# Package 'epialleleR'

December 7, 2023

**Title** Fast, Epiallele-Aware Methylation Caller and Reporter

**Version** 1.11.2

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

**Description** Epialleles are specific DNA methylation patterns that are mitotically and/or meiotically inherited. This package calls and reports cytosine methylation as well as frequencies of hypermethylated epialleles at the level of genomic regions or individual cytosines in next-generation sequencing data using binary alignment map (BAM) files as an input. Among other things, this package can also extract methylation patterns and assess allele specificity of methylation.

**SystemRequirements** C++17, GNU make

**NeedsCompilation** yes

**Depends** R (>= 4.1)

**Imports** stats, methods, utils, GenomicRanges, BiocGenerics, GenomeInfoDb, SummarizedExperiment, VariantAnnotation, data.table, Rcpp

**LinkingTo** Rcpp, BH, Rhtslib, zlibbioc

**Suggests** RUnit, knitr, rmarkdown, ggplot2, ggstance, gridExtra

**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** 7.2.3

**VignetteBuilder** knitr

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

**git_branch** devel

**git_last_commit** 077e142

**git_last_commit_date** 2023-11-01

**Repository** Bioconductor 3.19

**Date/Publication** 2023-12-07

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

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

# Table of contents:



---

`callMethylation` *callMethylation*

---

## Description

This function calls cytosine methylation and stores calls in BAM files.

## Usage

```
callMethylation(
  input.bam.file,
  output.bam.file,
  genome,
  nthreads = 1,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `input.bam.file` | input BAM file location string. |
| `output.bam.file` | output BAM file location string. |
| `genome` | reference (genomic) sequences file location string or an output of `preprocessGenome`. |
| `nthreads` | non-negative integer for the number of additional HTSlib threads to be used during file decompression (default: 1). |
| `verbose` | boolean to report progress and timings (default: TRUE). |

## Details

The function makes cytosine methylation calls for input BAM file and writes them in the XM tag of the output BAM file. Calls are made on the basis of reference (e.g., genomic) sequence and observed sequence and cytosine context of reads. Data reading/processing is done by means of HTSlib, therefore it is possible to significantly (>5x) speed up the calling using several (4-8) HTSlib decompression threads.

Methylation calling is only possible if genomic strand the read was aligned to is known. This information is typically stored in XG tag of Bismark/Illumina BAM files, or in YC tag of BWA-meth alignment files. ‘epialleleR‘ is aware of that and will use the whichever tag is available.

The sequence context of cytosines (h/H for "CHH", x/X for "CHG", z/Z for "CG") is determined based on the actual (observed) sequence of the read. E.g., if read "ACGT" was aligned to the forward strand of reference sequence "ACaaGT" with the CIGAR string "2M2D2M" (2 bases match, 2 reference bases are deleted, 2 bases match), then methylation call string will be ".Z.." (in contrast to the reference's one of ".H...."). This makes cytosine calls nearly identical to ones produced by Bismark Bisulfite Read Mapper and Methylation Caller or Illumina DRAGEN Bio IT Platform, however with one important distinction: ‘epialleleR‘ reports sequence context of cytosines followed by unknown bases ("CNN") as "H.." instead of "U.." (unknown; as for example Illumina DRAGEN Bio IT Platform does). Similarly, forward strand context of "CNG" is reported as "X..", forward strand context of "CGN" -> "Z..", reverse strand context of "NNG" -> "..H", reverse strand context of "CNG" -> "..X", reverse strand context of "NCG" -> "..Z". Both lowercase and uppercase ACGTN symbols in reference sequence are allowed and correctly recognised, however all the other symbols (e.g., extended IUPAC symbols, MRSVWYHKDB) within sequences are converted to N.

As a reference sequence, the function expects either location of (preferably ‘bgzip‘ped) FASTA file or an object obtained by `preprocessGenome`. The latter is preferred if methylation calling is performed on multiple BAM files.

The alignment records of the output BAM file will contain additional XM tag with the methylation call string for every mapped read which did not have XM tag available. Besides that, XG tag with reference sequence strand ("CT" or "GA") is added to such reads in case it wasn't present.

Please note that for the purpose of methylation calling, the very same reference genome must be used for both alignment (when BAM is produced) and calling cytosine methylation by `callMethylation` method. Exception is thrown if reference sequence header of BAM file doesn't match reference sequence data provided (this matching is performed on the basis of names and lengths of reference sequences).

## Value

list object with simple statistics of processed ("nrecs") records and calls made ("ncalled"). Even though "ncalled" can be less than "nrecs" (e.g., because not all reads are mapped), all records from the input BAM are written to the output BAM.

## See Also

`preprocessGenome` for preloading reference sequences and ‘epialleleR‘ vignettes for the description of usage and sample data.

Bismark Bisulfite Read Mapper and Methylation Caller, bwa-meth for fast and accurate alignment of long bisulfite-seq reads, or info on Illumina DRAGEN Bio IT Platform.

## Examples

