Package ‘DMRcaller’

Type Package

Title Differentially Methylated Regions caller

Version 1.36.0

Date 2021-11-17

Author Nicolae Radu Zabet <n.r.zabet@gen.cam.ac.uk>, Jonathan Michael Foonlan Tsang <jmftz@cam.ac.uk>, Alessandro Pio Greco <apgrec@essex.ac.uk> and Ryan Merritt <rmerri@essex.ac.uk>

Maintainer Nicolae Radu Zabet <n.r.zabet@gen.cam.ac.uk>

Description Uses Bisulfite sequencing data in two conditions and identifies differentially methylated regions between the conditions in CG and non-CG context. The input is the CX report files produced by Bismark and the output is a list of DMRs stored as GRanges objects.

License GPL-3

LazyLoad yes

Imports parallel, Rcpp, RcppRoll, betareg, grDevices, graphics, methods, stats, utils

Depends R (>= 3.5), GenomicRanges, IRanges, S4Vectors (>= 0.23.10)

Suggests knitr, RUnit, BiocGenerics

biocViews DifferentialMethylation, DNAMethylation, Software, Sequencing, Coverage

VignetteBuilder knitr

NeedsCompilation no

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

git_branch RELEASE_3_19

git_last_commit 0b1facf

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-29
analyseReadsInsideRegionsForCondition

Analyse reads inside regions for condition

Description

This function extracts from the methylation data the total number of reads, the number of methylated reads and the number of cytosines in the specific context from a region (e.g. DMRs)

Usage

analyseReadsInsideRegionsForCondition(regions, methylationData, context, label = "", cores = 1)
Arguments

- **regions** a GRanges object with a list of regions on the genome; e.g. could be a list of DMRs
- **methylationData** the methylation data in one condition (see methylationDataList).
- **context** the context in which to extract the reads ("CG", "CHG" or "CHH").
- **label** a string to be added to the columns to identify the condition
- **cores** the number of cores used to compute the DMRs.

Value

- a GRanges object with additional four metadata columns
  - **sumReadsM** the number of methylated reads
  - **sumReadsN** the total number of reads
  - **proportion** the proportion methylated reads
  - **cytosinesCount** the number of cytosines in the regions

Author(s)

Nicolae Radu Zabet

See Also

filterDMRs, computeDMRs, DMRsNoiseFilterCG, and mergeDMRsIteratively

Examples

```r
# load the methylation data
data(methylationDataList)

# load the DMRs in CG context. These DMRs were computed with minGap = 200.
data(DMRsNoiseFilterCG)

# retrieve the number of reads in CHH context in WT
DMRsNoiseFilterCGreadsCHH <- analyseReadsInsideRegionsForCondition(
  DMRsNoiseFilterCG[1:10],
  methylationDataList[["WT"]], context = "CHH",
  label = "WT")
```
computeDMRs

**Description**

This function computes the differentially methylated regions between two conditions.

**Usage**

```r
computeDMRs(methylationData1, methylationData2, regions = NULL, context = "CG", method = "noise_filter", windowSize = 100, kernelFunction = "triangular", lambda = 0.5, binSize = 100, test = "fisher", pValueThreshold = 0.01, minCytosinesCount = 4, minProportionDifference = 0.4, minGap = 200, minSize = 50, minReadsPerCytosine = 4, cores = 1)
```

**Arguments**

- `methylationData1` the methylation data in condition 1 (see `methylationDataList`).
- `methylationData2` the methylation data in condition 2 (see `methylationDataList`).
- `regions` a `GRanges` object with the regions where to compute the DMRs. If `NULL`, the DMRs are computed genome-wide.
- `context` the context in which the DMRs are computed ("CG", "CHG" or "CHH").
- `method` the method used to compute the DMRs ("noise_filter", "neighbourhood" or "bins"). The "noise_filter" method uses a triangular kernel to smooth the number of reads and then performs a statistical test to determine which regions display different levels of methylation in the two conditions. The "neighbourhood" method computes differentially methylated cytosines. Finally, the "bins" method partitions the genome into equal sized tiling bins and performs the statistical test between the two conditions in each bin. For all three methods, the cytosines or bins are then merged into DMRs without affecting the initial parameters used when calling the differentially methylated cytosines/bins (p-value, difference in methylation levels, minimum number of reads per cytosine).
- `windowSize` the size of the triangle base measured in nucleotides. This parameter is required only if the selected method is "noise_filter".
- `kernelFunction` a character indicating which kernel function to be used. Can be one of "uniform", "triangular", "gaussian" or "epanechnicov". This is required only if the selected method is "noise_filter".
- `lambda` numeric value required for the Gaussian filter ($K(x) = \exp(-\lambda x^2)$). This is required only if the selected method is "noise_filter" and the selected kernel function is "gaussian".
- `binSize` the size of the tiling bins in nucleotides. This parameter is required only if the selected method is "bins".
computeDMRs

test  the statistical test used to call DMRs ("fisher" for Fisher's exact test or "score" for Score test).

pValueThreshold  DMRs with p-values (when performing the statistical test; see test) higher or equal than pValueThreshold are discarded. Note that we adjust the p-values using the Benjamini and Hochberg’s method to control the false discovery rate.

minCytosinesCount  DMRs with less cytosines in the specified context than minCytosinesCount will be discarded.

minProportionDifference  DMRs where the difference in methylation proportion between the two conditions is lower than minProportionDifference are discarded.

minGap  DMRs separated by a gap of at least minGap are not merged. Note that only DMRs where the change in methylation is in the same direction are joined.

minSize  DMRs with a size smaller than minSize are discarded.

minReadsPerCytosine  DMRs with the average number of reads lower than minReadsPerCytosine are discarded.

cores  the number of cores used to compute the DMRs.

Value

the DMRs stored as a GRanges object with the following metadata columns:

direction  a number indicating whether the region lost (-1) or gain (+1) methylation in condition 2 compared to condition 1.

context  the context in which the DMRs was computed ("CG", "CHG" or "CHH").

sumReadsM1  the number of methylated reads in condition 1.

sumReadsN1  the total number of reads in condition 1.

proportion1  the proportion methylated reads in condition 1.

sumReadsM2  the number of methylated reads in condition 2.

sumReadsN2  the total number reads in condition 2.

proportion2  the proportion methylated reads in condition 2.

cytosinesCount  the number of cytosines in the DMR.

regionType  a string indicating whether the region lost ("loss") or gained ("gain") methylation in condition 2 compared to condition 1.

pValue  the p-value (adjusted to control the false discovery rate with the Benjamini and Hochberg’s method) of the statistical test when the DMR was called.

Author(s)

Nicolae Radu Zabet and Jonathan Michael Foonlan Tsang

See Also

filterDMRs, mergeDMRsIteratively, analyseReadsInsideRegionsForCondition and DMRsNoiseFilterCG
Examples

```r
# load the methylation data
data(methylationDataList)

# the regions where to compute the DMRs
regions <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(1,1E5))

# compute the DMRs in CG context with noise_filter method
DMRsNoiseFilterCG <- computeDMRs(methylationDataList[['WT']],
methylationDataList[['met1-3']], regions = regions,
context = "CG", method = "noise_filter",
winSize = 100, kernelFunction = "triangular",
test = "score", pValueThreshold = 0.01,
minCytosinesCount = 4, minProportionDifference = 0.4,
minGap = 200, minSize = 50, minReadsPerCytosine = 4,
cores = 1)

# compute the DMRs in CG context with neighbourhood method
DMRsNeighbourhoodCG <- computeDMRs(methylationDataList[['WT']],
methylationDataList[['met1-3']], regions = regions,
context = "CG", method = "neighbourhood",
test = "score", pValueThreshold = 0.01,
minCytosinesCount = 4, minProportionDifference = 0.4,
minGap = 200, minSize = 50, minReadsPerCytosine = 4,
cores = 1)

# compute the DMRs in CG context with bins method
DMRsBinsCG <- computeDMRs(methylationDataList[['WT']],
methylationDataList[['met1-3']], regions = regions,
context = "CG", method = "bins", binSize = 100,
test = "score", pValueThreshold = 0.01, minCytosinesCount = 4,
minProportionDifference = 0.4, minGap = 200, minSize = 50,
minReadsPerCytosine = 4, cores = 1)

## End(Not run)
```

computeDMRsReplicates  Compute DMRs

Description

This function computes the differentially methylated regions between replicates with two conditions.

