Package ‘Lheuristic’

May 26, 2026

Title Detection of scatterplots with L-shaped pattern

Version 1.5.0

Description The Lheuristic package identifies scatterpots that follow and L-shaped, negative distribution. It can be used to identify genes regulated by methylation by integration of an expression and a methylation array. The package uses two different methods to detect expression and methyaltion L-shapped scatterplots. The parameters can be changed to detect other scatterplot patterns.

License MIT + file LICENSE

Depends R (>= 4.4.0)

biocViews DNAMethylation, StatisticalMethod, MethylationArray

Imports Hmisc, stats, energy, grDevices, graphics, utils, MultiAssayExperiment, ggplot2, ggpubr

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

URL `https://github.com/ASPresearch/Lheuristic`

BugReports `https://github.com/ASPresearch/Lheuristic/issues`

VignetteBuilder knitr

Encoding UTF-8

LazyData false

RoxygenNote 7.3.2

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/Lheuristic

git_branch devel

git_last_commit 6f0f0ae

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-25

Author Sanchez Pla Alex [aut, cre] (ORCID: <`https://orcid.org/0000-0002-8673-7737`>), Miro Cau Berta [aut] (ORCID: <`https://orcid.org/0000-0001-6049-8697`>)

Maintainer Sanchez Pla Alex <`asanchez@ub.edu`>

Contents

`binScore` *binScore*

Description

`binScore` can be used to score scatterplots by directly comparing the sample counts with a matrix of minimal or maximal percentages/counts to be found in each cell. It implements the three bands rule implicitly by setting threshold values.

Usage

```
binScore(aGrid, aReq)
```

Arguments

`aGrid`	A matrix of counts as computed by ‘calcFreqs‘ function.
`aReq`	A matrix of minimum or maximum counts to be found in each cell if L-shape is TRUE.

Value

a score value for each scatterplot

Examples

```
# Generate some example data
aGrid <- matrix(c(20, 3, 0, 10, 2, 2, 20, 10, 20),
    nrow = 3, ncol = 3, byrow = TRUE
)
aReq <- matrix(c(15, 5, 0, 0, 5, 5, 10, 10, 15),
    nrow = 3, ncol = 3, byrow = TRUE
)
```

```
# Calculate the bin score
binScore(aGrid, aReq)
```

---

calcFreqs *calcFreqs*

Description

`calcFreqs` Given a MultiAssayExperiment with methylation and expression data, the function overlays a grid on a YMet ~ Xmet scatterplot and returns a 3x3 matrix with point counts per grid cell based on the vertical and horizontal lines.

Usage

```
calcFreqs(
  mae,
  geneNum,
  x1,
  x2,
  y1 = NULL,
  y2 = NULL,
  percY1 = 1/3,
  percY2 = 2/3
)
```

Arguments

`mae`	MultiAssayExperiment object containing methylation and expression matrices.
`geneNum`	row of expression/methylation matrix for which the frequencies will be computed.
`x1, x2`	Coordinates of vertical points in the X axis. Because it is expected to contain. methylation values that vary between 0 and 1, default values are 1/3 and 2/3.
`y1, y2`	Coordinates of vertical points on the Y-axis. If set to NULL, they will be automatically assigned as the percentiles of ‘y‘ defined by ‘percY1‘ and ‘percY2‘.
`percY1, percY2`	Default values for ‘y1‘ and ‘y2‘ when set to ‘NULL‘

Value

a matrix with calculated frequencies

Examples

```
# Methylation data
methylData <- matrix(runif(50), nrow = 10)
colnames(methylData) <- paste0(
    "samp",
    1:ncol(methylData)
)
rownames(methylData) <- paste0(
    "gene",
```

```
    1:nrow(methylData)
)
# Expression data
expresData <- matrix(rnorm(50), nrow = 10)
colnames(expresData) <- paste0(
    "samp",
    1:ncol(methylData)
)
rownames(expresData) <- paste0(
    "gene",
    1:nrow(methylData)
)
# ColData
colDat <- data.frame(
    sampleID = colnames(methylData),
    name = letters[1:ncol(methylData)]
)

rownames(colDat) <- colDat$sampleID
mae <- MultiAssayExperiment::MultiAssayExperiment(
    experiments = list(
        methylation = methylData,
        expression = expresData
    ),
    colData = colDat
)
geneRow <- 1
x1 <- 1 / 3
x2 <- 2 / 3
y1 <- NULL
y2 <- NULL
percY1 <- 1 / 3
percY2 <- 2 / 3

calcFreqs(
    mae = mae, geneNum = geneRow,
    x1, x2, y1, y2, percY1, percY2
)
```

