Package ‘epialleleR’

October 14, 2021

Title Fast, Epiallele-Aware Methylation Reporter

Version 1.0.0

CommentMaintainer Oleksii Nikolaienko <oleksii.nikolaienko@gmail.com>

Description Epialleles are specific DNA methylation patterns that are mitotically and/or meiotically inherited. This package calls hypermethylated epiallele frequencies at the level of genomic regions or individual cytosines in next-generation sequencing data using binary alignment map (BAM) files as an input. Other functionality includes computing the empirical cumulative distribution function for per-read beta values, and testing the significance of the association between epiallele methylation status and base frequencies at particular genomic positions (SNPs).

CommentSystemRequirements C++17

NeedsCompilation yes

Depends R (>= 4.1)

Imports stats, methods, utils, Rsamtools, GenomicRanges, BiocGenerics, GenomeInfoDb, SummarizedExperiment, VariantAnnotation, stringi, data.table

LinkingTo Rcpp, BH

Suggests RUnit, knitr, rmarkdown

License Artistic-2.0

URL https://github.com/BBCG/epialleleR

BugReports https://github.com/BBCG/epialleleR/issues

Encoding UTF-8

biocViews DNAMethylation, Epigenetics, MethylSeq

RoxygenNote 7.1.1

VignetteBuilder knitr

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

git_branch RELEASE_3_13

git_last_commit b01940d

git_last_commit_date 2021-05-19

Date/Publication 2021-10-14

Author Oleksii Nikolaienko [aut, cre]
(<https://orcid.org/0000-0002-5910-4934>)

Maintainer Oleksii Nikolaienko <oleksii.nikolaienko@gmail.com>

R topics documented:

`generateBedEcdf` *generateBedEcdf*

Description

This function computes empirical cumulative distribution functions (eCDF) for per-read beta values of the sequencing reads.

Usage

```
generateBedEcdf(
  bam,
  bed,
  bed.type = c("amplicon", "capture"),
  bed.rows = c(1),
  zero.based.bed = FALSE,
  match.tolerance = 1,
  match.min.overlap = 1,
  ecdf.context = c("CG", "CHG", "CHH", "CxG", "CX"),
  min.mapq = 0,
  skip.duplicates = FALSE,
  verbose = TRUE
)
```

Arguments

`bam`	BAM file location string OR preprocessed output of `preprocessBam` function.
`bed`	Browser Extensible Data (BED) file location string OR object of class `GRanges` holding genomic coordinates for regions of interest. It is used to match sequencing reads to the genomic regions prior to eCDF computation. The seqlevels of BED file/object must match the seqlevels of the BAM file/object used.

`bed.type`	character string for the type of assay that was used to produce sequencing reads: • "amplicon" (the default) – used for amplicon-based next-generation sequencing when exact coordinates of sequenced fragments are known. Matching of reads to genomic ranges are then performed by the read's start or end positions, either of which should be no further than 'match.tolerance' bases away from the start or end position of genomic ranges given in BED file/`GRanges` object • "capture" – used for capture-based next-generation sequencing when reads partially overlap with the capture target regions. Read is considered to match the genomic range when their overlap is more or equal to 'match.min.overlap'. If read matches two or more BED genomic regions, only the first match is taken (input `GRanges` are **not** sorted internally)
`bed.rows`	integer vector specifying what 'bed' regions should be included in the output. If 'c(1)' (the default), then function returns eCDFs for the first region of 'bed', if NULL - eCDF functions for all 'bed' genomic regions as well as for the reads that didn't match any of the regions (last element of the return value; only if there are such reads).
`zero.based.bed`	boolean defining if BED coordinates are zero based (default: FALSE).
`match.tolerance`	integer for the largest difference between read's and BED `GRanges` start or end positions during matching of amplicon-based NGS reads (default: 1).
`match.min.overlap`	integer for the smallest overlap between read's and BED `GRanges` start or end positions during matching of capture-based NGS reads (default: 1). If read matches two or more BED genomic regions, only the first match is taken (input `GRanges` are **not** sorted internally).
`ecdf.context`	string defining cytosine methylation context used for computing within-the-context and out-of-context eCDFs: • "CG" (the default) – within-the-context: CpG cytosines (called as zZ), out-of-context: all the other cytosines (hHxX) • "CHG" – within-the-context: CHG cytosines (xX), out-of-context: hHzZ • "CHH" – within-the-context: CHH cytosines (hH), out-of-context: xXzZ • "CxG" – within-the-context: CG and CHG cytosines (zZxX), out-of-context: CHH cytosines (hH) • "CX" – all cytosines are considered within-the-context
`min.mapq`	non-negative integer threshold for minimum read mapping quality (default: 0). Option has no effect if preprocessed BAM data was supplied as an input.
`skip.duplicates`	boolean defining if duplicate aligned reads should be skipped (default: FALSE). Option has no effect if preprocessed BAM data was supplied as an input OR duplicate reads were not marked by alignment software.
`verbose`	boolean to report progress and timings (default: TRUE).

