Package 'BTR'

September 17, 2016

Type Package

Title Training and Analysing Asynchronous Boolean Models

Version 1.2.4

Date 2016-9-13

Author Chee Yee Lim

Maintainer Chee Yee Lim <cyl49@cam.ac.uk>

Description Tools for inferring asynchronous Boolean models from single-cell expression data.

Depends R (>= 3.0.3), methods

Imports parallel, Rcpp (>= 0.11.4), foreach (>= 1.4.1), doParallel (>= 1.0.8), poweRlaw (>= 0.30.0), diptest (>= 0.75-7), igraph (>= 1.0.1), infotheo (>= 1.2.0), entropy (>= 1.2.1)

LinkingTo Rcpp

License GPL-3

LazyData true

Suggests bnlearn (>= 3.8.1), knitr, rmarkdown

VignetteBuilder knitr, rmarkdown

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-09-17 18:03:21

R topics documented:

`amat_to_bm` *Convert adjacency matrix into BoolModel object*

Description

This function converts adjacency matrix to BoolModel object. Able to take in adjacency matrix with -1, which encodes for inhibitory interaction.

Usage

```
amat_to_bm(amat, random = F)
```

Arguments

`amat`	matrix. directed adjacency matrix.
`random`	logical. Randomly assign to either act or inh rules, when the adjacency matrix only has values 0 and 1, but not -1.

`bm_to_amat` *Convert BoolModel into adjacency matrix*

Description

This function converts a BoolModel object into an adjacency matrix.

Usage

```
bm_to_amat(x, directed = T)
```

Arguments

`x`	S4 BoolModel object.
`directed`	logical. Whether to return directed or undirected adjacency matrix. Default to TRUE.

bm_to_df *Convert BoolModel object into BoolNet readable data frame*

Description

This method converts BoolModel object into a data frame, which is readable by BoolNet.

Usage

```
bm_to_df(bmodel)
```

Arguments

bmodel	BoolModel object.

bon_bmodel *HSC Boolean Model from Bonzanni et al.*

Description

A Boolean model describing HSC in mice. It contains 11 genes. Its steady state is a cyclic loop of 32 states.

Usage

```
data(bon_bmodel)
```

Format

A data frame with 11 rows and 2 columns.

Rows: each row consists of 1 gene and its associated Boolean rule. Column 1: target gene Column 2: associated Boolean rule

bon_istate *Initial state from Bonzanni et al.*

Description

An intial state specified in Bonzanni et al. It contains a set of Boolean values for 11 genes.

Usage

```
data(bon_istate)
```

Format

A data frame with 1 row and 11 columns.

Rows: each row consists of 1 set of Boolean state. Columns: each column is for 1 gene/variable.

BoolModel-class *An S4 class to represent a Boolean Model*

Description

This class represents Boolean Model in a S4 BoolModel object.

Fields

`target` character vector. It should contain gene symbols.

`targer_var` character vector. It should contain the internal representation of gene variables, in the form of "v[0-9]+s"

`rule_act` list of character vectors. Each element in the list should contain the activating gene variables for a particular target gene.

`rule_inh` list of character vectors. Each element in the list should contain the inhibiting gene variables for a particular target gene.

BTR *BTR: A package for studying asynchronous Boolean models*

Description

This package contains tools for Boolean model manipulation, as well as the search for the best Boolean model.

calc_mscore *Calculating Boolean model score wrt to a dataset*

Description

This function calculates a score for a Boolean model wrt to a dataset.

Usage

```
calc_mscore(bmodel, istate, fcdata, overlap_gene, max_varperrule,
  model_encoding, detail = F)
```

Arguments

bmodel	S4 BoolModel object. Model to be evaluated.
istate	data frame. Must have only 1 row, which represents 1 initial state.
fcdata	matrix. Represents the expression data df.
overlap_gene	character vector. Specify which genes are present in both model and data inputs.
max_varperrule	integer. Maximum number of terms per rule (combining both act and inh rule). Note that this number must not be smaller than number of variables. Default to 6.
model_encoding	numeric. Only required if the bmodel is encoded.
detail	logical. Whether to give more details in score calculation. Default to FALSE.

calc_roc *Calculate precision, recall, f-score, accuracy and specificity*

Description

This function calculates the precision, recall, f-score, accuracy and specificity from the output of validate_adjmat().

