation is used in the HMM, `"C_globalest"` is the C-port of this version. See `details` for a description.

Details

dat Data matrix. Rows correspond to measured proteins/genes etc. Columns contain all experiments, i.e. separate perturbations. Each experiment `i` consists of `T_i` time points and each time point is assumed to be measured in `R_i` replicates. The time is indicated as a numeric value, separated by an underscore in the column name. Example:

	EGF_1	EGF_1	EGF_2	EGF_2	EGF&X_1	EGF&X_2	EGF&X_2	EGF&X_2
EGF	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	0	0
AKT	1.45	1.8	0.99	1.6	1.78	1.8	1.56	1.58
ERK	1.33	1.7	1.57	1.3	0.68	0.34	0.62	0.47
MEK	0.45	0.8	0.99	0.6	0.78	0.8	0.56	0.58

For example, `EGF_1` means EGF treatment at time 1, `EGF&X_2` means simultaneous treatment with EGF and X at time 2 etc. One could use function `addstimuli` to automatically add the additional rows for the treatments to the data matrix, if they are not present. Unequal numbers of time points and replicates are allowed for each experiment. See the vignette for more details on the format of the data matrix.

implementation Several implementations are provided, differing in the way that the Gaussian parameters are estimated in the HMM. The `"R"` and `"C"` implementations derive separate optimal state matrices for each provided experiment. The state matrices are then concatenated to estimate the Gaussians. An alternative experimental implementation `"R_globalest"` is available, which derives a single state matrix for all experiments in the HMM. For separate derivation, the corresponding gaussians for each experiment can be rather different, leading to rather inhomogeneous parameter estimates with large variances. Using only one HMM for all experiments overcomes this problem, since the states are chosen with respect to all experiments. However, deriving the combined state matrix leads to higher number of possible system states to be regarded in the viterbi algorithm, and this will slow down the HMM. The default is to use `"C"` with a reasonable trade off of quality and speed.

Value

For `netga`*, a list containing the following elements:*

`dat`	Double matrix. The data matrix.
`phi.activation.count`	Integer. Counts how often an edge is an activation in the population.
`phi.inhibition.count`	Integer. Counts how often an edge is an inhibition in the population.
`phi.orig`	Adjacency matrix. The reference network, if it was provided.
`phi`	Adjacency matrix. The inferred network
`weights`	Matrix. Each entry is the maximum of the conf.act/conf.inh entries. I.e. this describes the support for an edge in the final network.
`weights.tc`	Matrix. Similar to weights, but calculated ignoring the types of the edges.
`stats`	Matrix. Contains result statistics for each network in the population: TP, FP, TN, FN, Sensitivity(SN), Specificity(SP), precision, F1. Only present if a reference network `phi.orig` was provided in the function call to `ddepn`.
`conf.act`	Matrix. Calculated as phi.activation.count/p
`conf.inh`	Matrix. Calculated as phi.inhibition.count/p
`stimuli`	List. The list of the input stimuli in format `list(c(Stim1=1),c(Stim1=1,Stim2=2))`. The first element in this example list is a single stimulus, the second a combinatorial stimulus of Stim1 and Stim2. The numbers are the indices identifying the nodes, i.e. the index in `rownames(dat)`. This is generated automatically from the formatted data matrix (see section *details*).
`P`	List. The population of networks that was inferred, i.e. the return list of `netga`.
`scorestats`	Matrix. Contains traces of the scores during the genetic algorithm. See `netga`.

For `mcmc`*, a list containing two elements:*

`samplings`	List. Contains all sampling runs. Each sampling run itself is a list as obtained via `mcmc_ddepn`.
`ltraces`	Matrix. Contains the posterior traces, each trace stored in one column of the matrix.

Note

TODO

Author(s)

Christian Bender

References

DDEPN
Bender et. al. 2010: Dynamic deterministic effects propagation networks: learning signalling pathways from longitudinal protein array data; Bioinformatics, Vol. 26(18), pp. i596-i602

Laplace prior
Bender, C. 2011: Systematic analysis of time resolved high-throughput data using stochastic network inference methods; PhD Thesis, University of Heidelberg, Combined Faculties for the Natural Sciences and for Mathematics, 2011

Froehlich et. al. 2007, Large scale statistical inference of signaling pathways from RNAi and microarray data; BMC Bioinformatics, Vol. 8(11), pp. 386ff

Scale free prior
Kamimura and Shimodaira, A Scale-free Prior over Graph Structures for Bayesian Inference of Gene Networks

See Also

TODO

Examples

```
## Not run:
## load package
library(ddepn)

## sample a network
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
phit <- signet$phi
stimuli <- signet$stimuli

## sample data
dataset <- makedata(phit, stimuli, mu.bg=1200, sd.bg=400,
                    mu.signal.a=2000, sd.signal.a=1000)

## use original network as prior matrix
## reset all entries for inhibiting edges
## to -1
B <- phit
B[B==2] <- -1

## Genetic algorithm, no prior
ret1 <- ddepn(dataset$datx, phiorig=phit, inference="netga",
              maxiterations=30, p=15, q=0.3, m=0.8,
              usebics=TRUE)
x11()
plotdetailed(ret1$phi,stimuli=ret1$stimuli)
```

```
## mcmc, laplaceinhib prior
ret2 <- ddepn(dataset$datx,phiorig=phit, inference="mcmc",
              maxiterations=300, burnin=100,
              usebics=FALSE, lambda=0.01, B=B, gam=1,
              priortype="laplaceinhib")

x11()
plotdetailed(ret2$samplings[[1]]$phi,stimuli=ret2$samplings[[1]]$stimuli)

## use mcmc with multiple cores, i.e. perform two independent runs
## requires package multicore and, of course multiple cores in the hardware
## use the original net as prior
 if(require(parallel)) {
  ret3 <- ddepn(dataset$datx,phiorig=phit, inference="mcmc",
                multicores=TRUE, cores=2,
                maxiterations=300, burnin=100,
                usebics=FALSE, lambda=0.01, B=B, gam=1,
                priortype="laplaceinhib")
 }

## resuming the inference from an inhibMCMC run and add another 100 iterations
ret4 <- ddepn(dataset$datx,phiorig=phit, inference="mcmc",
maxiterations=100, burnin=30, lambda=0.01, B=B,
priortype="laplaceinhib", usebics=FALSE)
ret4 <- resume_ddepn(ret4,maxiterations=100)

## resuming the inference from an netga run and add another 30 iterations
ret5 <- ddepn(dataset$datx,phiorig=phit, inference="netga",
maxiterations=20, p=10, q=0.3, m=0.8, lambda=0.01, B=B,
priortype="laplaceinhib", usebics=FALSE)
ret5 <- resume_ddepn(ret5,maxiterations=30)


## End(Not run)
```

`format_ddepn` *Performs renaming of the colnames of dataset hcc1954 to make it suitable as input for the inference.*

Description

A little helper that strips of the information on biological replicates in the example data matrix `hcc1954`. After stripping the matrix can be directly used as input for `ddepn`.

Usage

```
format_ddepn(dat)
```

Arguments

dat	A data matrix in the format described in ddepn, except for the column names, which still contain a label for the biological replicate. See the details for an example.

Details

Example for column names holding information on biological replicates:

	EGF-3_1	EGF-5_1	EGF-3_1	EGF-5_1	EGF-3_2	EGF-5_2	EGF-3_2	EGF-5_2
EGF	...							
X	...							
AKT	...							

