

# Package ‘CBN2Path’

May 5, 2026

**Title** CBN2Path: an R/Bioconductor package for the analysis of cancer progression pathways using Conjunctive Bayesian Networks

**Version** 1.3.0

**Author** William Choi-Kim [aut, cre], Sayed-Rzgar Hosseini [aut, cre]

**Maintainer**

William Choi-Kim <william@williamck.com>, Sayed-Rzgar Hosseini <razgar@gmail.com>

**Description** CBN2Path package provides a unifying interface to facilitate CBN-based quantification, analysis and visualization of cancer progression pathways.

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

**BugReports** <https://github.com/rockwillck/CBN2Path/issues>

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Imports** R6, ggraph, tidygraph, ggplot2, patchwork, cowplot, magrittr, igraph, rlang, grDevices, coda, graphics, stats, TCGAbiolinks, BiocParallel

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

**Depends** R (>= 4.1.0)

**Config/testthat/edition** 3

**biocViews** Software, StatisticalMethod, GraphAndNetwork, Bayesian, Pathways

**VignetteBuilder** knitr

**SystemRequirements** GNU Scientific Library (GSL)

**OS\_type** unix

**Config/Bioconductor/UnsupportedPlatforms** windows

**git\_url** <https://git.bioconductor.org/packages/CBN2Path>

**git\_branch** devel

**git\_last\_commit** 4253c00

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-05-05

## Contents

CBN2Path-package . . . . .	3
base2Indexing . . . . .	3
base2IndVec . . . . .	4
bcbn . . . . .	4
ctcbn . . . . .	5
ctcbnSingle . . . . .	6
edgeMarginalized . . . . .	7
generateData . . . . .	8
generateMatrixGenotypes . . . . .	8
generateTCGAMatrix . . . . .	9
genotypeFeasibility . . . . .	10
genotypeMatrixMutator . . . . .	10
getExamples . . . . .	11
getRawTCGAData . . . . .	11
hcbn . . . . .	12
hcbnSingle . . . . .	13
jensenShannonDivergence . . . . .	13
pathEdgeMapper . . . . .	14
pathNormalization . . . . .	15
pathProbCBN . . . . .	15
pathProbQuartetBCBN . . . . .	16
pathProbQuartetCTCBN . . . . .	16
pathProbQuartetHCBN . . . . .	17
pathProbQuartetRCBN . . . . .	17
pathProbSSWM . . . . .	18
pathwayCompatibilityQuartet . . . . .	19
pathwayFeasibility . . . . .	19
pathwayGenotypeCompatibility . . . . .	20
pathwayWeightingRCBN . . . . .	20
permutations . . . . .	21
posetWeightingRCBN . . . . .	22
predictability . . . . .	22
readLambda . . . . .	23
readPattern . . . . .	23
readPoset . . . . .	24
readTime . . . . .	24
Spock . . . . .	25
transitiveClosure . . . . .	27
visualizeCBNModel . . . . .	27
visualizeFitnessLandscape . . . . .	28
visualizeProbabilities . . . . .	29

**Index**

**31**

---

CBN2Path-package	<i>CBN2Path: "CBN2Path: an R/Bioconductor package for the analysis of cancer progression pathways using Conjunctive Bayesian Networks"</i>
------------------	--

---

### Description

CBN2Path package provides a unifying interface to facilitate CBN-based quantification, analysis and visualization of cancer progression pathways.

CBN2Path package provides a unifying interface to facilitate CBN-based quantification, analysis and visualization of cancer progression pathways.

### 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>
- Report bugs at <https://github.com/rockwillck/CBN2Path/issues>

Useful links:

- <https://github.com/rockwillck/CBN2Path>
- <http://dx.doi.org/10.1093/biomet/asp023>
- <http://dx.doi.org/10.1093/bioinformatics/btp505>
- Report bugs at <https://github.com/rockwillck/CBN2Path/issues>