`checkPairing` *checkPairing*

Description

`checkPairing` function to check if two matrices have the same dimensions row and column names.

Usage

```
checkPairing(X, Y)
```

Arguments

`X`	matrix
`Y`	matrix

Value

a logical value indicating if the two matrices have the same row and column names.

Examples

```
(X <- round(matrix(rnorm(30) * 10, ncol = 6), 1)) + 1:10
(Y <- round(X + matrix(rnorm(30) * 10, ncol = 6), 1)) - 10:1
(rownames(X) <- rownames(Y) <- letters[1:nrow(X)])
(m1 <- checkPairing(X, Y))
```

`correlationSelection` *correlationSelection: A vector correlation function calculator*

Description

`correlationSelection` Uses the function matCorrs; given two matrices X (m,n), Y (m,n) this function computes Pearson and Spearman correlation coefficients and their significance p-values for every pair of row vectors.

Usage

```
correlationSelection(
  mae,
  type = "Spearman",
  adj = TRUE,
  pValCutoff = 0.05,
  rCutoff = 0,
  sortByCorrs = FALSE
)
```

Arguments

`mae`	a MultiAssayExperiment object containing the methylation
`type`	specifies the correlation to choose between Spearman and Pearson. Default is Spearman.
`adj`	logical variable indicating if the p-value returned should be adjusted or not. Default is TRUE, it returns an adjusted p-value.
`pValCutoff`	the upper limit used for the p-value. Default is 0.05.
`rCutoff`	the upper limit used for the correlation coefficient. Default is 0, no cut off.
`sortByCorrs`	logical; if TRUE, results are ordered by ascending p-value. Default set to FALSE.

Value

dataframe with the correlations selected

Examples

```
# Methylation data
methylData <- matrix(runif(50), nrow = 10)
colnames(methylData) <- paste0("samp", 1:ncol(methylData))
rownames(methylData) <- paste0("gene", 1:nrow(methylData))
# Expression data
expresData <- matrix(rnorm(50), nrow = 10)
colnames(expresData) <- paste0("samp", 1:ncol(methylData))
rownames(expresData) <- paste0("gene", 1:nrow(methylData))
# ColData
colDat <- data.frame(
    sampleID = colnames(methylData),
    name = letters[1:ncol(methylData)]
)

rownames(colDat) <- colDat$sampleID
mae <- MultiAssayExperiment::MultiAssayExperiment(
    experiments = list(
        methylation = methylData,
        expression = expresData
    ),
    colData = colDat
)
correlationSelection(mae, pValCutoff = 0.25, rCutoff = 0.1,
type = "Spearman", sortByCorrs = TRUE)
```

`lhCreateMAE` *lhCreateMAE: Create a MultiAssayExperiment with Methylation and Expression Data*

Description

This function constructs a MultiAssayExperiment object using methylation and expression data, automatically ensuring consistency in sample names, feature names, and exposing dataset names. It internally uses `checkPairing` to validate that the provided datasets have matching row and column names.

Usage

```
lhCreateMAE(
  xDat,
  yDat,
  xName = "methylation",
  yName = "expression",
  colData = NULL
)
```

Arguments

`xDat`	A numeric matrix containing methylation data (features in rows, samples in columns).
`yDat`	A numeric matrix containing expression data (features in rows, samples in columns).

`xName`	A string specifying the name of the methylation dataset. Default: "methylation".
`yName`	A string specifying the name of the expression dataset. Default: "expression".
`colData`	(Optional) A DataFrame containing sample-level metadata. Default: NULL.