The format is STIMULUS-biolreplicate_time, i.e. EGF-3_1 means experiment with EGF stimulation, biological replicate labeled as 3 at time point 1. The biological replicate should be labeled by numbers only.

Author(s)

Christian Bender

See Also

hcc1954

Examples

```
## Not run:
  ## load package
  library(ddepn)
  data(hcc1954)
  format_ddepn(hcc1954)

## End(Not run)
```

---

gelman_diag *Convergence check by Gelman's potential scale reduction.*

Description

Compute Gelman's potential scale reduction to estimate convergence in inhibMCMC.

Usage

```
gelman_diag(ret)
```

Arguments

ret	List. Object returned by ddepn, when inhibMCMC was used as inference method.

Details

The function is a wrapper calling the gelman.diag function from package coda.

Value

The result of the gelman.diag function.

Note

Note that multiple chains have to be performed to use this diagnostic.

References

Gelman, A., Carlin, JB., Stern, HS., Rubin, DB.: Bayesian Data Analysis, 2nd edition, Chapman & Hall/CRC, chapter 11.6, pp294-295.

See Also

coda package.

Examples

```
## Not run:
library(ddepn)
set.seed(12345)
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
net <- signet$phi
stimuli <- signet$stimuli
weights <- signet$weights

## sample data
dataset <- makedata(net, stimuli, mu.bg=1200, sd.bg=400, mu.signal.a=2000, sd.signal.a=1000)
data <- dataset$datx

# use the original network as prior probability matrix
B <- net
B[B==2] <- -1
# construct a prior matrix with uniform probabilities for each edge
if(require(multicore)) {
  ret <- ddepn(data, phiorig=net, inference="mcmc",
    maxiterations=3000, burnin=1000,
    usebics=FALSE, lambda=0.01, B=B,
    multicores=TRUE, cores=4, priortype="laplaceinhib")
}

## now produce the convergence diagnostic
```

```
gelman_diag(ret)


## End(Not run)
```

`get.phi.final` *Construct final network from GA or mcmc result.*

Description

`get.phi.final` takes the output of `netga` and constructs a final network from the population, `get.phi.final.mcmc` takes a list containing the returned lists of `mcmc_ddepn` and calculates a final network for each result list.

Usage

```
get.phi.final(lst, th = 0.8)
get.phi.final.mcmc(retlist,maxiterations,prob=.333,qu=.99999)
```

Arguments

`lst`	Output list from `netga`
`th`	Double in [0;1]. Threshold for inclusion of an edge into the final network.
`retlist`	A list returned by MCMC inference.
`maxiterations`	Integer. Number of MCMC sampling iterations.
`prob`	Double $\in [0;1]$.Success probability for binomial density.
`qu`	Double $\in [0;1]$. Quantile of the binomial distribution, used as significance cut-off for edge inclusion.

Details

`get.phi.final` Takes the population P from the GA resultlist and returns the list with the element *lst$phi* replaced by the new final network. *lst$weights* only contains the weights with *lst$weights* > *th*.
`get.phi.final.mcmc` Takes a list of MCMC samplings returned by `mcmc_ddepn` and extracts a final network for each result. It is assumed that any type of edge is expected to be found with probability $1/3 = 0.33$, thus for `maxiterations` samplings we expect the probability of seeing exactly `k` edges of a certain type follows a binomial distribution `B(maxiterations, prob, k)`. An edge is included, if the probability of seeing more than the observed number of occurrences is less or equal than `1-qu`, i.e. $P(k > Kobserved) <= 1 - qu$.

Value

Result list as in `netga`, with replaced *phi* and *weights* elements.

Author(s)

Christian Bender

See Also

netga

Examples

```
## Not run:
## load package
library(ddepn)
## sample a network
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
phit <- signet$phi
stimuli <- signet$stimuli
## sample data
dataset <- makedata(phit, stimuli, mu.bg=1200, sd.bg=400, mu.signal.a=2000, sd.signal.a=1000)
## Genetic algorithm
ret1 <- ddepn(dataset$datx, phiorig=phit, inference="netga",
              maxiterations=30, p=15, q=0.3, m=0.8, P=NULL,
              usebics=TRUE)

plotrepresult(ret1,outfile=NULL)
ret2 <- get.phi.final(ret1, th=0.9)
plotrepresult(ret2,outfile=NULL)

## mcmc
maxiterations <- 300
## use original network as prior matrix
## reset all entries for inhibiting edges
## to -1
B <- phit
B[B==2] <- -1

ret3 <- ddepn(dataset$datx,phiorig=phit,
      inference="mcmc", usebics=FALSE,
      maxiterations=maxiterations, burnin=100, lambda=0.01, B=B)

plotrepresult(ret3$samplings[[1]],outfile=NULL)
ret4 <- get.phi.final(ret3$samplings[[1]],th=0.9)
plotrepresult(ret4,outfile=NULL)


## End(Not run)
```

hcc1954 *hcc1954*

Description

Example phosphoproteomic data set. Phosphorylation of 16 proteins from the ERBB signalling network were measured in 10 time points (between 0 and 60 minutes) in HCC1954 cell line. Measurements were generated via Reverse Phase Protein Array technique.

Format

Matrix 16x540. Rows hold proteins, Columns hold the Experiments and time points.
Column naming: STIMULUS_biolreplicate_time
Rows naming: Protein_Phosphorylation site.

Details

Use `data(hcc1954)` to obtain the FCF normalised and replicate centered data matrix or `data(hcc1954raw)` to obtain the raw data matrices. The latter comprise the raw data `hcc1954raw`, the FCF values `hcc1954fcf` and the medians of the experiment FCF values `hcc1954fcfmedian`. See examples for how to transform the raw to the normalised data.

The following experiments were performed: stimulation with ...
...EGF
...HRG
...EGF&HRG

The following proteins were measured:

protein_phosphorylationsite	entrez id
pEGFR_Y1068	1956
pERBB2_Y1112	2064
pERK12_T202Y204	5595 5594
pAKT_S473	207 208
pPDK1_S241	5170
pMEK_S217S221	5604 5605
pPLCgamma_S1248	5335 5336
pPKCalpha_S657Y658	5578
pp38_T180Y182	1432
pSRC_Y416	6714
pmTOR_S2448	2475
pp70S6K_T389	6198
pGSK3_Y279Y216	2931 2932
pPRAS_T246	100136753
pERBB3_Y1289	2065
pERBB4_Y1162	2066

Note

To use these data matrices as inputs for ddepn, the information on the biological replicates in the column names have to be removed using the function format_ddepn, since all replicates are treated equally during inference. The information on the biological replicates was left in the data matrix, however, since it might be useful for different approaches, too.

References

Bender et. al. 2010: Dynamic deterministic effects propagation networks: learning signalling pathways from longitudinal protein array data; Bioinformatics, Vol. 26(18), pp. i596-i602

See Also

format_ddepn center_ddepn

Examples

```
## Not run:

library(ddepn)

## get the normalised data matrix
data(hcc1954)
colnames(hcc1954)
rownames(hcc1954)

## get the raw matrices
data(hcc1954raw)

## perform FCF normalisation
datfcf <- hcc1954raw / hcc1954fcf * hcc1954fcfmedian

## center the data, datc is then equal to the hcc1954 matrix
datc <- center_ddepn(datfcf)

## End(Not run)
```

---

kegggraphs *kegggraphs*

Description

Contains a list of selected KEGG pathways, downloaded in October 2010.

