Package ‘DMRScan’

January 31, 2024

Title Detection of Differentially Methylated Regions

Version 1.24.0

Description This package detects significant differentially methylated regions (for both qualitative and quantitative traits), using a scan statistic with underlying Poisson heuristics. The scan statistic will depend on a sequence of window sizes (# of CpGs within each window) and on a threshold for each window size. This threshold can be calculated by three different means: i) analytically using Siegmund et.al (2012) solution (preferred), ii) an important sampling as suggested by Zhang (2008), and a iii) full MCMC modeling of the data, choosing between a number of different options for modeling the dependency between each CpG.

biocViews Software, Technology, Sequencing, WholeGenome

Depends R (>= 3.6.0)

Imports Matrix, MASS, ReppRoll, GenomicRanges, IRanges, GenomeInfoDb, methods, mvtnorm, stats, parallel

License GPL-3

LazyData true

Author Christian M Page [aut, cre],
Linda Vos [aut],
Trine B Rounge [ctb, dtc],
Hanne F Harbo [ths],
Bettina K Andreassen [aut]

Maintainer Christian M Page <page.ntnu@gmail.com>

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, BiocStyle, BiocManager

VignetteBuilder knitr

URL https://github.com/christpa/DMRScan

BugReports https://github.com/christpa/DMRScan/issues

PackageStatus Active

git_url https://git.bioconductor.org/packages/DMRScan
R topics documented:

- dmrscan .................................................. 2
- DMRScan.methylationData ............................... 3
- DMRScan.phenotypes ..................................... 4
- DMRScan_package ........................................ 4
- estimateThreshold ....................................... 6
- makeCpGgenes ............................................ 7
- makeCpGregions .......................................... 8
- manyWindowSizeScanner ................................. 9
- oneWindowSizeScanner .................................. 9

Index 11

dmrscan

DMR Scan function

Description

DMR Scan function

Usage

dmrscan(observations, windowSize, windowThreshold = NULL, chr = NULL, pos = NULL, maxGap = 500, ...)

Arguments

- observations: An object of either: GRangesList made by makeCpGregions, a vector of the test statistic, a GRanges object, or a "minfi" object (soon to be supported).
- windowSize: A sequence of windowSizes for the slidingWindow. Must be an integer vector, with equal length as the number of windows.
- windowThreshold: Optional argument with corresponding cut-off for each window. Will be estimated if not supplied.
- chr: A vector of chromosomal position. Only used when the observations vector is a matrix of test statistic.
- pos: A vector of genomic coordinates for the CpGs to match the chr argument
- maxGap: The maximum allowed gap between two CpGs within the same region.
- ... : Optional arguments to be passed to estimateThreshold, if no grid is specified.
Value

An object of type \texttt{GRanges} with significantly differentially

Examples

```r
## methylation data from chromosome 22
data(DMRScan.methylationData)
## phenotype (end-point for methylation data)
data(DMRScan.phenotypes)

## Test for an association between phenotype and methylation
test.statistics <- apply(DMRScan.methylationData, 1, function(x, y)
  summary(glm(y ~ x, family = binomial(link = "logit")))$coefficients[2, 3],
  y = DMRScan.phenotypes)

## Set chromosomal position to each test-statistic
positions <- data.frame(matrix(as.integer(unlist(strsplit(names(test.statistics),
  split="chr|[.]")), ncol = 3, byrow = TRUE))[, -1]

## Set clustering features
min.cpg <- 4  ## Minimum number of CpGs in a tested cluster
max.gap <- 750  ## Maximum distance (in base-pairs) within a cluster
## before it is broken up into two separate cluster

## Identify all clusters, and generate a list for each cluster
regions <- makeCpGregions(observations = test.statistics,
  chr = positions[, 1], pos = positions[, 2],
  maxGaps = max.gap, minCpGs = min.cpg)

## Number of CpGs in the slidingWindows, can be either a single number
## or a sequence of windowSizes
windowSizes <- 3:7
nCpGs <- sum(sapply(regions, length))  ## Number of CpGs to be tested

## Estimate the windowThreshold, based on the number of CpGs and windowSizes
windowThresholds <- estimateWindowThreshold(nProbes = nCpGs,
  windowSize = windowSizes, method = "sampling", mcmc = 10000)

## Run the slidingWindow
DMRScanResults <- dmrscan(observations = regions,
  windowSize = windowSizes,
  windowThreshold = windowThresholds)

## Print the result
print(DMRScanResults)
```
Description

Bi-sulfite sequencing data from all known CpG islands at chromosome 22 from 100 the Finish teens study, sampled from extreme BMI quantiles. The data set is reduced to 25139 sites on chromosome 22. See "Genome-wide DNA methylation in saliva and body size of adolescent girls", TB Rounge, CM Page, M Lepisto, E Pekka, and BK Andreassen and E Weiderpass, _Epigenomics_ 8.11 (2016): 1495-1505 for a full overview of the data set.