Usage

```r
computeDMRsReplicates(methylationData, condition = NULL, regions = NULL,
```
computeDMRsReplicates

context = "CG", method = "neighbourhood", binSize = 100,
test = "betareg", pseudocountM = 1, pseudocountN = 2,
pValueThreshold = 0.01, minCytosinesCount = 4,
minProportionDifference = 0.4, minGap = 200, minSize = 50,
minReadsPerCytosine = 4, cores = 1)

Arguments

methylationData
the methylation data containing all the conditions for all the replicates.

condition
a vector of strings indicating the conditions for each sample in methylationData. Two different values are allowed (for the two conditions).

regions
a GRanges object with the regions where to compute the DMRs. If NULL, the DMRs are computed genome-wide.

context
the context in which the DMRs are computed ("CG", "CHG" or "CHH").

method
the method used to compute the DMRs "neighbourhood" or "bins"). The "neighbourhood" method computes differentially methylated cytosines. Finally, the "bins" method partitions the genome into equal sized tilling bins and performs the statistical test between the two conditions in each bin. For all three methods, the cytosines or bins are then merged into DMRs without affecting the initial parameters used when calling the differentially methylated cytosines/bins (p-value, difference in methylation levels, minimum number of reads per cytosine).

binSize
the size of the tiling bins in nucleotides. This parameter is required only if the selected method is "bins".

test
the statistical test used to call DMRs ("betareg" for Beta regression).

pseudocountM
numerical value to be added to the methylated reads before modelling beta regression.

pseudocountN
numerical value to be added to the total reads before modelling beta regression.

pValueThreshold
DMRs with p-values (when performing the statistical test; see test) higher or equal than pValueThreshold are discarded. Note that we adjust the p-values using the Benjamini and Hochberg’s method to control the false discovery rate.

minCytosinesCount
DMRs with less cytosines in the specified context than minCytosinesCount will be discarded.

minProportionDifference
DMRs where the difference in methylation proportion between the two conditions is lower than minProportionDifference are discarded.

minGap
DMRs separated by a gap of at least minGap are not merged. Note that only DMRs where the change in methylation is in the same direction are joined.

minSize
DMRs with a size smaller than minSize are discarded.

minReadsPerCytosine
DMRs with the average number of reads lower than minReadsPerCytosine are discarded.

cores
the number of cores used to compute the DMRs.
computeDMRsReplicates

**Value**

the DMRs stored as a `GRanges` object with the following metadata columns:

- **direction** a number indicating whether the region lost (-1) or gain (+1) methylation in condition 2 compared to condition 1.
- **context** the context in which the DMRs was computed ("CG", "CHG" or "CHH").
- **sumReadsM1** the number of methylated reads in condition 1.
- **sumReadsN1** the total number of reads in condition 1.
- **proportion1** the proportion methylated reads in condition 1.
- **sumReadsM2** the number of methylated reads in condition 2.
- **sumReadsN2** the total number reads in condition 2.
- **proportion2** the proportion methylated reads in condition 2.
- **cytosinesCount** the number of cytosines in the DMR.
- **regionType** a string indicating whether the region lost ("loss") or gained ("gain") methylation in condition 2 compared to condition 1.
- **pValue** the p-value (adjusted to control the false discovery rate with the Benjamini and Hochberg’s method) of the statistical test when the DMR was called.

**Author(s)**

Alessandro Pio Greco and Nicolae Radu Zabet

**Examples**

```r
## Not run:
data("syntheticDataReplicates")

# compute the DMRs in CG context with neighbourhood method

# creating condition vector
c(condition <- c("a", "a", "b", "b")

# computing DMRs using the neighbourhood method
DMRsReplicatesNeighbourhood <- computeDMRsReplicates(methylationData = methylationData, condition = condition, regions = NULL, context = "CHH", method = "neighbourhood", test = "betareg", pseudocountM = 1, pseudocountN = 2, pValueThreshold = 0.01, minCytosinesCount = 4, minProportionDifference = 0.4, minGap = 200, minSize = 50, minReadsPerCytosine = 4,
```
computeMethylationDataCoverage

## End(Not run)

### computeMethylationDataCoverage

**Compute methylation data coverage**

#### Description

This function computes the coverage for bisulfite sequencing data. It returns a vector with the proportion (or raw count) of cytosines that have the number of reads higher or equal than a vector of specified thresholds.

#### Usage

```r
computeMethylationDataCoverage(methylationData, regions = NULL, context = "CG", breaks = NULL, proportion = TRUE)
```

#### Arguments

- `methylationData`:
  
  the methylation data stored as a `GRanges` object with four metadata columns (see `methylationDataList`).

- `regions`:
  
  a `GRanges` object with the regions where to compute the coverage. If `NULL`, the coverage is computed genome-wide.

- `context`:
  
  the context in which the DMRs are computed ("CG", "CHG" or "CHH").

- `breaks`:
  
  a numeric vector specifying the different values for the thresholds when computing the coverage.

- `proportion`:
  
  a logical value indicating whether to compute the proportion (TRUE) or raw counts (FALSE).

#### Value

- a vector with the proportion (or raw count) of cytosines that have the number of reads higher or equal than the threshold values specified in the `breaks` vector.

#### Author(s)

Nicolae Radu Zabet and Jonathan Michael Foonlan Tsang

#### See Also

`plotMethylationDataCoverage`, `methylationDataList`
Examples

```
# load the methylation data
data(methylationDataList)

# compute coverage in CG context
breaks <- c(1,5,10,15)
coverage_CG_wt <- computeMethylationDataCoverage(methylationDataList["WT"],
                                                context="CG", breaks=breaks)
```

computeMethylationDataSpatialCorrelation

*Compute methylation data spatial correlation*

Description

This function computes the correlation of the methylation levels as a function of the distances between the Cytosines. The function returns a vector with the correlation of methylation levels at distance equal to a vector of specified thresholds.

Usage

```
computeMethylationDataSpatialCorrelation(methylationData, regions = NULL,
                                         context = "CG", distances = NULL)
```

Arguments

- **methylationData**
  
  the methylation data stored as a GRanges object with four metadata columns (see `methylationDataList`).

- **regions**
  
  a GRanges object with the regions where to compute the correlation. If NULL, the correlation is computed genome-wide.

- **context**
  
  the context in which the correlation is computed ("CG", "CHG" or "CHH").

- **distances**
  
  a numeric vector specifying the different values for the distances when computing the correlation.

Value

a vector with the correlation of the methylation levels for Cytosines located at distances specified in the distances vector.

Author(s)

Nicolae Radu Zabet

See Also

plotMethylationDataSpatialCorrelation, methylationDataList
Examples

```r
# load the methylation data
data(methylationDataList)

# compute spatial correlation in CG context
distances <- c(1,5,10,15)
correlation_CG_wt <- computeMethylationDataSpatialCorrelation(methylationDataList[["WT"]],
                                                                context="CG", distances=distances)
```

**computeMethylationProfile**

*Compute methylation profile*

**Description**

This function computes the low resolution profiles for the bisulfite sequencing data.