Value

A MultiAssayExperiment object containing the provided datasets with sample metadata.

Examples

```
library(MultiAssayExperiment)

# Create synthetic methylation and expression data
methylData <- matrix(runif(50), nrow = 10)
colnames(methylData) <- paste0("samp", 1:ncol(methylData))
rownames(methylData) <- paste0("gene", 1:nrow(methylData))

expresData <- matrix(rnorm(50), nrow = 10)
colnames(expresData) <- colnames(methylData)
rownames(expresData) <- rownames(methylData)

# Create sample metadata
colDat <- data.frame(sampleID = colnames(methylData),
name = letters[1:ncol(methylData)])
rownames(colDat) <- colDat$sampleID

# Construct MultiAssayExperiment
mae <- lhCreateMAE(methylData, expresData, colData = colDat)

# Display dataset names
names(experiments(mae))
```

`matCorrs` *A row-wise correlation function calculator*

Description

`matCorrs` Given a MultiAssayExperiment object, the function computes Pearson and Spearman correlation coefficients and their significance p-values for every pair of row vectors.

Usage

```
matCorrs(mae)
```

Arguments

`mae`	A MultiAssayExperiment object containing the methylation and expression data.

Value

matrix with a correlation value for each gene

Examples

```
# Methylation data
methylData <- matrix(runif(50), nrow = 10)
colnames(methylData) <- paste0("samp", 1:ncol(methylData))
rownames(methylData) <- paste0("gene", 1:nrow(methylData))
# Expression data
expresData <- matrix(rnorm(50), nrow = 10)
colnames(expresData) <- paste0("samp", 1:ncol(methylData))
rownames(expresData) <- paste0("gene", 1:nrow(methylData))
# ColData
colDat <- data.frame(
    sampleID = colnames(methylData),
    name = letters[1:ncol(methylData)]
)

rownames(colDat) <- colDat$sampleID
mae <- MultiAssayExperiment::MultiAssayExperiment(
    experiments = list(
        methylation = methylData,
        expression = expresData
    ),
    colData = colDat
)
matCorrs(mae)
```

`messageTitle` *messageTitle*

Description

`messageTitle` A wrapper function for sending messages to console.

Usage

```
messageTitle(aMessage, underChar = "-")
```

Arguments

`aMessage`	text of the message.
`underChar`	character to use for underlining the message.

Value

a message to console

Examples

```
messageTitle("Hello world", "$")
```

`numScore` *numScore*

Description

`numScore` A function to score scatterplot using a weight matrix. The scoring does not incorporate logical conditions such as 'if xij < C ...'

Usage

```
numScore(aGrid, LShaped, aWeightMifL, aWeightMifNonL)
```

Arguments

`aGrid`	A matrix of counts computed by the 'calcFreqs' function.
`LShaped`	A boolean indicating whether the scatterplot can be classified as "L-shaped".
`aWeightMifL`	A matrix of weights applied to score counts if the scatterplot is classified as "L".
`aWeightMifNonL`	A matrix of weights applied to score counts if the scatterplot is classified as "non-L".

Value

a numeric score for each scatterplot

Examples

```
# Methylation data
methylData <- matrix(runif(50), nrow = 10)
colnames(methylData) <- paste0("samp", 1:ncol(methylData))
rownames(methylData) <- paste0("gene", 1:nrow(methylData))
# Expression data
expresData <- matrix(rnorm(50), nrow = 10)
colnames(expresData) <- paste0("samp", 1:ncol(methylData))
rownames(expresData) <- paste0("gene", 1:nrow(methylData))
# ColData
colDat <- data.frame(
    sampleID = colnames(methylData),
    name = letters[1:ncol(methylData)]
)

rownames(colDat) <- colDat$sampleID
mae <- MultiAssayExperiment::MultiAssayExperiment(
    experiments = list(
        methylation = methylData,
        expression = expresData
    ),
    colData = colDat
)
trueFreq <- calcFreqs(mae, x1 = 1 / 3, x2 = 2 / 3)
LShaped <- FALSE
weightsIfL <- matrix(c(2, -1, -99, 1, 0, -1, 1, 1, 2),
    nrow = 3, byrow = TRUE
)
```

```
weightsIfNonL <- matrix(c(0, -1, -99, 0, 0, -1, 0, 0, 0),
    nrow = 3, byrow = TRUE
)
numScore(
    trueFreq, LShaped,
    weightsIfL, weightsIfNonL
)
```