Details

The function matches reads (or read pairs as a single entity) to the genomic regions provided in a BED file/GRanges object, computes average per-read beta values according to the cytosine context parameter ‘ecdf.context‘, and returns a list of eCDFs for within- and out-of-context average per-read beta values, which can be used for plotting.

The resulting eCDFs and their plots can be used to characterise the methylation pattern of a particular genomic region, e.g. if reads that match to that region are methylated in an "all-or-none" manner or if some intermediate methylation levels are more frequent.

Value

list with a number of elements equal to the length of ‘bed.rows‘ (if not NULL), or to the number of genomic regions within ‘bed‘ (if ‘bed.rows==NULL‘) plus one item for all reads not matching ‘bed‘ genomic regions (if any). Every list item is a list on it’s own, consisting of two eCDF functions for within- and out-of-context per-read beta values.

See Also

preprocessBam for preloading BAM data, generateCytosineReport for methylation statistics at the level of individual cytosines, generateBedReport for genomic region-based statistics, generateVcfReport for evaluating epiallele-SNV associations, and ‘epialleleR‘ vignettes for the description of usage and sample data.

Examples

```
# amplicon data
amplicon.bam <- system.file("extdata", "amplicon010meth.bam",
                            package="epialleleR")
amplicon.bed <- system.file("extdata", "amplicon.bed",
                            package="epialleleR")

# let's compute eCDF
amplicon.ecdfs <- generateBedEcdf(bam=amplicon.bam, bed=amplicon.bed,
                                  bed.rows=NULL)

# there are 5 items in amplicon.ecdfs, let's plot all of them
par(mfrow=c(1,length(amplicon.ecdfs)))

# cycle through items
for (x in 1:length(amplicon.ecdfs)) {
  # four of them have names corresponding to amplicon.bed genomic regions,
  # fifth - NA for all the reads that don't match to any of those regions
  main <- if (is.na(names(amplicon.ecdfs[x]))) "unmatched"
          else names(amplicon.ecdfs[x])

  # plotting eCDF for within-the-context per-read beta values (in red)
  plot(amplicon.ecdfs[[x]]$context, col="red", verticals=TRUE,
       do.points=FALSE, xlim=c(0,1), xlab="per-read beta value",
       ylab="cumulative density", main=main)
```

```
    # adding eCDF for out-of-context per-read beta values (in blue)
    plot(amplicon.ecdfs[[x]]$out.of.context, add=TRUE, col="blue",
         verticals=TRUE, do.points=FALSE)
  }
```

---

generateBedReport *generateBedReport*

Description

'generateBedReport', 'generateAmpliconReport', 'generateCaptureReport' – these functions match BAM reads to the set of genomic locations and return the fraction of reads with an average methylation level passing an arbitrary threshold.