---

base2Indexing	<i>base2Indexing</i>
---------------	----------------------

---

### 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 occurrence in the BCBN MCMC-sampling scheme.

**Examples**

```
set.seed(100)
mat <- matrix(sample(c(0, 1), 16, replace = TRUE), 4, 4)
base2Indexing(mat)
```

---

base2IndVec	<i>base2IndVec</i>
-------------	--------------------

---

**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	<i>B-CBN</i>
------	--------------

---

**Description**

B-CBN

**Usage**

```
bcbn(
  data = defaultData(),
  nSamples = 25000,
  theta = 0,
  epsilon = 0.05,
  nChains = 4,
  thin = 10,
  maxL = 1000,
  nCores = 1,
  progressBar = FALSE
)
```

**Arguments**

data	Generated data
nSamples	Number of samples <def: 25000>
theta	Theta <def: 0>
epsilon	Epsilon <def: 0.05>
nChains	N-Chains <def: 4>
thin	Thin <def: 10>
maxL	The maximum number of iteration <def: 1000>
nCores	Number of parallelized cores <def: 1>
progressBar	Print out progress bar; default is FALSE

**Value**

A matrix

**Examples**

```
bcbn()
```

---

ctcbn

*CT-CBN*

---

**Description**

CT-CBN

**Usage**

```
ctcbn(
  datasets,
  bootstrapSamples = 0,
  randomSeed = 1,
  samplingRate = 1,
  epsilon = 2,
  numDrawnSamples = 0,
  numEmRuns = 1,
  nCores = 1,
  progressBar = FALSE
)
```

**Arguments**

datasets	Vector of Spock objects with poset and pattern/lambda data or a Spock object (alias of ctcbnSingle).
bootstrapSamples	Number of bootstrap samples (requires epsilon > 0, numDrawnSamples = 0)
randomSeed	Random seed.
samplingRate	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 and the output Spock will include multiple resulting posets.
numDrawnSamples	If > 0, the number of samples to draw from the model. If zero (default), the model will be learned from data.
numEmRuns	Number of em runs.
nCores	Maximum number of threads to use to parallelize.
progressBar	Print out progress bar; default is FALSE

**Value**

A matrix of results.

**Examples**

```
examplePath <- getExamples()[3]
bc <- Spock$new(
  poset = readPoset(examplePath)$sets,
  numMutations = readPoset(examplePath)$mutations,
  genotypeMatrix = readPattern(examplePath)
)
ctcbn(bc)
ctcbn(c(bc, bc, bc))
```

---

ctcbnSingle

*CT-CBN Single Batch*


---

**Description**

CT-CBN Single Batch

**Usage**

```
ctcbnSingle(
  dataset,
  bootstrapSamples = 0,
  randomSeed = 1,
  samplingRate = 1,
  epsilon = 2,
  numDrawnSamples = 0,
  numEmRuns = 1
)
```

**Arguments**

dataset	Spock object with poset and pattern/lambda data.
bootstrapSamples	Number of bootstrap samples (requires epsilon > 0, numDrawnSamples = 0)
randomSeed	Random seed.

samplingRate	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 and the output Spock will include multiple resulting posets.
numDrawnSamples	If > 0, the number of samples to draw from the model. If zero (default), the model will be learned from data.
numEmRuns	Number of em runs.

**Value**

A list of output data.

**Examples**

```
examplePath <- getExamples()[1]
bc <- Spock$new(
  poset = readPoset(examplePath)$sets,
  numMutations = readPoset(examplePath)$mutations,
  genotypeMatrix = readPattern(examplePath)
)
ctcbnSingle(bc)
```

---

edgeMarginalized	<i>edgeMarginalized</i>
------------------	-------------------------

---

**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 <- pathProbCBN(dag, lambda, x)
edgeMarginalized(pathP, x)
```

---

generateData	<i>Generate Data</i>
--------------	----------------------

---

**Description**

Generate Data

**Usage**

```
generateData(poset, theta, eps, n)
```

**Arguments**

poset	Poset matrix
theta	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 <- transitiveClosure(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
generateData(tr, theta, eps, n)
```

---

generateMatrixGenotypes	<i>generateMatrixGenotypes</i>
-------------------------	--------------------------------