Usage

```
calc_roc(x)
```

Arguments

x	integer vector. Vector output by validate_adjmat().

check_and *Check if containing AND terms*

Description

This function checks if a particular Boolean model contains AND terms.

Usage

```
check_and(bmodel)
```

Arguments

bmodel	BoolModel object.

compress_bmodel *Compress BoolModel*

Description

This function compresses S4 BoolModel object by representing variables using numbers, and also only the act rules and inh rules are kept. Return a list of 3 vectors, corresponding to act rules and inh rules.

Usage

```
compress_bmodel(bmodel, encoding)
```

Arguments

bmodel	S4 BoolModel object.
encoding	named numerical vector returned by get_encodings().

decompress_bmodel *Decompress BoolModel*

Description

This function decompresses the bmodel compressed by compress_bmodel(). Return a S4 BoolModel object.

Usage

```
decompress_bmodel(x, encoding, format = "bmodel")
```

Arguments

x	vector returned by compress_bmodel.
encoding	named numerical vector returned by get_encodings().
format	character. Specifies which format to return. Possible values: 'bmodel', 'df', 'amat', 'simp_df'. Default to 'bmodel'.

decreate_boolmodel *Decreate Boolean model*

Description

This function converts a S4 BoolModel object into a list of 4 lists. (for use by Rcpp in simulate_model())

Usage

```
decreate_boolmodel(bmodel)
```

Arguments

bmodel	S4 BoolModel object.

df_to_bm *Convert a data frame into BoolModel object*

Description

This method converts a data frame into a BoolModel object. Note that the model should only has 1 NOT operator. More than 1 is STRICTLY NOT allowed.

Usage

```
df_to_bm(in_df)
```

Arguments

in_df	data frame with 2 columns, targets and factors

emodel1 *Example Boolean Model used in the vignette*

Description

A Boolean model used in the examples of the vignette.

Usage

```
data(example_models)
```

Format

Each Boolean model is a BoolModel object.

emodel2 *Example Boolean Model used in the vignette*

Description

A Boolean model used in the examples of the vignette.

Usage

```
data(example_models)
```

Format

Each Boolean model is a BoolModel object.

emodel3 *Example Boolean Model used in the vignette*

Description

A Boolean model used in the examples of the vignette.

Usage

```
data(example_models)
```

Format

Each Boolean model is a BoolModel object.

eval_bool *Evaluating Boolean rules*

Description

This function evaluates the Boolean rules (both act and inh) of one gene at a time. Return a logical value for that gene.

Usage

```
eval_bool(bmodel, val, ind)
```

Arguments

bmodel	S4 BoolModel object.
val	named logical vector. It should contain the values for all genes at that time point. Note that each value in the vector must be named by its corresponding gene name.
ind	integer. It indicates the state of which gene should be computed.

extract_term *Extract Boolean terms*

Description

This function extracts the terms within a Boolean rule, using OR as the separator. Bracketed variables are counted as one term.

Usage

```
extract_term(brule)
```

Arguments

brule	character vector. It should be either an activating or inhibiting rule for a target gene.

filter_dflist *Filter columns of df in a list*

Description

This function filters columns of multiple df in a list, when compared using a vector. Use through lapply().

Usage

```
filter_dflist(x, y, uniq_bool = T)
```

Arguments

x	data frame. It should be a list element if used from lapply.
y	character vector. It should contains gene names found in the colnames of the data frame.
uniq_bool	logical. Whether to return unique rows only.

gen_one_rmodel *Generate a random Boolean model*

Description

This function generates a random Boolean model. Returns an S4 BoolModel object. Note that this method will not give empty rule, i.e. 0 term in both act and inh rules.

Usage

```
gen_one_rmodel(var, exponent = 3, and_bool, self_loop = F,
  mvar = length(var), model_type = "random")
```

Arguments

var	character vector. A vector of single genes/variables to be used in the model.
exponent	integer. The exponent of power law distribution. Default to 3.
and_bool	logical. Indicates whether to include AND terms or not.
self_loop	logical. Indicates whether to allow self_loop. Default to F.
mvar	integer. Maximum number of variables in act or inh rule. Default to length(var).
model_type	character. Specifies the type of model generated.

Details

The number of terms in a function for a gene is modelled by power-law distribution.

gen_singlerule *Generate random act and inh rule for a single gene*

Description

This function generates one random Boolean rule (both act and inh) per run. Return a list of two vectors. Note that this method will not give empty rule, i.e. 0 term in both act and inh rules.