Format

List of 78, each Element holding the KEGG pathway id and an adjacency matrix.

Details

The KEGG Hierarchy was downloaded and transformed into graphNEL objects by using the KEGGgraph R-package. Each graphNEL was converted to simple adjacency matrices. A selection of pathways was performed to restrict to the signalling and disease pathways:

The following pathways are included:

KEGG pathway ID	Name
04010	MAPK signaling pathway
04012	ErbB signaling pathway
04020	Calcium signaling pathway
04060	Cytokine-cytokine receptor interaction
04080	Neuroactive ligand-receptor interaction
04110	Cell cycle
04115	p53 signaling pathway
04120	Ubiquitin mediated proteolysis
04130	SNARE interactions in vesicular transport
04140	Regulation of autophagy
04150	mTOR signaling pathway
04210	Apoptosis
04310	Wnt signaling pathway
04320	Dorso-ventral axis formation
04330	Notch signaling pathway
04340	Hedgehog signaling pathway
04350	TGF-beta signaling pathway
04360	Axon guidance
04370	VEGF signaling pathway
04510	Focal adhesion
04512	ECM-receptor interaction
04514	Cell adhesion molecules (CAMs)
04520	Adherens junction
04530	Tight junction
04540	Gap junction
04610	Complement and coagulation cascades
04612	Antigen processing and presentation
04614	Renin-angiotensin system
04620	Toll-like receptor signaling pathway
04630	Jak-STAT signaling pathway
04640	Hematopoietic cell lineage
04650	Natural killer cell mediated cytotoxicity
04660	T cell receptor signaling pathway
04662	B cell receptor signaling pathway
04664	Fc epsilon RI signaling pathway
04670	Leukocyte transendothelial migration
04710	Circadian rhythm
04720	Long-term potentiation
04730	Long-term depression

04740	Olfactory transduction
04742	Taste transduction
04810	Regulation of actin cytoskeleton
04910	Insulin signaling pathway
04912	GnRH signaling pathway
04914	Progesterone-mediated oocyte maturation
04916	Melanogenesis
04920	Adipocytokine signaling pathway
04930	Type II diabetes mellitus
04940	Type I diabetes mellitus
04950	Maturity onset diabetes of the young
05010	Alzheimer's disease
05012	Parkinson's disease
05014	Amyotrophic lateral sclerosis (ALS)
05110	Vibrio cholerae infection
05120	Epithelial cell signaling in Helicobacter pylori infection
05130	Pathogenic Escherichia coli infection - EHEC
05131	Pathogenic Escherichia coli infection - EPEC
05200	Pathways in cancer
05210	Colorectal cancer
05211	Renal cell carcinoma
05212	Pancreatic cancer
05213	Endometrial cancer
05214	Glioma
05215	Prostate cancer
05216	Thyroid cancer
05217	Basal cell carcinoma
05218	Melanoma
05219	Bladder cancer
05220	Chronic myeloid leukemia
05221	Acute myeloid leukemia
05222	Small cell lung cancer
05223	Non-small cell lung cancer
05310	Asthma
05320	Autoimmune thyroid disease
05322	Systemic lupus erythematosus
05330	Allograft rejection
05332	Graft-versus-host disease
05340	Primary immunodeficiency

Source

http://www.genome.jp/kegg/

References

Kanehisa: Kyoto Encyclopaedia of Genes and Genomes

Examples

```
## Not run:
library(ddepn)
data(kegggraphs)
names(kegggraphs)

## End(Not run)
```

`makedata` *Generate artificial dataset given a network and list of stimuli.*

Description

Dataset generation using an input network, list of stimuli and various experiment parameters.

Usage

```
makedata(phi, stimuli, R.t = 3, R.b = 3, TT = 10, mu.bg = 0, sd.bg = 0.1,
mu.signal.a = 2, sd.signal.a = 0.5, mu.signal.i = -2, sd.signal.i = 0.5,
allow.stim.off = FALSE)
```

Arguments

`phi`	Adjacency matrix.The input network.
`stimuli`	List containing all stimuli.
`R.t`	Integer. Number of technical replicates. So far, no distinction is done between technical and biological replicates, so in total a number of $R.t \cdot R.b$ replicates is generated. If given as vector of length equal to the length of `stimuli`, then each element denotes the numbers of replicates in each experiment, denoted by the corresponding element in `stimuli`.
`R.b`	Integer. Number of biological replicates. So far, no distinction is done between technical and biological replicates, so in total a number of $R.t \cdot R.b$ replicates is generated. Can also be given as vector, with the same effect as for `R.t`.
`TT`	Integer.Number of timepoints in the experiment. Analogously to `R.t` and `R.b`, can be given as vector, denoting the numbers of time points in each separate experiment.
`mu.bg`	Double. Mean background intensity (for passive state of the protein.
`sd.bg`	Double. Sd for background intensity.
`mu.signal.a`	Double. Mean intensity for activation.
`sd.signal.a`	Double. Sd for activation intensity.
`mu.signal.i`	Double. Mean intensity for inhibition.
`sd.signal.i`	Double. Sd for inhibition intensity.
`allow.stim.off`	Boolean. If TRUE, a stimulus can become inactive at some time point, if FALSE, the stimulus will be always active.

Details

Generates a dataset from the given network, stimuli and parameters.

Value

A list containing the dataset:

datx	The data matrix.
gammax	Matrix of true state transitions.
stimuli	The list of stimuli.
phi	The network.
R.t	Technical replicates.
R.b	Biological replicates.
TT	Number of timepoints.

Author(s)

Christian Bender

Examples

```
## Not run:
library(ddepn)
dataset <- makedata(matrix(sample(c(0,1,2),9,replace=TRUE),nrow=3,
dimnames=list(LETTERS[1:3],LETTERS[1:3])), list(A=1))

## End(Not run)
```

---

mcmc_ddepn *mcmc_ddepn - Perform MCMC sampling for DDEPN.*

Description

MCMC sampling for DDEPN. Takes an initial network and samples from the posterior. runmcmc is a wrapper function for multiple calls of mcmc_ddepn, in case that multiple cores are used for parallel MCMC runs.

Usage

```
mcmc_ddepn(dat, phiorig=NULL, phi=NULL, stimuli=NULL,
    th=0.8, multicores=FALSE, outfile=NULL, maxiterations=10000,
    usebics=FALSE, cores=2, lambda=NULL, B=NULL,Z=NULL,
    samplelambda=NULL, hmmiterations=30, fanin=4,
    gam=NULL, it=NULL, K=NULL, burnin=1000, priortype="laplaceinhib",
    plotresults=TRUE,always_sample_sf=FALSE, scale_lik=FALSE,
    allow.stim.off=TRUE,debug=0,retobj=NULL, implementation="C")
```

```
runmcmc(x,dat,phiorig,phi,stimuli,th,multicores,outfile,maxiterations,
usebics,cores,lambda,B,Z,samplelambda,hmmiterations,
fanin,gam,it,K,burnin,priortype,
plotresults=TRUE,always_sample_sf=FALSE,
scale_lik=FALSE, allow.stim.off=TRUE,debug=0,
retobj=NULL, implementation="C")
```