---

**Description**

generateMatrixGenotypes

**Usage**

```
generateMatrixGenotypes(g)
```

**Arguments**

g                    genotype length

**Value**

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

**Examples**

```
generateMatrixGenotypes(4)
```

---

`generateTCGAMatrix`      *Generate TCGA Genotype Matrix*

---

**Description**

Generate TCGA Genotype Matrix

**Usage**

```
generateTCGAMatrix(  
  rawData = suppressMessages(getRawTCGAData("TCGA-BLCA")),  
  genes = c("TP53", "ARID1A", "KDM6A", "PIK3CA", "RB1", "EP300", "FGFR3", "CREBBP",  
            "STAG2", "ATM")  
)
```

**Arguments**

rawData            Raw TCGA data generated using getRawTCGAData  
genes              Genes to generate genotype matrix on

**Value**

A genotype matrix where each row is a patient and each column is a gene

**Examples**

```
generateTCGAMatrix(rawData = data.frame())  
# generateTCGAMatrix()
```

---

`genotypeFeasibility`    *genotypeFeasibility*

---

**Description**

`genotypeFeasibility`

**Usage**

```
genotypeFeasibility(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 <- generateMatrixGenotypes(4)
dag <- matrix(c(4, 4, 4, 1, 2, 3), 3, 2)
x <- 4
genoF4 <- genotypeFeasibility(geno4, dag, x)
```

---

`genotypeMatrixMutator`    *genotypeMatrixMutator*

---

**Description**

`genotypeMatrixMutator`

**Usage**

```
genotypeMatrixMutator(mat, fp, fn)
```

**Arguments**

`mat`                The genotype matrix including sampled genotypes, which need to be mutated.  
`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)
gMatMut <- genotypeMatrixMutator(gMat, 0.2, 0.2)
```

---

getExamples	<i>Get paths to examples</i>
-------------	------------------------------

---

**Description**

Get paths to examples

**Usage**

```
getExamples()
```

**Value**

A vector of paths

**Examples**

```
getExamples()
```

---

getRawTCGAData	<i>Get Raw TCGA Data</i>
----------------	--------------------------

---

**Description**

Get Raw TCGA Data

**Usage**

```
getRawTCGAData(project)
```

**Arguments**

project            TCGA project ID; pass "help" to see list of all project IDs

**Value**

data frame of TCGA data for given project

**Examples**

```
getRawTCGAData("help")
```

hcbn

*H-CBN***Description**

H-CBN

**Usage**

```
hcbn(
  datasets,
  anneal = FALSE,
  temp = 0,
  annealingSteps = 0,
  epsilon = 2,
  nCores = 1,
  progressBar = FALSE
)
```

**Arguments**

datasets	Vector of Spock objects with poset and pattern/lambda data or a Spock object (alias of hcbnSingle).
anneal	If TRUE, performs a simulated annealing run starting from the poset
temp	Temperature of simulated annealing.
annealingSteps	Number of simulated annealing steps.
epsilon	Value of eps for CT-CBN model selection. Requires both pattern and lambda data in input Spock.
nCores	Maximum number of threads to use to parallelize.
progressBar	Print out progress bar; default is FALSE

**Value**

A matrix of results.

**Examples**

```
examplePath <- getExamples()[3]
bc <- Spock$new(
  poset = readPoset(examplePath)$sets,
  numMutations = readPoset(examplePath)$mutations,
  genotypeMatrix = readPattern(examplePath)
)
hcbn(bc)
hcbn(c(bc, bc, bc))
```

---

hcbnSingle	<i>H-CBN Single Batch</i>
------------	---------------------------

---

**Description**

H-CBN Single Batch

**Usage**

```
hcbnSingle(
  datasetObj,
  anneal = FALSE,
  temp = 0,
  annealingSteps = 0,
  epsilon = 2
)
```

**Arguments**

datasetObj	Spock object with poset and pattern/lambda data.
anneal	If TRUE, performs a simulated annealing run starting from the poset
temp	Temperature of simulated annealing.
annealingSteps	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**

```
examplePath <- getExamples()[1]
bc <- Spock$new(
  poset = readPoset(examplePath)$sets,
  numMutations = readPoset(examplePath)$mutations,
  genotypeMatrix = readPattern(examplePath)
)
hcbnSingle(bc)
```

---

jensenShannonDivergence	<i>jensenShannonDivergence</i>
-------------------------	--------------------------------

---

**Description**

jensenShannonDivergence

**Usage**

