Package 'Rediscover'

July 21, 2025

Type Package

Title Identify Mutually Exclusive Mutations

Version 0.3.2

Author Juan A. Ferrer-Bonsoms, Laura Jareno, and Angel Rubio

Maintainer Juan A. Ferrer-Bonsoms <jafhernandez@tecnun.es>

Description An optimized method for identifying mutually exclusive genomic events. Its main contribution is a statistical analysis based on the Poisson-Binomial distribution that takes into account that some samples are more mutated than others. See [Canisius, Sander, John WM Martens, and Lodewyk FA Wessels. (2016) ``A novel independence test for somatic alterations in cancer shows that biology drives mutual exclusivity but chance explains most co-occurrence." Genome biology 17.1 : 1-17. <doi:10.1186/s13059-016-1114-x>]. The mutations matrices are sparse matrices. The method developed takes advantage of the advantages of this type of matrix to save time and computing resources.

Depends R (>= 4.0), Matrix, PoissonBinomial, ShiftConvolvePoibin, utils, matrixStats

Imports maftools, data.table, parallel, RColorBrewer, methods

Suggests knitr, rmarkdown, RUnit, BiocStyle, BiocGenerics, dplyr, kableExtra, magick, stats, qvalue

License Artistic-2.0

LazyData true

RoxygenNote 7.2.3

Encoding UTF-8

biocViews mutex

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2023-04-14 12:10:06 UTC

Contents

AMP_COAD . . . 2
A_example . . . 3
A_Matrix . . . 3
B_example . . . 4
B_Matrix . . . 4
discoversomaticInteractions . . . 5
getMutex . . . 7
getMutexAB . . . 9
getMutexGroup . . . 11
getPM . . . 12
PMatrix-class . . . 13
PM_AMP_COAD . . . 13
PM_COAD . . . 14
TCGA_COAD . . . 14

Index **16**

`AMP_COAD` *AMP_COAD data*

Description

A binary matrix, with information about amplifications in Colon Adenocarcinoma, created by applying `GDCquery` and used as real example in `getMutexAB`.

Usage

```
data("AMP_COAD")
```

Format

The format is:

num [1:1000, 1:391] 0 0 0 0 0 0 0 0 0 0 ...

- attr(*, "dimnames")=List of 2

..$: chr [1:1000] "ENSG00000212993.4" "ENSG00000279524.1" "ENSG00000136997.13" "ENSG00000101294.15" ...

..$: chr [1:391] "TCGA-CA-6718" "TCGA-D5-6931" "TCGA-AZ-6601" "TCGA-G4-6320" ...

Examples

```
data(AMP_COAD)
## maybe str(AMP_COAD)
```

A_example *A_example data*

Description

A binary matrix of class `matrix` used as toy example in `getPM` and `getMutex` and `getMutexAB` and `getMutexGroup`.

Usage

```
data("A_example")
```

Format

The format is: num [1:1000, 1:500] 0 0 0 0 0 1 0 0 0 1 ...

Examples

```
data(A_example)
```

A_Matrix *A_Matrix data*

Description

A binary dgCMatrix matrix used as toy example in `getPM` and `getMutex` and `getMutexAB` and `getMutexGroup`

Usage

```
data("A_Matrix")
```

Format

The format is:

Formal class 'dgCMatrix' [package "Matrix"] with 6 slots

..@ i : int [1:249838] 5 9 10 11 13 14 18 20 23 24 ...

..@ p : int [1:501] 0 503 1010 1506 1995 2497 2981 3488 4002 4474 ...

..@ Dim : int [1:2] 1000 500

..@ Dimnames:List of 2

.. ..$: NULL

.. ..$: NULL

..@ x : num [1:249838] 1 1 1 1 1 1 1 1 1 1 ...

..@ factors : list()

Examples

```
data(A_Matrix)
```

`B_example` *B_example data*

Description

A binary matrix of class `matrix` used as toy example in `getPM` and `getMutex` and `getMutexAB` and `getMutexGroup`.

