Package 'CBN2Path'

July 25, 2025

Title Conjunctive Bayesian Networks

Version 0.1.0

Description A family of R functions based on the CBN family of functions created at ETH-Zurich.

URL https://github.com/rockwillck/CBN2Path,
http://dx.doi.org/10.1093/biomet/asp023,
http://dx.doi.org/10.1093/bioinformatics/btp505

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports R6, foreach, doMC, ggraph, tidygraph, ggplot2, patchwork, cowplot, magrittr, igraph, rlang, grDevices

Suggests testthat (>= 3.0.0), BiocStyle, knitr, rmarkdown, rBCBN

Depends R (>= 3.5.0)

Config/testthat/edition 3

biocViews Software, StatisticalMethod, GraphAndNetwork

VignetteBuilder knitr

NeedsCompilation yes

Author William Choi-Kim [aut, cre] (<https://orcid.org/0009-0000-3902-4745>), Sayed-Rzgar Hosseini [aut] (<https://orcid.org/0000-0002-2308-6754>)

Maintainer William Choi-Kim <william@williamck.com>

Contents

`CBN2Path-package` *CBN2Path: Conjunctive Bayesian Networks*

Description

A family of R functions based on the CBN family of functions created at ETH-Zurich.

A family of R functions based on the CBN family of functions created at ETH-Zurich.

Author(s)

Maintainer: William Choi-Kim <william@williamck.com> (ORCID)

Authors:

- Sayed-Rzgar Hosseini <rzgar.hosseini@indstate.edu> (ORCID)

See Also

Useful links:

- https://github.com/rockwillck/CBN2Path
- http://dx.doi.org/10.1093/biomet/asp023
- http://dx.doi.org/10.1093/bioinformatics/btp505

Useful links:

- https://github.com/rockwillck/CBN2Path
- http://dx.doi.org/10.1093/biomet/asp023
- http://dx.doi.org/10.1093/bioinformatics/btp505

`Base2Indexing` *Base2Indexing*

Description

Base2Indexing

Usage

```
Base2Indexing(mat)
```

Arguments

`mat`	A given poset represented by a binary matrix (in B-CBN)

Value

#Poset weight vectors based on the frequency of occurence in the BCBN MCMC-sampling scheme.

Examples

```
set.seed(100)
mat<-matrix(sample(c(0,1),16,replace=TRUE),4,4)
Index<-Base2Indexing(mat)
```

Base2IndVec *Base2IndVec*

Description

Base2IndVec

Usage

```
Base2IndVec(vec)
```

Arguments

vec	a binary genotype vector

Value

a number used for indexing a given genotype

Examples

```
vec<-c(0,1,0,1)
Base2IndVec(vec)
```

bcbn *B-CBN*

Description

B-CBN

Usage

```
bcbn(
  data = NULL,
  n_samples = 25000,
  theta = 0,
  epsilon = 0.05,
  n_chains = 4,
  thin = 10,
  Max_L = 1000,
  n_cores = 1
)
```

Arguments

data	Generated data
n_samples	Number of samples <def: 25000>
theta	Theta <def: 0>
epsilon	Epsilon <def: 0.05>
n_chains	N-Chains <def: 4>
thin	Thin <def: 10>
Max_L	The maximum number of iteration <def: 1000>
n_cores	Number of parallelized cores <def: 1>

Value

A matrix

Examples

```
if (!require("rBCBN")) {
    install.packages(CBN2Path::getBCBNinstall(), repos = NULL, type = "source")
}
bcbn()
```

---

checkBCBNVersion *Check if installed BCBN is correct*

Description

Check if installed BCBN is correct

Usage

```
checkBCBNVersion()
```

---

combinations *combinations*

Description

combinations

Usage

```
combinations(n, r, v = 1:n, set = TRUE, repeats.allowed = FALSE)
```

Arguments

n	total number of elements in the set
r	subset size
v	1:n
set	Logical flag indicating whether duplicates should be removed from the source vector v. Defaults to TRUE.
repeats.allowed	Logical flag indicating whether the constructed vectors may include duplicated values. Defaults to FALSE.

