

Package: SingleMoleculeGenomicsIO (via r-universe)

October 26, 2026

Title Tools for reading single molecule genomics data

Version 0.1.9

Description Provides tools for reading of single-molecule genomics data from different file formats in R. The package includes functions for reading collapsed or read-level data from modBAM, bedMethyl and `modkit extract` tabular files, as well as C-to-T conversion based BAM files, as `SummarizedExperiment` objects.

License MIT + file LICENSE

URL <https://github.com/fmicompbio/SingleMoleculeGenomicsIO>,
<https://fmicompbio.github.io/SingleMoleculeGenomicsIO>

BugReports <https://github.com/fmicompbio/SingleMoleculeGenomicsIO/issues>

Depends R (>= 4.1.0)

Imports BiocGenerics, BiocParallel, Biostrings, BSgenome, cli, data.table, dplyr, GenomicRanges, ggplot2, IRanges, methods, Rcpp, rlang, S4Vectors, Seqinfo, SparseArray, SummarizedExperiment, tibble, tidyr, utils

Suggests BiocIO, BiocManager, knitr, MatrixGenerics, R.utils, rmarkdown, Rsamtools, sessioninfo, SimpleUpset, testthat (>= 3.0.0), withr

LinkingTo cli, Rcpp, Rhtslib

VignetteBuilder knitr

biocViews Epigenetics, LongRead

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

SystemRequirements GNU make

Config/roxygen2/version 8.1.1

Config/pak/sysreqs

make libbz2-dev libicu-dev liblzma-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository <https://fmicompbio.r-universe.dev>

Date/Publication 2026-10-19 07:51:20 UTC

RemoteUrl <https://github.com/fmicompbio/SingleMoleculeGenomicsIO>

RemoteRef HEAD

RemoteSha 8c2db76b5182cc8223fff51c362662f1778f5cb5

Contents

addReadLevelAssay	3
addSeqContext	4
annotateReadSegments	5
calcReadStats	6
checkSEValidity	9
countAlignmentsInBam	10
countMismatchStatePairs	11
countStatePairs	13
estimateNoise	15
estimateSNR	16
expandSEToBaseSpace	17
extractColDataColumns	18
extractSeqContext	19
filterPositions	21
filterReads	22
filterReadsModBam	25
flattenReadLevelAssay	27
getBaseCoverageForBam	29
getReadLevelAssayNames	30
getReadLevelColDataNames	31
getReadNamesBySample	31
plotReadStats	32
readBedMethyl	33
readMismatchBam	35
readModBam	38
readModkitExtract	41
refargToDNAStrngSet	43
regionStringToGRanges	44
regroupReads	45
sampleEntropy	46
subsetReads	47

Index	50
--------------	-----------

addReadLevelAssay	<i>Add a read-level assay to a SummarizedExperiment object</i>
-------------------	--

Description

Add a read-level assay to a SummarizedExperiment object

Usage

```
addReadLevelAssay(
  se,
  assayDF,
  assayName,
  replaceExisting = FALSE,
  verbose = FALSE
)
```

Arguments

se	A SummarizedExperiment object.
assayDF	A DataFrame with the same dimensionality as se. Each column should be a read-level NaMatrix for a single sample.
assayName	A character scalar giving the name of the new assay.
replaceExisting	A logical scalar indicating whether to replace an existing assay with the same name or not.
verbose	Logical scalar. If TRUE, report on progress.

Value

A [SummarizedExperiment](#) object with the new read-level assay added.

Author(s)

Charlotte Soneson

Examples

```
library(SummarizedExperiment)
library(S4Vectors)
modbamfile <- system.file("extdata", "6mA_1_10reads.bam",
  package = "SingleMoleculeGenomicsIO")
se <- readModBam(bamfiles = modbamfile, regions = "chr1:6940000-6955000",
  modbase = "a", verbose = TRUE,
  BPPARAM = BiocParallel::SerialParam())
# duplicate the 'mod_prob' assay into a new assay named 'new_assay'
se <- addReadLevelAssay(se, assayDF = assay(se, "mod_prob"),
```

```

                                assayName = "new_assay")
assayNames(se)
metadata(se)$readLevelData

```

addSeqContext	<i>Add sequence context around positions of interest to a SummarizedExperiment</i>
---------------	--

Description

Convenience function to extract sequence context around positions of interest (the [rowRanges](#) of a [RangedSummarizedExperiment](#)) and add them to the [SummarizedExperiment](#)'s row data (`rowData(se)$sequenceContext`). The extracted sequences will correspond to the regions defined as `resize(rowRanges(x), width = sequenceContextWidth, fix = "center")`. Sequence contexts are extracted using [extractSeqContext](#).