```
callMethylation(
 input.bam.file=system.file("extdata", "test", "dragen-se-unsort-xg.bam", package="epialleleR"),
  output.bam.file=tempfile(pattern="output-", fileext=".bam"),
  genome=system.file("extdata", "test", "reference.fasta.gz", package="epialleleR")
)
```

---

`extractPatterns` *extractPatterns*

---

## Description

This function extracts methylation patterns (epialleles) for a given genomic region of interest.

## Usage

```
extractPatterns(
  bam,
  bed,
  bed.row = 1,
  zero.based.bed = FALSE,
  match.min.overlap = 1,
  extract.context = c("CG", "CHG", "CHH", "CxG", "CX"),
  min.context.freq = 0.01,
  clip.patterns = FALSE,
  strand.offset = c(CG = 1, CHG = 2, CHH = 0, CxG = 0, CX = 0)[extract.context],
  highlight.positions = c(),
  ...,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `bam` | BAM file location string OR preprocessed output of `preprocessBam` function. BAM file alignment records must contain XG tag (strand information for the reference genome) and methylation call string (XM tag). Read more about these and other requirements and BAM preprocessing at `preprocessBam`. |
| `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 style of seqlevels of BED file/object must match the style of seqlevels of the BAM file/object used. The BED/`GRanges` rows are **not** sorted internally. |
| `bed.row` | single non-negative integer specifying what 'bed' region should be included in the output (default: 1). |
| `zero.based.bed` | boolean defining if BED coordinates are zero based (default: FALSE). |

`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).

`extract.context`
: string defining cytosine methylation context used to report:

- "CG" (the default) – CpG cytosines (called as zZ)
- "CHG" – CHG cytosines (xX)
- "CHH" – CHH cytosines (hH)
- "CxG" – CG and CHG cytosines (zZxX)
- "CX" – all cytosines

`min.context.freq`
: real number in the range [0;1] (default: 0.01). Genomic positions that are covered by smaller fraction of patterns (e.g., with erroneous context) won't be included in the report.

`clip.patterns`
: boolean if patterns should not extend over the edge of 'bed' region (default: FALSE).

`strand.offset`
: single non-negative integer specifying the offset of bases at the reverse (-) strand compared to the forward (+) strand. Allows to "merge" genomic positions when methylation is symmetric (in CG and CHG contexts). By default, equals 1 for 'extract.context'=="CG", 2 for "CHG", or 0 otherwise.

`highlight.positions`
: integer vector with genomic positions of bases to include in every overlapping pattern. Allows to visualize the distribution of single-nucleotide variations (SNVs) among methylation patterns. 'highlight.positions' takes precedence if any of these positions overlap with within-the-context positions of methylation pattern.

`...`
: other parameters to pass to the `preprocessBam` function. Options have no effect if preprocessed BAM data was supplied as an input.

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

## Details

The function matches reads (for paired-end sequencing alignment files - read pairs as a single entity) to the genomic region provided in a BED file/GRanges object, extracts methylation statuses of bases within those reads, and returns a data frame which can be used for plotting of DNA methylation patterns.