Examples

```r
data(DMRScan.methylationData)
head(DMRScan.methylationData)
```

Description


Examples

```r
data(DMRScan.phenotypes)
table(DMRScan.phenotypes)
```

Description

DMRScan: An R-package for identification of Differentially Metylated Regions

Arguments

- `observations`: An object of type GRangesList from makeCpGregions
- `windowSize`: A sequence of windowSizes for the slidingWindow, must be an integer
- `windowThreshold`: Optional argument with corresponding cut-off for each window. Will be estimated if not supplied.
- `...`: Optional arguments to be passed to `estimateThreshold`, if no grid is specified.
Value

An object of type `GRanges` with significantly differentially

Author(s)

Christian Page, <page.ntnu@gmail.com>

References

Not Published yet (Under revision)

Examples

```r
## nProbeoad methylation data from chromosome 22
data(DMRScan.methylationData)
## nProbeoad phenotype (end-point for methylation data)
data(DMRScan.phenotypes)

## Test for an association between phenotype and Methylation
test.statistics <- apply(DMRScan.methylationData,1,function(x,y)
  summary(glm(y ~ x, family = binomial(link = "logit")))$coefficients[2,3],
  y = DMRScan.phenotypes)

## Set chromosomal position to each test-statistic
positions <- data.frame(matrix(as.integer(unlist(strsplit(names(test.statistics), split="chr|\[\].\)"))), ncol = 3, byrow = TRUE))

## Set clustering features
min.cpg <- 4 ## Minimum number of CpGs in a tested cluster
max.gap <- 750

## Identify all clusters, and generate a list for each cluster
regions <- makeCpGregions(observations = test.statistics,
  chr = positions[,1], pos = positions[,2],
  maxGap = max.gap, minCpG = min.cpg)

## Number of CpGs in the slidingWindows, can be either a single number
## or a sequence of windowSizes
windowSizes <- 3:7
nCpG <- sum(sapply(regions, length)) ## Number of CpGs to be tested

# Estimate the windowThreshold, based on the number of CpGs and windowSizes
windowThresholds <- estimateWindowThreshold(nProbe = nCpG,
  windowSize = windowSizes, method = "sampling", mcmc = 10000)

## Run the slidingWindow
DMRScanResults <- dmrscan(observations = regions,
  windowSize = windowSizes,
  windowThreshold = windowThresholds)

## Print the result
print(DMRScanResults)
```
estimateThreshold  EstimateWindowThresholds

Description

Estimate window thresholds for sliding window, one unique value for each window size

Usage

```r
estimateWindowThreshold(nProbe, windowSize, method = "siegmund",
                        mcmc = 1000, nCPU = 1, submethod = "ar", ...)
```

Arguments

- `nProbe` The number of probes (CpGs) in the study.
- `windowSize` The different window sizes to be tested. Must be either one, or an ordered sequence of integers.
- `method` Gives the method by which the threshold is calculated. Can be either an analytical solution "siegmund", provided by Siegumud et.al (2012), or an iterative process; either importance sampling "sampling", as suggested by Zhang (2012) or a full MCMC model "mcmc" which can account for any dependency structure, which is passed to arima.sim, with ...
- `mcmc` The number of MCMC iterations to be used, when using either Important Sampling ("zhang") or MCMC estimation of the threshold.
- `nCPU` When calculating the thresholds on a cluster, how many CPUs should be used. This option is only compatible with the 'mcmc' method.
- `submethod` A character string indicating if an AR(5) or ARIMA model should be used. In the AR(5), the index runs from -2 to 2. A regular AR(p) model can be obtained using ARIMA(p,0,0) instead.
- `...` Optional parameters passed on to `arima`, when simulating data using the mcmc option, see arima.sim()

Value

Returns a vector of the threshold for each window size

Examples

```r
thresholdGrid <- estimateWindowThreshold(nProbe = 1000,
                                          windowSize = 3:8, method = "siegmund")
```
makeCpGgenes

Cluster

Description
Cluster CpGs together in genes based on annotation

Usage
makeCpGgenes(observations, chr, pos, gene, minCpG = 2)

Arguments
observations   Vector of corresponding observed T-value for each CpG, must be ordered in the same way as chr and pos
chr            Vector of chromosome location for each CpG
pos            Vector giving base pair position for each CpG If unsorted, use order(chr,pos) to sort the genomic positions within each chromosome.
gene           A vector assigning each probe to a gene.
minCpG         Minimum number of CpGs allowed in each region to be considered. Default is set to at least 2 CpGs within each region.

Value
The supplied observations ordered into into a list, with one entry for each CpG region.