**Usage**

```r
computeMethylationProfile(methylationData, region,
                          windowSize = floor(width(region)/500), context = "CG")
```

**Arguments**

- `methylationData` the methylation data stored as a `GRanges` object with four metadata columns (see `methylationDataList`).
- `region` a `GRanges` object with the regions where to compute the DMRs.
- `windowSize` a numeric value indicating the size of the window in which methylation is averaged.
- `context` the context in which the DMRs are computed ("CG", "CHG" or "CHH").

**Value**

A `GRanges` object with equal sized tiles of the `region`. The object consists of the following metadata

- `sumReadsM` the number of methylated reads.
- `sumReadsN` the total number of reads.
- `Proportion` the proportion of methylated reads.
- `context` the context ("CG", "CHG" or "CHH").

**Author(s)**

Nicolaie Radu Zabet and Jonathan Michael Foonlan Tsang
computeOverlapProfile

Description

This function computes the distribution of a subset of regions (GRanges object) over a large region (GRanges object)

Usage

computeOverlapProfile(subRegions, largeRegion,
    windowSize = floor(width(largeRegion)/500), binary = TRUE, cores = 1)

Arguments

- subRegions: a GRanges object with the sub regions; e.g. can be the DMRs.
- largeRegion: a GRanges object with the region where to compute the overlaps; e.g. a chromosome.
- windowSize: The largeRegion is partitioned into equal sized tiles of width windowSize.
- binary: a value indicating whether to count 1 for each overlap or to compute the width of the overlap.
- cores: the number of cores used to compute the DMRs.

Examples

# load the methylation data
data(methylationDataList)

# the region where to compute the profile
region <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(1,1E6))

# compute low resolution profile in 20 Kb windows
lowResProfileWTCHH <- computeMethylationProfile(methylationDataList["WT"],
    region, windowSize = 20000, context = "CHH")

## Not run:
# compute low resolution profile in 10 Kb windows
lowResProfileWTCG <- computeMethylationProfile(methylationDataList["WT"],
    region, windowSize = 10000, context = "CG")

lowResProfileMet13CG <- computeMethylationProfile(
    methylationDataList["met1-3"], region,
    windowSize = 10000, context = "CG")

## End(Not run)
Value

A `GRanges` object with equal sized tiles of the regions. The object has one metadata file `score` which represents: the number of subRegions overlapping with the tile, in the case of `binary = TRUE`, and the width of the subRegions overlapping with the tile, in the case of `binary = FALSE`.

Author(s)

Nicolae Radu Zabet

See Also

`plotOverlapProfile`, `filterDMRs`, `computeDMRs`, and `mergeDMRsIteratively`

Examples

```r
# load the methylation data
data(methylationDataList)

# load the DMRs in CG context
data(DMRsNoiseFilterCG)

# the coordinates of the area to be plotted
largeRegion <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(1,1E5))

# compute overlaps distribution
hotspots <- computeOverlapProfile(DMRsNoiseFilterCG, largeRegion,
                                  windowSize = 10000, binary = FALSE)
```

---

**DMRcaller**  
*Call Differentially Methylated Regions (DMRs) between two samples*

**Description**

Uses bisulfite sequencing data in two conditions and identifies differentially methylated regions between the conditions in CG and non-CG context. The input is the CX report files produced by Bismark and the output is a list of DMRs stored as GRanges objects.

**Details**

The most important functions in the **DMRcaller** are:

- `readBismark`: Reads the Bismark CX report files in a `GRanges` object.
- `readBismarkPool`: Reads multiple CX report files and pools them together.
- `saveBismark`: Saves the methylation data stored in a `GRanges` object into a Bismark CX report file.
- `poolMethylationDatasets`: Pools together multiple methylation datasets.
- `poolTwoMethylationDatasets`: Pools together two methylation datasets.
- `computeMethylationDataCoverage`: Computes the coverage for the bisulfite sequencing data.
plotMethylationDataCoverage Plots the coverage for the bisulfite sequencing data.
computeMethylationDataSpatialCorrelation Computes the correlation between methylation
levels as a function of the distances between the Cytosines.
plotMethylationDataSpatialCorrelation Plots the correlation of methylation levels for Cy-
tosines located at a certain distance apart.
computeMethylationProfile Computes the low resolution profiles for the bisulfite sequencing
data at certain locations.
plotMethylationProfile Plots the low resolution profiles for the bisulfite sequencing data at
certain locations.
plotMethylationProfileFromData Plots the low resolution profiles for the loaded bisulfite se-
quencing data.
computeDMRs Computes the differentially methylated regions between two conditions.
filterDMRs Filters a list of (potential) differentially methylated regions.
mergeDMRsIteratively Merge DMRs iteratively.
analyseReadsInsideRegionsForCondition Analyse reads inside regions for condition.
plotLocalMethylationProfile Plots the methylation profile at one locus for the bisulfite se-
quencing data.
computeOverlapProfile Computes the distribution of a set of subregions on a large region.
plotOverlapProfile Plots the distribution of a set of subregions on a large region.
getWholeChromosomes Computes the GRanges objects with each chromosome as an element from
the methylationData.
joinReplicates Merges two GRanges objects with single reads columns. It is necessary to start
the analysis of DMRs with biological replicates.
computeDMRsReplicates Computes the differentially methylated regions between two conditions
with multiple biological replicates.

Author(s)
Nicolae Radu Zabet <n.r.zabet@gen.cam.ac.uk>, Jonathan Michael Foonlan Tsang <jmft2@cam.ac.uk>
Alessandro Pio Greco <apgreco@essex.ac.uk>
Maintainer: Nicolae Radu Zabet <n.r.zabet@gen.cam.ac.uk>

See Also
See vignette("rd", package = "DMRcaller") for an overview of the package.

Examples
## Not run:
# load the methylation data
data(methylationDataList)

# plot the low resolution profile at 5 Kb resolution
par(mar=c(4, 4, 3, 1)+0.1)
plotMethylationProfileFromData(methylationDataList[["WT"]],
methylDataList[["met1-3"]],
conditionsNames=c("WT", "met1-3"),
windowSize = 5000, autoscale = TRUE,
context = c("CG", "CHG", "CHH"),
labels = LETTERS))

# compute low resolution profile in 10 Kb windows in CG context
lowResProfileWTCG <- computeMethylationProfile(methylDataList[["WT"]],
region, windowSize = 10000, context = "CG")

lowResProfileMet13CG <- computeMethylationProfile(
methylDataList[["met1-3"]], region,
windowSize = 10000, context = "CG")

lowResProfileCG <- GRangesList("WT" = lowResProfileWTCG,
"met1-3" = lowResProfileMet13CG)

# compute low resolution profile in 10 Kb windows in CHG context
lowResProfileWTCHG <- computeMethylationProfile(methylDataList[["WT"]],
region, windowSize = 10000, context = "CHG")

lowResProfileMet13CHG <- computeMethylationProfile(
methylDataList[["met1-3"]], region,
windowSize = 10000, context = "CHG")

lowResProfileCHG <- GRangesList("WT" = lowResProfileWTCHG,
"met1-3" = lowResProfileMet13CHG)

# plot the low resolution profile
par(mar=c(4, 4, 3, 1)+0.1)
par(mfrow=c(2,1))
plotMethylationProfile(lowResProfileCG, autoscale = FALSE,
labels = LETTERS[1],
title="CG methylation on Chromosome 3",
col=c("#D55E00","#E69F00"), pch = c(1,0),
lty = c(4,1))

plotMethylationProfile(lowResProfileCHG, autoscale = FALSE,
labels = LETTERS[2],
title="CHG methylation on Chromosome 3",
col=c("#0072B2","#56B4E9"), pch = c(16,2),
lty = c(3,2))