Arguments

dat	The data matrix.
phiorig	The reference network to compare to. Can be NULL.
phi	The start network. Empty if NULL.
stimuli	The stimuli list.
th	Threshold for inclusion of an edge in the final network.
multicores	Use multiple cores. Not used here.
outfile	File to which the network should be drawn.
maxiterations	Integer. Maximum number of MCMC iterations.
usebics	Use bics for model selection.
cores	Not used here.
lambda	NULL, Numeric or NA. The Prior influence hyperparameter for the laplace prior. If numeric, used as fixed prior strength or starting value for prior strength sampling (when samplelambda is numeric, too). If NA, lambda is integrated out in the calculation of the prior. If NULL, no laplace prior is used.
B	The Prior information matrix. See prior for details.
Z	Normalisation factor for prior.
hmmiterations	Maximum iterations in the HMM.
fanin	Integer: maximal indegree for nodes.
gam	Prior influence strength in scalefree prior.
it	Number of iterations to generate background distribution in scalefree prior.
K	Proportionality factor in scalefree prior
samplelambda	Numeric or NULL. If NULL, the Laplace hyperparameter lambda is kept fix during the MCMC inference. If numeric, lambda is sampled uniformly around the initial value of lambda, with an interval size defined by samplelambda.
x	List containing two items: An adjacency matrix phi and a string outfile describing a path.
burnin	Integer. Specifies the number of iterations used as burnin phase for mcmc_ddepn.
priortype	Character. One of none, laplaceinhib, laplace or scalefree for use of the respective prior type. Ignored if usebics=TRUE. For netga, usebics=FALSE, priortype="none" means optimising the likelihood directly, for mcmc_ddepn, none is not allowed.
plotresults	Boolean. If TRUE, some statistics are plotted while inhibMCMC is running.

`always_sample_sf`
: Boolean. Update scaling factor in inhibMCMC sampling through the whole sampling if TRUE. Keep scaling factor fixed after burn-in if FALSE.

`scale_lik`
: Boolean. Perform scaling of the likelihood according to how many data points were used to calculate the overall likelihood.

`allow.stim.off`
: Boolean. If TRUE, the stimulus can become passive at some time. This will generate additional reachable system states, in particular all states from the normal state matrix, generated by the propagation, but with the stimulus node set to 0.

`debug`
: Numeric. If 0, a status bar indicates the progress of the algorithm. If 1 or 2, extra information is printed to the console (for `debug=2` more information than for `debug=1`).

`retobj`
: List. The output generated during an inhibMCMC run (see `ddepn` for argument `inference="mcmc"`). Passed by function `resume_ddepn` to resume the inference.

`implementation`
: String. One of `"C"`,`"R"`,`"R_globalest"`,`"C_globalest"`. Different implementations of the HMM in `perform.hmmsearch`. If `"R"`, the original pure R-implementation is used, if `"C"`, a ported C-implementation is used. If `"R_globalest"`, an experimental version of the parameter estimation is used in the HMM, `"C_globalest"` is the C-port of this version. See `details` for a description.

Details

Usually this function is called internally by `ddepn`.

Value

A list of the following elements:

`phi`
: The inferred network.

`L`
: Likelihood.

`aic`
: Akaikes Information Criterion.

`bic`
: Bayesian Information Criterion.

`posterior`
: Posterior probability.

`dat`
: The data matrix.

`theta`
: The parameter matrix for the gaussians.

`gamma`
: The state transition matrix.

`gammaposs`
: The theoretical state transition matrix, as generated by the effect propagation.

`tps`
: A list. Each element is a vector of time points for each experiment in the data matrix.

`stimuli`
: List of stimuli.

`reps`
: Number of replicates for each experiment.

`hmmiterations`
: Maximum number of iterations during an HMM run.

`lastmove`
: Type of the last change that was performed.

coords	Position in the network where the last change was performed.
lambda	Laplace prior hyperparameter.
B	Laplace prior matrix.
Z	Laplace prior normalisation factor. (Not used at the moment.)
pegm	Probability of performing the last move.
pegmundo	Probability of reverting the last move.
nummoves	Total number of possible moves in the current step.
fanin	Maximal indegree for nodes.
gam	Sparsity prior hyperparameter.
it	Sparsity prior iterations.
K	Sparsity prior scaling factor.
conf.act	Matrix of beliefs that an edge is an activation. (equals $freqa/eoccur$)
conf.inh	Matrix of beliefs that an edge is an inhibition. (equals $freqi/eoccur$)
eoccur	Matrix of total occurrences of edges.
phi.orig	Adjacency of reference network, if given.
stats	Matrix of scores and statistics recorded during MCMC.
freqa	Counts how often an edge was an activation.
freqi	Counts how often an edge was an activation.
mu_run	The running mean values of the parameters in theta.
Qi	A helper for the calculation of sd_run.
sd_run	The running standard deviations of the parameters in theta.

Note

TODO

Author(s)

Christian Bender

References

TODO

See Also

ddepn

Examples

```
## Not run:
## load package
library(ddepn)

## sample a network
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
phit <- signet$phi
stimuli <- signet$stimuli

## sample data
dataset <- makedata(phit, stimuli, mu.bg=1200, sd.bg=400,
mu.signal.a=2000, sd.signal.a=1000)

## prior normalisation factor
lambda <- 0.01

## network to start with
V <- rownames(dataset$datx)
phistart <- matrix(0, nrow=n, ncol=n, dimnames=list(V,V))

## use original network as prior matrix
## reset all entries for inhibiting edges
## to -1
B <- phit
B[B==2] <- -1

## now the sampling
ret <- mcmc_ddepn(dataset$datx, phiorig=phit, phi=phistart, stimuli=stimuli,
            th=0.8, multicores=FALSE, outfile=NULL, maxiterations=300,
            usebics=FALSE, cores=1, lambda=lambda, B=B,
            hmmiterations=100, fanin=4, burnin=100, priortype="laplaceinhib")

## End(Not run)
```

`netga` *netga - Perform network search using a genetic algorithm.*

Description

Uses a genetic algorithm for network search. Combines selection of best models, crossing over of nodes and connected edges between pairs of nodes and mutation of edges to sample the search space.

Usage

```
netga(dat, stimuli, P=NULL, maxiterations=1000, p=100,
```

```
q=0.3, m=0.8, hmmiterations=30, multicores=FALSE, usebics=FALSE,
cores=2, lambda=NULL, B=NULL, Z=NULL, scorefile=NULL, fanin = 4,
gam=NULL,it=NULL,K=NULL,quantL=.5,quantBIC=.5,priortype="none",
plotresults=TRUE, scale_lik=FALSE, allow.stim.off=TRUE,
debug=0,retobj=NULL, implementation="C")
```

Arguments

dat	The data matrix. See ddepn
stimuli	The list of stimuli. See ddepn
P	A list containing a population of network models. If NULL, an initial population of networks is sampled including the empty and fully connected network.
maxiterations	Integer, maximal number of generations.
p	The size of the population. Used only if P is not provided.
q	Double $\in [0;1]$. The rate of selection or crossing over. At most $(1-q)\cdot\|P\|$ nodes are selected to persist into the next generation, but only if the fitness score is more optimal than the median of scores in the population. If less than $(1-q)*\|P\|$ fulfill this criterion, the number $q\cdot\|P\|$ of crossing overs is increased accordingly.
m	Double $\in [0;1]$. The mutation rate - $m \cdot \|P\|$ networks out of the Population are mutated *after* selection and crossing over. Mutations happen for edges and change the type of the edge randomly.
hmmiterations	Maximum number of iterations during the HMM.
multicores	Use multiple processors.
usebics	Use BIC statistic for model selection.
cores	Number of cores to use in case of multicores=TRUE
lambda	The Prior influence strength in the laplace prior.
B	The Prior information matrix. Corresponds to prior edge probabilities in the final network.
Z	The normalisation factor for the prior distribution.
fanin	Integer: maximal indegree for nodes.
gam	Prior influence strength in scalefree prior. Used as exponent in laplaceinhib prior: see prior for details.
it	Number of iterations to generate background distribution in scalefree prior.
K	Proportionality factor in scalefree prior
quantL	Quantile of Population likelihood. Defines the selection threshold in netga. Also used as posterior selection quantile. Note that the Likelihood or Posterior have to be maximised, so all networks with a likelihood/posterior *greater* than the threshold are selected.
quantBIC	Quantile of Population likelihood. Defines the selection threshold in netga. Note that the BIC is minimised, so all networks with BIC *less* than the threshold are selected.