Value

a matrix with (n choose r) rows and r columns

Examples

```
COMB<-combinations(10,4)
```

---

ctcbn *CT-CBN*

Description

CT-CBN

Usage

```
ctcbn(
  datasets,
  bootstrap_samples = 0,
  random_seed = 1,
  sampling_rate = 1,
  epsilon = 2,
  num_drawn_samples = 0,
  num_em_runs = 1,
  n_cores = 1
)
```

Arguments

datasets	Vector of Spock objects with poset and pattern/lambda data or a Spock object (alias of ctcbn_single).
bootstrap_samples	Number of bootstrap samples (requires epsilon > 0, num_drawn_samples = 0)
random_seed	Random seed.
sampling_rate	Sampling rate.
epsilon	If between 0 and 1, the fraction of violations allowed per edge. If negative, the interval 0 to 0.5 will be sampled equidistantly with N points.

num_drawn_samples
: If > 0, the number of samples to draw from the model. If zero (default), the model will be learned from data.

num_em_runs
: Number of em runs.

n_cores
: Maximum number of threads to use to parallelize.

Value

A matrix of results.

Examples

```
example_path <- get_examples()[1]
bc <- Spock$new(
    poset = read_poset(example_path)$sets,
    numMutations = read_poset(example_path)$mutations,
    genotypeMatrix = read_pattern(example_path)
)
ctcbn(bc)
```

---

ctcbn_single *CT-CBN Single Batch*

Description

CT-CBN Single Batch

Usage

```
ctcbn_single(
  dataset,
  bootstrap_samples = 0,
  random_seed = 1,
  sampling_rate = 1,
  epsilon = 0,
  num_drawn_samples = 0,
  num_em_runs = 1
)
```

Arguments

dataset
: Spock object with poset and pattern/lambda data.

bootstrap_samples
: Number of bootstrap samples (requires epsilon > 0, num_drawn_samples = 0)

random_seed
: Random seed.

sampling_rate
: Sampling rate.

epsilon
: If between 0 and 1, the fraction of violations allowed per edge. If negative, the interval 0 to 0.5 will be sampled equidistantly with N points.

num_drawn_samples
: If > 0, the number of samples to draw from the model. If zero (default), the model will be learned from data.

num_em_runs
: Number of em runs.

Value

A list of output data.

Examples

```
example_path <- get_examples()[1]
bc <- Spock$new(
    poset = read_poset(example_path)$sets,
    numMutations = read_poset(example_path)$mutations,
    genotypeMatrix = read_pattern(example_path)
)
ctcbn_single(bc)
```

EdgeMarginalized *EdgeMarginalized*

Description

EdgeMarginalized

Usage

```
EdgeMarginalized(PathProb, x)
```

Arguments

PathProb	The pathway probabilities returned in the step 3 of the R-CBN algorithm
x	The number of mutations to consider

Value

returns the marginal probability of all the potential edges

Examples

```
DAG<-matrix(c(2,2,4,1,3,3),3,2)
LAMBDA<-c(1,4,3,2.5,2)
x<-4
PathP<-PathProb_CBN(DAG, LAMBDA, x)
EdgeProb<-EdgeMarginalized(PathP,x)
```

`generate_data` *Generate Data*

Description

Generate Data

Usage

```
generate_data(poset, thetas, eps, N)
```

Arguments

`poset`	Poset matrix
`thetas`	Vector of theta values
`eps`	Epsilon
`N`	N

Value

A matrix

Examples

```
poset <- matrix(0, 10, 10)

poset[1, 2] <- 1
poset[2, 3] <- 1
poset[3, 4] <- 1
poset[5, 4] <- 1
poset[6, 7] <- 1
poset[8, 9] <- 1
poset[8, 10] <- 1
poset[6, 9] <- 1

tr <- transitive_closure(poset)
theta <- c(0.8, 0.7, 0.6, 0.7, 0.4, 0.25, 0.6, 0.75, 0.5, 0.2)
eps <- 0.1
N <- 400

generate_data(tr, theta, eps, N)
```

generate_matrix_genotypes *generate_matrix_genotypes*

Description

generate_matrix_genotypes

Usage

```
generate_matrix_genotypes(g)
```

Arguments

g	genotype length

Value

a genotype matrix with ncol=g and nrow=2^g

Examples

```
Geno4<-generate_matrix_genotypes(4)
```

GenotypeMatrix_Mutator *GenotypeMatrix_Mutator*

Description

GenotypeMatrix_Mutator

Usage

```
GenotypeMatrix_Mutator(mat, FP, FN)
```

Arguments

mat	The genotype matrix including sampled genotypes, which need to be muatated.
FP	False positive rate
FN	False negative rate