Usage

```
addSeqContext(x, sequenceContextWidth, sequenceReference)
```

Arguments

<code>x</code>	A RangedSummarizedExperiment .
<code>sequenceContextWidth</code>	A numeric scalar giving the width of the sequence context to be extracted from the reference (<code>sequenceReference</code> argument). This must be an odd number so that the sequence can be centered on the modified base. If <code>sequenceContextWidth = 0</code> (the default), no sequence context will be extracted.
<code>sequenceReference</code>	A BSgenome object, or a character scalar giving the path to a fasta formatted file with reference sequences, or a DNAStringSet object. The sequence context (see <code>sequenceContextWidth</code> argument) will be extracted from these sequences.

Value

A [RangedSummarizedExperiment](#) object with sequence contexts added as a [DNAStringSet](#) object to `rowData(x)$sequenceContext`.

Author(s)

Michael Stadler

See Also

[extractSeqContext](#)

Examples

```
# load package
library(SummarizedExperiment)

# file with sequence in fasta format of length 6957060
reffile <- system.file("extdata", "reference.fa.gz",
                      package = "SingleMoleculeGenomicsIO")

# define some regions at the end of the reference sequence
se <- SummarizedExperiment(
  assays = matrix(1:3, ncol=1),
  rowRanges = GRanges(
    "chr1", IRanges(start = 6957060 - c(4, 2, 0),
                    width = 1, names = c("a", "b", "c")),
    strand = "-")

# add sequence context (note the padding with N's)
rowRanges(se)
se <- addSeqContext(se, 7, reffile)
rowRanges(se)
```

annotateReadSegments *Add read segment annotation*

Description

Add segments for individual reads to the colData of a SummarizedExperiment object with read-level data.

Usage

```
annotateReadSegments(se, irlList, name)
```

Arguments

se	A SummarizedExperiment object.
irlList	A list of IRangesList objects. Each list element corresponds to a sample (a column in se), with the ranges in an IRangesList element giving the segments of an individual read (coordinates refer to the genome).
name	A character scalar giving the name of the colData(se) column, in which the annotated segments should be stored. If name already exists in colData(se), it will be overwritten.

Value

A SummarizedExperiment object corresponding to se, with annotated segments added to colData(se)[[name]]

Examples

```
library(IRanges)
modbamfile <- system.file("extdata", "6mA_1_10reads.bam",
                           package = "SingleMoleculeGenomicsIO")
se <- readModBam(bamfiles = modbamfile, regions = "chr1:6935400-6936300",
                 modbase = "a", verbose = FALSE,
                 BPPARAM = BiocParallel::SerialParam())

segs <- list(s1 = IRangesList(
  "s1-d52a5f6a-a60a-4f85-913e-eada84bfbfb9" = IRanges(
    start = 6925834, width = 140),
  "s1-6cf74134-e550-4c02-bd2b-91385422ee25" = IRanges(
    start = c(6926000, 6926200), width = 140)))

se <- annotateReadSegments(se, segs, "nucl")
se$nucl
lengths(se$nucl$s1)
```

calcReadStats	<i>Calculate or add summary statistics for read-level base modification data</i>
---------------	--

Description

calcReadStats calculates various per-read summary statistics on modification probabilities or calls from a [SummarizedExperiment](#) object with genomic positions in rows and samples in columns. addReadStats adds them to the colData under name. See details for more information on the statistics that are calculated.