scorefile	Name of pdf-file where intermediate traces of the BIC, the differences of BICs in two succeeding generations and a comparison of optimum and average fitness are drawn. If NULL, the plot is drawn on the standard device for plotting.
priortype	Character. One of none, laplaceinhib, laplace or scalefree for use of the respective prior type. Ignored if usebics=TRUE. For netga, usebics=FALSE, priortype="none" means optimising the likelihood directly, for mcmc_ddepn, none is not allowed.
plotresults	Boolean. If TRUE, some statistics are plotted while netga is running.
scale_lik	Boolean. Perform scaling of the likelihood according to how many data points were used to calculate the overall likelihood.
allow.stim.off	Boolean. If TRUE, the stimulus can become passive at some time. This will generate additional reachable system states, in particular all states from the normal state matrix, generated by the propagation, but with the stimulus node set to 0.
debug	Numeric. If 0, a status bar indicates the progress of the algorithm. If 1 or 2, extra information is printed to the console (for debug=2 more information than for debug=1).
retobj	List. The output generated during a GA run (see function ddepn for argument inference="netga"). Passed by function resume_ddepn to resume the inference.
implementation	String. One of "C","R","R_globalest","C_globalest". Different implementations of the HMM in perform.hmmsearch. If "R", the original pure R-implementation is used, if "C", a ported C-implementation is used. If "R_globalest", an experimental version of the parameter estimation is used in the HMM, "C_globalest" is the C-port of this version. See details for a description.

Details

Usually this function is called internally by ddepn.

Value

A list containing the following objects:
A list named P holding the model objects for each candidate network in the population. Each entry in the list contains the following elements:

phi	Adjacency matrix. The inferred network.
L	Likelihood.
aic	Akaikes Information Criterion.
bic	Bayesian Information Criterion.
posterior	Posterior probability. Only present if usebics=FALSE, i.e. when a prior model is used during inference.
dat	The data matrix.
theta	The parameter matrix for the Gaussians.
gamma	The state transition matrix.
gammaposs	The theoretical state transition matrix, as generated by the effect propagation.

tps	A list. Each element is a vector of time points for each experiment in the data matrix.
stimuli	List of stimuli.
reps	Number of replicates for each experiment.
hmmiterations	Maximum number of iterations during an HMM run.
lastmove	String. Type of the last change that was performed.
coords	Position in the network where the last change was performed.
lambda	Laplace prior hyperparameter.
B	The Prior information matrix. See prior for details.
Z	Laplace prior normalisation factor. (Not used at the moment.)
fanin	Maximal indegree for nodes.
gam	Sparsity prior hyperparameter.
it	Sparsity prior iterations.
K	Sparsity prior scaling factor.
priortype	The prior that was used.

A matrix named scorestats holding in each column the traces of the following statistics, each row corresponds to one iteration:

dL_total	Median of total Likelihood change.
dP_total	Median of total Prior change.
dL_crossover	Median of Likelihood change for crossover.
dL_mutation	Median of Likelihood change for mutation.
dP_crossover	Median of Prior change for crossover.
dP_mutation	Median of Prior change for mutation.
dL_total_abs	As above, but absolute values.
dP_total_abs	As above, but absolute values.
dL_crossover_abs	As above, but absolute values.
dL_mutation_abs	As above, but absolute values.
dP_crossover_abs	As above, but absolute values.
dP_mutation_abs	As above, but absolute values.
score	Trace of the total median score. The score depends on the type of inference, if usebics=TRUE it is the BIC score, if usebics=FALSE and priortype="none", it is the Liklihood score, otherwise the posterior score.

Note

TODO

Author(s)

Christian Bender

References

TODO

See Also

ddepn

Examples

```
## Not run:
## load package
library(ddepn)

## sample a network
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
phit <- signet$phi
stimuli <- signet$stimuli

## sample data
dataset <- makedata(phit, stimuli, mu.bg=1200, sd.bg=400, mu.signal.a=2000, sd.signal.a=1000)

## now the GA
ret <- netga(dataset$dat, stimuli, P=NULL, maxiterations=30, p=15,
    q=0.3, m=0.8, hmmiterations=100, multicores=FALSE, usebics=TRUE, cores=1,
    lambda=NULL, B=NULL, scorefile=NULL, priortype="none")
## now the GA with prior
## use original network as prior matrix
## reset all entries for inhibiting edges
## to -1
B <- phit
B[B==2] <- -1
ret <- netga(dataset$dat, stimuli, P=NULL, maxiterations=30, p=15,
    q=0.3, m=0.8, hmmiterations=100, multicores=FALSE, usebics=FALSE, cores=1,
    lambda=0.01, B=B, scorefile=NULL, priortype="laplaceinhib")



## End(Not run)
```

plotdetailed *plotdetailed, layout.ellipsis*

Description

plotdetailed uses igraph to plot a graph containing activations and inhibitions. Input is an adjacency matrix with activation edge represented by a 1, inhibition by a 2.

Usage

```
plotdetailed(phi, weights = NULL, main = "", stimuli = NULL,
  node.color = "grey", node.size1 = 30, node.size2 = 7, edge.width = 1,
  edge.arrowsize = 0.5, layout = layout.circle, pdf = NULL, pointsize = 12,
  edge.width.inhib = 1.5, plot.legend = TRUE,
  label.cex = 1, vlabel.cex = 0.6, tk = FALSE, fontsize=20, rescale=TRUE)

layout.ellipsis(ig, a=1, b=1.5)
```

Arguments

phi	Adjacency matrix. The network to be plotted.
weights	Optional matrix of edge weights.
main	Optional character string containing main title for the plot.
stimuli	List of input stimuli. Are marked as red filled nodes in the plot.
fontsize	Deprecated. Still there for compatibility issues with plotdetailed_Rgraphviz. Use cex.label to control edge label size.
node.color	Define coloring of nodes.
node.size1	Width of a rectangular node.
node.size2	Height of a rectangular node.
edge.width	Width of edges.
edge.arrowsize	Size of edge ends.
layout	Graph layout algorithm or matrix with node coordinates.
pdf	Pdf output file name.
pointsize	Pointsize argument used in pdf device.
edge.width.inhib	Width of inhibitory edges.
plot.legend	Define if legend should be drawn or not.
label.cex	Edge label size factor.
vlabel.cex	Node label size factor.
tk	Use tk-plotting engine or not.

rescale	Rescaling coordinates or not.
ig	Igraph object.
a	Semi major axis of ellipse.
b	Semi minor axis of ellipse.

Details

Plot a graph encoded in adjacency matrix phi using the igraph library. Activation edges are drawn as solid lines, inhibition edges as dashed lines. Stimuli nodes can be colored red using the stimuli argument.