`plotGeneByName` *plotGeneByName*

Description

`plotGeneByName` plots points on a scatterplot with a 3x3 grid superimposed. The name of a the gene is provided jointly with the MultiAssayExperiment object and used to select the row to be plotted.

Usage

```
plotGeneByName(
  geneName,
  mae,
  filename = NULL,
  text4Title = NULL,
  plotGrid = TRUE,
  figs = c(2, 2)
)
```

Arguments

`geneName`	The name of the gene to be plotted.
`mae`	A MultiAssayExperiment object containing the methylation and expression data for the specified gene.
`filename`	If provided, the name of the file to save the results as a PDF; defaults to NULL.
`text4Title`	A string used as the main title for the plot. Defaults to ‘geneName‘ if set to NULL.
`plotGrid`	A boolean parameter indicating whether to pass the grid option to the ‘plotGeneSel‘ function.
`figs`	A two-component vector defining the 2-dimensional structure of the plots to be generated.

Value

a pdf with scatterplots of selected by gene name

Examples

```
# Plot gene by name based on example data
# Methylation data
methylData <- matrix(runif(50), nrow = 10)
colnames(methylData) <- paste0("samp", 1:ncol(methylData))
rownames(methylData) <- paste0("gene", 1:nrow(methylData))
# Expression data
expresData <- matrix(rnorm(50), nrow = 10)
colnames(expresData) <- paste0("samp", 1:ncol(methylData))
rownames(expresData) <- paste0("gene", 1:nrow(methylData))
# ColData
colDat <- data.frame(
    sampleID = colnames(methylData),
    name = letters[1:ncol(methylData)]
)

rownames(colDat) <- colDat$sampleID
mae <- MultiAssayExperiment::MultiAssayExperiment(
    experiments = list(
        methylation = methylData,
        expression = expresData
    ),
    colData = colDat
)
plotGeneByName(gene = "gene7", mae = mae)
```

`plotGeneSel` *plotGeneSel*

Description

`plotGeneSel` plots points on a scatterplot with a 3x3 grid overimposed.

Usage

```
plotGeneSel(
  mae,
  genePos,
  titleText,
  x1 = 1/3,
  x2 = 2/3,
  y1 = NULL,
  y2 = NULL,
  percY1 = 1/3,
  percY2 = 2/3,
  plotGrid = TRUE
)
```

Arguments

`mae`	A MultiAssayExperiment object containing the methylation and expression data for the specified gene.

`genePos`	The index of the gene to be plotted within the MultiAssayExperiment object.
`titleText`	plot title.
`x1, x2`	Coordinates of vertical points in the X axis. Because it is expected to contain methylation values that vary between 0 and 1. The default values are 1/3 and 2/3.
`y1, y2`	Coordinates of vertical points in the Y axis. Leaving them as NULL assigns them the percentiles of yVec defined by 'percY1' and 'percY2'.
`percY1, percY2`	Values used to act as default for 'y1'and 'y2' when these are set to 'NULL'
`plotGrid`	logical. Defautl to TRUE will plot gridlines over the scatterplot.

Value

a pdf with scatterplots for selected genes

Examples

```
# Methylation data
methylData <- matrix(runif(50), nrow = 10)
colnames(methylData) <- paste0("samp", 1:ncol(methylData))
rownames(methylData) <- paste0("gene", 1:nrow(methylData))
# Expression data
expresData <- matrix(rnorm(50), nrow = 10)
colnames(expresData) <- paste0("samp", 1:ncol(methylData))
rownames(expresData) <- paste0("gene", 1:nrow(methylData))
# ColData
colDat <- data.frame(
    sampleID = colnames(methylData),
    name = letters[1:ncol(methylData)]
)

rownames(colDat) <- colDat$sampleID
mae <- MultiAssayExperiment::MultiAssayExperiment(
    experiments = list(
        methylation = methylData,
        expression = expresData
    ),
    colData = colDat
)

plotGeneSel(mae, genePos = 7, titleText = "L-shaped gene")
```

`plotGenesMat` *plotGenesMat*

Description

`plotGenesMat` wrapper function for plotting the scatterplots associated with two matrices.