Usage

```
calcReadStats(
  se,
  assayName = "mod_prob",
  stats = NULL,
  regions = NULL,
  sequenceContext = NULL,
  minNobsPpos = 0,
  minNobsPread = 0,
  LowConf = 0.7,
  LagRange = c(12, 64),
  EntrControl = NULL,
  BPPARAM = MulticoreParam(4L, RNGseed = 42L),
  verbose = FALSE
)

addReadStats(se, ..., name = "QC")
```

Arguments

se	A RangedSummarizedExperiment object with assay assayName typically returned by readModkitExtract or readModBam .
assayName	A character scalar specifying the assay of se containing the read-level data to be summarized. Typically, this assay contains modification probabilities.
stats	Character vector specifying which statistics to calculate. When set to NULL all available statistics except the sample entropy are calculated. See details for available read statistics.
regions	A GRanges object limiting the positions included in the calculations to the ones overlapping the corresponding genomic regions. Alternatively, regions can be specified as a character vector (e.g. "chr1:1200-1300") that can be coerced into a GRanges object.
sequenceContext	A character vector with sequence context(s) to include in the calculations. Only positions that match one of the provided sequence contexts will be included. Sequence contexts can be provided using IUPAC redundancy codes. The sequence contexts of modified bases are obtained from <code>rowData(se)\$sequenceContext</code> and thus requires that se contains the appropriate information, for example by setting the <code>sequenceContextWidth</code> and <code>sequenceReference</code> arguments of readModkitExtract when it was generated, or by adding it using addSeqContext .
minNobsPpos	A numeric scalar value ≥ 1 indicating the minimum coverage on individual positions for them to be included in the calculations. In high coverage data this is an effective filter for removing spurious modbases, typically the result of erroneous basecalling.
minNobsPread	A numeric scalar with the minimum number of observed modifiable bases per read for it to be included in the calculations. NA values are returned for the reads that do not pass this threshold.
LowConf	A numeric scalar with the minimum call confidence below which calls are considered "low confidence".
LagRange	A numeric vector of two values (minimum and maximum) defining the range of lags for the calculation of autocorrelation and partial autocorrelation (see details section).
EntrControl	Optional named list with elements <code>m</code> , <code>r</code> , <code>maxStarts</code> , <code>nThreads</code> to control Sample Entropy calculation. Missing elements use defaults: <code>m=2L</code> , <code>r=0.2</code> , <code>maxStarts=1000</code> , <code>nThreads=1L</code> . See also sampleEntropy
BPPARAM	A BiocParallelParam object that controls the number of parallel CPU threads to use for calculating read statistics. The default value is (MulticoreParam (4L, <code>RNGseed = 42L</code>)).
verbose	If TRUE, report on progress.
...	For <code>addReadStats</code> only: Additional arguments passed on to <code>calcReadStats</code> .
name	For <code>addReadStats</code> only: A character scalar specifying the name to be used to store the result in the <code>colData</code> of the output.

Details

calcReadStats calculates a collection of location/scatter statistics and information theoretic/signal-processing metrics for the modification probability, confidence or modification call value vectors across individual reads (data in assay assayName). Only bases matching the criteria given byregions, sequenceContext, minNobsPpos and minNobsPread are included in the calculations. The values of these filtering parameters are stored in the attribute of the output.

stats selects the summaries to be calculated. Currently available values are:

MeanModProb : Mean modification probability across the read.

FracMod : Fraction of confidently called bases (either modified or unmodified), that are modified. By default, all bases are considered confidently called.

MeanConf : Mean call confidence across the read.

MeanConfUnm : Mean call confidence confined to unmodified bases (modifiable bases with modification probability < 0.5).

MeanConfMod : Mean call confidence confined to modified bases (modifiable bases with modification probability >= 0.5).

FracLowConf : Fraction of modifiable bases called with low confidence (call confidence < LowConf).

IQRModProb : Interquartile range of modification probabilities across the read.

sdModProb : Standard deviation of modification probabilities across the read.

SEntrModProb : Sample entropy of the modification probability signal. Sample entropy is a metric assessing the complexity of one-dimensional physiological signals. The higher the sample entropy, the more irregular, unpredictable and therefore complex the signal. See [wikipedia:Sample_entropy](#) for more details.