# plot the coverage in all three contexts
plotMethylationDataCoverage(methylDataList[["WT"]],
methylDataList[["met1-3"]],
breaks = 1:15, regions = NULL,
conditionsNames = c("WT","met1-3"),
context = c("CG","CHG","CHH"),
proportion = TRUE, labels = LETTERS, col = NULL,
pch = c(1,0,16,2,15,17), lty = c(4,1,3,2,6,5),
contextPerRow = FALSE)

# plot the correlation of methylation levels as a function of distance
plotMethylationDataSpatialCorrelation(methylationDataList[['WT']],
  distances = c(1,5,10,15), regions = NULL,
  conditionsNames = c("WT","met1-3"),
  context = c("CG"),
  labels = LETTERS, col = NULL,
  pch = c(1,0,16,2,15,17), lty = c(4,1,3,2,6,5),
  contextPerRow = FALSE)

# the regions where to compute the DMRs
regions <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(1,1E6))

# compute the DMRs in CG context with noise_filter method
DMRsNoiseFilterCG <- computeDMRs(methylationDataList[['WT']],
  methylationDataList[['met1-3']], regions = regions,
  context = "CG", method = "noise_filter",
  windowSize = 100, kernelFunction = "triangular",
  test = "score", pValueThreshold = 0.01,
  minCytosinesCount = 4, minProportionDifference = 0.4,
  minGap = 200, minSize = 50, minReadsPerCytosine = 4,
  cores = 1)

# compute the DMRs in CG context with neighbourhood method
DMRsNeighbourhoodCG <- computeDMRs(methylationDataList[['WT']],
  methylationDataList[['met1-3']], regions = regions,
  context = "CG", method = "neighbourhood",
  test = "score", pValueThreshold = 0.01,
  minCytosinesCount = 4, minProportionDifference = 0.4,
  minGap = 200, minSize = 50, minReadsPerCytosine = 4,
  cores = 1)

# compute the DMRs in CG context with bins method
DMRsBinsCG <- computeDMRs(methylationDataList[['WT']],
  methylationDataList[['met1-3']], regions = regions,
  context = "CG", method = "bins", binSize = 100,
  test = "score", pValueThreshold = 0.01,
  minCytosinesCount = 4, minProportionDifference = 0.4,
  minGap = 200, minSize = 50, minReadsPerCytosine = 4,
  cores = 1)

# load the gene annotation data
data(GEs)

# select the genes
genes <- GEs[which(GEs$type == "gene")]

# the regions where to compute the DMRs
genes <- genes[overlapsAny(genes, regions)]

# filter genes that are differentially methylated in the two conditions
DMRsGenesCG <- filterDMRs(methylationDataList[['WT']],
  methylationDataList[['met1-3']], potentialDMRs = genes,
  context = "CG", test = "score", pValueThreshold = 0.01,
  minCytosinesCount = 4, minProportionDifference = 0.4,
  minReadsPerCytosine = 3, cores = 1)
#merge the DMRs

```
DMRsNoiseFilterCGLarger <- mergeDMRsIteratively(DMRsNoiseFilterCG,
    minGap = 500, respectSigns = TRUE,
    methylationDataList[["WT"]],
    methylationDataList[["met1-3"]],
    context = "CG", minProportionDifference=0.4,
    minReadsPerCytosine = 1, pValueThreshold=0.01,
    test="score",alternative = "two.sided")
```

#select the genes

```
genes <- GEs[which(GEs$type == "gene")]
```

# the coordinates of the area to be plotted

```
chr3Reg <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(510000,530000))
```

# load the DMRs in CG context

data(DMRsNoiseFilterCG)

```
DMRsCGlist <- list("noise filter"=DMRsNoiseFilterCG,
                   "neighbourhood"=DMRsNeighbourhoodCG,
                   "bins"=DMRsBinsCG,
                   "genes"=DMRsGenesCG)
```

# plot the CG methylation

```
par(mar=c(4, 4, 3, 1)+0.1)
par(mfrow=c(1,1))
plotLocalMethylationProfile(methylationDataList[["WT"]],
    methylationDataList[["met1-3"]], chr3Reg,
    DMRsCGlist, c("WT", "met1-3"), GEs,
    windowSize=100, main="CG methylation")
```

```
hotspotsHypo <- computeOverlapProfile(
    DMRsNoiseFilterCG[(DMRsNoiseFilterCG$regionType == "loss")],
    region, windowSize=2000, binary=TRUE, cores=1)
```

```
hotspotsHyper <- computeOverlapProfile(
    DMRsNoiseFilterCG[(DMRsNoiseFilterCG$regionType == "gain")],
    region, windowSize=2000, binary=TRUE, cores=1)
```

```
plotOverlapProfile(GRangesList("Chr3"=hotspotsHypo),
    GRangesList("Chr3"=hotspotsHyper),
    names=c("loss", "gain"), title="CG methylation")
```

# loading synthetic data

data("syntheticDataReplicates")

# creating condition vector

```
condition <- c("a", "a", "b", "b")
```

# computing DMRs using the neighbourhood method

```
DMRsReplicatesNeighbourhood <- computeDMRsReplicates(methylationData = methylationData,
```
extractGC

This function extracts GC sites in the genome

**Usage**

```
extractGC(methylationData, genome, contexts = c("ALL", "CG", "CHG", "CHH"))
```
filterDMRs

Arguments

methylationData  the methylation data stored as a GRanges object with four metadata columns (see methylationDataList).

genome  a BSgenome with the DNA sequence of the organism

contexts  the context in which the DMRs are computed ("ALL", "CG", "CHG" or "CHH").

Value

the a subset of methylationData consisting of all GC sites.

Author(s)

Ryan Merritt

Examples

## Not run:
# load the genome sequence
if(!require("BSgenome.Athaliana.TAIR.TAIR9", character.only = TRUE)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Athaliana.TAIR.TAIR9")
}
library(BSgenome.Athaliana.TAIR.TAIR9)

# load the methylation data
data(methylationDataList)

methylationDataWTGpCpG <- extractGC(methylationDataList["WT"],
  BSgenome.Athaliana.TAIR.TAIR9,
  "CG")

## End(Not run)

---

filterDMRs  Filter DMRs

Description

This function verifies whether a set of potential DMRs (e.g. genes, transposons, CpG islands) are differentially methylated or not.

Usage

filterDMRs(methylationData1, methylationData2, potentialDMRs, context = "CG",
  test = "fisher", pValueThreshold = 0.01, minCytosinesCount = 4,
  minProportionDifference = 0.4, minReadsPerCytosine = 3, cores = 1)
Arguments

methylationData1
the methylation data in condition 1 (see methylationDataList).
methylationData2
the methylation data in condition 2 (see methylationDataList).
potentialDMRs
a GRanges object with potential DMRs where to compute the DMRs. This can be a list of gene and/or transposable elements coordinates.
context
the context in which the DMRs are computed ("CG", "CHG" or "CHH").
test
the statistical test used to call DMRs ("fisher" for Fisher’s exact test or "score" for Score test).
pValueThreshold
DMRs with p-values (when performing the statistical test; see test) higher or equal than pValueThreshold are discarded. Note that we adjust the p-values using the Benjamini and Hochberg’s method to control the false discovery rate.
minCytosinesCount
DMRs with less cytosines in the specified context than minCytosinesCount will be discarded.
minProportionDifference
DMRs where the difference in methylation proportion between the two conditions is lower than minProportionDifference are discarded.
minReadsPerCytosine
DMRs with the average number of reads lower than minReadsPerCytosine are discarded.
cores
the number of cores used to compute the DMRs.