The stimuli list describes the input stimuli or inhibitions. Each of these external treatment nodes are drawn as red nodes in the graph. The format of the list should be:\ Example: Assume two treatments EGF and X, a data matrix data:

	EGF_1	EGF_1	EGF_2	EGF_2	EGF&X_1	EGF&X_2	EGF&X_2	EGF&X_2
EGF	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	0	0
AKT	1.45	1.8	0.99	1.6	1.78	1.8	1.56	1.58

The stimuli list should be in the format: `stimuli <- list(c(EGF=1),c(EGF=1,X=2))`, i.e. each element in the list corresponds to a treatment. Each treatment is a named numeric vector describing the row indices of the treatments in the data matrices.

layout.ellipsis is another graph layout making elliptical graphs possible. Can be used as layout argument for plotting. Use rescale=FALSE if using layout.ellipsis.

Value

A list object holding the igraph object ig and the layout coordinates layout:

ig	Igraph object created from the adjacency matrix.
layout	Matrix with two columns, holding the x and y coordinates of the elements in the graph.

Author(s)

Christian Bender

References

none

See Also

plotdetailed_Rgraphviz

Examples

```
## Not run:
library(ddepn)
mat <- matrix(sample(c(0,1,2),25,replace=TRUE), nrow=5,ncol=5,
dimnames=list(LETTERS[1:5],LETTERS[1:5]))
diag(mat) <- 0
weights <- matrix(sample(c(1:25),25,replace=TRUE), nrow=5,ncol=5,
dimnames=list(LETTERS[1:5],LETTERS[1:5]))
stimuli <- list(list(A=1))
plotdetailed(mat,weights,"Some random graph",stimuli,layout=layout.graphopt)
layoutmat <- cbind(c(1,2,3,4,5), c(5,4,3,2,1))
plotdetailed(mat,weights,"Some random graph",stimuli,layout=layoutmat)

## End(Not run)
```

---

plotdetailed_Rgraphviz

plotdetailed_Rgraphviz get.labels get.arrowhead get.arrowtail

---

Description

plotdetailed_Rgraphviz uses Rgraphviz to plot a graph containing activations and inhibitions. Input is an adjacency matrix with activation edge represented by a 1, inhibition by a 2.

Usage

```
plotdetailed_Rgraphviz(phi,weights=NULL,main="",stimuli=NULL,
layoutType = "dot", fontsize=20)
get.labels(phi)
get.arrowhead(phi)
get.arrowtail(phi)
```

Arguments

phi	Adjacency matrix. The network to be plotted.
weights	Optional matrix of edge weights.
main	Optional character string containing main title for the plot.
stimuli	List of input stimuli. Are marked as red filled nodes in the plot.
layoutType	The graphviz layout for drawing the network.
fontsize	Fontsize of text in the plot.

Details

`get.labels`, `get.arrowhead` and `get.arrowtail` are helper functions for the plot construction and usually not called directly.

The `stimuli` list describes the input stimuli or inhibitions. Each of these external treatment nodes are drawn as red nodes in the graph. The format of the list should be:\ Example: Assume two treatments `EGF` and `X`, a data matrix data:

	EGF_1	EGF_1	EGF_2	EGF_2	EGF&X_1	EGF&X_2	EGF&X_2	EGF&X_2
EGF	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	0	0
AKT	1.45	1.8	0.99	1.6	1.78	1.8	1.56	1.58

The stimuli list should be in the format: `stimuli <- list(c(EGF=1),c(EGF=1,X=2))`, i.e. each element in the list corresponds to a treatment. Each treatment is a named numeric vector describing the row indices of the treatments in the data matrices.

Value

none

Author(s)

Christian Bender

Examples

```
## Not run:
library(ddepn)
mat <- matrix(sample(c(0,1,2),25,replace=TRUE), nrow=5,ncol=5,
dimnames=list(LETTERS[1:5],LETTERS[1:5]))
weights <- matrix(sample(c(1:25),25,replace=TRUE), nrow=5,ncol=5,
dimnames=list(LETTERS[1:5],LETTERS[1:5]))
stimuli <- list(list(A=1))
plotdetailed_Rgraphviz(mat,weights,"Some random graph",stimuli,layoutType="dot")
plotdetailed_Rgraphviz(mat,weights,"Some random graph",stimuli,layoutType="neato")

## End(Not run)
```

---

plotrepresult *Plot result of reconstruction.*

Description

Summary plot of the result of a network reconstruction.

Usage

```
plotrepresult(lst, outfile, fontsize=15)
```

Arguments

`lst`	Return list of `netga`
`outfile`	Path to pdf file where to plot.
`fontsize`	Font size of text in the plot.

Details

TODO

Value

TODO

Note

TODO

Author(s)

Christian Bender

References

TODO

See Also

TODO

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

---

plot_edgeconfidences *Boxplots of edge confidences obtained in inhibMCMC.*

Description

Plot the confidences for each edge, obtained from L independent MCMC chains.

Usage

```
plot_edgeconfidences(ret, start = 1, stop = NULL, act = "conf.act",
inh = "conf.inh", cex.axis = 1.5, ...)

make_edge_df(samp)
```

Arguments

ret	List. Either the returned object of the ddepn call using the inhibMCMC inference (argument inference="mcmc"), or the sublist samplings from this object.
start	Numeric. Optional, defines the start of the subset of the iterations in the MCMC chains which are used to generate the boxplots.
stop	Numeric. Optional, defines the end of the subset of the iterations in the MCMC chains which are used to generate the boxplots.
act	String. Defines the statistic to be used from the MCMC results. One of "conf.act" or "freqa", for activation confidence or frequency of activations.
inh	String. Defines the statistic to be used from the MCMC results. One of "conf.inh" or "freqi", for inhibition confidence or frequency of inhibitions.
samp	List. Contains the sampling runs from each MCMC chain. Note that the number of chains $L := length(samp)$.
cex.axis	Numeric. Scale factor for the axis labels.
...	Further plotting options passed to the lattice panel.bwplot function.

Details

Create a summary plot of edge confidences or counts over *L* MCMC runs. Assume that *N* is the number of nodes in the inferred network. Each panel in the summary plot contains *N* boxes showing the activation and *N* boxes showing the inhibition confidences/frequencies. Activation boxes are shown in blue, inhibition boxes in red. The column name in each panel defines the source node, from which an edge originates. The panel name denotes the destination node to which the edge points.

Value

none

Author(s)

Christian Bender

Examples

```
## Not run:
## load package
library(ddepn)
library(multicore)

## sample a network and data
set.seed(1234)
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
phit <- signet$phi
stimuli <- signet$stimuli
dataset <- makedata(phit, stimuli, mu.bg=1200, sd.bg=400, mu.signal.a=2000, sd.signal.a=1000)
```

```
## use original network as prior matrix
## reset all entries for inhibiting edges
## to -1
B <- phit
B[B==2] <- -1

## perform inhibMCMC inference, using 4 CPU cores to get 4 MCMC chains
ret <- ddepn(dataset$datx, phiorig=phit, maxiterations=300, burnin=50,
            plotresults=FALSE, inference="mcmc",
            usebics=FALSE, priortype="laplaceinhib", lambda=0.01, B=B,
            multicores=TRUE, cores=4)

plot_edgeconfidences(ret, act="conf.act", inh="conf.inh",pch="|")


## End(Not run)
```

`plot_mcmctraces` *Trace plot for inhibMCMC posterior traces.*

Description

Plots the traces of posterior probabilities during an inhibMCMC run.