Usage

```
generateAmpliconReport(
  bam,
  bed,
  report.file = NULL,
  zero.based.bed = FALSE,
  match.tolerance = 1,
  threshold.reads = TRUE,
  threshold.context = c("CG", "CHG", "CHH", "CxG", "CX"),
  min.context.sites = 2,
  min.context.beta = 0.5,
  max.outofcontext.beta = 0.1,
  min.mapq = 0,
  skip.duplicates = FALSE,
  gzip = FALSE,
  verbose = TRUE
)

generateCaptureReport(
  bam,
  bed,
  report.file = NULL,
  zero.based.bed = FALSE,
  match.min.overlap = 1,
  threshold.reads = TRUE,
  threshold.context = c("CG", "CHG", "CHH", "CxG", "CX"),
  min.context.sites = 2,
  min.context.beta = 0.5,
  max.outofcontext.beta = 0.1,
  min.mapq = 0,
  skip.duplicates = FALSE,
  gzip = FALSE,
  verbose = TRUE
)

generateBedReport(
  bam,
  bed,
  report.file = NULL,
  zero.based.bed = FALSE,
  bed.type = c("amplicon", "capture"),
  match.tolerance = 1,
  match.min.overlap = 1,
  threshold.reads = TRUE,
  threshold.context = c("CG", "CHG", "CHH", "CxG", "CX"),
  min.context.sites = 2,
  min.context.beta = 0.5,
  max.outofcontext.beta = 0.1,
  min.mapq = 0,
  skip.duplicates = FALSE,
  gzip = FALSE,
  verbose = TRUE
)
```

Arguments

`bam`	BAM file location string OR preprocessed output of `preprocessBam` function.
`bed`	Browser Extensible Data (BED) file location string OR object of class `GRanges` holding genomic coordinates for regions of interest. The seqlevels of BED file/object must be the same as seqlevels of BAM file/object used.
`report.file`	file location string to write the BED report. If NULL (the default) then report is returned as a `data.table` object.
`zero.based.bed`	boolean defining if BED coordinates are zero based (default: FALSE).
`match.tolerance`	integer for the largest difference between read's and BED `GRanges` start or end positions during matching of amplicon-based NGS reads (default: 1).
`threshold.reads`	boolean defining if sequence reads should be thresholded before counting reads belonging to variant epialleles (default: TRUE). Disabling thresholding is possible but makes no sense in this context as all the reads will be assigned to the variant epiallele, which will result in VEF==1 (in such case ‘NA‘ VEF values are returned in order to avoid confusion).
`threshold.context`	string defining cytosine methylation context used for thresholding the reads: • "CG" (the default) – within-the-context: CpG cytosines (called as zZ), out-of-context: all the other cytosines (hHxX) • "CHG" – within-the-context: CHG cytosines (xX), out-of-context: hHzZ • "CHH" – within-the-context: CHH cytosines (hH), out-of-context: xXzZ

- "CxG" – within-the-context: CG and CHG cytosines (zZxX), out-of-context: CHH cytosines (hH)
- "CX" – all cytosines are considered within-the-context, this effectively results in no thresholding

This option has no effect when read thresholding is disabled.

`min.context.sites`
: non-negative integer for minimum number of cytosines within the ‘threshold.context‘ (default: 2). Reads containing **fewer** within-the-context cytosines are considered completely unmethylated (thus belonging to the reference epiallele). This option has no effect when read thresholding is disabled.

`min.context.beta`
: real number in the range [0;1] (default: 0.5). Reads with average beta value for within-the-context cytosines **below** this threshold are considered completely unmethylated (thus belonging to the reference epiallele). This option has no effect when read thresholding is disabled.

`max.outofcontext.beta`
: real number in the range [0;1] (default: 0.1). Reads with average beta value for out-of-context cytosines **above** this threshold are considered completely unmethylated (thus belonging to the reference epiallele). This option has no effect when read thresholding is disabled.

`min.mapq` non-negative integer threshold for minimum read mapping quality (default: 0). Option has no effect if preprocessed BAM data was supplied as an input.

`skip.duplicates`
: boolean defining if duplicate aligned reads should be skipped (default: FALSE). Option has no effect if preprocessed BAM data was supplied as an input OR duplicate reads were not marked by alignment software.

`gzip` boolean to compress the report (default: FALSE).

`verbose` boolean to report progress and timings (default: TRUE).