Usage

```
data("B_example")
```

Format

The format is:

int [1:1000, 1:500] 0 1 1 0 1 0 0 0 1 1 ...

Examples

```
data(B_example)
```

`B_Matrix` *B_Matrix data*

Description

A binary dgCMatrix matrix used as toy example in `getPM` and `getMutex` and `getMutexAB` and `getMutexGroup`.

Usage

```
data("B_Matrix")
```

Format

The format is:

Formal class 'dgCMatrix' [package "Matrix"] with 6 slots

..@ i : int [1:249526] 1 2 4 8 9 11 13 15 18 20 ...

..@ p : int [1:501] 0 498 1014 1527 2048 2558 3036 3511 4035 4537 ...

..@ Dim : int [1:2] 1000 500

..@ Dimnames:List of 2

.. ..$: NULL

.. ..$: NULL

..@ x : num [1:249526] 1 1 1 1 1 1 1 1 1 1 ...

..@ factors : list()

Examples

```
data(B_Matrix)
```

---

discoversomaticInteractions *discoversomaticInteractions*

Description

Function adapted to maftools where given a .maf file, it graphs the somatic interactions between a group of genes, i.e., the combination of gene expression and mutation data to detect mutually exclusive or co-ocurring events.

Usage

```
discoversomaticInteractions(
  maf,
  top = 25,
  genes = NULL,
  pvalue = c(0.05, 0.01),
  getMutexMethod = "ShiftedBinomial",
  getMutexMixed = TRUE,
  returnAll = TRUE,
  geneOrder = NULL,
  fontSize = 0.8,
  showSigSymbols = TRUE,
  showCounts = FALSE,
  countStats = "all",
  countType = "all",
  countsFontSize = 0.8,
  countsFontColor = "black",
```

```
  colPal = "BrBG",
  showSum = TRUE,
  colNC = 9,
  nShiftSymbols = 5,
  sigSymbolsSize = 2,
  sigSymbolsFontSize = 0.9,
  pvSymbols = c(46, 42),
  limitColorBreaks = TRUE
)
```

Arguments

maf	maf object generated by read.maf
top	check for interactions among top 'n' number of genes. Defaults to top 25. genes
genes	List of genes among which interactions should be tested. If not provided, test will be performed between top 25 genes.
pvalue	Default c(0.05, 0.01) p-value threshold. You can provide two values for upper and lower threshold.
getMutexMethod	Method for the 'getMutex' function (by default "ShiftedBinomial")
getMutexMixed	Mixed parameter for the 'getMutex' function (by default TRUE)
returnAll	If TRUE returns test statistics for all pair of tested genes. Default FALSE, returns for only genes below pvalue threshold.
geneOrder	Plot the results in given order. Default NULL.
fontSize	cex for gene names. Default 0.8
showSigSymbols	Default TRUE. Heighlight significant pairs
showCounts	Default FALSE. Include number of events in the plot
countStats	Default 'all'. Can be 'all' or 'sig'
countType	Default 'all'. Can be 'all', 'cooccur', 'mutexcl'
countsFontSize	Default 0.8
countsFontColor	Default 'black'
colPal	colPalBrewer palettes. Default 'BrBG'. See RColorBrewer::display.brewer.all() for details
showSum	show [sum] with gene names in plot, Default TRUE
colNC	Number of different colors in the palette, minimum 3, default 9
nShiftSymbols	shift if positive shift SigSymbols by n to the left, default 5
sigSymbolsSize	size of symbols in the matrix and in legend. Default 2
sigSymbolsFontSize	size of font in legends. Default 0.9
pvSymbols	vector of pch numbers for symbols of p-value for upper and lower thresholds c(upper, lower). Default c(46, 42)
limitColorBreaks	limit color to extreme values. Default TRUE

Details

Internally, this function run the getMutex function. With the 'getMutexMethod' parameter user might select the 'method' parameter of the getMutex function. For more details run '?getMutex'

#' @return A list of data.tables and it will print a heatmap with the results.