Value

The mutated version of the genotype matrix

Examples

```
set.seed(100)
gMat<-matrix(sample(c(0,1),800,replace = TRUE),200,4)
gMat_mut<-GenotypeMatrix_Mutator(gMat,0.2,0.2)
```

Genotype_Feasibility *Genotype_Feasibility*

Description

Genotype_Feasibility

Usage

```
Genotype_Feasibility(genotypes, DAG, x)
```

Arguments

genotypes	the full set of potential binary genotypes of a given length.
DAG	matrix representing the DAG of restrictions.
x	the number of mutations considered.

Value

a binary vector, which indicates feasibility or infeasibility of a set of genotypes

Examples

```
Geno4<-generate_matrix_genotypes(4)
DAG<-matrix(c(4,4,4,1,2,3),3,2)
x<-4
GenoF4<-Genotype_Feasibility(Geno4,DAG,x)
```

getBCBNinstall *Get BCBN .tgz path*

Description

Get BCBN .tgz path

Usage

```
getBCBNinstall()
```

Value

internal .tgz path to install rBCBN

Examples

```
getBCBNinstall()
```

getBCBNVersion *Get needed BCBN version*

Description

Get needed BCBN version

Usage

```
getBCBNVersion()
```

get_examples *Get paths to examples*

Description

Get paths to examples

Usage

```
get_examples()
```

Value

A vector of paths

Examples

```
get_examples()
```

hcbn *H-CBN*

Description

H-CBN

Usage

```
hcbn(
  datasets,
  anneal = FALSE,
  temp = 0,
  annealing_steps = 0,
  epsilon = 2,
  n_cores = 1
)
```

Arguments

datasets	Vector of Spock objects with poset and pattern/lambda data or a Spock object (alias of hcbn_single).
anneal	If TRUE, performes a simulated annealing run starting from the poset
temp	Temperature of simulated annealing.
annealing_steps	Number of simulated annealing steps.
epsilon	Value of eps for CT-CBN model selection. Requires both pattern and lambda data in input Spock.
n_cores	Maximum number of threads to use to parallelize.

Value

A matrix of results.

Examples

```
example_path <- get_examples()[1]
bc <- Spock$new(
    poset = read_poset(example_path)$sets,
    numMutations = read_poset(example_path)$mutations,
    genotypeMatrix = read_pattern(example_path)
)
hcbn(bc)
hcbn(c(bc, bc, bc))
```

---

hcbn_single *H-CBN Single Batch*

Description

H-CBN Single Batch

Usage

```
hcbn_single(
  datasetObj,
  anneal = FALSE,
  temp = 0,
  annealing_steps = 0,
  epsilon = 2
)
```

Arguments

datasetObj	Spock object with poset and pattern/lambda data.
anneal	If TRUE, performes a simulated annealing run starting from the poset
temp	Temperature of simulated annealing.
annealing_steps	Number of simulated annealing steps.
epsilon	Value of eps for CT-CBN model selection. Requires both pattern and lambda data in input Spock.

Value

A list of output data.