`match.min.overlap`
: integer for the smallest overlap between read’s and BED `GRanges` start or end positions during matching of capture-based NGS reads (default: 1). If read matches two or more BED genomic regions, only the first match is taken (input `GRanges` are **not** sorted internally).

`bed.type` character string for the type of assay that was used to produce sequencing reads:

- "amplicon" (the default) – used for amplicon-based next-generation sequencing when exact coordinates of sequenced fragments are known. Matching of reads to genomic ranges are then performed by the read’s start or end positions, either of which should be no further than ‘match.tolerance‘ bases away from the start or end position of genomic ranges given in BED file/`GRanges` object
- "capture" – used for capture-based next-generation sequencing when reads partially overlap with the capture target regions. Read is considered to match the genomic range when their overlap is more or equal to ‘match.min.overlap‘. If read matches two or more BED genomic regions, only the first match is taken (input `GRanges` are **not** sorted internally)

Details

Functions report hypermethylated variant epiallele frequencies (VEF) per genomic region of interest using BAM and BED files as input. Reads (or read pairs as a single entity) are matched to genomic locations by exact coordinates (‘generateAmpliconReport‘ or ‘generateBedReport‘ with an option bed.type="amplicon") or minimum overlap (‘generateCaptureReport‘ or ‘generateBedReport‘ with an option bed.type="capture") – the former to be used for amplicon-based NGS data, while the latter – for the capture-based NGS data. The function’s logic is explained below.

Let’s suppose we have a BAM file with four reads, all mapped to the "+" strand of chromosome 1, positions 1-16. The genomic range is supplied as a parameter ‘bed = as("chr1:1-100", "GRanges")‘. Assuming the default values for the thresholding parameters (threshold.reads = TRUE, threshold.context = "CG", min.context.sites = 2, min.context.beta = 0.5, max.outofcontext.beta = 0.1), the input and results will look as following:

methylation string	threshold	explained
...Z..x+.h..x..h.	below	min.context.sites < 2 (only one zZ base)
...Z..z.h..x..h.	above	pass all criteria
...Z..z.h..X..h.	below	max.outofcontext.beta > 0.1 (1XH / 3xXhH = 0.33)
...Z..z.h..z-.h.	below	min.context.beta < 0.5 (1Z / 3zZ = 0.33)

Only the second read will satisfy all of the thresholding criteria, leading to the following BED report (given that all reads map to chr1:+:1-16):

seqnames	start	end	width	strand	nreads+	nreads-	VEF
chr1	1	100	100	*	4	0	0.25

Value

`data.table` object containing VEF report for BED `GRanges` or NULL if report.file was specified. If BAM file contains reads that would not match to any of BED `GRanges`, the last row in the report will contain information on such reads (with seqnames, start and end equal to NA). The report columns are:

- seqnames – reference sequence name
- start – start of genomic region
- end – end of genomic region
- width – width of genomic region
- strand – strand
- ... – other columns that were present in BED or metadata columns of `GRanges` object
- nreads+ – number of reads mapped to the forward ("+") strand
- nreads- – number of reads mapped to the reverse ("-") strand
- VEF – frequency of reads passing the threshold

See Also

`preprocessBam` for preloading BAM data, `generateCytosineReport` for methylation statistics at the level of individual cytosines, `generateVcfReport` for evaluating epiallele-SNV associations, generateBedEcdf for analysing the distribution of per-read beta values, and ‘epialleleR‘ vignettes for the description of usage and sample data.

GRanges class for working with genomic ranges, seqlevelsStyle function for getting or setting the seqlevels style.

Examples

```
# amplicon data
amplicon.bam    <- system.file("extdata", "amplicon010meth.bam",
                               package="epialleleR")
amplicon.bed    <- system.file("extdata", "amplicon.bed",
                               package="epialleleR")
amplicon.report <- generateAmpliconReport(bam=amplicon.bam,
                                          bed=amplicon.bed)

# capture NGS
capture.bam    <- system.file("extdata", "capture.bam",
                              package="epialleleR")
capture.bed    <- system.file("extdata", "capture.bed",
                              package="epialleleR")
capture.report <- generateCaptureReport(bam=capture.bam, bed=capture.bed)

# generateAmpliconReport and generateCaptureReport are just aliases
# of the generateBedReport
bed.report <- generateBedReport(bam=capture.bam, bed=capture.bed,
                                bed.type="capture")
identical(capture.report, bed.report)
```

---

generateCytosineReport   *generateCytosineReport*

Description

This function counts methylated and unmethylated DNA bases taking into the account average methylation level of the entire sequence read.