Usage

```
plot_mcmctraces(ret, thin = 1)
```

Arguments

`ret`	List. Object returned by ddepn call, using argument `inference="mcmc"` for inhibMCMC sampling.
`thin`	The thinning interval between consecutive observations.

Details

Plot the MCMC posterior traces of all runs during inhibMCMC inference. Uses a thinning interval defined in `thin`.

Value

none

Note

Uses the coda package and its mcmc object and plot functions.

Author(s)

Christian Bender

See Also

ddepn

Examples

```
## Not run:
## load package
library(ddepn)
library(multicore)
library(coda)

## sample a network and data
set.seed(1234)
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
phit <- signet$phi
stimuli <- signet$stimuli
dataset <- makedata(phit, stimuli, mu.bg=1200, sd.bg=400, mu.signal.a=2000, sd.signal.a=1000)

## use original network as prior matrix
## reset all entries for inhibiting edges
## to -1
B <- phit
B[B==2] <- -1

## perform inhibMCMC inference, using 4 CPU cores to get 4 MCMC chains
ret <- ddepn(dataset$datx, phiorig=phit, maxiterations=300, burnin=50,
            plotresults=FALSE, inference="mcmc",
            usebics=FALSE, priortype="laplaceinhib", lambda=0.01, B=B,
            multicores=TRUE, cores=4)

plot_mcmctraces(ret,thin=1)
plot_mcmctraces(ret,thin=10)


## End(Not run)
```

---

plot_profiles *Plot the data and inferred model parameters.*

Description

Creates a time profile for each experiment and protein in the data set. A spline fit is shown representing the trend of the intensities over time. Finally, model parameter estimations obtained in the inference are laid over the plots to show the active or passive states.

Usage

```
plot_profiles(ret, log=FALSE, ord=NULL, mfrow=c(4,4),
              plotcurves=TRUE, plothist=TRUE, selection.criterion="aic")

get_theta_consensus(ret)

get_gamma_consensus(ret)

heatmapcolors(dat,ncol,lowcol="green",highcol="red",middlecol="white")
```

Arguments

ret	List or numeric data matrix. Either an object returned by ddepn with arguments inference="netga" for Genetic Algorithm optimisation or inference="mcmc" for inhibMCMC sampling. Can also be a raw data matrix, in which case no parameter estimates are shown.
log	Boolean. Transform data to log scale.
ord	Vector of Strings. Optional, defines a node order for the plots.
mfrow	Vector of numerics. Two values giving the number of rows and columns in which the output plot should be arranged.
plotcurves	Boolean. Show the data profiles and fitted splines or not.
plothist	Boolean. Show additional histogram representation of the data or not.
selection.criterion	For the spline fits to the data, choose a model selection criterion from "aic", "bic" or "pvalue" (for Akaike Information Criterion, Bayesian Information Criterion or p-value selection. This will affect how many degrees of freedom are used for the spline fit.
dat	The data to be represented by the colour values. Is used to retrieve the range of values for which the colour palette is created.
ncol	Integer. The number of colours in the panel.
lowcol	A colour string, either keyword or hex-representation. Defining the colour used for the passive state.
middlecol	A colour string, either keyword or hex-representation. Defining the colour in the middle of the palette.
highcol	A colour string, either keyword or hex-representation. Defining the colour used for the active state.

Details

Plots for each protein and experiment the data points along the time axis (as boxplots). Fits a smoothing spline to the data (see bestgam), using model selection criterion selection.criterion for deciding on the number of degrees of freedom. If ret is a list (from GA or inhibMCMC inference), the estimated model parameters for the Gaussian active and passive distributions are extracted and shown in the timecourse plots (active: red, passive:green). If ret is a numeric matrix,

only the data will be plotted. Boxplots are coloured using a diverging colourpanel (created by `heatmapcolors` from green to red, indicating the state of the node at the respective time point.

Value

none

Note

Needs package *gam* for the spline fitting.

Author(s)

Christian Bender

See Also

`bestgam`

Examples

```
## Not run:
## load package
library(ddepn)
library(multicore)
library(gam)

## sample a network and data
set.seed(1234)
n <- 6
signet <- signalnetwork(n=n, nstim=2, cstim=0, prop.inh=0.2)
phit <- signet$phi
stimuli <- signet$stimuli
dataset <- makedata(phit, stimuli, mu.bg=1200, sd.bg=400, mu.signal.a=2000, sd.signal.a=1000)

## use original network as prior matrix
## reset all entries for inhibiting edges
## to -1
B <- phit
B[B==2] <- -1

## perform inhibMCMC inference, using 4 CPU cores to get 4 MCMC chains
ret <- ddepn(dataset$datx, phiorig=phit, maxiterations=300, burnin=50,
            plotresults=FALSE, inference="mcmc",
            usebics=FALSE, priortype="laplaceinhib", lambda=0.01, B=B,
            multicores=TRUE, cores=4)

## perform netga inference, using 4 CPU cores
ret2 <- ddepn(dataset$datx, phiorig=phit, maxiterations=20, p=15, q=0.3,
m=0.8, plotresults=FALSE, inference="netga", usebics=FALSE,
priortype="laplaceinhib", lambda=0.01, B=B,
multicores=TRUE, cores=4)
```

```
## plot the data and fits
## mcmc
plot_profiles(ret, mfrow=c(3,4))
## netga
plot_profiles(ret2, mfrow=c(3,4))
## data only
plot_profiles(ret2$dat, mfrow=c(3,4))



## End(Not run)
```

`prior` *Calculate a structure prior. Usually called internally.*

Description

Provides three types of structure priors. `laplaceinhib` and `laplace` penalise the difference between the actual network and a reference network. `scalefree` penalises high node degrees.

Usage

```
prior(phi, lambda = NULL, B = NULL, Z = NULL,
   gam = NULL, it = NULL, K = NULL, priortype = "laplaceinhib")

calcpr(lambda, B, phi, gam)
```

Arguments

`phi`	The candidate network.
`lambda`	Laplace prior hyperparameter describing the prior influence strength.
`B`	Laplace prior probability matrix.
`Z`	Laplace prior normalisation factor for the prior. (Not used at the moment.)
`gam`	Scalefree prior degree distribution coefficient: $P(k)\tilde{}k^gam$ or exponent for difference term in laplaceinhib prior.
`K`	Scale-free prior scaling factor/Strength
`it`	Scale-free prior number of iterations for prior sampling.
`priortype`	String. Either `uniform`, `laplaceinhib`, `laplace` or `scalefree`. `uniform` assumes uniform prior network distributions. `laplaceinhib` calculates the difference between the candidate network and a reference matrix containing edge probabilites, while both edge types (activation and inhibition) are included. `laplace` is the same, except for ignoring the edge type. `scalefree` calculates a probability of a network following a scale free network architecture. See the references for a detailed description of the priors.

Details

For the *laplaceinhib* and *laplace* prior types, the matrix B is of central importance. The matrix has the same dimensions as the network to be inferred, each entry corresponding to a confidence in the existence of the respective edge. This confidence can be aquired by using external pathway sources, e.g. the KEGG database. See the vignette for a description of how to get the prior matrices. No matter how the confidence scores are obtained, there are two options, either use the *laplaceinhib* prior type, in which knowledge about the type of the edges is present in the external pathway source. Each confidence score for an edge, that is found as inhibiting edge in the reference pathways, is multiplied by -1 to obtain a negative value for the inhibiting edges. The entries of the prior matrix B thus lie in the interval $[-1; 1]$, where -1 means strong confidence that an edge is an inhibition, 1 means strong confidence that the edge is an activation and 0 means that nothing is known about the presence or type of the edge.