Usage

```
gen_singlerule(x, np, tar_ind, and_bool, self_loop = F,
  rule_type = "random")
```

Arguments

x	character vector. A vector of all single terms to be used.
np	integer. Number of gene variables in a rule. NOT max_varperrule here.
tar_ind	numerical. Indicate which gene is the rule for. Used in preventing self-loop.
and_bool	logical. Indicates whether to include AND terms or not.
self_loop	logical. Indicates whether to allow self_loop. Default to F.
rule_type	character. Types of rules. Defaults to random.

---

gen_two_rmodel *Generate two random Boolean models with a specified number of steps apart*

Description

This function generates a random Boolean model, then get another random Boolean model that is a specified number of steps apart by adding and/or removing genes. Returns a list of two S4 BoolModel objects.

Usage

```
gen_two_rmodel(var, steps, mvar = length(var), and_bool, in_bmodel = NULL,
  self_loop = F)
```

Arguments

var	character vector. A vector of single genes/variables to be used in the model.
steps	integer. Number of steps apart between the two models. If steps=0, give completely random starting model.
mvar	integer. Maximum number of variables in act or inh rule. Default to length(var).
and_bool	logical. Indicates whether to include AND terms or not.
in_bmodel	BoolModel object. The starting model supplied.
self_loop	logical. Indicates whether to allow self_loop. Default to F.

Details

The number of terms in a function for a gene is modelled by power-law distribution.

gen_two_rmodel_dag *Generate two random DAG Boolean models with a specified number of steps apart*

Description

This function generates a random DAG Boolean model, then get another random DAG Boolean model that is a specified number of steps apart by adding and/or removing genes. Difficult to generate completely directed graph with a specified number of steps apart.

Usage

```
gen_two_rmodel_dag(var, steps, mvar = length(var), in_amat = NULL,
  acyclic = T)
```

Arguments

var	character vector. A vector of single genes/variables to be used in the model.
steps	integer. Number of steps apart between the two models. If steps=0, give completely random starting model.
mvar	integer. Maximum number of variables in act or inh rule. Default to length(var).
in_amat	matrix. Provide adjacency matrix of first model.
acyclic	logical. Whether to restrict the model to being acyclic or not. Defaults to TRUE.

get_encodings *Get corresponding encodings for compression or decompression.*

Description

This function gets all possible terms (single or double terms) and their corresponding encodings. This function limits the number of possible variables in the model to 999.

Usage

```
get_encodings(bmodel, force_and = T)
```

Arguments

bmodel	S4 BoolModel object.
force_and	logical. Whether to include ANDs in the encoding even if no AND is present in the bmodel object. Defaults to T.

grow_bmodel *Add extra genes to a Boolean model*

Description

This function adds extra genes to a Boolean model. Return a list of BoolModel object and an initial state.

Usage

```
grow_bmodel(in_gene, in_model)
```

Arguments

in_gene	character vector. Genes to be added into the model.
in_model	data frame or BoolModel object. If it is a data frame, it must have 2 columns, which are targets and update functions.

initialise_data *Initialise data*

Description

This function initialises data frame of Boolean state space. Returns initialised data frame.

Usage

```
initialise_data(state, aslogic = F)
```

Arguments

state	data frame. It should contain either 0/1 or F/T data of gene expression.
aslogic	logical. It specifies whether to convert the input data into Boolean values. Default to FALSE.

initialise_model *Initialise model*

Description

This function initialises a Boolean model. Returns initialised S4 BoolModel object. Note that the model should only has 1 NOT operator. More than 1 is STRICTLY NOT allowed.

Usage

```
initialise_model(init_model)
```

Arguments

init_model	data frame of Boolean model. It should contain two columns, targets and functions.

initialise_raw_data *Initialise raw data*

Description

This function initialise raw gene expression values in a matrix. Return either a matrix of (1) continuous values or (2) binary values. Note that kmeans clustering as binarisation only works well if the data has a bimodal distribution.