Value

a GRanges object with 11 metadata columns that contain the DMRs; see computeDMRs.

Author(s)

Nicolae Radu Zabet

See Also

DMRsNoiseFilterCG, computeDMRs, analyseReadsInsideRegionsForCondition and mergeDMRsIteratively

Examples

# load the methylation data
data(methylationDataList)
# load the gene annotation data
data(GEs)

# select the genes
genes <- GEs[which(GEs$type == "gene")]

# the regions where to compute the DMRs
regions <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(1,1E5))
genes <- genes[overlapsAny(genes, regions)]

# filter genes that are differentially methylated in the two conditions
DMRsGenesCG <- filterDMRs(methylationDataList["WT"],
                          methylationDataList["met1-3"], potentialDMRs = genes,
                          context = "CG", test = "score", pValueThreshold = 0.01,
                          minCytosinesCount = 4, minProportionDifference = 0.4,
                          minReadsPerCytosine = 3, cores = 1)

GEs

The genetic elements data

Description

A GRanges object containing the annotation of the Arabidopsis thaliana

Format

A GRanges object

Source

The object was created by calling import.gff3 function from rtracklayer package for ftp://ftp.arabidopsis.org/Maps/gbrowse_data/TAIR10/TAIR10_GFF3_genes_transposons.gff

getWholeChromosomes

Get whole chromosomes from methylation data

Description

Returns a GRanges object spanning from the first cytocine until the last one on each chromosome

Usage

getWholeChromosomes(methylationData)

Arguments

methylationData

the methylation data stored as a GRanges object with four metadata columns (see methylationDataList).

Value

a GRanges object will all chromosomes.
Author(s)
Nicolae Radu Zabet

Examples

```r
# load the methylation data
data(methylationDataList)

# get all chromosomes
chromosomes <- getWholeChromosomes(methylationDataList[["WT"]])
```

---

**joinReplicates**

*Joins together two GRanges objects in a single containing all the replicates*

---

Description

This function joins together data that come from biological replicates to perform analysis.

Usage

```r
joinReplicates(methylationData1, methylationData2, usecomplete = FALSE)
```

Arguments

- `methylationData1`: the methylation data stored as a GRanges object with four metadata columns (see `methylationDataList`).
- `methylationData2`: the methylation data stored as a GRanges object with four metadata columns (see `methylationDataList`).
- `usecomplete`: Boolean, determine wheter, when the two dataset differ for number of cytosines, if the smaller dataset should be added with zero reads to match the bigger dataset.

Value

returns a GRanges object containing multiple metadata columns with the reads from each object passed as parameter

Author(s)

Alessandro Pio Greco and Nicolae Radu Zabet
mergeDMRsIteratively

Examples

```r
## Not run:
# load the methylation data
data(methylationDataList)

# Joins the wildtype and the mutant in a single object
joined_data <- joinReplicates(methylationDataList["WT"],
                               methylationDataList["met1-3"], FALSE)

## End(Not run)
```

mergeDMRsIteratively  Merge DMRs iteratively

Description

This function takes a list of DMRs and attempts to merge DMRs while keeping the new DMRs statistically significant.

Usage

```r
mergeDMRsIteratively(DMRs, minGap, respectSigns = TRUE, methylationData1,
                      methylationData2, context = "CG", minProportionDifference = 0.4,
                      minReadsPerCytosine = 4, pValueThreshold = 0.01, test = "fisher",
                      alternative = "two.sided", cores = 1)
```

Arguments

- **DMRs**
  the list of DMRs as a GRanges object; e.g. see computeDMRs
- **minGap**
  DMRs separated by a gap of at least minGap are not merged.
- **respectSigns**
  logical value indicating whether to respect the sign when joining DMRs.
- **methylationData1**
  the methylation data in condition 1 (see methylationDataList).
- **methylationData2**
  the methylation data in condition 2 (see methylationDataList).
- **context**
  the context in which the DMRs are computed ("CG", "CHG" or "CHH").
- **minProportionDifference**
  two adjacent DMRs are merged only if the difference in methylation proportion of the new DMR is higher than minProportionDifference.
- **minReadsPerCytosine**
  two adjacent DMRs are merged only if the number of reads per cytosine of the new DMR is higher than minReadsPerCytosine.
- **pValueThreshold**
  two adjacent DMRs are merged only if the p-value of the new DMR (see test below) is lower than pValueThreshold. Note that we adjust the p-values using the Benjamini and Hochberg’s method to control the false discovery rate.
- **test**
  the test statistic to use for significance testing.
- **alternative**
  the alternative hypothesis for the test.
- **cores**
  the number of cores to use for parallel processing.
mergeDMRsIteratively

test  the statistical test used to call DMRs ("fisher" for Fisher's exact test or "score" for Score test).

alternative  indicates the alternative hypothesis and must be one of "two.sided", "greater" or "less".

cores  the number of cores used to compute the DMRs.

Value

the reduced list of DMRs as a GRanges object; e.g. see computeDMRs

Author(s)

Nicolae Radu Zabet

See Also

filterDMRs, computeDMRs, analyseReadsInsideRegionsForCondition and DMRsNoiseFilterCG

Examples

# load the methylation data
data(methylationDataList)

#load the DMRs in CG context they were computed with minGap = 200
data(DMRsNoiseFilterCG)

#merge the DMRs
DMRsNoiseFilterCGLarger <- mergeDMRsIteratively(DMRsNoiseFilterCG[1:100],
  minGap = 500, respectSigns = TRUE,
  methylationDataList[['WT']],
  methylationDataList[['met1-3']],
  context = "CG", minProportionDifference=0.4,
  minReadsPerCytosine = 1, pValueThreshold=0.01,
  test="score",alternative = "two.sided")

## Not run:
#set genomic coordinates where to compute DMRs
regions <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(1,1E5))

# compute DMRs and remove gaps smaller than 200 bp
DMRsNoiseFilterCG200 <- computeDMRs(methylationDataList[['WT']],
  methylationDataList[['met1-3']], regions = regions,
  context = "CG", method = "noise_filter",
  windowSize = 100, kernelFunction = "triangular",
  test = "score", pValueThreshold = 0.01,
  minCytosinesCount = 1, minProportionDifference = 0.4,
  minGap = 200, minSize = 0, minReadsPerCytosine = 1,
  cores = 1)
methylationDataList <- computeDMRs(methylationDataList[["WT"]],
methylationDataList[["met1-3"]], regions = regions,
context = "CG", method = "noise_filter",
WindowSize = 100, kernelFunction = "triangular",
test = "score", pValueThreshold = 0.01,
minCytosinesCount = 1, minProportionDifference = 0.4,
minGap = 0, minSize = 0, minReadsPerCytosine = 1,
cores = 1)

DMRsNoiseFilterCG0Merged200 <- mergeDMRsIteratively(DMRsNoiseFilterCG0,
               minGap = 200, respectSigns = TRUE,
methylationDataList[["WT"]],
methylationDataList[["met1-3"]],
context = "CG", minProportionDifference=0.4,
minReadsPerCytosine = 1, pValueThreshold=0.01,
test="score",alternative = "two.sided")

#check that all newly computed DMRs are identical
print(all(DMRsNoiseFilterCG200 == DMRsNoiseFilterCG0Merged200))

## End(Not run)

methylationDataList  The methylation data list

Description

A GRangesList object containing the methylation data at each cytosine location in the genome in Wild Type (WT) and met1-3 mutant (met1-3) in Arabidopsis thaliana. The data only contains the first 1 Mbp from Chromosome 3.

Format

The GRanges elements contain four metadata columns

context    the context in which the DMRs are computed ("CG", "CHG" or "CHH").
readsM     the number of methylated reads.
readsN     the total number of reads.
trinucleotide_context the specific context of the cytosine (H is replaced by the actual nucleotide).