```
jensenShannonDivergence(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), 12, replace = TRUE), 3, 4)
pathCT <- pathProbQuartetCTCBN(gMat)
pathH <- pathProbQuartetHCBN(gMat)
jensenShannonDivergence(pathCT, pathH)
```

---

pathEdgeMapper

*pathEdgeMapper*

---

**Description**

pathEdgeMapper

**Usage**

```
pathEdgeMapper(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 <- pathEdgeMapper(4)
```

---

pathNormalization      *pathNormalization*

---

**Description**

pathNormalization

**Usage**

```
pathNormalization(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 <- pathProbCBN(dag, lambda, x)
pathN <- pathNormalization(pathP, x)
```

---

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

---

**Description**

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

**Usage**

```
pathProbCBN(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 <- pathProbCBN(dag, lambda, x)
```

---

`pathProbQuartetBCBN`    *pathProbQuartetBCBN*

---

**Description**

`pathProbQuartetBCBN`

**Usage**

`pathProbQuartetBCBN(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), 12, replace = TRUE), 3, 4)
pathProbQuartetBCBN(gMat)
```

---

`pathProbQuartetCTCBN`    *pathProbQuartetCTCBN*

---

**Description**

`pathProbQuartetCTCBN`

**Usage**

`pathProbQuartetCTCBN(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), 12, replace = TRUE), 3, 4)
pathProbQuartetCTCBN(gMat)
```

---

pathProbQuartetHCBN    *pathProbQuartetHCBN*

---

**Description**

pathProbQuartetHCBN

**Usage**

```
pathProbQuartetHCBN(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), 12, replace = TRUE), 3, 4)
pathProbQuartetHCBN(gMat)
```

---

pathProbQuartetRCBN    *pathProbQuartetRCBN*

---

**Description**

pathProbQuartetRCBN

**Usage**

```
pathProbQuartetRCBN(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), 12, replace = TRUE), 3, 4)
pathProbQuartetRCBN(gMat)
```

---

`pathProbSSWM`

*pathProbSSWM*

---

**Description**

`pathProbSSWM`

**Usage**

```
pathProbSSWM(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 <- pathProbSSWM(f, x)
```

---

pathwayCompatibilityQuartet  
*pathwayCompatibilityQuartet*

---

**Description**

pathwayCompatibilityQuartet

**Usage**

```
pathwayCompatibilityQuartet(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)
pathwayCompatibilityQuartet(gMat)
```

---

pathwayFeasibility    *pathwayFeasibility*

---

**Description**

pathwayFeasibility

**Usage**

```
pathwayFeasibility(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
pathwayFeasibility(dag, x)
```

---

pathwayGenotypeCompatibility  
*pathwayGenotypeCompatibility*

---

**Description**

pathwayGenotypeCompatibility

**Usage**

```
pathwayGenotypeCompatibility(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)
pathwayGenotypeCompatibility(path, geno1)
pathwayGenotypeCompatibility(path, geno2)
```

---

pathwayWeightingRCBN    *pathwayWeightingRCBN*

---

**Description**

pathwayWeightingRCBN

**Usage**

```
pathwayWeightingRCBN(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 <- pathProbCBN(dag, lambda, x)
edgeProb <- edgeMarginalized(pathP, x)
peMap <- pathEdgeMapper(4)
pathwayWeightingRCBN(edgeProb, peMap)
```

---

permutations

*permutations*

---

**Description**

permutations

**Usage**

```
permutations(n, r, v = 1:n, set = TRUE, repeatsAllowed = 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.
repeatsAllowed	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)
```

---

posetWeightingRCBN      *posetWeightingRCBN*

---

**Description**

posetWeightingRCBN

**Usage**

```
posetWeightingRCBN(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 <- posetWeightingRCBN(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), 12, replace = TRUE), 3, 4)
pathCT <- pathProbQuartetCTCBN(gMat)
pathH <- pathProbQuartetHCBN(gMat)
predC <- predictability(pathCT, 4)
predictability(pathH, 4)
```

---

readLambda	<i>Read a .lambda file</i>
------------	----------------------------

---

**Description**

Read a .lambda file

**Usage**

```
readLambda(fileStem)
```

**Arguments**

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

**Value**

A matrix.

**Examples**