ACModProb : Autocorrelation of the modification probability values for lags in the range LagRange. This range typically covers the signal of nucleosome periodicity.

PACModProb : Partial autocorrelation of the modification probability values for lags in the range LagRange. This range typically covers the signal of nucleosome periodicity.

NoiseVar : raw **Read Noise variance** estimated as $0.5 \text{Var}(\Delta x)$, where Δx are lag-1 differences that may skip up to k consecutive NAs. A floor is applied: $\text{noise} = \max(\text{rawNoise}, b_0 + b_1 \bar{x})$.

SignalVar : **Read Signal variance** $\max(\text{totalVar} - \text{NoiseVar}, \varepsilon)$.

SNR : **Read Signal-to-Noise Ratio** $\log_2(\text{SignalVar}/\text{NoiseVar})$.

When SNR-related statistics are requested, a robust noise floor $b_0 + b_1 * \text{mean}(x)$ is fitted per sample (see *NoiseVar*). The fitted coefficients are stored in the result metadata (see below). If SNR-related statistics are not computed, NA/NA placeholders are stored for uniformity.

Value

For calcReadStats, a SimpleList object with summary statistics for the samples (columns) in se.

For addReadStats, a SummarizedExperiment object with summary statistics added to the name column of colData.

Author(s)

Panagiotis Papapasaikas, Charlotte Soneson, Michael Stadler

Examples

```
# load example data
library(SummarizedExperiment)
modbamfile <- system.file("extdata", "6mA_1_10reads.bam",
                          package = "SingleMoleculeGenomicsIO")
se <- readModBam(bamfile = modbamfile, regions = "chr1:6940000-6955000",
               modbase = "a", verbose = TRUE,
               BPPARAM = BiocParallel::SerialParam())

readStats <- calcReadStats(se, BPPARAM = BiocParallel::SerialParam())
readStats$s1

se_withReadStats <- addReadStats(se, name = "QC",
                                BPPARAM = BiocParallel::SerialParam())
se_withReadStats$QC$s1
metadata(se_withReadStats$QC$s1)
```

checkSEValidity

*Check internal consistency of SummarizedExperiment object***Description**

All assays with read-level data must have the same number and order of the reads, which must also agree with the order in `se$QC` if that exists. All assays must have the same column names, which must also agree with the column names of the object, and the sample column in the `colData`.

Usage

```
checkSEValidity(se, verbose = FALSE)
```

Arguments

`se` A SummarizedExperiment object.

`verbose` A logical scalar. If TRUE, report on progress.

Value

Silently returns NULL. If the object is not valid, an error will be raised.

Author(s)

Charlotte Soneson

Examples

```
library(GenomicRanges)
modbamfiles <- system.file("extdata", c("6mA_1_10reads.bam",
                                         "6mA_2_10reads.bam"),
                           package = "SingleMoleculeGenomicsIO")
se <- readModBam(bamfiles = modbamfiles, regions = "chr1:6940000-6955000",
                modbase = "a", verbose = FALSE,
                variantPositions = GPos(seqnames = "chr1",
                                         pos = c(6940000, 6940500)),
                BPPARAM = BiocParallel::SerialParam())
checkSEValidity(se)
```

countAlignmentsInBam	<i>Count alignments in an indexed bam file</i>
----------------------	--

Description

Report the number of mapped and unmapped alignments in a sorted and indexed bam file.

Usage

```
countAlignmentsInBam(bamfile)
```

Arguments

bamfile	Character scalar giving the name of the bam file to count alignments in. The file must have an associated index.
---------	--

Value

A named list with elements "mapped" and "unmapped", where the "mapped" element is a named vector with the number of alignments per chromosome, and the "unmapped" element is a numeric scalar giving the number of unmapped reads in the bam file.

Examples

```
countAlignmentsInBam(system.file("extdata/6mA_1_10reads.bam",
                                  package = "SingleMoleculeGenomicsIO"))
```

countMismatchStatePairs

Count pairs of modified bases by distance and modification state for bam files where modifications are indicated by sequence mismatches

Description

For all pairs of bases with modification calls in a read (pair), tabulate the number of pairs at a given distance with a given modification state.