Source

Description

This function plots the methylation profile at one locus for the bisulfite sequencing data. The points on the graph represent methylation proportion of individual cytosines, their colour which sample they belong to and the intensity of the the colour how many reads that particular cytosine had. This means that darker colors indicate stronger evidence that the corresponding cytosine has the corresponding methylation proportion, while lighter colors indicate a weaker evidence. The solid lines represent the smoothed profiles and the intensity of the line the coverage at the corresponding position (darker colors indicate more reads while lighter ones less reads). The boxes on top represent the DMRs, where a filled box will represent a DMR which gained methylation while a box with a pattern represent a DMR that lost methylation. The DMRs need to have a metadafield "regionType" which can be either "gain" (where there is more methylation in condition 2 compared to condition 1) or "loss" (where there is less methylation in condition 2 compared to condition 1). In case this metadafield is missing all DMRs are drawn using a filled box. Finally, we also allow annotation of the DNA sequence. We represent by a black boxes all the exons, which are joined by a horizontal black line, thus, marking the full body of the gene. With grey boxes we mark the transposable elements. Both for genes and transposable elements we plot them over a mid line if they are on the positive strand and under the mid line if they are on the negative strand.

Usage

plotLocalMethylationProfile(methylationData1, methylationData2, region, DMRs = NULL, conditionsNames = NULL, gff = NULL, windowSize = 150, context = "CG", labels = NULL, col = NULL, main = "", plotMeanLines = TRUE, plotPoints = TRUE)

Arguments

methylationData1

the methylation data in condition 1 (see methylationDataList).

methylationData2

the methylation data in condition 2 (see methylationDataList).

region

a GRanges object with the region where to plot the high resolution profile.

DMRs

a GRangesList object or a list with the list of DMRs (see computeDMRs or filterDMRs).

conditionsNames

the names of the two conditions. This will be used to plot the legend.

gff

a GRanges object with all elements usually imported from a GFF3 file. The gff file needs to have an metafield "type". Only the elements of type "gene", "exon" and "transposable_element" are plotted. Genes are represented as horizontal black lines, exons as a black rectangle and transposable elements as a grey rectangle. The elements are plotted on the corresponding strand (+ or -).
plotLocalMethylationProfile

windowSize   the size of the triangle base used to smooth the average methylation profile.

context  the context in which the DMRs are computed ("CG", "CHG" or "CHH").

labels  a vector of character used to add a subfigure characters to the plot. If NULL nothing is added.

col  a character vector with the colors. It needs to contain a minimum of 4 length(DMRs) colors. If not or if NULL, the defalut colors will be used.

main  a character with the title of the plot

plotMeanLines  a logical value indicating whether to plot the mean lines or not.

plotPoints  a logical value indicating whether to plot the points or not.

Value

Invisibly returns NULL

Author(s)

Nicolae Radu Zabet

Examples

# load the methylation data
data(methylationDataList)
# load the gene annotation data
data(GEs)

#select the genes
genes <- GEs[which(GEs$type == "gene")]

# the coordinates of the area to be plotted
chr3Reg <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(510000,530000))

# load the DMRs in CG context
data(DMRsNoiseFilterCG)

DMRsCGlist <- list("noise filter"=DMRsNoiseFilterCG)

# plot the CG methylation
par(mar=c(4, 4, 3, 1)+0.1)
par(mfrow=c(1,1))
plotLocalMethylationProfile(methylationDataList[["WT"]],
methylationDataList[["met1-3"]], chr3Reg,
DMRsCGlist, c("WT", "met1-3"), GEs,
windowSize=100, main="CG methylation")
plotMethylationDataCoverage

*Plot methylation data coverage*

**Description**

This function plots the coverage for the bisulfite sequencing data.

**Usage**

```r
plotMethylationDataCoverage(methylationData1, methylationData2 = NULL, breaks, regions = NULL, conditionsNames = NULL, context = "CG", proportion = TRUE, labels = NULL, col = NULL, pch = c(1, 0, 16, 2, 15, 17), lty = c(4, 1, 3, 2, 6, 5), contextPerRow = FALSE)
```

**Arguments**

- `methylationData1` the methylation data in condition 1 (see `methylahionDataList`).
- `methylationData2` the methylation data in condition 2 (see `methylahionDataList`). This is optional.
- `breaks` a numeric vector specifying the different values for the thresholds when computing the coverage.
- `regions` a `GRanges` object with the regions where to compute the coverage. If `NULL`, the coverage is computed genome-wide.
- `conditionsNames` a vector of character with the names of the conditions for `methylationData1` and `methylationData2`.
- `context` the context in which the DMRs are computed ("CG", "CHG" or "CHH").
- `proportion` a logical value indicating whether proportion or counts will be plotted.
- `labels` a vector of character used to add a subfigure character to the plot. If `NULL` nothing is added.
- `col` a character vector with the colors. It needs to contain a minimum of 2 colors per condition. If not or if `NULL`, the default colors will be used.
- `pch` the R symbols used to plot the data. It needs to contain a minimum of 2 symbols per condition. If not or if `NULL`, the default symbols will be used.
- `lty` the line types used to plot the data. It needs to contain a minimum of 2 line types per condition. If not or if `NULL`, the default line types will be used.
- `contextPerRow` a logical value indicating if the each row represents an individual context. If `FALSE`, each column will represent an individual context.
plotMethylationDataCoverage

Details

This function plots the proportion of cytosines in a specific context that have at least a certain number of reads (x-axis).

Value

Invisibly returns NULL.

Author(s)

Nicolae Radu Zabet

See Also

computeMethylationDataCoverage, methylationDataList

Examples

# load the methylation data
data(methylationDataList)

# plot the coverage in CG context
par(mar=c(4, 4, 3, 1)+0.1)
plotMethylationDataCoverage(methylationDataList[["WT"]],
                           methylationDataList[["met1-3"]],
breaks = c(1,5,10,15), regions = NULL,
conditionsNames = c("WT","met1-3"),
context = c("CG"), proportion = TRUE,
labels = LETTERS, col = NULL,
pch = c(1,0,16,2,15,17), lty = c(4,1,3,2,6,5),
contextPerRow = FALSE)

## Not run:
# plot the coverage in all three contexts
plotMethylationDataCoverage(methylationDataList[["WT"]],
                           methylationDataList[["met1-3"]],
breaks = 1:15, regions = NULL,
conditionsNames = c("WT","met1-3"),
context = c("CG", "CHG", "CHH"),
proportion = TRUE, labels = LETTERS, col = NULL,
pch = c(1,0,16,2,15,17), lty = c(4,1,3,2,6,5),
contextPerRow = FALSE)

## End(Not run)
plotMethylationDataSpatialCorrelation

Plot methylation data spatial correlation

Description

This function plots the correlation of methylation levels for Cytosines located at a certain distance apart.