## Value

`data.table` object containing per-read (pair) base methylation information for the genomic region of interest. The report columns are:

- seqnames – read (pair) reference sequence name
- strand – read (pair) strand
- start – start of the read (pair)
- end – end of the read (pair)

- nbase – number of within-the-context bases for this read (pair)
- beta – beta value of this read (pair), calculated as a ratio of the number of methylated within-the-context bases to the total number of within-the-context bases
- pattern – hex representation of 64-bit FNV-1a hash calculated for all reported base positions and bases in this read (pair). This hash value depends only on included genomic positions and their methylation call string chars (hHxXzZ) or nucleotides (ACGT, for highlighted bases only), thus it is expected to be unique for every methylation pattern, although equal for identical methylation patterns independently on read (pair) start, end, or strand (when correct 'strand.offset' is given)
- ... – columns for each genomic position that hold corresponding methylation call string char, or NA if position is not present in the read (pair)

## 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, generateBedEcdf for analysing the distribution of per-read beta values, 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 get our patterns
patterns <- extractPatterns(bam=amplicon.bam, bed=amplicon.bed, bed.row=3)
nrow(patterns)  # read pairs overlap genomic region of interest

# these are positions of bases
base.positions <- grep("^[0-9]+$", colnames(patterns), value=TRUE)

# let's make a summary table with counts of every pattern
patterns.summary <- patterns[, c(lapply(.SD, unique), .N),
                             by=.(pattern, beta), .SDcols=base.positions]
nrow(patterns.summary)  # unique methylation patterns

# let's melt and plot them
plot.data <- data.table::melt.data.table(patterns.summary,
  measure.vars=base.positions, variable.name="pos", value.name="base")

# continuous positions, nonunique patterns according to their counts
if (require("ggplot2", quietly=TRUE) & require("ggstance", quietly=TRUE)) {
  ggplot(na.omit(plot.data)[N>1],
         aes(x=as.numeric(as.character(pos)), y=factor(N),
             group=pattern, color=factor(base, levels=c("z","Z")))) +
    geom_line(color="grey", position=position_dodgev(height=0.5)) +
    geom_point(position=position_dodgev(height=0.5)) +
    scale_colour_grey(start=0.8, end=0) +
```

```
    theme_light() +
    labs(x="position", y="count", title="epialleles", color="base")
}

# upset-like plot of all patterns, categorical positions, sorted by counts
if (require("ggplot2", quietly=TRUE) & require("gridExtra", quietly=TRUE)){
  grid.arrange(
    ggplot(na.omit(plot.data),
           aes(x=pos, y=reorder(pattern,N),
               color=factor(base, levels=c("z","Z")))) +
      geom_line(color="grey") +
      geom_point() +
      scale_colour_grey(start=0.8, end=0) +
      theme_light() +
      scale_x_discrete(breaks=function(x){x[c(rep(FALSE,5), TRUE)]}) +
      theme(axis.text.y=element_blank(), legend.position="none") +
      labs(x="position", y=NULL, title="epialleles", color="base"),

    ggplot(unique(na.omit(plot.data)[, .(pattern, N, beta)]),
           aes(x=N+0.5, y=reorder(pattern,N), alpha=beta, label=N)) +
      geom_col() +
      geom_text(alpha=0.5, nudge_x=0.2, size=3) +
      scale_x_log10() +
      theme_minimal() +
      theme(axis.text.y=element_blank(), legend.position="none") +
      labs(x="count", y=NULL, title=""),
    ncol=2, widths=c(0.75, 0.25)
  )
}
```

---

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"),
  ...,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| bam | BAM file location string OR preprocessed output of preprocessBam function. BAM file alignment records must contain XG tag (strand information for the reference genome) and methylation call string (XM tag). Read more about these and other requirements and BAM preprocessing at preprocessBam. |
| 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 style of seqlevels of BED file/object must match the style of seqlevels of the BAM file/object used. |
| bed.type | character string for the type of assay that was used to produce sequencing reads:<br>• "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<br>• "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:<br>• "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

`...` other parameters to pass to the `preprocessBam` function. Options have no effect if preprocessed BAM data was supplied as an input.

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

## Details

The function matches reads (for paired-end sequencing alignment files - 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-CpGs-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, `extractPatterns` for exploring methylation patterns, 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 them all
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)
}

# recover default plotting parameters
par(mfrow=c(1,1))
```

---

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,
  ...,
  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,
  ...,
  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,
  ...,
  gzip = FALSE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| bam | BAM file location string OR preprocessed output of preprocessBam function. BAM file alignment records must contain XG tag (strand information for the reference genome) and methylation call string (XM tag). Read more about these and other requirements and BAM preprocessing at preprocessBam. |
| bed | Browser Extensible Data (BED) file location string OR object of class GRanges holding genomic coordinates for regions of interest. The style of seqlevels of BED file/object must be the same as the style of 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.