Usage

```
generateCytosineReport(
  bam,
  report.file = NULL,
  threshold.reads = TRUE,
  threshold.context = c("CG", "CHG", "CHH", "CxG", "CX"),
  min.context.sites = 2,
  min.context.beta = 0.5,
  max.outofcontext.beta = 0.1,
  report.context = threshold.context,
  min.mapq = 0,
  skip.duplicates = FALSE,
  gzip = FALSE,
  verbose = TRUE
)
```

Arguments

`bam`	BAM file location string OR preprocessed output of `preprocessBam` function.
`report.file`	file location string to write the cytosine report. If NULL (the default) then report is returned as a `data.table` object.
`threshold.reads`	boolean defining if sequence reads (read pairs) should be thresholded before counting methylated cytosines (default: TRUE). Disabling thresholding makes the report virtually indistinguishable from the ones generated by other software, such as Bismark or Illumina DRAGEN Bio IT Platform.
`threshold.context`	string defining cytosine methylation context used for thresholding the reads: • "CG" (the default) – within-the-context: CpG cytosines (called as zZ), out-of-context: all the other cytosines (hHxX) • "CHG" – within-the-context: CHG cytosines (xX), out-of-context: hHzZ • "CHH" – within-the-context: CHH cytosines (hH), out-of-context: xXzZ • "CxG" – within-the-context: CG and CHG cytosines (zZxX), out-of-context: CHH cytosines (hH) • "CX" – all cytosines are considered within-the-context, this effectively results in no thresholding This option has no effect when read thresholding is disabled.
`min.context.sites`	non-negative integer for minimum number of cytosines within the 'threshold.context' (default: 2). Reads containing **fewer** within-the-context cytosines are considered completely unmethylated (all C are counted as T). This option has no effect when read thresholding is disabled.
`min.context.beta`	real number in the range [0;1] (default: 0.5). Reads with average beta value for within-the-context cytosines **below** this threshold are considered completely unmethylated (all C are counted as T). This option has no effect when read thresholding is disabled.
`max.outofcontext.beta`	real number in the range [0;1] (default: 0.1). Reads with average beta value for out-of-context cytosines **above** this threshold are considered completely unmethylated (all C are counted as T). This option has no effect when read thresholding is disabled.
`report.context`	string defining cytosine methylation context to report (default: value of 'threshold.context').
`min.mapq`	non-negative integer threshold for minimum read mapping quality (default: 0). Option has no effect if preprocessed BAM data was supplied as an input.

`skip.duplicates`	boolean defining if duplicate aligned reads should be skipped (default: FALSE). Option has no effect if preprocessed BAM data was supplied as an input OR duplicate reads were not marked by alignment software.
`gzip`	boolean to compress the report (default: FALSE).
`verbose`	boolean to report progress and timings (default: TRUE).

Details

The function reports cytosine methylation information using BAM file as an input. In contrast to the other currently available software, reads (or read pairs as a single entity) can be thresholded by their average methylation level before counting methylated bases, effectively resulting in hyper-methylated variant epiallele frequency (VEF) being reported instead of beta value. The function's logic is explained below.