Usage

plotMethylationDataSpatialCorrelation(methylationData1, methylationData2 = NULL, distances, regions = NULL, conditionsNames = NULL, context = "CG", labels = NULL, col = NULL, pch = c(1, 0, 16, 2, 15, 17), lty = c(4, 1, 3, 2, 6, 5), contextPerRow = FALSE, log = "")

Arguments

methylationData1
  the methylation data in condition 1 (see methylationDataList).
methylationData2
  the methylation data in condition 2 (see methylationDataList). This is optional.
distances
  a numeric vector specifying the different values for the distances when computing the correlation.
regions
  a GRanges object with the regions where to compute the correlation. If NULL, the coverage is computed genome-wide.
conditionsNames
  a vector of character with the names of the conditions for methylationData1 and methylationData2.
context
  the context in which the DMRs are computed ("CG", "CHG" or "CHH").
labels
  a vector of character used to add a subfigure character to the plot. If NULL nothing is added.
col
  a character vector with the colors. It needs to contain a minimum of 2 colors per condition. If not or if NULL, the default colors will be used.
pch
  the R symbols used to plot the data. It needs to contain a minimum of 2 symbols per condition. If not or if NULL, the default symbols will be used.
lty
  the line types used to plot the data. It needs to contain a minimum of 2 line types per condition. If not or if NULL, the default line types will be used.
contextPerRow
  a logical value indicating if the each row represents an individual context. If FALSE, each column will represent an individual context.
log
  a character indicating if any of the axes will be displayed on log scale. This argument will be passed to plot function.
This function plots the proportion of cytosines in a specific context that have at least a certain number of reads (x-axis).

Invisibly returns NULL

Nicolae Radu Zabet

computeMethylationDataSpatialCorrelation, methylationDataList

## Not run:
# load the methylation data
data(methylationDataList)

# plot the spatial correlation in CG context
par(mar=c(4, 4, 3, 1)+0.1)
plotMethylationDataSpatialCorrelation(methylationDataList[["WT"]],
    distances = c(1,5,10,15), regions = NULL,
    conditionsNames = c("WT","met1-3"),
    context = c("CG"),
    labels = LETTERS, col = NULL,
    pch = c(1,0,16,2,15,17), lty = c(4,1,3,2,6,5),
    contextPerRow = FALSE)

# plot the spatial correlation in all three contexts
plotMethylationDataSpatialCorrelation(methylationDataList[["WT"]],
methylationDataList[["met1-3"]],
    distances = c(1,5,10,15,20,50,100,150,200,500,1000),
    regions = NULL, conditionsNames = c("WT","met1-3"),
    context = c("CG", "CHG", "CHH"),
    labels = LETTERS, col = NULL,
    pch = c(1,0,16,2,15,17), lty = c(4,1,3,2,6,5),
    contextPerRow = FALSE, log="x")

## End(Not run)
plotMethylationProfile

Plot Methylation Profile

Description

This function plots the low resolution profiles for the bisulfite sequencing data.

Usage

plotMethylationProfile(methylationProfiles, autoscale = FALSE,
labels = NULL, title = "", col = NULL, pch = c(1, 0, 16, 2, 15, 17),
lty = c(4, 1, 3, 2, 6, 5))

Arguments

methylationProfiles
  a GRangesList object. Each GRanges object in the list is generated by calling
  the function computeMethylationProfile.
autoscale
  a logical value indicating whether the values are autoscaled for each context
  or not.
labels
  a vector of character used to add a subfigure characters to the plot. If NULL
  nothing is added.
title
  the plot title.
col
  a character vector with the colours. It needs to contain a minimum of 2 colours
  per context. If not or if NULL, the default colours will be used.
pch
  the R symbols used to plot the data.
lty
  the line types used to plot the data.

Value

Invisibly returns NULL

Author(s)

Nicolae Radu Zabet

See Also

plotMethylationProfileFromData, computeMethylationProfile and methylationDataList
Examples

# load the methylation data
data(methylationDataList)

# the region where to compute the profile
region <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(1,1E6))

# compute low resolution profile in 20 Kb windows
lowResProfileWTCG <- computeMethylationProfile(methylationDataList["WT"],
region, windowSize = 20000, context = "CG")

lowResProfilesCG <- GRangesList("WT" = lowResProfileWTCG)

# plot the low resolution profile
par(mar=c(4, 4, 3, 1)+0.1)
par(mfrow=c(1,1))
plotMethylationProfile(lowResProfilesCG, autoscale = FALSE,
title="CG methylation on Chromosome 3",
col=c("#D55E00","#E69F00"), pch = c(1,0),
lty = c(4,1))

## Not run:
# compute low resolution profile in 10 Kb windows in CG context
lowResProfileWTCG <- computeMethylationProfile(methylationDataList["WT"],
region, windowSize = 10000, context = "CG")

lowResProfileMet13CG <- computeMethylationProfile(methylationDataList["met1-3"],
region, windowSize = 10000, context = "CG")

lowResProfileCG <- GRangesList("WT" = lowResProfileWTCG,
"met1-3" = lowResProfileMet13CG)

# compute low resolution profile in 10 Kb windows in CHG context
lowResProfileWTCHG <- computeMethylationProfile(methylationDataList["WT"],
region, windowSize = 10000, context = "CHG")

lowResProfileMet13CHG <- computeMethylationProfile(methylationDataList["met1-3"],
region, windowSize = 10000, context = "CHG")

lowResProfileCHG <- GRangesList("WT" = lowResProfileWTCHG,
"met1-3" = lowResProfileMet13CHG)

# plot the low resolution profile
par(mar=c(4, 4, 3, 1)+0.1)
par(mfrow=c(2,1))
plotMethylationProfile(lowResProfileCG, autoscale = FALSE,
labels = LETTERS[1],
title="CG methylation on Chromosome 3",
col=c("#D55E00","#E69F00"), pch = c(1,0),
lty = c(4,1))
plotMethylationProfileFromData

Plot methylation profile from data

Description

This function plots the low resolution profiles for all bisulfite sequencing data.

Usage

plotMethylationProfileFromData(methylationData1, methylationData2 = NULL, regions = NULL, conditionsNames = NULL, context = "CG", windowSize = NULL, autoscale = FALSE, labels = NULL, col = NULL, pch = c(16,2), lty = c(3,2))

Arguments

methylationData1

the methylation data in condition 1 (see methylationDataList).

methylationData2

the methylation data in condition 2 (see methylationDataList). This is optional.

regions

a GRanges object with the regions where to plot the profiles.

conditionsNames

the names of the two conditions. This will be used to plot the legend.

class context

a vector with all contexts in which the DMRs are computed ("CG", "CHG" or "CHH").

windowSize

a numeric value indicating the size of the window in which methylation is averaged.

autoscale

a logical value indicating whether the values are autoscaled for each context or not.

labels

a vector of character used to add a subfigure character to the plot. If NULL nothing is added.

col

a character vector with the colours. It needs to contain a minimum of 2 colours per condition. If not or if NULL, the default colours will be used.
plotMethylationProfileFromData

**pch**
the R symbols used to plot the data. It needs to contain a minimum of 2 symbols per condition. If not or if NULL, the default symbols will be used.

**lty**
the line types used to plot the data. It needs to contain a minimum of 2 line types per condition. If not or if NULL, the default line types will be used.

**contextPerRow**
a logical value indicating if each row represents an individual context. If FALSE, each column will represent an individual context.

**Value**

Invisibly returns NULL

**Author(s)**
Nicolae Radu Zabet

**See Also**

plotMethylationProfile, computeMethylationProfile and methylationDataList

**Examples**

```r
# load the methylation data
data(methylationDataList)

# plot the low resolution profile at 10 Kb resolution
par(mar=c(4, 4, 3, 1)+0.1)
plotMethylationProfileFromData(methylationDataList[["WT"],
methylationDataList[["met1-3"]],
conditionsNames=c("WT", "met1-3"),
windowSize = 20000, autoscale = TRUE,
context = c("CHG"))

## Not run:
# plot the low resolution profile at 5 Kb resolution
par(mar=c(4, 4, 3, 1)+0.1)
plotMethylationProfileFromData(methylationDataList[["WT"],
methylationDataList[["met1-3"]],
conditionsNames=c("WT", "met1-3"),
windowSize = 5000, autoscale = TRUE,
context = c("CG", "CHG", "CHH"),
labels = LETTERS)

## End(Not run)
```
plotOverlapProfile  

Plot overlap profile

Description

This function plots the distribution of a set of subregions on a large region.