`...`
: other parameters to pass to the `preprocessBam` function. Options have no effect if preprocessed BAM data was supplied as an input.

`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 (for paired-end sequencing alignment files - 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 (pairs) mapped to the forward ("+") strand
- nreads- – number of reads (pairs) 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, extractPatterns for exploring methylation patterns, 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,
  ...,
  gzip = FALSE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `bam` | BAM file location string OR preprocessed output of `preprocessBam` function. BAM file alignment records must contain XG tag (strand information for the reference genome) and methylation call string (XM tag). Read more about these and other requirements and BAM preprocessing at `preprocessBam`. |
| `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:<br>• "CG" (the default) — within-the-context: CpG cytosines (called as zZ), out-of-context: all the other cytosines (hHxX)<br>• "CHG" — within-the-context: CHG cytosines (xX), out-of-context: hHzZ<br>• "CHH" — within-the-context: CHH cytosines (hH), out-of-context: xXzZ<br>• "CxG" — within-the-context: CG and CHG cytosines (zZxX), out-of-context: CHH cytosines (hH)<br>• "CX" — all cytosines are considered within-the-context, this effectively results in no thresholding<br>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').

`...`
: other parameters to pass to the `preprocessBam` function. Options have no effect if preprocessed BAM data was supplied as an input.

`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 or data as an input. In contrast to the other currently available software, reads (for paired-end sequencing alignment files - read pairs as a single entity) can be thresholded by their average methylation level before counting methylated bases, effectively resulting in hypermethylated 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 (Z at position 4) 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 |
|---|---|---|---|---|---|
| chr1 | + | 4 | CG | 1 | 3 |
| chr1 | + | 7 | CG | 0 | 3 |

| | | | | | |
|---|---|---|---|---|---|
| chr1 | + | 9 | CHH | 0 | 4 |
| chr1 | + | 12 | CHG | 0 | 3 |
| chr1 | + | 15 | CHH | 0 | 4 |

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 |
|---|---|---|---|---|---|
| chr1 | + | 4 | CG | 4 | 0 |
| chr1 | + | 7 | CG | 0 | 3 |
| chr1 | + | 9 | CHH | 0 | 4 |
| chr1 | + | 12 | CHG | 1 | 2 |
| chr1 | + | 15 | CHH | 0 | 4 |

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 the effect of 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. Therefore this sequence is not reported, and this won't change until such information will be considered as worth adding.

## Value

`data.table` object containing cytosine report in Bismark-like 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

## See Also

'values' vignette for a comparison and visualisation of epialleleR output values for various input files. 'epialleleR' vignette for the description of usage and sample data.

`preprocessBam` for preloading BAM data, `generateBedReport` for genomic region-based statistics, `generateVcfReport` for evaluating epiallele-SNV associations, `extractPatterns` for exploring methylation patterns, `generateBedEcdf` for analysing the distribution of per-read beta values.

## 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")
```

---

`generateMhlReport` *generateMhlReport*

---

## Description

This function computes *Linearised* Methylated Haplotype Load ($lMHL$) per genomic position.

## Usage

```
generateMhlReport(
  bam,
  report.file = NULL,
  haplotype.context = c("CG", "CHG", "CHH", "CxG", "CX"),
  max.haplotype.window = 0,
  min.haplotype.length = 0,
  ...,
  gzip = FALSE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `bam` | BAM file location string OR preprocessed output of `preprocessBam` function. BAM file alignment records must contain XG tag (strand information for the reference genome) and methylation call string (XM tag). Read more about these and other requirements and BAM preprocessing at `preprocessBam`. |
| `report.file` | file location string to write the $lMHL$ report. If NULL (the default) then report is returned as a `data.table` object. |
| `haplotype.context` | string for a cytosine context that defines a haplotype:<br>• "CG" (the default) – CpG cytosines only (called as zZ)<br>• "CHG" – CHG cytosines only (xX)<br>• "CHH" – CHH cytosines only (hH)<br>• "CxG" – CG and CHG cytosines (zZxX)<br>• "CX" – all cytosines; this, as well as the other non-CG contexts, may have little sense but still included for consistency |