Usage

```
initialise_raw_data(x, max_expr = "high", uni_thre = 0.2, scale = T,
  discretised = F)
```

Arguments

x	matrix. Numeric data of gene expression.
max_expr	character. Specify whether max expression value is the lowest (as in qPCR), or the highest (as in RNAseq and microarray). Option: 'low', 'high'. Default to 'high'.
uni_thre	numerical. Speficy threshold for unimodality test. Default to 0.2.
scale	logical. Whether to scale the data to a range of 0-1. Default to T.
discretised	logical. Whether to return discretised data. Default to F.

krum_bmodel *Myeloid Boolean Model from Krumsiek et al.*

Description

A Boolean model describing myeloid development in mice. It contains 11 genes. Its steady states are 4 static attractors.

Usage

```
data(krum_bmodel)
```

Format

A data frame with 11 rows and 2 columns.

Rows: each row consists of 1 gene and its associated Boolean rule. Column 1: target gene Column 2: associated Boolean rule

krum_istate *Initial state from Krumsiek et al.*

Description

An intial state specified in Krumsiek et al. It contains a set of Boolean values for 11 genes.

Usage

```
data(krum_istate)
```

Format

A data frame with 1 row and 11 columns.

Rows: each row consists of 1 set of Boolean state. Columns: each column is for 1 gene/variable.

man_dist *Calculates pairwise Manhattan distances between two matrices*

Description

This function calculates pairwise Manhattan distances between two matrices.

Usage

```
man_dist(x, y)
```

Arguments

x	matrix
y	matrix

match_term *Check for matching terms*

Description

This function checks if the first term is found within a vector of terms. Return a logical value. It is smarter than simple string matching, e.g. match_term(v1s&v2s, v2s&v1s) == T, match_term(v1s, v1s&v2s) == T, match_term(v1s&v2s, v1s) == T.

Usage

```
match_term(t1, t2, mode = "logic")
```

Arguments

t1	character vector of length 1 or more. It should be a vector of gene variable.
t2	character vector of length 1 or more. It should be a vector of gene variables.
mode	character. Indicates the mode of action. Options: 'logic', 'unique'. Default to 'logic'.

minmod_internal *Inner function of minimal modification of whole Boolean model*

Description

This function generates all possible boolean models minimally modified. Returns a lists of 2 lists, deleted models and added models

Usage

```
minmod_internal(bm, index, encoded, model_encoding, and_bool, self_loop = F)
```

Arguments

bm	S4 BoolModel object.
index	integer. Specifying rule of which gene to modify.
encoded	logical. Return Boolean models in encoded form to save space.
model_encoding	list. Only used if encoded=T.
and_bool	logical. Default to check_and() if unspecified.
self_loop	logical. Whether to allow self_loop in random starting model. Default to F.

minmod_model *Minimal modification of whole Boolean model*

Description

This function generates all possible boolean models minimally modified. Returns a lists of 2 lists, deleted models and added models

Usage

```
minmod_model(bm, index = NULL, overlap_gene = NULL, sep_list = F,
  encoded = F, model_encoding, and_bool = NULL, self_loop = F)
```

Arguments

bm	S4 BoolModel object.
index	integer. Specifying rule of which gene to modify. If NULL, modifies all rules in the model. Defaults to NULL.
overlap_gene	character vector. Specify which genes are present in both model and data inputs.
sep_list	logical. Separate add and del lists.
encoded	logical. Return Boolean models in encoded form to save space.
model_encoding	list. Only used if encoded=T.
and_bool	logical. Default to check_and().
self_loop	logical. Whether to allow self_loop in random starting model. Default to F.

model_consensus *Intersection of genes*

Description

This function finds the intersection of genes and provide a score for them. Return a consensus model or a vector of scores.

Usage

```
model_consensus(bmodel_list, model_encoding, format = "vec")
```

Arguments

bmodel_list	list of BoolModel.
model_encoding	list. Use for compressing and decompressing Boolean models.
format	character. Specifies which format to return. Possible values: 'vec', 'df'. Default to 'vec'.

model_dist *Calculate distance between Boolean models*

Description

This method takes in two models and calculate the distance between them. The value return indicate the number of steps between the two models.

Usage

```
model_dist(x, y)
```

Arguments

x	S4 BoolModel object. Test model.
y	S4 BoolModel object. Reference model.

model_setdiff *Find the set difference between two Boolean models*

Description

This method takes in two models and find the set difference between them. Return a vector with the set difference.

Usage

```
model_setdiff(x, y, directed = F)
```

Arguments

x	S4 BoolModel object. Test model.
y	S4 BoolModel object. Reference model.
directed	logical. If TRUE, return the difference in terms with respect to x.

model_train *Training Model*

Description

This function performs model training to find the best model, using information from data. It requires an initial state supplied to perform the search, and an initial model can also be supplied to be included in the initial population. Note that if a model is supplied, and the genes in the model is different from the genes in the data, only the genes overlapping between model and data will be retained for further analysis.