Examples

```
data(DMRScan.methylationData) ## Load methylation data from chromosome 22
data(DMRScan.phenotypes) ## Load phenotype (end-point for methylation data)

## Test for an association between phenotype and Methylation
testStatistics <- apply(DMRScan.methylationData,1,function(x,y)
  summary(glm(y ~ x, family = binomial(link = "logit"))$coefficients[2,3],
  y = DMRScan.phenotypes)

## Set chromosomal position to each test-statistic
pos <- data.frame(matrix(as.integer(unlist(strsplit(names(testStatistics),
  split="chr[.]"))), ncol = 3, byrow = TRUE))[,-1]

## Set clustering features
minCpG <- 3 ## Minimum number of CpGs in a tested cluster
gene <- sample(paste("Gene",1:100,sep=""),
  length(testStatistics),replace=TRUE)
regions <- makeCpGgenes(observations = testStatistics,
  chr = pos[,1], pos = pos[,2],
  gene = gene, minCpG = minCpG)
```
**Description**

Cluster CpGs together in regions based on proximity.

**Usage**

```r
makeCpGregions(observations, chr, pos, maxGap = 500, minCpG = 2)
```

**Arguments**

- `observations`: Vector of corresponding observed T-value for each CpG, must be ordered in the same way as `chr` and `pos`.
- `chr`: Vector of chromosome location for each CpG.
- `pos`: Vector giving base pair position for each CpG. If unsorted, use `order(chr,pos)` to sort the genomic positions within each chromosome.
- `maxGap`: Maximum allowed base pair gap within a cluster. Default is set to 500.
- `minCpG`: Minimum number of CpGs allowed in each region to be considered. Default is set to at least 2 CpGs within each region.

**Value**

The supplied observations ordered into a GRangesList object. To be parsed further into dmrscan.

**Examples**

```r
data(DMRScan.methylationData) # Load methylation data from chromosome 22
data(DMRScan.phenotypes) # Load phenotype (end-point for methylation data)

# Test for an association between phenotype and Methylation
testStatistics <- apply(DMRScan.methylationData,1,function(x,y)
    summary(glm(y ~ x, family = binomial(link = "logit"))$coefficients[2,3],
    y = DMRScan.phenotypes)

# Set chromosomal position to each test-statistic
pos<- data.frame(matrix(as.integer(unlist(strsplit(names(testStatistics),
    split="chr[.]"))), ncol = 3, byrow = TRUE))[,,-1]

# Set clustering features
minCpG <- 3 # Minimum number of CpGs in a tested cluster
# Maximum distance (in base-pairs) within a cluster before it is
# broken up into two separate cluster
maxGap <- 750
regions <- makeCpGregions(observations = testStatistics, chr = pos[,1],
    pos = pos[,2], maxGap = maxGap, minCpG = minCpG)
```
manyWindowSizeScanner

Method Fixed window size scan for a sequence of window sizes

Description

Method Fixed window size scan for a sequence of window sizes

Usage

manyWindowSizeScanner(region, windowThreshold, windowSize)

## S4 method for signature 'GRangesList'
manyWindowSizeScanner(region, windowThreshold, windowSize)

## S4 method for signature 'GRanges'
manyWindowSizeScanner(region, windowThreshold, windowSize)

Arguments

region Object of type GRanges
windowThreshold Vector of window thresholds
windowSize Vector of window sizes to be tested on regions

Value

A list of the windows that are significant

Examples

## Not run

oneWindowSizeScanner

Method Fixed window size scan for one window size

Description

Method Fixed window size scan for one window size
Usage

oneWindowSizeScanner(region, windowThreshold, windowSize)

## S4 method for signature 'GRangesList'
oneWindowSizeScanner(region, windowThreshold, windowSize)

## S4 method for signature 'GRanges'
oneWindowSizeScanner(region, windowThreshold, windowSize)

Arguments

region Object of type GRanges
windowThreshold Vector of window thresholds
windowSize Vector of window sizes to be tested on regions

Value

A list of the windows that are significant

Examples

## Not run
Index

* CpG
  - makeCpGgenes, 7
  - makeCpGregions, 8

* DMR,
  - DMRScan_package, 4

* DMRScan
  - dmrscan, 2
  - DMRScan_package, 4

* Regions
  - makeCpGgenes, 7
  - makeCpGregions, 8

* datasets
  - DMRScan.methylationData, 3

* dataset
  - DMRScan.phenotypes, 4

arima, 6

dmrscan, 2, 8
DMRScan.methylationData, 3
DMRScan.phenotypes, 4
DMRScan_package, 4
DMRScan_package-package
  - (DMRScan_package), 4

estimateThreshold, 2, 4, 6
estimateWindowThreshold
  - (estimateThreshold), 6

GRanges, 2, 3, 5
GRangesList, 2

makeCpGgenes, 7
makeCpGregions, 2, 8

manyWindowSizeScanner, 9

manyWindowSizeScanner, DMRScan-package
  - (manyWindowSizeScanner), 9

manyWindowSizeScanner, DMRScan-list
  - (manyWindowSizeScanner), 9

oneWindowSizeScanner, DMRScan-package
  - (oneWindowSizeScanner), 9

oneWindowSizeScanner, DMRScan-list
  - (oneWindowSizeScanner), 9

Rt (oneWindowSizeScanner), 9
Rt, GRanges-method
  - (oneWindowSizeScanner), 9
Rt, GRangesList-method
  - (oneWindowSizeScanner), 9

St (manyWindowSizeScanner), 9