If $lMHL$ calculations are needed for all three possible cytosine contexts *independently*, one has to run this function for each required 'haplotype.context' separately, because 'haplotype.context'=="CX" assumes that *any* cytosine context is allowed within the same haplotype. This behaviour may change in the future.

`max.haplotype.window`
: non-negative integer for maximum value of $L'$ in $lMHL$ formula. When 0 (the default), calculations are performed for the full haplotype length ($L' = L$, although the maximum value is currently limited to 65535). Having no length restrictions make sense for short-read sequencing when the length of the read is comparable to the length of a typical methylated block, the depth of coverage is high, and the lengths of all reads are roughly equal. However, calculations using non-restricted haplotype length are meaningless for long-read sequencing — when the same read may cover a number of regions with very different methylation properties, and reads themselves can be of a very different length. In the latter case it is advised to limit the 'max.haplotype.window' to a number of cytosines in a typical hypermethylated region. For thorough explanation and more examples, see Details section and vignette.

`min.haplotype.length`
: non-negative integer for minimum length of a haplotype (default: 0 will include haplotypes of any length). When 'min.haplotype.length'>0, reads (read pairs) with fewer than 'min.haplotype.length' cytosines within the 'haplotype.context' are skipped.

`...`
: other parameters to pass to the `preprocessBam` function. Options have no effect if preprocessed BAM data was supplied as an input.

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

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

## Details

The function reports *Linearised* Methylated Haplotype Load ($lMHL$) at the level of individual cytosines using BAM file location or preprocessed data as an input. Function uses the following formula:

$$lMHL = \frac{\sum_{i=1}^{L'} w_i \times MH_i}{\sum_{i=1}^{L'} w_i \times H_i}$$

where $L'$ is the length of a calculation window (e.g., number of CpGs; $L' \leq L$, where $L$ is the length of a haplotype covering current genomic position), $MH_i$ is a number of fully successive methylated stretches with $i$ loci within a methylated stretch that overlaps current genomic position, $H_i$ is a number of fully successive stretches with $i$ loci, $w_i$ is a weight for $i$-locus haplotype ($w_i = i$).

This formula is a modification of the original Methylated Haplotype Load (MHL) formula that was first described by Guo et al., 2017 (doi: 10.1038/ng.3805):

$$MHL = \frac{\sum_{i=1}^{L} w_i \times \frac{MH_i}{H_i}}{\sum_{i=1}^{L} w_i}$$

where $L$ is the length of a longest haplotype covering current genomic position, $\frac{MH_i}{H_i} = P(MH_i)$ is the fraction of fully successive methylated stretches with $i$ loci, $w_i$ is a weight for $i$-locus haplotype ($w_i = i$).

The modifications to original formula are made in order to:

- **provide granularity of values** — the original MHL formula gives the same MHL value for every cytosine of a partially methylated haplotype (e.g., MHL=0.358 for each cytosine within a read with methylation call string "zZZZ"). In contrast, $lMHL$==0 for the non-methylated cytosines (e.g., $lMHL$==c(0, 0.5, 0.5, 0.5) for cytosines within a read with methylation call string "zZZZ").
- **enable calculations for long-read sequencing alignments** — $lMHL$ calculation window can be limited to a particular number of cytosines. This allows to use the formula for very long haplotypes as well as to compare values for sequencing data of varying read length.
- **reduce the complexity of MHL calculation** for data of high breadth and depth — $lMHL$ values for all genomic positions can be calculated using a single pass (cycling through reads just once) as the linearised calculations of numerator and denominator for $lMHL$ do not require prior knowledge on how many reads cover a particular position. This is achieved by moving $H_i$ multiplier to the denominator of the $lMHL$ formula.