Usage

```r
plotOverlapProfile(overlapsProfiles1, overlapsProfiles2 = NULL,
                  names = NULL, labels = NULL, col = NULL, title = "",
                  logscale = FALSE, maxValue = NULL)
```

Arguments

- `overlapsProfiles1`: a `GRanges` object with the overlaps profile; see `computeOverlapProfile`.
- `overlapsProfiles2`: a `GRanges` object with the overlaps profile; see `computeOverlapProfile`. This is optional. For example, one can be use `overlapsProfiles1` to display hypomethylated regions and `overlapsProfiles2` the hypermethylated regions.
- `names`: a vector of character to add labels for the two `overlapsProfiles`. This is an optional parameter.
- `labels`: a vector of character used to add a subfigure character to the plot. If `NULL` nothing is added.
- `col`: a character vector with the colours. It needs to contain 2 colours. If not or if `NULL`, the default colours will be used.
- `title`: the title of the plot.
- `logscale`: a logical value indicating if the colours are on logscale or not.
- `maxValue`: a maximum value in a region. Used for the colour scheme.

Value

Invisibly returns `NULL`.

Author(s)

Nicolae Radu Zabet

See Also

`computeOverlapProfile`, `filterDMRs`, `computeDMRs` and `mergeDMRsIteratively`
Examples

# load the methylation data
data(methylationDataList)

# load the DMRs in CG context
data(DMRsNoiseFilterCG)

# the coordinates of the area to be plotted
largeRegion <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(1,1E5))

# compute overlaps distribution
hotspotsHypo <- computeOverlapProfile(DMRsNoiseFilterCG, largeRegion,
  windowSize = 10000, binary = FALSE)

plotOverlapProfile(GRangesList("Chr3"=hotspotsHypo),
  names = c("hypomethylated"), title = "CG methylation")

## Not run:

largeRegion <- GRanges(seqnames = Rle("Chr3"), ranges = IRanges(1,1E6))

hotspotsHypo <- computeOverlapProfile(
  DMRsNoiseFilterCG[(DMRsNoiseFilterCG$regionType == "loss")],
  largeRegion, windowSize=2000, binary=TRUE, cores=1)

hotspotsHyper <- computeOverlapProfile(
  DMRsNoiseFilterCG[(DMRsNoiseFilterCG$regionType == "gain")],
  largeRegion, windowSize=2000, binary=TRUE, cores=1)

plotOverlapProfile(GRangesList("Chr3"=hotspotsHypo),
  GRangesList("Chr3"=hotspotsHyper),
  names=c("loss", "gain"), title="CG methylation")

## End(Not run)

poolMethylationDatasets

Pool methylation data

Description

This function pools together multiple methylation datasets.

Usage

poolMethylationDatasets(methylationDataList)
Arguments

methylationDataList

a GRangesList object where each element of the list is a GRanges object with the methylation data in the corresponding condition (see methylationDataList).

Value

the methylation data stored as a GRanges object with four metadata columns (see methylationDataList).

Author(s)

Nicolae Radu Zabet

Examples

# load methylation data object
data(methylationDataList)

# pools the two datasets together
pooledMethylationData <- poolTwoMethylationDatasets(methylationDataList)
Examples

# load methylation data object
data(methylationDataList)

# save the two datasets together
pooledMethylationData <- poolTwoMethylationDatasets(methylationDataList[[1]],
methylationDataList[[2]])

readBismark

Description

This function takes as input a CX report file produced by Bismark and returns a Granges object
with four metadata columns. The file represents the bisulfite sequencing methylation data.

Usage

readBismark(file)

Arguments

file The filename (including path) of the methylation (CX report generated by Bismark) to be read.

Value

the methylation data stored as a Granges object with four metadata columns (see methylationDataList).

Author(s)

Nicolae Radu Zabet and Jonathan Michael Foonlan Tsang

Examples

# load methylation data object
data(methylationDataList)

# save the two datasets together
pooledMethylationData <- poolTwoMethylationDatasets(methylationDataList[[1]],
methylationDataList[[2]])

# load the data
methylationDataWT <- readBismark("chr3test_a_thaliana_wt.CX_report")

# check that the loading worked
all(methylationDataWT == methylationDataList[["WT"]])
readBismarkPool

Description

This function takes as input a vector of CX report file produced by Bismark and returns a GRanges object with four metadata columns (see methylationDataList). The file represents the pooled bisulfite sequencing data.

Usage

readBismarkPool(files)

Arguments

files The filenames (including path) of the methylation (CX report generated with Bismark) to be read

Value

the methylation data stored as a GRanges object with four metadata columns (see methylationDataList).

Author(s)

Nicolae Radu Zabet and Jonathan Michael Foonlan Tsang

Examples

# load methylation data object
data(methylationDataList)

# save the two datasets
saveBismark(methylationDataList[["WT"]],
            "chr3test_a_thaliana_wt.CX_report")
saveBismark(methylationDataList[["met1-3"]],
            "chr3test_a_thaliana_met13.CX_report")

# reload the two datasets and pool them
filenames <- c("chr3test_a_thaliana_wt.CX_report",
               "chr3test_a_thaliana_met13.CX_report")
methylationDataPool <- readBismarkPool(filenames)
Description

This function takes as input a GRanges object generated with readBismark and saves the output to a file using Bismark CX report format.

Usage

saveBismark(methylationData, filename)

Arguments

methylationData
the methylation data stored as a GRanges object with four metadata columns (see methylationDataList).

filename
the filename where the data will be saved.

Value

Invisibly returns NULL

Author(s)

Nicolae Radu Zabet

Examples

# load methylation data object
data(methylationDataList)

# save one dataset to a file
saveBismark(methylationDataList[["WT"]], "chr3test_a_thaliana_wt.CX_report")

syntheticDataReplicates

Simulated data for biological replicates

Description

A GRanges object containing simulated data for methylation in four samples. The conditions associated with each sample are a, a, b and b.
Format

A `GRanges` object containing multiple metadata columns with the reads from each object passed as parameter.

Source

The object was created by calling `joinReplicates` function.
Index

analyseReadsInsideRegionsForCondition, 2, 5, 14, 18, 20, 24
computeDMRs, 3, 4, 13, 14, 18, 20, 23, 24, 26, 36
computeDMRsReplicates, 6, 14
computeMethylationDataCoverage, 9, 13, 29
computeMethylationDataSpatialCorrelation, 10, 14, 31
computeMethylationProfile, 11, 14, 32, 35
computeOverlapProfile, 12, 14, 36
DMRcaller, 13
DMRcaller-package (DMRcaller), 13
DMRsNoiseFilterCG, 3, 5, 18, 20, 24
extractGC, 18

filterDMRs, 3, 5, 13, 14, 18, 19, 24, 36

GEs, 21
getWholeChromosomes, 14, 21
GRanges, 3–5, 7–13, 18–24, 26, 28, 30, 32, 34, 36, 38–42
GRangesList, 26, 38

joinReplicates, 14, 22, 42

mergeDMRsIteratively, 3, 5, 13, 14, 18, 20, 23, 36
methylationDataList, 3, 4, 9–12, 18–23, 25, 26, 28–32, 34, 35, 38–41

plot, 30
plotLocalMethylationProfile, 14, 26
plotMethylationDataCoverage, 9, 14, 28
plotMethylationDataSpatialCorrelation, 10, 14, 30
plotMethylationProfile, 12, 14, 32, 35

poolMethylationDatasets, 13, 37
poolTwoMethylationDatasets, 13, 38
readBismark, 13, 25, 39, 41
readBismarkPool, 13, 40
saveBismark, 13, 41
syntheticDataReplicates, 41