References

Mayakonda A, Lin DC, Assenov Y, Plass C, Koeffler HP. 2018. Maftools: efficient and comprehensive analysis of somatic variants in cancer. Genome Research. http://dx.doi.org/10.1101/gr.239244.118

Examples

```
## Not run:

  #An example of how to perform the function,
  #using data from TCGA, Colon Adenocarcinoma in this case.

  #coad.maf <- GDCquery_Maf("COAD", pipelines = "muse") %>% read.maf
  coad.maf <- read.maf(GDCquery_Maf("COAD", pipelines = "muse"))
  discoversomaticInteractions(maf = coad.maf, top = 35, pvalue = c(1e-2, 2e-3))

## End(Not run)
```

---

getMutex *getMutex function*

Description

Given a binary matrix and its corresponding probability matrix pij, compute the Poisson Binomial method to estimate mutual exclusive events.

Usage

```
getMutex(
  A = NULL,
  PM = getPM(A),
  lower.tail = TRUE,
  method = "ShiftedBinomial",
  mixed = TRUE,
  th = 0.05,
  verbose = FALSE,
  parallel = FALSE,
  no_cores = NULL
)
```

Arguments

A	The binary matrix
PM	The corresponding probability matrix of A. It can be computed using function getPM. By default equal to getPM(A)
lower.tail	True if mutually exclusive test. False for co-ocurrence. By default is TRUE.
method	one of the following: "ShiftedBinomial" (default),"Exact", "Binomial", and "RefinedNormal".
mixed	option to compute lower p-values with an exact method. By default TRUE
th	upper threshold of p.value to apply the exact method.
verbose	The verbosity of the output
parallel	If the exact method is executed with a parallel process.
no_cores	number of cores. If not stated number of cores of the CPU - 1

Details

we implemented three different approximations of the Poison-Binomial distribution function:

- "ShiftedBinomial" (by default) that correspond to a shifted Binomial with three parameters (Peköz, Shwartz, Christiansen, & Berlowitz, 2010).
- "Exact" that use the exact formula using the ‘PoissonBinomial‘ Rpackage based on the work from (Biscarri, Zhao, & Brunner, 2018).
- "Binomial" with two parameters (Cam, 1960).
- "RefinedNormal" that is based on the work from (Volkova, 1996).

If ‘mixed‘ option is selected (by default is FALSE), the "Exact" method is computed for p-values lower than a threshold (‘th‘ parameter, that by default is 0.05). When the exact method is computed, it is possible to parallelize the process by selecting the option ‘parallel‘ (by default FALSE) and setting the number of cores (‘no_cores‘ parameter)

Value

A symmetric matrix with the p-values of the corresponding test.

Examples

```
#This first example is a basic
#example of how to perform getMutex.

data("A_example")
PMA <- getPM(A_example)
mismutex <- getMutex(A=A_example,PM=PMA)


#The next example, is the same as the first one but,
# using a matrix of class Matrix.

data("A_Matrix")
```

```
  A_Matrix <- A_Matrix[1:100,1:50]
  #small for the example
  PMA_Matrix <- getPM(A_Matrix)
  mismutex <- getMutex(A=A_Matrix,PM=PMA_Matrix)


  ## Not run:
  #Finally, the last example, shows a real
  #example of how to perform this function when using
  #data from TCGA, Colon Adenocarcinoma in this case.

  data("TCGA_COAD")
  data("PM_COAD")

  PM_COAD <- getMutex(TCGA_COAD, PM_COAD)

## End(Not run)
```

---

getMutexAB *getMutexAB function*

Description

Given two binary matrices and its corresponding probability matrices PAij and PBij, compute the Poisson Binomial method to estimate mutual exclusive events between A and B

Usage

```
getMutexAB(
  A,
  PMA = getPM(A),
  B,
  PMB = getPM(B),
  lower.tail = TRUE,
  method = "ShiftedBinomial",
  mixed = TRUE,
  th = 0.05,
  verbose = FALSE,
  parallel = FALSE,
  no_cores = NULL
)
```