These modifications make $lMHL$ calculation similar though *non-equivalent* to the original MHL. However, the most important property of MHL — emphasis on hypermethylated blocks — is retained. And in return, $lMHL$ gets better applicability for analysis of sequencing data of varying depth and read length.

Other notes on function's behaviour:

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.

Cytosine 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.

## Value

`data.table` object containing $lMHL$ report 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
- coverage – number of reads (read pairs) that include this position
- length – average length of a haplotype, i.e., average number of cytosines within 'haplotype.context' for reads (read pairs) that include this position
- lmhl – $lMHL$ value

## See Also

'values' vignette for a comparison and visualisation of epialleleR output values for various input files. 'epialleleR' vignette for the description of usage and sample data.

preprocessBam for preloading BAM data, generateCytosineReport for other methylation statistics at the level of individual cytosines, generateBedReport for genomic region-based statistics, generateVcfReport for evaluating epiallele-SNV associations, extractPatterns for exploring methylation patterns, generateBedEcdf for analysing the distribution of per-read beta values.

## Examples

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

# lMHL report
mhl.report <- generateMhlReport(capture.bam)

# lMHL report with a `max.haplotype.window` of 1 is identical to a
# conventional cytosine report (or nearly identical when sequencing errors
# are present)
mhl.report <- generateMhlReport(capture.bam, max.haplotype.window=1)
cg.report  <- generateCytosineReport(capture.bam, threshold.reads=FALSE)
identical(
  mhl.report[, .(rname, strand, pos, context, value=lmhl)],
  cg.report[ , .(rname, strand, pos, context, value=meth/(meth+unmeth))]
)
```

---

# 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,
```

```
  ...,
  gzip = FALSE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| bam | BAM file location string OR preprocessed output of preprocessBam function. BAM file alignment records must contain XG tag (strand information for the reference genome) and methylation call string (XM tag). Read more about these and other requirements and BAM preprocessing at preprocessBam. |
| vcf | Variant Call Format (VCF) file location string OR a VCF object returned by readVcf function. If VCF object is supplied, the style of its seqlevels must match the style of 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:<br>• NULL (the default) – seqlevels in BED file/object and VCF file are the same<br>• "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 style of seqlevels of BED file/object must match the style of 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:<br>• "CG" (the default) – within-the-context: CpG cytosines (called as zZ), out-of-context: all the other cytosines (hHxX)<br>• "CHG" – within-the-context: CHG cytosines (xX), out-of-context: hHzZ<br>• "CHH" – within-the-context: CHH cytosines (hH), out-of-context: xXzZ<br>• "CxG" – within-the-context: CG and CHG cytosines (zZxX), out-of-context: CHH cytosines (hH)<br>• "CX" – all cytosines are considered within-the-context, this effectively results in no thresholding<br>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.

`...`
: other parameters to pass to the `preprocessBam` function. Options have no effect if preprocessed BAM data was supplied as an input.

`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 (for paired-end sequencing alignment files - 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 style for seqlevels (i.e. chromosome names).

After counting, function checks if certain bases occur more often within reads belonging to certain epialleles using Fisher Exact test (HTSlib's own implementation) 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, the default ('min.baseq=0') output of 'generateVcfReport' is equivalent to the one of 'samtools mplieup -Q 0 ...', and therefore may result in false SNVs caused by misalignments. Remember to increase 'min.baseq' ('samtools mplieup -Q' default value is 13) to obtain higher-quality results.

## 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 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 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, `extractPatterns` for exploring methylation patterns, `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,
  paired = NULL,
  min.mapq = 0,
  min.baseq = 0,
  skip.duplicates = FALSE,
  nthreads = 1,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| bam.file | BAM file location string. |
| paired | boolean for expected alignment endness: 'TRUE' for paired-end, 'FALSE' for single-end, or 'NULL' for auto detect (the default). |
| min.mapq | non-negative integer threshold for minimum read mapping quality (default: 0). |
| min.baseq | non-negative integer threshold for minimum nucleotide base 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. |
| nthreads | non-negative integer for the number of additional HTSlib threads to be used during BAM file decompression (default: 1). Two threads (and usually no more than two) make sense for the files larger than 100 MB. |
| 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, HTSlib is used to read the data, therefore it is possible to speed up the loading by means of HTSlib decompression threads.

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

Please note that for BAM preprocessing as well as all its reporting methods, 'epialleleR' requires genomic strand (XG tag) and a methylation call string (XM tag) to be present in a BAM file - i.e., methylation calling must be performed after read mapping/alignment by your software of choice. It

is the case for BAM files produced by Bismark Bisulfite Read Mapper and Methylation Caller, Illumina DRAGEN, Illumina Cloud analysis solutions, as well as contemporary Illumina sequencing instruments with on-board read mapping/alignment (NextSeq 1000/2000, NovaSeq X), therefore such files can be analysed without additional steps. For alignments produced by other tools, e.g., BWA-meth, methylation calling must be performed prior to BAM loading / reporting, by means of `callMethylation`.

'preprocessBam' always tests if BAM file is paired- or single-ended and has all necessary tags (XM/XG) available. It is recommended to use 'verbose' processing and check messages for correct identification of alignment endness. Otherwise, if the 'paired' parameter is set explicitly, exception is thrown when expected endness differs from the auto detected one.

During preprocessing of paired-end alignments, paired reads are merged according to their base quality: nucleotide base with the highest value in the QUAL string is taken, unless its quality is less than 'min.baseq', which results in no information for that particular position ("-"/"N"). These merged reads are then processed as a single entity in all 'epialleleR' methods. Due to merging, overlapping bases in read pairs are counted only once, and the base with the highest quality is taken.

During preprocessing of single-end alignments, no read merging is performed. Only bases with quality of at least 'min.baseq' are considered. Lower base quality results in no information for that particular position ("-"/"N").

It is also a requirement currently that paired-end BAM file must be sorted by QNAME instead of genomic location (i.e., "unsorted") to perform merging of paired-end reads. Error message is shown if it is sorted by genomic location, in this case please sort it by QNAME using 'samtools sort -n -o out.bam in.bam'.

## Value

`data.table` object containing preprocessed BAM data.

## See Also

`preprocessGenome` for preloading reference sequences and `callMethylation` for methylation calling.

`generateCytosineReport` for methylation statistics at the level of individual cytosines, `generateBedReport` for genomic region-based statistics, `generateVcfReport` for evaluating epiallele-SNV associations, `extractPatterns` for exploring methylation patterns, `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)
```