If no information on the type of the edges is available in the external data source, *priortype="laplace"* should be used, where the edge confidence ranges in the interval $[0; 1]$, where 1 means strong confidence that the edge is present and 0 means that nothing is known about the presence of the edge.

Argument `gam` is used either as exponent in the scalefree prior, as it is described in the reference, or in `laplaceinhib` and `laplace` as exponent in the following formula:
$P(phi_{ij}|lambda, gam, B) = \frac{1}{2 \cdot lambda} exp(\frac{-|phi_{ij} - B_{ij}|^{gam}}{lambda})$
It controls how strong the differences between an inferred edge and the probability for seeing this edge in a reference set of networks are to be weighted. Defaults to 1, if omitted. The prior curve rapidly decayes with increasing difference $|phi_{ij} - B_{ij}|$, while for `gam` larger than 1, the prior curve is changed to an s-shaped curve with a plateau at the upper bound of $P(phi_{ij}|lambda, gam, B)$ and an exponential decay when a certain threshold of $|phi_{ij} - B_{ij}|$ is reached.

See also the references for a description of the priors.

The `calcpr` function is a helper that calculates the laplace prior probability.

Value

Returns a double for the prior probability of network structure `phi`.

Note

TODO

Author(s)

Christian Bender

References

Laplace prior
Froehlich et. al. 2007, Large scale statistical inference of signaling pathways from RNAi and microarray data.

Scale free prior
Kamimura and Shimodaira, A Scale-free Prior over Graph Structures for Bayesian Inference of Gene Networks

See Also

ddepn

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

---

run_checks *DDEPN function tests.*

Description

Perform function tests for the different inference types and prior models.

Usage

```
run_checks(outfile = NULL, plotresults = TRUE)

wrapddepn(dataset, phit, stimuli, debug=FALSE, plotresults=TRUE, ...)

cls()
```

Arguments

outfile	String. A file name giving the file where output plots should be stored. If NULL, the user is asked for a file name via CLI. Unused, if plotresults=FALSE.
plotresults	Boolean. If TRUE, result plots are generated for each test and stored in file outfile. If FALSE, plots are omitted.
dataset	A sample dataset, e.g. generated by makedata.
phit	A sample network adjacency matrix, e.g. generated by signalnetwork.
stimuli	A list with sample stimuli, e.g. generated by signalnetwork.
debug	Boolean. Debug mode switch.
...	Further arguments passed to the ddepn call.

Details

run_checks Run a set of tests to check whether all ddepn functions are working. The following tests are performed:

```
 [1]  "GA, BIC optimisation"
 [2]  "GA, uniform prior"
 [3]  "GA, laplaceinhib, two types, lambda fix"
 [4]  "GA, laplaceinhib, two types, lambda integrated"
 [5]  "GA, laplace, one type, lambda fix"
 [6]  "GA, laplace, one type, lambda integrated"
 [7]  "GA, scalefree"
 [8]  "inhibMCMC, uniform prior"
 [9]  "inhibMCMC, laplaceinhib, two types, lambda fix"
 [10] "inhibMCMC, laplaceinhib, two types, lambda integrated"
 [11] "inhibMCMC, laplace, one type, lambda fix"
 [12] "inhibMCMC, laplace, one type, lambda integrated"
 [13] "inhibMCMC, scalefree"
 [14] "inhibMCMC, laplaceinhib, two types, sample lambda"
```

wrapddepn A simple ddepn wrap function defining a default call of ddepn for the checks.

cls Function for clearing the R-console.

Value

Matrix with the test results, where the first column holds the test and the second the result (passed or failed).

Author(s)

Christian Bender

Examples

```
## Not run:
library(ddepn)
tests <- run_checks(plotresults=FALSE)

## End(Not run)
```

`signalnetwork` *Sample a random signalling network.*

Description

For n nodes, sample a network with $nstim$ stimuli and $cstim$ combinatorial stimuli.

Usage

```
signalnetwork(n=10, nstim=2, cstim=0, prop.inh=.2,plot=F,gamma=1,B=NULL,
V=NULL,stimuli=NULL)
```

Arguments

n	Integer. Number of nodes.
nstim	Integer. Number of stimuli.
cstim	Integer. Number of combinatorial stimuli.
prop.inh	Proportion (in [0;1]) of the number of activating edges to be included as inhibiting edges in the network.
plot	Boolean. If TRUE, a plot of the generated graph is drawn.
gamma	Double. Strength of power law decay. Used for simulating the number of outgoing edges.
B	The prior edge probability matrix.
V	Vector of strings. Names of the nodes.
stimuli	List. See ddepn for an explanation.

Details

Simulates an artificial signalling network. Starts at $nstim$ random stimuli and selects random children, to which activation edges are drawn. These children are the new stimuli and the procedure is repeated until all nodes were reached by activating edges. Finally, $prop.inh * numedges$ inhibiting edges are added randomly. The number of stimuli combinations $cstim$ is limited by $sum_{k=2}^{n} \binom{k}{n}$. If defined, B gives a matrix containing prior probabilities for each possible edge in the network.

Value

List containing the adjacency list phi and the list of all stimuli.

Author(s)

Christian Bender

See Also

simulatedata

Examples

```
## Not run:
library(ddepn)
signalnetwork(n=10, nstim=4, cstim=4, prop.inh=.4, plot=TRUE)

## End(Not run)
```

simulatedata *simulatedata*

Description

Generate artificial timecourse data for network inference.

Usage

```
simulatedata(phi, mu.bg=0, sd.bg=0.1,
mu.signal.a=1, sd.signal.a=0.5,
mu.signal.i=-1, sd.signal.i=0.5,
stimulus=sample(nrow(phi),2),TT=10,R.t=4,R.b=3,
plot=FALSE,stimuli=NULL, allow.stim.off=TRUE)
```

Arguments

phi	The network for which data should be simulated.
mu.bg	mean for passive state.
sd.bg	sd for passive state.
mu.signal.a	mean for active state of type activation.
sd.signal.a	sd for active state of type activation.
mu.signal.i	mean for active state of type inhibition.
sd.signal.i	sd for active state of type inhibition.
stimulus	Where the network gets stimulated. Are set to 1 for the effect propagation.
TT	Number of timepoints.
R.t	Number of technical replicates.
R.b	Number of biological replicates.
plot	Should a plot be generated after data generation.
stimuli	List of input stimuli.
allow.stim.off	Boolean. Allow the stimuli to become inactive at some point. See also ddepn.

Details

TODO

Value

Artificial datasets for a given network.

Author(s)

Christian Bender

Examples

```
## Not run:
library(ddepn)
n <- 8
phi <- matrix(sample(c(0,1,2),n*n,replace=TRUE),nrow=n,dimnames=list(LETTERS[1:n],LETTERS[1:n]))
simulatedata(phi, mu.bg=0, sd.bg=0.1,
mu.signal.a=1, sd.signal.a=0.5,
mu.signal.i=-1, sd.signal.i=0.5,
stimulus=sample(nrow(phi),2),TT=10,R.t=4,R.b=3,
plot=TRUE)

## End(Not run)
```

Index