Arguments

A	The binary matrix of events A
PMA	The corresponding probability matrix of A. It can be computed using function getPM. By default equal to getPM(A)

B	The binary matrix of events B
PMB	The corresponding probability matrix of B. It can be computed using function getPM. By default equal to getPM(B)
lower.tail	True if mutually exclusive test. False for co-ocurrence. By default is TRUE.
method	one of the following: "ShiftedBinomial" (default),"Exact", "RefinedNormal", and "Binomial".
mixed	option to compute lower p-values with an exact method. By default TRUE
th	upper threshold of p-value to apply the exact method.
verbose	The verbosity of the output
parallel	If the exact method is executed with a parallel process.
no_cores	number of cores. If not stated number of cores of the CPU - 1

Details

we implemented three different approximations of the Poison-Binomial distribution function:

- "ShiftedBinomial" (by default) that correspond to a shifted Binomial with three parameters (Peköz, Shwartz, Christiansen, & Berlowitz, 2010).
- "Exact" that use the exact formula using the ‘PoissonBinomial‘ Rpackage based on the work from (Biscarri, Zhao, & Brunner, 2018).
- "Binomial" with two parameters (Cam, 1960).
- "RefinedNormal" that is based on the work from (Volkova, 1996).

If ‘mixed‘ option is selected (by default is FALSE), the "Exact" method is computed for p-values lower than a threshold (‘th‘ parameter, that by default is 0.05). When the exact method is computed, it is possible to parallelize the process by selecting the option ‘parallel‘ (by default FALSE) and setting the number of cores (‘no_cores‘ parameter)

Value

A matrix with the p-values of the corresponding test.

Examples

```
#The next example, is the same as the first
# one but, using a matrix of class Matrix.

data("A_Matrix")
data("B_Matrix")
PMA <- getPM(A_Matrix)
PMB <- getPM(B_Matrix)
mismutex <- getMutexAB(A=A_Matrix, PM=PMA, B=B_Matrix, PMB = PMB)

#Finally, the last example, shows a
#real example of how to perform this function
# when using data from TCGA, Colon Adenocarcinoma in this case.
```

```
## Not run:
data("TCGA_COAD_AMP")
data("AMP_COAD")
data("PM_TCGA_COAD_AMP")
data("PM_AMP_COAD")

mismutex <- getMutexAB(A=TCGA_COAD_AMP,
                       PMA=PM_TCGA_COAD_AMP,
                       B=AMP_COAD,
                       PMB = PM_AMP_COAD)

## End(Not run)
```

---

getMutexGroup *getMutexGroup function*

Description

Given a binary matrix and its corresponding probability matrix pij, compute the Poisson Binomial method to estimate mutual exclusive events.

Usage

```
getMutexGroup(A = NULL, PM = NULL, type = "Impurity", lower.tail = TRUE)
```

Arguments

A	The binary matrix
PM	The corresponding probability matrix of A. It can be computed using function getPM. By default equal to getPM(A)
type	one of Coverage, Exclusivity or Impurity. By default is Impurity
lower.tail	True if mutually exclusive test. False for co-ocurrence. By default is TRUE.

Value

A symmetric matrix with the p.value of the corresponding test.

Examples

```
#This first example is a basic
#example of how to perform getMutexGroup

data("A_example")
A2 <- A_example[,1:30]
A2[1,1:10] <- 1
```

```
A2[2,1:10] <- 0
A2[3,1:10] <- 0
A2[1,11:20] <- 0
A2[2,11:20] <- 1
A2[3,11:20] <- 0
A2[1,21:30] <- 0
A2[2,21:30] <- 0
A2[3,21:30] <- 1
PM2 <- getPM(A2)
A <- A2[1:3,]
PM <- PM2[1:3,]

getMutexGroup(A, PM, "Impurity")
getMutexGroup(A, PM, "Coverage")
getMutexGroup(A, PM, "Exclusivity")
```

---

getPM *getPM function*

Description

Given a binary matrix estimates the corresponding probability matrix pij.