Examples

```
example_path <- get_examples()[1]
bc <- Spock$new(
    poset = read_poset(example_path)$sets,
    numMutations = read_poset(example_path)$mutations,
    genotypeMatrix = read_pattern(example_path)
)
hcbn_single(bc)
```

`Jensen_Shannon_Divergence` *Jensen_Shannon_Divergence*

Description

Jensen_Shannon_Divergence

Usage

```
Jensen_Shannon_Divergence(Prob1, Prob2)
```

Arguments

`Prob1`	The first (discrete) probability distribution (vector)
`Prob2`	The second (discrete) probability distribution (vector)

Value

Jensen Shannon Divergence between the two (discrete) probability distributions

Examples

```
set.seed(100)
gMat<-matrix(sample(c(0,1),800,replace = TRUE),200,4)
PathCT<-PathProb_Quartet_CTCBN(gMat)
PathH<-PathProb_Quartet_HCBN(gMat)
JSD<-Jensen_Shannon_Divergence(PathCT,PathH)
```

lambda_example_data *Example .lambda files*

Description

Model parameters, if N > 0

Details

These files are included in the package under `inst/extdata/` and can be accessed using `system.file()` or `get_examples()`. They are unchanged from their original source.

Source

https://bsse.ethz.ch/cbg/software/ct-cbn.html

PathProb_CBN *PathProb_CBN: quantifies pathway probabilities using the output of CT-CBN or H-CBN*

Description

PathProb_CBN: quantifies pathway probabilities using the output of CT-CBN or H-CBN

Usage

```
PathProb_CBN(DAG, LAMBDA, x)
```

Arguments

DAG	matrix representing the DAG of restrictions.
LAMBDA	the lambda values, which are produced by the CBN model.
x	the number of mutations considered.

Value

vector of probabilities assigned to all potential pathways of length x

Examples

```
DAG<-matrix(c(2,2,4,1,3,3),3,2)
LAMBDA<-c(1,4,3,2.5,2)
x<-4
PathP<-PathProb_CBN(DAG, LAMBDA, x)
```

PathProb_Quartet_BCBN *PathProb_Quartet_BCBN*

Description

PathProb_Quartet_BCBN

Usage

```
PathProb_Quartet_BCBN(gMat)
```

Arguments

gMat	The n by 4 binary genotype matrix representing a given quartet for a sample of n genotypes.

Value

The probability distribution (returned by the B-CBN model), which is represented as a vector of length 24.

Examples

```
set.seed(100)
gMat<-matrix(sample(c(0,1),800,replace = TRUE),200,4)
PathB<-PathProb_Quartet_BCBN(gMat)
```

PathProb_Quartet_CTCBN *PathProb_Quartet_CTCBN*

Description

PathProb_Quartet_CTCBN

Usage

```
PathProb_Quartet_CTCBN(gMat)
```

Arguments

gMat	The n by 4 binary genotype matrix representing a given quartet for a sample of n genotypes.

Value

The probability distribution (returned by the CT-CBN model), which is represented as a vector of length 24.

Examples

```
set.seed(100)
gMat<-matrix(sample(c(0,1),800,replace = TRUE),200,4)
PathCT<-PathProb_Quartet_CTCBN(gMat)
```

PathProb_Quartet_HCBN *PathProb_Quartet_HCBN*

Description

PathProb_Quartet_HCBN

Usage

```
PathProb_Quartet_HCBN(gMat)
```

Arguments

gMat	The n by 4 binary genotype matrix representing a given quartet for a sample of n genotypes.

Value

The probability distribution (returned by the H-CBN model), which is represented as a vector of length 24.

Examples

```
set.seed(100)
gMat<-matrix(sample(c(0,1),800,replace = TRUE),200,4)
PathH<-PathProb_Quartet_HCBN(gMat)
```

PathProb_Quartet_RCBN *PathProb_Quartet_RCBN*

Description

PathProb_Quartet_RCBN

Usage

```
PathProb_Quartet_RCBN(gMat)
```

Arguments

gMat	The n by 4 binary genotype matrix representing a given quartet for a sample of n genotypes.

Value

The probability distribution (returned by the R-CBN model), which is represented as a vector of length 24

Examples

```
set.seed(100)
gMat<-matrix(sample(c(0,1),800,replace = TRUE),200,4)
PathR<-PathProb_Quartet_RCBN(gMat)
```

PathProb_SSWM *PathProb_SSWM*

Description

PathProb_SSWM

Usage

```
PathProb_SSWM(FITNESS, x)
```

Arguments

FITNESS	A vector of length 2^x, each element of which representing the fitness assigned to one of the 2^x genotypes.
x	The number of mutations considered.