Usage

```
countMismatchStatePairs(
  bamfile,
  bamFormat = "QuasR",
  regions = ".",
  sequenceContext = "GCH",
  nAlnsToSample = 0,
  seqnamesToSampleFrom = character(0),
  readBaseUnmod = "T",
  readBaseMod = "C",
  windowSize = 200,
  minMapQ = 0,
  minAlignedLength = 0,
  seqinfo = NULL,
  sequenceReference = NULL,
  BPPARAM = MulticoreParam(4L),
  verbose = FALSE
)
```

Arguments

bamfile	Character scalar giving the path to a BAM file, containing alignments with specific (e.g., C-to-T) mismatches. The bamfile must have an index.
bamFormat	A character scalar giving the format of BAM files. Currently supported are BAM files that have been created with "QuasR" (using qAlign in bisulfite mode) or "Bismark".
regions	A GRanges object specifying which genomic regions to extract the reads from. Alternatively, regions can be specified as a character vector (e.g. "chr1:1200-1300", "chr2:-6000", "chr1:10-", "chrM" or ".").
sequenceContext	A character scalar with an odd number of characters, specifying the genomic context on the plus strand, for which to report (mis-)matches. The string may contain IUPAC ambiguity codes, e.g. sequenceContext="GCH" would correspond to all GpC dinucleotides, excluding C's that are in CpG dinucleotides. The call will be made by comparing the genomic base in the middle of sequenceContext

	to the aligned base in the read and interpreted according to <code>readBaseUnmod</code> and <code>readBaseMod</code> .
<code>nAlnsToSample</code>	A numeric scalar. If non-zero, <code>regions</code> is ignored and approximately <code>nAlnsToSample</code> randomly selected alignments on <code>seqnamesToSampleFrom</code> are read from the bamfile. If bamfile contains paired alignments, they are included or excluded as pairs. Alignments are counted individually towards <code>nAlnsToSample</code> , thus for paired-end data this parameter has to be set to twice the number of pairs to be sampled. In order to make the results reproducible, make sure to set the random number seed using <code>set.seed</code> . Please note that secondary and supplementary alignments in bamfiles contribute to the total number of alignments but will not be sampled, thus the number of used alignments may be lower than <code>nAlnsToSample</code> .
<code>seqnamesToSampleFrom</code>	A character vector with one or several sequence names (chromosomes) from which to sample alignments from (only used if <code>nAlnsToSample</code> is greater than zero). If set to <code>character(0)</code> , all chromosomes present in any of the bam files will be used.
<code>readBaseUnmod</code> , <code>readBaseMod</code>	Character scalars defining the read bases that are interpreted as unmodified or modified, respectively, when aligned to the middle base of <code>sequenceContext</code> .
<code>windowSize</code>	Numeric scalar giving the maximum window size covering pairs of modified bases to consider in pair-counting mode. A window size of 1 corresponds to a single base, a size of 2 to directly adjacent bases, etc.
<code>minMapQ</code>	Numeric scalar giving the minimal mapping quality to include alignments in pair-counting mode.
<code>minAlignedLength</code>	Numeric scalar giving the minimal alignment length to include alignments in pair-counting mode.
<code>seqinfo</code>	NULL or a Seqinfo object containing information about the set of genomic sequences (chromosomes). Alternatively, a named numeric vector with genomic sequence names and lengths. Useful to set the sorting order of sequence names.
<code>sequenceReference</code>	A BSgenome object, or a character scalar giving the path to a fasta formatted file with reference sequences, or a DNAStringSet object. The sequence context (see <code>sequenceContextWidth</code> argument) will be extracted from these sequences.
<code>BPPARAM</code>	A BiocParallelParam object that controls the number of parallel CPU threads to use for some of the steps in <code>readModBam()</code> . The default value is (MulticoreParam (4L, <code>RNGseed</code> = 42L)). If randomly sampling reads (<code>nAlnsToSample</code> > 0), make sure to set the <code>RNGseed</code> argument when constructing the <code>BPPARAM</code> object for reproducible results (see also <code>vignette("Random_Numbers", package = "BiocParallel")</code>).
<code>verbose</code>	A logical scalar. If TRUE, report on progress.