Usage

```
model_train(cdata, bmodel = NULL, istate = NULL, max_varperrule = 6,
  and_bool = T, self_loop = F, con_thre = 0.3, tol = 1e-06,
  verbose = F, detailed_output = F)
```

Arguments

cdata	data frame of expression data. Should have state(row) x gene(column).
bmodel	Boolean model in data frame. If NULL, use a random Boolean model. Defaults to NULL.
istate	data frame. Must have only 1 row, which represents 1 initial state. Defaults to NULL.
max_varperrule	integer. Maximum number of terms per rule (combining both act and inh rule). Note that this number must be higher than number of genes. Defaults to 3.

and_bool	logical. Whether to consider AND terms. IF bmodel is not NULL, defaults to whether AND interaction is included in bmodel. If bmodel is NULL, then defaults to TRUE.
self_loop	logical. Whether to allow self_loop in random starting model. Default to F.
con_thre	numerical. Threshold used to generating the final consensus model. Must be between 0 and 1.
tol	numeric. Tolerance in ending condition. Default to 1e-6. It cannot be lower than .Machine$double.eps ^ 0.5.
verbose	logical. Whether to give detailed output to the screen. Defaults to F.
detailed_output	logical. Whether to return only the model inferred, or all the details obtained during optimisation. Defaults to F.

Examples

```
data(wilson_raw_data)
cdata = initialise_raw_data(wilson_raw_data, max_expr = 'low')

#select only relevant cells.
cell_ind = grepl('cmp', rownames(cdata)) | grepl('gmp', rownames(cdata))
fcdata = cdata[cell_ind,]

#select genes to be included.
gene_ind = c('fli1', 'gata1', 'gata2', 'gfi1', 'scl', 'sfpi1')
fcdata = fcdata[, gene_ind]

final_model = model_train(cdata=fcdata, max_varperrule=2)
plotBM(final_model)
```

---

outgenysis_model *Output a Boolean Model into Genysis readable format*

Description

This function outputs a Boolean Model in a format that is readable by Genysis. Return invisibly the formatted vector.

Usage

```
outgenysis_model(bmodel, path = getwd(), file = NULL)
```

Arguments

bmodel	S4 BoolModel object.
path	character. Specify path (AND NOT file name). Default to current working directory, i.e. getwd(). Set to NULL to disable file output.
file	character. Specify file name. Default to NULL for default file names.

outgraph_model *Output a Boolean Model into Cytoscape & Gephi readable format*

Description

This function outputs a Boolean Model in a format that is readable by Cytoscape and Gephi. Return invisibly the edges (with edge attributes) and node attributes. (i.e. list of 2 dfs)

Usage

```
outgraph_model(bmodel, path = getwd(), file = NULL, and_node = T)
```

Arguments

bmodel	S4 BoolModel object.
path	character. Specify path (AND NOT file name). Default to current working directory, i.e. getwd(). Set to NULL to disable file output.
file	character. Specify file name. Default to NULL for default file names.
and_node	logical. Specify AND as an individual node. Default to T.

outstate_graph *Generate state transition graph*

Description

This function generates a state transition graph using a Boolean model and its state space. Each node represent a state. All nodes in this graph is linked by an edge only if the 2 states have different value in only 1 gene. The output is readable by Cytoscape and Gephi.

Usage

```
outstate_graph(mstate, bmodel, directed = F, record.both = F,
  filepath = getwd())
```

Arguments

mstate	data frame. It should be a state(row) x gene(column) df.
bmodel	S4 BoolModel object.
directed	logical. Indicates whether to make directed edges or not. Default to FALSE.
record.both	logical. Indicates whether to also record nodes that have no edges. Default to FALSE.
filepath	character vector. Specify path (AND NOT file name). Default to current working directory, i.e. getwd(). Set to NULL to disable file output.

plotBM *Plot Boolean Model*

Description

This method plots the network underlying Boolean models by using igraph for quick visualisation. Require igraph.