Value

vector of probabilities assigned to all potential pathways of length x

Examples

```
F<-c(0,0.1,0.2,0.1,0.2,0.4,0.3,0.2,0.2,0.1,0,0.6,0.4,0.3,0.2,1)
x<-4
PathP<-PathProb_SSWM(F,x)
```

Pathway_Compatibility_Quartet
Pathway_Compatibility_Quartet

Description

Pathway_Compatibility_Quartet

Usage

```
Pathway_Compatibility_Quartet(gMat)
```

Arguments

gMat	The n by 4 binary genotype matrix representing a given quartet for a sample of n genotypes.

Value

The compatibility score, which is represented as a vector of length 24, each element of which corresponds to one of the 24 pathways of length 4.

Examples

```
set.seed(100)
gMat<-matrix(sample(c(0,1),800,replace = TRUE),200,4)
Pathway_Compatibility_Quartet(gMat)
```

Pathway_Feasibility *Pathway_Feasibility*

Description

Pathway_Feasibility

Usage

```
Pathway_Feasibility(DAG, x)
```

Arguments

DAG	matrix representing the DAG of restrictions.
x	the number of mutations considered.

Value

a binary vector, which indicates feasibility or infeasibility of a set of pathways

Examples

```
DAG<-matrix(c(4,4,4,1,2,3),3,2)
x<-4
PathF<-Pathway_Feasibility(DAG, x)
```

Pathway_Genotype_Compatiblility
Pathway_Genotype_Compatiblility

Description

Pathway_Genotype_Compatiblility

Usage

```
Pathway_Genotype_Compatiblility(Pathway, Genotype)
```

Arguments

Pathway	a vector representing the given pathway.
Genotype	a binary vector representing the given genotype.

Value

returns 1 (if the given genotype is compatible with the given pathway), and 0 otherwise

Examples

```
Geno1<-c(1,0,1,0)
Geno2<-c(1,1,0,0)
Path<-c(1,2,3,4)
Pathway_Genotype_Compatiblility(Path,Geno1)
Pathway_Genotype_Compatiblility(Path,Geno2)
```

Pathway_Weighting_RCBN
Pathway_Weighting_RCBN

Description

Pathway_Weighting_RCBN

Usage

```
Pathway_Weighting_RCBN(EdgeProb, PEmap)
```

Arguments

EdgeProb	Marginal edge probabilities
PEmap	Pathway-edge compatibility matrix

Value

The pathway weights (step 4 of the R-CBN algorithm)

Examples

```
DAG<-matrix(c(2,2,4,1,3,3),3,2)
LAMBDA<-c(1,4,3,2.5,2)
x<-4
PathP<-PathProb_CBN(DAG, LAMBDA, x)
EdgeProb<-EdgeMarginalized(PathP,x)
PEmap<-Path_Edge_Mapper(4)
W2<-Pathway_Weighting_RCBN(EdgeProb,PEmap)
```

Path_Edge_Mapper *Path_Edge_Mapper*

Description

Path_Edge_Mapper

Usage

```
Path_Edge_Mapper(x)
```

Arguments

x	number of mutations to consider

Value

Pathway to edge compatibility matrix, each element of which indicates whether a given edge is included in the transitive closure of a given pathway (1) or not (0).

Examples

```
PEmap<-Path_Edge_Mapper(4)
```

Path_Normalization *Path_Normalization*

Description

Path_Normalization

Usage

```
Path_Normalization(PathProb, x)
```

Arguments

PathProb	The pathway probabilities returned in the step 3 of the R-CBN algorithm
x	The number of mutations to consider

Value

The updated pathway probabilities (the step 5 of the R-CBN algorithm)

Examples

```
DAG<-matrix(c(2,2,4,1,3,3),3,2)
LAMBDA<-c(1,4,3,2.5,2)
x<-4
PathP<-PathProb_CBN(DAG, LAMBDA, x)
PathN<-Path_Normalization(PathP, x)
```

---

pat_example_data *Example .pat and .sim.pat files*

Description

Mutational patterns (genotypes), unless N > 0

Details

These files are included in the package under `inst/extdata/` and can be accessed using `system.file()` or `get_examples()`. They are unchanged from their original source.