Value

A `DataFrame` with `windowSize` rows and five columns. The metadata slot contains header information from bamfile.

Author(s)

Charlotte Soneson, Michael Stadler

Examples

```
bamfile <- system.file("extdata", "BisSeq_quasr_single.bam",
  package = "SingleMoleculeGenomicsIO")
reffile <- system.file("extdata", "reference.fa.gz",
  package = "SingleMoleculeGenomicsIO")
tbl <- countMismatchStatePairs(bamfile = bamfile, bamFormat = "QuasR",
  regions = "chr1:6940000-6955000",
  sequenceContext = "C",
  readBaseUnmod = "T", readBaseMod = "C",
  windowSize = 200,
  sequenceReference = reffile,
  BPPARAM = BiocParallel::SerialParam(),
  verbose = TRUE)

tbl
```

countStatePairs

Count pairs of modified bases by distance and modification state

Description

For all pairs of bases with modification calls in a read, tabulate the number of pairs at a given distance with a given modification state.

Usage

```
countStatePairs(
  bamfile,
  regions = ".",
  modbase,
  nAlnsToSample = 0,
  seqnamesToSampleFrom = character(0),
  threshUnmod = 0.5,
  threshMod = 0.5,
  windowSize = 200,
  minMapQ = 0,
  minAlignedLength = 0,
  BPPARAM = MulticoreParam(4L),
  verbose = FALSE
)
```

Arguments

bamfile	Character scalar with the path to a modBAM file, containing information about base modifications in MM and ML tags. The bamfile must have an index.
regions	A GRanges object specifying which genomic regions to extract the reads from. Alternatively, regions can be specified as a character vector (e.g. "chr1:1200-1300", "chr2:-6000", "chr1:10-", "chrM" or ".").
modbase	Character scalar defining the modified base.
nAlnsToSample	A numeric scalar. If non-zero, regions is ignored and approximately nAlnsToSample randomly selected alignments on seqnamesToSampleFrom are read from the bamfile. In order to make the results reproducible, make sure to set the random number seed using set.seed. Please note that secondary and supplementary alignments in bamfiles contribute to the total number of alignments but will not be sampled, thus the number of used alignments may be lower than nAlnsToSample.
seqnamesToSampleFrom	A character vector with one or several sequence names (chromosomes) from which to sample alignments from (only used if nAlnsToSample is greater than zero). If set to character(0), all chromosomes present in any of the bam files will be used.
threshUnmod, threshMod	Numeric scalars defining how to convert modification probabilities p to modification states. Bases with $p < \text{threshUnmod}$ will be considered unmodified, and bases with $p \geq \text{threshMod}$ modified. All remaining bases will be ignored.
windowSize	Numeric scalar giving the maximum window size covering pairs of modified bases to consider in pair-counting mode. A window size of 1 corresponds to a single base, a size of 2 to directly adjacent bases, etc.
minMapQ	Numeric scalar giving the minimal mapping quality to include alignments in pair-counting mode.
minAlignedLength	Numeric scalar giving the minimal alignment length to include alignments in pair-counting mode.
BPPARAM	A BiocParallelParam object that controls the number of parallel CPU threads to use for some of the steps in readModBam(). The default value is (MulticoreParam (4L, RNGseed = 42L)). If randomly sampling reads (nAlnsToSample > 0), make sure to set the RNGseed argument when constructing the BPPARAM object for reproducible results (see also vignette("Random_Numbers", package = "BiocParallel")).
verbose	A logical scalar. If TRUE, report on progress.

Value

A `DataFrame` with `windowSize` rows and five columns. The metadata slot contains header information from bamfile.