preprocessGenome *preprocessGenome*

## Description

This function reads and preprocesses (optionally 'bgzip'ped) FASTA file with reference sequences.

## Usage

```
preprocessGenome(genome.file, nthreads = 1, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| genome.file | reference (genomic) sequences file location string. |
| nthreads | non-negative integer for the number of additional HTSlib threads to be used during file decompression (default: 1). |
| verbose | boolean to report progress and timings (default: TRUE). |

## Details

The function loads and preprocesses reference (genomic) sequences, saving time when methylation calling needs to be performed on multiple BAM files. Currently, reading the data is done by means of HTSlib, therefore it is possible to speed up the loading by means of HTSlib decompression threads when FASTA file is compressed by 'bgzip'.

This function is also called internally when file location is supplied as an input for callMethylation method.

'preprocessGenome' checks if index file is present, and if not, creates it automatically. It is possible and recommended to use compressed FASTA file as an input, but the file must be compressed by 'bgzip' (part of samtools/HTSlib). When FASTA file is compressed, faster loading can be achieved using (typically one) additional HTSlib decompression thread.

During loading, both lowercase and uppercase ACGTN symbols are allowed and correctly recognised, however all the other symbols (e.g., extended IUPAC symbols, MRSVWYHKDB) within sequences are converted to N.

Please also note that for the purpose of methylation calling, the very same reference genome must be used for both alignment (when BAM is produced) and calling cytosine methylation by callMethylation method.

## Value

list object containing preprocessed reference sequence data.

## See Also

callMethylation for methylation calling, and 'epialleleR' vignettes for the description of usage and sample data.

Block compression/decompression utility bgzip.

## Examples

```
genome.file <- system.file("extdata", "test", "reference.fasta.gz", package="epialleleR")
genome.data <- preprocessGenome(genome.file)
```

---

`simulateBam` *simulateBam*

---

## Description

This function creates sample BAM files given mandatory and optional BAM fields.

## Usage