Source

https://bsse.ethz.ch/cbg/software/ct-cbn.html

---

permutations *permutations*

Description

permutations

Usage

```
permutations(n, r, v = 1:n, set = TRUE, repeats.allowed = FALSE)
```

Arguments

`n`	total number of elements in the set
`r`	subset size
`v`	1:n
`set`	Logical flag indicating whether duplicates should be removed from the source vector v. Defaults to TRUE.
`repeats.allowed`	Logical flag indicating whether the constructed vectors may include duplicated values. Defaults to FALSE.

Value

a matrix with (n!/(n-r)!) rows and r columns

Examples

```
PERM<-permutations(4,4)
```

---

poset_example_data *Example .poset files*

Description

Event poset used if -e is not set; if -e is set, the file is used for determining the number of events as specified in the first row

Details

These files are included in the package under `inst/extdata/` and can be accessed using `system.file()` or `get_examples()`. They are unchanged from their original source.

Source

https://bsse.ethz.ch/cbg/software/ct-cbn.html

Poset_Weighting_RCBN *Poset_Weighting_RCBN*

Description

Poset_Weighting_RCBN

Usage

```
Poset_Weighting_RCBN(vec)
```

Arguments

vec	The likelihood vector corresponding to a given set of posets

Value

The poset weight vector determined using the reciprocal ranking method

Examples

```
set.seed(100)
LogLik<-runif(219)
W1<-Poset_Weighting_RCBN(LogLik)
```

Predictability *Predictability*

Description

Predictability

Usage

```
Predictability(Prob, x)
```

Arguments

Prob	Pathway probability vector
x	The length of genotype vectors

Value

Predictability

Examples

```
set.seed(100)
gMat<-matrix(sample(c(0,1),800,replace = TRUE),200,4)
PathCT<-PathProb_Quartet_CTCBN(gMat)
PathH<-PathProb_Quartet_HCBN(gMat)
PredC<-Predictability(PathCT,4)
PredH<-Predictability(PathH,4)
```

read_lambda *Read a .lambda file*

Description

Read a .lambda file

Usage

```
read_lambda(filestem)
```

Arguments

filestem	The filename of the .lambda file without the .lambda suffix.

Value

A matrix.

Examples

```
bcPath <- get_examples()[1]
read_lambda(bcPath)
```

read_pattern *Read a .pat file*

Description

Read a .pat file

Usage

```
read_pattern(filestem)
```

Arguments

filestem	The filename of the .pat file without the .pat suffix.

Value

A matrix.

Examples

```
bcPath <- get_examples()[1]
read_pattern(bcPath)
```

read_poset *Read a .poset file*

Description

Read a .poset file

Usage

```
read_poset(filestem)
```

Arguments

filestem	The filename of the .poset file without the .poset suffix.

Value

A list containing the number of mutations and a matrix.

Examples

```
bcPath <- get_examples()[1]
read_poset(bcPath)
```

read_time *Read a .time file*

Description

Read a .time file

Usage

```
read_time(filestem)
```

Arguments

filestem	The filename of the .time file without the .time suffix.

Value

A matrix.

Examples

```
bcPath <- get_examples()[1]
read_pattern(bcPath)
```

Spock *Poset and pattern/lambda data*

Description

A data class containing poset and pattern/lambda matrices.

Details

Use the read_ methods to feed data from files.

Public fields

`poset` Poset matrix.

`numMutations` Number of mutations.

`genotypeMatrix` Genotype matrix.

`lambda` Lambda list.

Methods

Public methods:

- `Spock$new()`
- `Spock$getSize()`
- `Spock$getPoset()`
- `Spock$getSecond()`
- `Spock$getPattern()`
- `Spock$getLambda()`
- `Spock$clone()`

Method `new()`**:** Create a new Spock object.

Usage:

```
Spock$new(poset, numMutations, genotypeMatrix, lambda = NULL)
```

Arguments:

`poset` Poset matrix or list of poset matrices.

`numMutations` Number of mutations.

`genotypeMatrix` Genotype matrix.

`lambda` Lambda list.

Returns: A new `Spock` object.

Method `getSize()`**:** Get the number of posets.