Author(s)

Charlotte Sonesson, Michael Stadler

Examples

```

modbamfile <- system.file("extdata", "6mA_1_10reads.bam",
                          package = "SingleMoleculeGenomicsIO")
res <- countStatePairs(bamfile = modbamfile,
                      regions = "chr1",
                      modbase = "a", windowSize = 300,
                      BPPARAM = BiocParallel::SerialParam())
res

```

estimateNoise	<i>Estimation of noise variance for time series.</i>
---------------	--

Description

Given a time series signal, potentially with missing/unobserved values, this function estimates the level of noise under the assumption of local continuity/smoothness.

Usage

```
estimateNoise(probs, read_pos, k, min_diffs)
```

Arguments

probs	Numeric vector of (observed) time series measurements.
read_pos	Integer vector of measurement positions in the time series (same length as probs).
k	Integer, maximum gap size tolerated when computing Δx .
min_diffs	Integer, minimum number of lag-1 differences required to produce an estimate (default -1 = auto).

Details

Noisevariance $\approx 0.5 \text{ Var}(\Delta x)$ where Δx are lag-1 differences that may skip up to k missing values. This follows from error propagation and the assumption of low varying x in adjacent measurements.

Value

A vector with the following items:

1. **Mean signal**
2. **Total variance**
3. **Noise variance estimate**
4. **Number of adjacent positions used for the estimate**

Examples

```
estimateNoise(
  c(0.1, 0.25, 0.3, 0.45, 0.5, 0.7, 0.7),
  c(1L, 2L, 3L, 4L, 6L, 8L, 11L),
  2L, 1)
```

estimateSNR	<i>Estimate a time series Signal to Noise Ratio (SNR) given total and noise variance components and (optionally) a background noise model.</i>
-------------	--

Description

Given time series variance components (total variance, raw noise variance), compute the final noise, signal, and SNR. For noise estimation use one of:

1. **A raw noise variance** e.g the result of estimateNoise
2. **A background noise model** based on linear predictors
3. **A flooring rule:** final noise = max(raw, background)

The background noise model is a *general* linear noise baseline: $\text{baseline} = \sum_i \text{beta}[i] * \text{feat}[i]$.

Usage

```
estimateSNR(totalVar, noiseRaw, eps, betas, features, noise_mode = "floor")
```

Arguments

totalVar	Total time series variance. Typically calculated with estimateNoise
noiseRaw	noise variance estimate. Typically calculated with estimateNoise
eps	Numeric. Lower bound applied to both noise and signal.
betas, features	Coefficients (<i>betas</i>) and predictors (<i>features</i>) for the background noise model. Ignored when noise_mode = "raw". Both vectors must have the same length when used. The predictors form a <i>design vector</i> : include a constant 1 if the model has an intercept. For example, for $\text{baseline} = b_0 + b_1 * \text{mean}$, use $\text{betas} = \text{c}(b_0, b_1)$, $\text{features} = \text{c}(1, \text{mean})$.
noise_mode	Character scalar: one of "raw", "model", "floor". • "raw" -> use noise_raw • "model" -> use baseline = sum(betas * features) • "floor" -> use max(noise_raw, baseline)

Value

Named numeric vector with elements: - snr (log2 scale) - signal - noise final estimate used for signal and snr calculation - noise baseline (sum(betas * features); NA if not applied) - noise raw

Examples

```
estimateSNR(totalVar = 1.0, noiseRaw = 1.0, eps = 0.01,
            betas = c(0.5, 2.0), features = c(2.0, 4.5),
            noise_mode = "model")
```

expandSEToBaseSpace	<i>Expand the rows of a RangedSummarizedExperiment to single base resolution.</i>
---------------------	---

Description

Expand the rows of a RangedSummarizedExperiment to single base resolution.