```
bcPath <- getExamples()[1]
readLambda(bcPath)
```

---

readPattern	<i>Read a .pat file</i>
-------------	-------------------------

---

**Description**

Read a .pat file

**Usage**

```
readPattern(fileStem)
```

**Arguments**

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

**Value**

A matrix.

**Examples**

```
bcPath <- getExamples()[1]
readPattern(bcPath)
```

---

readPoset	<i>Read a .poset file</i>
-----------	---------------------------

---

**Description**

Read a .poset file

**Usage**

```
readPoset(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 <- getExamples()[1]
readPoset(bcPath)
```

---

readTime	<i>Read a .time file</i>
----------	--------------------------

---

**Description**

Read a .time file

**Usage**

```
readTime(fileStem)
```

**Arguments**

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

**Value**

A matrix.

**Examples**

```
bcPath <- getExamples()[1]
readPattern(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.

## Value

a Spock object

## 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\\$show\(\)](#)
- [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** `show()`: Print summary information to console.

*Usage:*

```
Spock$show(verbose = FALSE)
```

*Arguments:*

verbose Method prints contents as well as dimensions to console if TRUE.

*Returns:* Nothing.

**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.

## Examples

```
examplePath <- getExamples()[1]
bc <- Spock$new(
  poset = readPoset(examplePath)$sets,
  numMutations = readPoset(examplePath)$mutations,
  genotypeMatrix = readPattern(examplePath)
)
```

---

transitiveClosure	<i>Transitive Closure</i>
-------------------	---------------------------

---

**Description**

Transitive Closure

**Usage**

```
transitiveClosure(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
transitiveClosure(poset)
```

---

visualizeCBNModel	<i>Visualize CBN Model</i>
-------------------	----------------------------

---

**Description**

Visualize CBN Model

**Usage**

```
visualizeCBNModel(
  poset,
  nodeColor = "darkgreen",
  numNodes = max(4, max(poset))
)
```

**Arguments**

poset	Poset object to visualize
nodeColor	Color of nodes in resulting graph
numNodes	Number of nodes (default is the larger number between 4 and the largest index given in the poset)

**Value**

Plot (gg object) visualization of CBN model

**Examples**

```
poset <- readPoset(getExamples()[1])
visualizeCBNModel(poset$sets)
```

---

```
visualizeFitnessLandscape
      Visualize Fitness Landscape
```

---

**Description**

Visualize Fitness Landscape

**Usage**

```
visualizeFitnessLandscape(
  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)
visualizeFitnessLandscape(colIntensity)

```

---

visualizeProbabilities

*Visualize Pathway Probabilities*


---

**Description**

Visualize Pathway Probabilities

**Usage**

```

visualizeProbabilities(
  probabilities,
  outputFile = NULL,
  geneNames = as.character(1:inverseFactorial(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**

```
visualizeProbabilities(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),
visualizeProbabilities(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)
visualizeProbabilities(mat, columnTitles = TRUE)
```

# Index

## \* internal

CBN2Path-package, 3

base2Indexing, 3

base2IndVec, 4

bcbn, 4

CBN2Path-package, 3

ctcbn, 5

ctcbnSingle, 6

edgeMarginalized, 7

generateData, 8

generateMatrixGenotypes, 8

generateTCGAMatrix, 9

genotypeFeasibility, 10

genotypeMatrixMutator, 10

getExamples, 11

getRawTCGAData, 11

hcbn, 12

hcbnSingle, 13

jensenShannonDivergence, 13

pathEdgeMapper, 14

pathNormalization, 15

pathProbCBN, 15

pathProbQuartetBCBN, 16

pathProbQuartetCTCBN, 16

pathProbQuartetHCBN, 17

pathProbQuartetRCBN, 17

pathProbSSWM, 18

pathwayCompatibilityQuartet, 19

pathwayFeasibility, 19

pathwayGenotypeCompatibility, 20

pathwayWeightingRCBN, 20

permutations, 21

posetWeightingRCBN, 22

predictability, 22

readLambda, 23

readPattern, 23

readPoset, 24

readTime, 24

Spock, 25

transitiveClosure, 27

visualizeCBNModel, 27

visualizeFitnessLandscape, 28

visualizeProbabilities, 29