Usage

```
getPM(A)
```

Arguments

A	The binary matrix

Value

A 'PMatrix' object with the corresponding probability estimations. This 'PMatrix' object stored the corresponding coefficients of the logistic regression computed. With this coefficients it is possible to build the complete matrix of probabilities.

Examples

```
#This first example is a basic example of how to perform getPM:

data("A_example")
PMA <- getPM(A_example)

#The next example, is the same as the first one but,
#using a matrix of class Matrix:
```

```
  data("A_Matrix")
  PMA_Matrix <- getPM(A_Matrix)

  ## Not run:
  #Finally, the last example, shows a real example
  #of how to perform this function when when using
  #data from TCGA, Colon Adenocarcinoma in this case:
  data("TCGA_COAD")
  PM_COAD <- getPM(TCGA_COAD)

## End(Not run)
```

`PMatrix-class` *An S4 class to store the probabilities*

Description

An S4 class to store the probabilities of gene i being mutated in sample j

Slots

`rowExps` Sample depending estimated coefficients obtained from the logistic regression

`colExps` gene depending estimated coefficients obtained from the logistic regression

---

`PM_AMP_COAD` *PM_AMP_COAD data*

Description

Probability matrix, with information of genes being amplified in samples in Colon Adenocarcinoma, created by AMP_COAD.rda applying `getPM` and used as real example and `getMutexAB`.

Usage

```
data("PM_AMP_COAD")
```

Format

The format is:

num [1:1000, 1:391] 0.118 0.118 0.118 0.118 0.114 ...

Examples

```
data(PM_AMP_COAD)
```

PM_COAD *PM_COAD data*

Description

Probability matrix, with information of genes being mutated in samples in Colon Adenocarcinoma, created by TCGA_COAD.rda applying getPM and used as real example in getMutex and getMutexAB and getMutexGroup.

Usage

```
data("PM_COAD")
```

Format

The format is:

Formal class 'PMatrix' [package "Rediscover"] with 2 slots

..@ rowExps: num [1:399] 13.1 1.02 7.43 3.26 0.4 ...

..@ colExps: num [1:17616] 2.54 1.78 1.76 1.35 0.6 ...

Examples

```
data(PM_COAD)
```

TCGA_COAD *TCGA_COAD*

Description

A binary matrix, with information about genes mutations in Colon Adenocarcinoma, created by applying maftools to .maf file and used as real example in getPM and getMutex and getMutexAB and getMutexGroup.

Usage

```
data("TCGA_COAD")
```

Format

The format is:

num [1:399, 1:17616] 1 1 1 1 1 1 1 1 1 1 ...

- attr(*, "dimnames")=List of 2

..$: chr [1:399] "TCGA-CA-6718" "TCGA-D5-6931" "TCGA-AZ-6601" "TCGA-G4-6320" ...

..$: chr [1:17616] "APC" "TP53" "TTN" "KRAS" ...

Examples

```
data(TCGA_COAD)
```

Index

∗ **datasets**
A_example, 3
A_Matrix, 3
AMP_COAD, 2
B_example, 4
B_Matrix, 4
PM_AMP_COAD, 13
PM_COAD, 14
TCGA_COAD, 14
[,PMatrix,matrix,missing,ANY-method *(*PMatrix-class*)*, 13
[,PMatrix,missing,numeric,ANY-method *(*PMatrix-class*)*, 13
[,PMatrix,numeric,missing,ANY-method *(*PMatrix-class*)*, 13
[,PMatrix,numeric,numeric,ANY-method *(*PMatrix-class*)*, 13

A_example, 3
A_Matrix, 3
AMP_COAD, 2

B_example, 4
B_Matrix, 4

dim,PMatrix-method *(*PMatrix-class*)*, 13
discoversomaticInteractions, 5

getMutex, 7
getMutexAB, 9
getMutexGroup, 11
getPM, 12

PM_AMP_COAD, 13
PM_COAD, 14
PMatrix *(*PMatrix-class*)*, 13
PMatrix-class, 13

TCGA_COAD, 14