Usage:

```
Spock$getSize()
```

Returns: Number of posets.

Method `getPoset()`**:** Write poset data to a tempfile.

Usage:

```
Spock$getPoset(index = 1)
```

Arguments:

`index` Index of poset.

Returns: File path to tempfile.

Method `getSecond()`**:** Write pattern/lambda data to a tempfile.

Usage:

```
Spock$getSecond(n)
```

Arguments:

`n` Number of drawn samples.

Returns: File path to tempfile.

Method `getPattern()`**:** Write pattern data to a tempfile.

Usage:

```
Spock$getPattern()
```

Returns: File path to tempfile.

Method `getLambda()`**:** Write lambda data to a tempfile.

Usage:

```
Spock$getLambda()
```

Returns: File path to tempfile.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Spock$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`transitive_closure` *Transitive Closure*

Description

Transitive Closure

Usage

```
transitive_closure(poset)
```

Arguments

`poset`	Poset matrix

Value

Poset matrix

Examples

```
poset <- matrix(0, 10, 10)

poset[1, 2] <- 1
poset[2, 3] <- 1
poset[3, 4] <- 1
poset[5, 4] <- 1
poset[6, 7] <- 1
poset[8, 9] <- 1
poset[8, 10] <- 1
poset[6, 9] <- 1
transitive_closure(poset)
```

visualize_cbn_model *Visualize CBN Model*

Description

Visualize CBN Model

Usage

```
visualize_cbn_model(poset, nodeColor = "darkgreen")
```

Arguments

poset	Poset object to visualize
nodeColor	Color of nodes in resulting graph

Value

Plot (gg object) visualization of CBN model

Examples

```
poset <- read_poset(get_examples()[1])
visualize_cbn_model(poset)
```

visualize_fitness_landscape *Visualize Fitness Landscape*

Description

Visualize Fitness Landscape

Usage

```
visualize_fitness_landscape(
  fitness,
  selectNodes = NULL,
  nGenes = 4,
  lowColor = "white",
  highColor = "blue"
)
```

Arguments

fitness	Fitness vectors for each genotype provided in selectNodes or for all genotypes if none selected
selectNodes	Select genotypes to visualize
nGenes	Length of each genotype
lowColor	Color for wild type genotype
highColor	Color for fully mutated genotype

Value

Plot (gg object) visualization of fitness landscape

Examples

```
Genotypes <- c(
    "0000",
    "1000",
    "0100",
    "0010",
    "0001",
    "1100",
    "1010",
    "1001",
    "0110",
    "0101",
    "0011",
    "1110",
    "1101",
    "1011",
    "0111",
    "1111"
)
#
COLintensity <- c(0, rep(0.25, 4), rep(0.5, 6), rep(0.75, 4), 1)
visualize_fitness_landscape(COLintensity)
```

---

visualize_probabilities

Visualize Pathway Probabilities

Description

Visualize Pathway Probabilities

Usage

```
visualize_probabilities(
  probabilities,
  outputFile = NULL,
  geneNames = as.character(1:inverse_factorial(length(probabilities))),
  geneColors = rainbow(length(geneNames), v = 0.5),
  columnTitles = TRUE
)
```

Arguments

probabilities	List or matrix of probabilities for each pathway (matrix if multiple models)
outputFile	File to output to; if none provided, a plot will be returned
geneNames	Gene names; if single character, rendered in circles
geneColors	Gene colors
columnTitles	Include column titles

Value

Plot or file name

Examples

```
visualize_probabilities(c(0.05, 0.03, 0.12, 0.04, 0.02, 0, 0.05, 0.04, 0.05, 0.06, 0.04, 0.02, 0.03, 0.02, 0.05

visualize_probabilities(c(0.05, 0.03, 0.12, 0.04, 0.02, 0, 0.05, 0.04, 0.05, 0.06, 0.04, 0.02, 0.03, 0.02, 0.05

mat <- matrix(c(0.1, 0.3, 0, 0.2, 0.4, 0, 0.2, 0.2, 0.1, 0, 0.2, 0.3), ncol = 2)
visualize_probabilities(mat, columnTitles = TRUE)
```