Usage

```
plotGenesMat(
  mae,
  geneNames = NULL,
  fileName = "Scatplot_Sel_Genes.pdf",
  text4Title = NULL,
  x1 = 1/3,
  x2 = 2/3,
  y1 = NULL,
  y2 = NULL,
  percY1 = 1/3,
  percY2 = 2/3,
  plotGrid = TRUE,
  logicSc = NULL,
  saveToPDF = FALSE,
  plotsPerPage = 9
)
```

Arguments

`mae`	A MultiAssayExperiment object containing the methylation and expression data.
`geneNames`	A character vector of gene names to plot. If NULL, all genes are plotted.
`fileName`	The name of the file used to save the results as a PDF. If NULL, the plot is displayed on the screen.
`text4Title`	An optional title for the plot, incorporating the gene name and L-shape score. Defaults to NULL.
`x1`	The x-coordinate of vertical points on the X-axis. Expected to contain methylation values ranging between 0 and 1, with default values set to 1/3 and 2/3.
`x2`	The x-coordinate of vertical points on the X-axis. Expected to contain methylation values ranging between 0 and 1, with default values set to 1/3 and 2/3.
`y1`	The y-coordinate of vertical points on the Y-axis. If NULL, these are set to the percentiles of yVec defined by percY1 and percY2.
`y2`	The y-coordinate of vertical points on the Y-axis. If NULL, these are set to the percentiles of yVec defined by percY1 and percY2.
`percY1`	Values used as defaults for y1 when it is set to NULL.
`percY2`	Values used as defaults for y2 when it is set to NULL.
`plotGrid`	A logical value; defaults to TRUE, indicating whether to plot gridlines on the graph.
`logicSc`	A numeric score representing the L-shape score. Defaults to NULL.
`saveToPDF`	Logical, if TRUE, saves the plots to a PDF file specified by fileName. If FALSE (default), plots are displayed interactively without saving.
`plotsPerPage`	A numeric value indicating the number of plots to be printed in each page. Default set to 9.

Value

a pdf with scatterplots for all genes

Examples

```
# Methylation data
methylData <- matrix(runif(50), nrow = 10)
colnames(methylData) <- paste0("samp", 1:ncol(methylData))
rownames(methylData) <- paste0("gene", 1:nrow(methylData))
# Expression data
expresData <- matrix(rnorm(50), nrow = 10)
colnames(expresData) <- paste0("samp", 1:ncol(methylData))
rownames(expresData) <- paste0("gene", 1:nrow(methylData))
# ColData
colDat <- data.frame(
sampleID = colnames(methylData),
name = letters[1:ncol(methylData)])

rownames(colDat) <- colDat$sampleID
mae <- MultiAssayExperiment::MultiAssayExperiment(
    experiments = list(
        methylation = methylData,
        expression = expresData
    ),
    colData = colDat
)
selectedGenes <- c("gene1", "gene5")
plotGenesMat(mae, geneNames = selectedGenes,
saveToPDF = FALSE)
```

`scoreGenesMat` *scoreGenesMat*

Description

`scoreGenesMat` scores scatterplots using a binary and a numeric schemes row-wise.

Usage

```
scoreGenesMat(
  mae,
  x1 = 1/3,
  x2 = 2/3,
  y1 = NULL,
  y2 = NULL,
  percY1 = 1/3,
  percY2 = 2/3,
  aReqPercentsMat,
  aWeightMifL = 0.5,
  aWeightMifNonL = 0.25
)
```

Arguments

`mae`	MultiAssayExperiment object containing methylation and expression matrices.