Usage

```
plotBM(bmodel, makePlot = T, ...)
```

Arguments

bmodel	S4 BoolModel object.
makePlot	logical. Whether to make plot or just return the object. Default to T.
...	Additional parameters to plot.igraph.

printBM *Print Boolean Model*

Description

This method converts the S4 BoolModel object back into a human-readable data frame, with two columns: (1) target genes, (2) Boolean rules.

Usage

```
printBM(bmodel, gene.names = F)
```

Arguments

bmodel	S4 BoolModel object.
gene.names	logical. Specify whether to write rules in terms of genes or internal variables. Default to FALSE.

rcpp_simulate *Simulate a Boolean model.*

Description

(&&&Not for public use&&&)This function simulates the Boolean model using an initial state. For use within simulate_model(). Returns a matrix of full asynchronous state space.

Usage

```
rcpp_simulate(bmodel, fstate, verbose = FALSE)
```

Arguments

bmodel	list. A list of 4 lists created by decreate_model().
fstate	data frame. It must have been initialised by initialise_data(), and has gene names as column names. Must contain only 1 row.
verbose	logical. Indicates whether to output progress.

rcpp_validate *Calculating validation scores between two adjacency matrices*

Description

This function calculates the validation scores between two adjacency matrices.

Usage

```
rcpp_validate(inf_mat, true_mat)
```

Arguments

inf_mat	matrix. It should be adjacency matrix of inferred network.
true_mat	matrix. It should be adjacency matrix of true network.

simulate_model *Simulating Boolean model*

Description

This function simulates the Boolean model using an initial state. Returns the full asynchronous state space, and point steady states.

Usage

```
simulate_model(bmodel, istate, steady_bool = F)
```

Arguments

bmodel	S4 BoolModel object.
istate	data frame. It must have been initialised by initialise_data(), and has gene names as column names. Must contain only 1 row.
steady_bool	logical. Specifies whether to return point steady states or not. Default to F.

unique_raw_data *Remove raw data duplicated wrt to the model state*

Description

This function removes the 'duplicates' in an expression wrt to the model state.

Usage

```
unique_raw_data(dx, cx)
```

Arguments

dx	matrix. Initialised and discretised numeric data of gene expression.
cx	matrix. Initialised and continuous numeric data of gene expression.

validate_adjmat *Calculate true positive, true negative, false positive and false negative*

Description

This function calculates the true positive, true negative, false positive and false negative values from the adjacency matrices.

Usage

```
validate_adjmat(inf_mat, true_mat)
```

Arguments

inf_mat	matrix. It should be adjacency matrix of inferred network.
true_mat	matrix. It should be adjacency matrix of true network.

vcat *Verbose cat*

Description

This function is a simple wrapper for cat with a Boolean value for turning it on/off.

Usage

```
vcat(string, bool)
```

Arguments

string	character vector. String intended to be printed.
bool	logical. Specify whether to print the string to stout or not.+

which.random.min *Pick a random minimum value*

Description

This function locates the minimum value in a vector (similar to which.min), however it will randomly break ties when there are multiple minimum values.

Usage

```
which.random.min(x, favour_first = F)
```

Arguments

x	numeric vector.
favour_first	logical. If this is TRUE, and the first value in the vector supplies is one of the min, the index for the first value will always be returned.

wilson_raw_data *Raw single cell qRT-PCR expression data from Wilson et al.*

Description

A raw single cell expression data obtained from multiple cell types.

Usage

```
data(wilson_raw_data)
```

Format

A data frame with 1626 rows and 44 columns.

Rows: each row consists of raw expression values from 1 cell. Columns: each column is for 1 gene/variable.

wilson_raw_rnaseq *Raw single cell RNAseq expression data from Wilson et al.*

Description

A raw single cell expression data obtained from multiple cell types.

Usage

```
data(wilson_raw_rnaseq)
```

Format

A data frame with 96 rows and 38498 columns.

Rows: each row consists of raw expression values from 1 cell. Columns: each column is for 1 gene/variable.

writeBM *Write Boolean Model*

Description

This method writes the S4 BoolModel object into a CSV file. This method is a wrapper for print.BoolModel. The output is a data frame, with two columns: (1) target genes, (2) Boolean rules.

Usage

```
writeBM(bmodel, file, gene.names = F, rownames = F)
```

Arguments

bmodel	S4 BoolModel object.
file	file name with path, or a file connection object.
gene.names	logical. Specify whether to write rules in terms of genes or internal variables. Default to FALSE.
rownames	logical. It specifies whether to write row names.

Index