Let's suppose we have a BAM file with four reads, all mapped to the "+" strand of chromosome 1, positions 1-16. Assuming the default values for the thresholding parameters (threshold.reads = TRUE, threshold.context = "CG", min.context.sites = 2, min.context.beta = 0.5, max.outofcontext.beta = 0.1), the input and results will look as following:

methylation string	threshold	explained	methylation reported
...Z..x+.h..x..h.	below	min.context.sites < 2 (only one zZ base)	all cytosines unmethylated
...Z..z.h..x..h.	above	pass all criteria	only C4 is methylated
...Z..z.h..X..h.	below	max.outofcontext.beta > 0.1 (1XH / 3xXhH = 0.33)	all cytosines unmethylated
...Z..z.h..z-.h.	below	min.context.beta < 0.5 (1Z / 3zZ = 0.33)	all cytosines unmethylated

Only the second read will satisfy all of the thresholding criteria, leading to the following CX report (given that all reads map to chr1:+:1-16):

rname	strand	pos	context	meth	unmeth	triad
chr1	+	4	CG	1	3	NNN
chr1	+	7	CG	0	3	NNN
chr1	+	9	CHH	0	4	NNN
chr1	+	12	CHG	0	3	NNN
chr1	+	15	CHH	0	4	NNN

With the thresholding disabled (threshold.reads = FALSE) all methylated bases will retain their status, so the CX report will be similar to the reports produced by other methylation callers (such as Bismark or Illumina DRAGEN Bio IT Platform):

rname	strand	pos	context	meth	unmeth	triad
chr1	+	4	CG	4	0	NNN
chr1	+	7	CG	0	3	NNN
chr1	+	9	CHH	0	4	NNN
chr1	+	12	CHG	1	2	NNN
chr1	+	15	CHH	0	4	NNN

Other notes:

Methylation string bases in unknown context ("uU") are simply ignored, which, to the best of our knowledge, is consistent with the behaviour of other tools.

In order to mitigate sequencing errors (leading to rare variations in the methylation context, as in reads 1 and 4 above), the context present in more than 50% of the reads is assumed to be correct, while all bases at the same position but having other methylation context are simply ignored. This allows reports to be prepared without using the reference genome sequence.

The downside of not using the reference genome sequence is the inability to determine the actual sequence of triplet for every base in the cytosine report. For now the sequence is reported as "NNN" and this will stay until such information will be considered as worth adding.

Value

`data.table` object containing cytosine report in Bismark format or NULL if report.file was specified. The report columns are:

- rname – reference sequence name (as in BAM)
- strand – strand
- pos – cytosine position
- context – methylation context
- meth – number of methylated cytosines
- unmeth – number of unmethylated cytosines
- triad – sequence spanning region [<pos>:<pos+2>]. Always equals to "NNN" due to the reference genome-independent reporting

See Also

`preprocessBam` for preloading BAM data, `generateBedReport` for genomic region-based statistics, `generateVcfReport` for evaluating epiallele-SNV associations, `generateBedEcdf` for analysing the distribution of per-read beta values, and ‘epialleleR‘ vignettes for the description of usage and sample data.

Examples

```
capture.bam <- system.file("extdata", "capture.bam", package="epialleleR")

# CpG report with thresholding
cg.report <- generateCytosineReport(capture.bam)

# CX report without thresholding
cx.report <- generateCytosineReport(capture.bam, threshold.reads=FALSE,
             report.context="CX")
```

`generateVcfReport` *generateVcfReport*

Description

This function calculates base frequencies at particular genomic positions and tests their association with the methylation status of the sequencing reads.

Usage

```
generateVcfReport(
  bam,
  vcf,
  vcf.style = NULL,
  bed = NULL,
  report.file = NULL,
  zero.based.bed = FALSE,
  threshold.reads = TRUE,
  threshold.context = c("CG", "CHG", "CHH", "CxG", "CX"),
  min.context.sites = 2,
  min.context.beta = 0.5,
  max.outofcontext.beta = 0.1,
  min.mapq = 0,
  skip.duplicates = FALSE,
  gzip = FALSE,
  verbose = TRUE
)
```

Arguments

`bam`	BAM file location string OR preprocessed output of `preprocessBam` function.
`vcf`	Variant Call Format (VCF) file location string OR a VCF object returned by `readVcf` function. If VCF object is supplied, its seqlevels must match the seqlevels of the BAM file/object used.
`vcf.style`	string for the seqlevels style of the VCF file, if different from BED file/object. Only has effect when ‘vcf‘ parameter points to the VCF file location and ‘bed‘ is not NULL. Possible values: • NULL (the default) – seqlevels in BED file/object and VCF file are the same • "NCBI", "UCSC", ... – valid parameters of `seqlevelsStyle` function
`bed`	Browser Extensible Data (BED) file location string OR object of class `GRanges` holding genomic coordinates for regions of interest. It is used to include only the specific genomic ranges when the VCF file is loaded. This option has no effect when VCF object is supplied as a ‘vcf‘ parameter. The seqlevels of BED file/object must match the seqlevels of the BAM file/object used.