```
simulateBam(
  output.bam.file = NULL,
  qname = NULL,
  flag = NULL,
  rname = NULL,
  pos = NULL,
  mapq = NULL,
  cigar = NULL,
  rnext = NULL,
  pnext = NULL,
  tlen = NULL,
  seq = NULL,
  qual = NULL,
  ...,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `output.bam.file` | output BAM file location string. If NULL (default), records are not written to BAM but returned as a `data.table` object for review. |
| `qname` | character vector of query names. When default (NULL), names like "q0001".."qnnnn" will be assigned. |
| `flag` | integer vector of bitwise flags (a combination of the BAM_F* constants). When default (NULL), zero (i.e., unique, valid, single-end, aligned read) is assigned for every record. |
| `rname` | character vector of chromosome (reference) names. When default (NULL), "chrS" is assigned for every record. |
| `pos` | integer vector of 1-based leftmost coordinates of the queries. When default (NULL), 1 is assigned for every record. |
| `mapq` | integer vector of mapping qualities. When default (NULL), 60 is assigned for every record. |

| | |
|---|---|
| cigar | character vector of CIGAR strings. When default (NULL), "lM" is assigned for every record, where 'l' is the length of the query ('seq'). |
| rnext | character vector of chromosome (reference) names for next read in template. When default (NULL), "chrS" is assigned for every record. |
| pnext | integer vector of 1-based leftmost coordinates of next read in template. When default (NULL), 1 is assigned for every record. |
| tlen | integer vector of observed template lengths. When default (NULL), the length of the corresponding query ('seq') is assigned for every record. |
| seq | character vector of query sequences. When default (NULL), random sequence is assigned. The lengths of these random sequences equal to the lengths of methylation call strings from the 'XM' optional parameter (if supplied), or to the 'tlen' parameter (if defined). If none of these parameters is supplied, length of every 'seq' will equal 10. |
| qual | query sequence quality strings (ASCII of base QUALity plus 33). When default (NULL), quality of every base is assigned to "F" (QUALity of 47 + 33). The lengths of these quality strings equal to the length of the corresponding query sequences ('seq') for every record. |
| ... | optional tags to add to the records, in the form 'tag=value'. Can be either integer vector (e.g., for "NM" tag), or character vector (e.g., "XM" tag for methylation call string, "XG"/"YC" tag for reference strand read was aligned to). |
| verbose | boolean to report progress and timings (default: TRUE). |

## Details

The function creates sample alignment records and saves them in BAM file. Output can be used to test epialleleR methods as well as other tools for methylation analysis. This method can significantly simplify calculation of methylation metrics on example data (beta, VEF, and lMHL values of epialleleR; methylation heterogeneity metrics of other tools).

The number of records written will be equal to the largest length of any supplied (nondefault) parameter or 1 if no parameters were supplied. If lengths of supplied parameters differ, shorter vectors will be recycled (a whole number of times or with remainder if necessary).

Please note that function performs almost no validity checks for supplied fields. In particular, be extra careful constructing paired-end BAM alignments, and if necessary use 'samtools' to perform validity check or manual editing after BAM->SAM conversion.

## Value

number of BAM records written (if 'output.bam.file' is not NULL) or `data.table` object containing final records prepared for writing. NB: this object has 0-based coordinates and numerically encoded reference names.

## See Also

`generateCytosineReport` and `generateMhlReport` for methylation reports at the level of individual cytosines, as well as 'epialleleR' vignettes for the description of usage and sample data.

Samtools for viewing BAM files. SAMv1 file format specifications. Specifications of optional SAM tags. metheor for ultrafast DNA methylation heterogeneity calculation from bisulfite alignments.

## Examples

```
out.bam <- tempfile(pattern="simulated", fileext=".bam")
simulateBam(
  output.bam.file=out.bam,
  pos=c(1, 2),
  XM=c("ZZZzzZZZ", "ZZzzzzZZ"),
  XG=c("CT", "AG")
)
generateCytosineReport(out.bam, threshold.reads=FALSE)
```

# Index