`x1, x2`	Coordinates of vertical points on the X-axis. Expected to contain methylation values ranging between 0 and 1, with default values set to 1/3 and 2/3.
`y1, y2`	Coordinates of vertical points on the Y-axis. If set to NULL, they default to the percentiles of yVec as defined by ‘percY1‘ and ‘percY2‘.
`percY1, percY2`	Values used to act as default for ‘y1‘and ‘y2‘ when these are set to ‘NULL‘.
`aReqPercentsMat`	A matrix specifying the minimum and maximum percentage counts required in each cell.
`aWeightMifL`	A matrix of weights applied to score counts when the scatterplot is classified as "L".
`aWeightMifNonL`	A matrix of weights applied to score counts when the scatterplot is classified as "non-L".

Value

A data frame with two columns: ’logicSc’ (logical score indicating if the gene is ’active’) and ’numericSc’ (a numerical score).

Examples

```
# Score genes based on example data

# Methylation data
methylData <- matrix(runif(50), nrow = 10)
colnames(methylData) <- paste0("samp", 1:ncol(methylData))
rownames(methylData) <- paste0("gene", 1:nrow(methylData))
# Expression data
expresData <- matrix(rnorm(50), nrow = 10)
colnames(expresData) <- paste0("samp", 1:ncol(methylData))
rownames(expresData) <- paste0("gene", 1:nrow(methylData))
# ColData
colDat <- data.frame(
    sampleID = colnames(methylData),
    name = letters[1:ncol(methylData)]
)

rownames(colDat) <- colDat$sampleID
mae <- MultiAssayExperiment::MultiAssayExperiment(
    experiments = list(
        methylation = methylData,
        expression = expresData
    ),
    colData = colDat
)
sampleSize <- ncol(MultiAssayExperiment::experiments(mae)[[1]])
reqPercentages <- matrix(c(3, 20, 5, 5, 40, 20, 4, 1, 2),
    nrow = 3, byrow = TRUE
)
(theWeightMifL <- matrix(c(2, -2, -sampleSize / 5, 1, 0, -2, 1, 1, 2),
    nrow = 3, byrow = TRUE
))
(theWeightMifNonL <- matrix(c(0, -2, -sampleSize / 5, 0, 0, -2, 0, 0, 0),
    nrow = 3, byrow = TRUE
))
```

```
scoreGenesMat(mae,
    x1 = 1 / 3, x2 = 2 / 3,
    y1 = NULL, y2 = NULL, percY1 = 1 / 3, percY2 = 2 / 3,
    aReqPercentsMat = reqPercentages,
    aWeightMifL = theWeightMifL,
    aWeightMifNonL = theWeightMifNonL
)
```

`TCGAexpression` *Expression data matrix obtained from TCGA, TCGA-COAD dataset*

Description

A mxn matrix containing normalized gene expressions from TCGA

Usage

```
data(TCGAexpression)
```

Format

An object of class `matrix` (inherits from `array`) with 1000 rows and 30 columns.

Value

data matrix with gene expression data

Source

TCGA.org

`TCGAmethylation` *Methylation data matrix obtained from TCGA, TCGA-COAD dataset*

Description

A mxn matrix containing normalized methylation values from TCGA

Usage

```
data(TCGAmethylation)
```

Format

An object of class `matrix` (inherits from `array`) with 1000 rows and 30 columns.

Value

data matrix with gene methylation data

Source

TCGA.org

`toReqMat` *toReqMat*

Description

`toReqMat` can be used to turn a matrix of required percentages into a matrix of required counts to facilitate its scoring.

Usage

```
toReqMat(numPoints, aReqPercentMat)
```

Arguments

`numPoints`	Number of points in a scatterplot. Used to turn the required percentages into required counts.
`aReqPercentMat`	Matrix of required percentages.

Value

a counts matrix

Examples

```
reqPercentages <- matrix(
    c(
        3, 20, 5,
        5, 40, 20,
        4, 1, 2
    ),
    nrow = 3, byrow = TRUE
)
numberOfPoints <- 100
reqMat <- toReqMat(
    numPoints = numberOfPoints,
    aReqPercentMat = reqPercentages
)
```

Index