`report.file`	file location string to write the VCF report. If NULL (the default) then report is returned as a `data.table` object.
`zero.based.bed`	boolean defining if BED coordinates are zero based (default: FALSE).
`threshold.reads`	boolean defining if sequence reads should be thresholded before counting bases in reference and variant epialleles (default: TRUE). Disabling thresholding is possible but makes no sense in this context as all the reads will be assigned to the variant epiallele, which will result in Fisher's Exact test p-value of 1 (in columns 'FEp+' and 'FEP-').
`threshold.context`	string defining cytosine methylation context used for thresholding the reads: • "CG" (the default) – within-the-context: CpG cytosines (called as zZ), out-of-context: all the other cytosines (hHxX) • "CHG" – within-the-context: CHG cytosines (xX), out-of-context: hHzZ • "CHH" – within-the-context: CHH cytosines (hH), out-of-context: xXzZ • "CxG" – within-the-context: CG and CHG cytosines (zZxX), out-of-context: CHH cytosines (hH) • "CX" – all cytosines are considered within-the-context, this effectively results in no thresholding This option has no effect when read thresholding is disabled.
`min.context.sites`	non-negative integer for minimum number of cytosines within the 'threshold.context' (default: 2). Reads containing **fewer** within-the-context cytosines are considered completely unmethylated (thus belonging to the reference epiallele). This option has no effect when read thresholding is disabled.
`min.context.beta`	real number in the range [0;1] (default: 0.5). Reads with average beta value for within-the-context cytosines **below** this threshold are considered completely unmethylated (thus belonging to the reference epiallele). This option has no effect when read thresholding is disabled.
`max.outofcontext.beta`	real number in the range [0;1] (default: 0.1). Reads with average beta value for out-of-context cytosines **above** this threshold are considered completely unmethylated (thus belonging to the reference epiallele). This option has no effect when read thresholding is disabled.
`min.mapq`	non-negative integer threshold for minimum read mapping quality (default: 0). Option has no effect if preprocessed BAM data was supplied as an input.
`skip.duplicates`	boolean defining if duplicate aligned reads should be skipped (default: FALSE). Option has no effect if preprocessed BAM data was supplied as an input OR duplicate reads were not marked by alignment software.
`gzip`	boolean to compress the report (default: FALSE).
`verbose`	boolean to report progress and timings (default: TRUE).

Details

Using BAM reads and sequence variation information as an input, ‘generateVcfReport‘ function thresholds the reads (or read pairs as a single entity) according to supplied parameters and calculates the occurrence of **Ref**erence and **Alt**ernative bases within reads, taking into the account DNA strand the read mapped to and average methylation level (epiallele status) of the read.

The information on sequence variation can be supplied as a Variant Call Format (VCF) file location or an object of class VCF, returned by the `readVcf` function call. As whole-genome VCF files can be extremely large, it is strongly advised to use only relevant subset of their data, prefiltering the VCF object manually before calling ‘generateVcfReport‘ or specifying ‘bed‘ parameter when ‘vcf‘ points to the location of such large VCF file. Please note that all the BAM, BED and VCF files must use the same seqlevels (i.e. chromosome names).

After counting, function checks if certain bases occur more often within reads belonging to certain epialleles using Fisher’s Exact test (`fisher.test`) and reports separate p-values for reads mapped to **"+"** (forward) and **"-"** (reverse) DNA strands.

Please note that the final report currently includes only the VCF entries with single-base REF and ALT alleles. Also, there is no filtering by the base quality at the moment, thus the output of ‘generateVcfReport‘ is equivalent to the one of ‘samtools mplieup -Q 0 ...‘, and may result in false SNVs caused by misalignments. These will be fixed in the future.

Value

`data.table` object containing VCF report or NULL if report.file was specified. The report columns are:

- name – variation identifier (e.g. "rs123456789")
- seqnames – reference sequence name
- range – genomic coordinates of the variation
- REF – base at the reference allele
- ALT – base at the alternative allele
- [M|U][+|-][Ref|Alt] – number of **Ref**erence or **Alt**ernative bases that were found at this particular position within **M**ethylated (above threshold) or **U**nmethylated (below threshold) reads that were mapped to **"+"** (forward) or **"-"** (reverse) DNA strand. NA values mean that it is not possible to determine the number of bases due to the bisulfite conversion-related limitations (C->T variants on "+" and G->A on "-" strands)
- SumRef – sum of all **Ref**erence base counts
- SumAlt – sum of all **Alt**ernative base counts
- FEp+ – Fisher’s Exact test p-value for association of a variation with methylation status of the reads that map to the **"+"** (forward) DNA strand. Calculated using following contingency table:

M+Ref	M+Alt
U+Ref	U+Alt

- FEp- – Fisher’s Exact test p-value for association of a variation with methylation status of the reads that map to the **"-"** (reverse) DNA strand. Calculated using following contingency table:

M-Ref	M-Alt
U-Ref	U-Alt

See Also

preprocessBam for preloading BAM data, generateCytosineReport for methylation statistics at the level of individual cytosines, generateBedReport for genomic region-based statistics, generateBedEcdf for analysing the distribution of per-read beta values, and 'epialleleR' vignettes for the description of usage and sample data.

GRanges class for working with genomic ranges, readVcf function for loading VCF data, seqlevelsStyle function for getting or setting the seqlevels style.

Examples

```
capture.bam <- system.file("extdata", "capture.bam", package="epialleleR")
capture.bed <- system.file("extdata", "capture.bed", package="epialleleR")
capture.vcf <- system.file("extdata", "capture.vcf.gz",
                           package="epialleleR")

# VCF report
vcf.report <- generateVcfReport(bam=capture.bam, bed=capture.bed,
                                vcf=capture.vcf)
```

---

preprocessBam *preprocessBam*

Description

This function reads and preprocesses BAM file.

Usage

```
preprocessBam(bam.file, min.mapq = 0, skip.duplicates = FALSE, verbose = TRUE)
```

Arguments

bam.file	BAM file location string.
min.mapq	non-negative integer threshold for minimum read mapping quality (default: 0).
skip.duplicates	boolean defining if duplicate aligned reads should be skipped (default: FALSE). Option has no effect if duplicate reads were not marked by alignment software.
verbose	boolean to report progress and timings (default: TRUE).

Details

The function loads and preprocesses BAM file, saving time when multiple analyses are to be performed on large input files. Currently, Rsamtools package is used to read the data, but this will change in the future with a goal of speeding up this step even further.

This function is also used internally when BAM file location is supplied as an input for other 'epialleleR' methods.

'preprocessBam' automatically determines whether BAM is derived from single-end or paired-end sequencing. When the latter is the case, paired reads are merged so that the overlapping fragments of second read are clipped (because quality of the second read is usually lower than of the first). These merged reads are then processed as a single entity in all 'epialleleR' methods.

Please also note that for all its methods, 'epialleleR' requires methylation call string to be present in a BAM file - i.e. methylation calling must be performed after read mapping/alignment by your software of choice.

Value

`data.table` object containing preprocessed BAM data.

See Also

`generateCytosineReport` for methylation statistics at the level of individual cytosines, `generateBedReport` for genomic region-based statistics, `generateVcfReport` for evaluating epiallele-SNV associations, `generateBedEcdf` for analysing the distribution of per-read beta values, and 'epialleleR' vignettes for the description of usage and sample data.

Sequence Alignment/Map format specifications, duplicate alignments marking by Samtools and Illumina DRAGEN Bio IT Platform.

Examples

```
capture.bam <- system.file("extdata", "capture.bam", package="epialleleR")
bam.data    <- preprocessBam(capture.bam)
```

Index