Usage

```
expandSEToBaseSpace(
  se,
  regions = NULL,
  flank = 0L,
  seqinfo = NULL,
  keepAssays = getReadLevelAssayNames(se),
  ignore.strand = TRUE,
  assayNameAmbig = "mod_prob",
  verbose = FALSE
)
```

Arguments

se	RangedSummarizedExperiment object to be expanded to single base resolution.
regions	A GRanges object with one or more regions defining the ranges for expanding se. Alternatively, the regions can be specified as character scalars (e.g. "chr1:1200-1300") that can be coerced into a GRanges object. If NULL (the default), regions is set to the range of the data in se.
flank	Integer scalar. If regions=NULL and flank is non-zero, flank positions will be added on both sides of each range in the derived regions. This is useful to add a buffer of NA values between observed values in non-overlapping regions.
seqinfo	NULL or a Seqinfo object containing information about the set of genomic sequences (chromosomes). Alternatively, a named numeric vector with genomic sequence names and lengths. Used to convert a character region to a GRanges object.
keepAssays	Character vector indicating which (read-level) assays to expand to base space. Only these assays will be present in the returned object.
ignore.strand	A logical scalar defining whether to ignore the strand of the rowRanges(se). If FALSE, se is first filtered (using filterPositions(..., filters = "repeated.positions")) to resolve any positions represented on both strands.

assayNameAmbig A character scalar indicating the assay to use to decide which row to retain if multiple rows represent the same genomic position (on different strands). The row with the largest row sum in this assay is retained. Only used if `ignore.strand` is `FALSE`.

verbose Logical scalar indicating whether to print out progress messages.

Value

A single-base resolution [RangedSummarizedExperiment](#) corresponding to `se`. If `ignore.strand` is `TRUE`, the return object is unstranded. If `ignore.strand` is `FALSE`, all positions present in the input object will retain the assigned strand, while all other positions will be unstranded.

Author(s)

Charlotte Soneson, Michael Stadler

Examples

```
modbamfile <- system.file("extdata", "6mA_1_10reads.bam",
                          package = "SingleMoleculeGenomicsIO")
se <- readModBam(bamfiles = modbamfile, regions = "chr1:6940000-6955000",
               modbase = "a", verbose = TRUE,
               BPPARAM = BiocParallel::SerialParam())
se_exp <- expandSEToBaseSpace(se)
dim(se)
dim(se_exp)
```

`extractColDataColumns` *Extract one or more (possibly nested) columns from the colData of a SummarizedExperiment object*

Description

Extract one or more (possibly nested) columns from the `colData` of a `SummarizedExperiment` object

Usage

```
extractColDataColumns(
  se,
  colNames,
  alignWith = ifelse(any(grepl(":", colNames)), "read", "sample")
)
```

Arguments

<code>se</code>	A SummarizedExperiment object.
<code>colNames</code>	A character vector corresponding to the names of annotation (<code>colData</code>) columns to extract and consolidate into a <code>data.frame</code> . The names can be either columns of <code>colData(se)</code> itself, or columns in a nested (read-level) annotation column of <code>colData(se)</code> . In the latter case, the <code>colNames</code> should be of the form <code>outerColName:innerColName</code> , where <code>outerColName</code> is a column name in <code>colData(se)</code> , and <code>innerColName</code> is a column name in each element of <code>colData(se)[[outerColName]]</code> .
<code>alignWith</code>	A character scalar indicating whether the rows in the returned <code>data.frame</code> correspond to reads ("read") or samples ("sample"). If any element of <code>colNames</code> refers to a nested column, <code>alignWith</code> must be "read".

Value

A `data.frame` with columns `sample`, `read` (if `alignWith = "read"`) and `make.names(colNames)`.

Author(s)

Michael Stadler, Charlotte Soneson

Examples

```
extractfiles <- system.file("extdata",
                           c("modkit_extract_rc_6mA_1.tsv.gz",
                             "modkit_extract_rc_6mA_2.tsv.gz"),
                           package = "SingleMoleculeGenomicsIO")
se <- readModkitExtract(extractfiles, modbase = "a", filter = "modkit",
                       BPPARAM = BiocParallel::SerialParam())
df <- extractColDataColumns(se = se,
                           colNames = c("modbase", "readInfo:ref_strand"),
                           alignWith = "read")

head(df)
dim(df)
df <- extractColDataColumns(se = se,
                           colNames = "modbase",
                           alignWith = "sample")

head(df)
dim(df)
```

<code>extractSeqContext</code>	<i>Extract the sequence context around positions of interest</i>
--------------------------------	--

Description

This function will extract a sequence context of `sequenceContextWidth` bases around the center of the regions defined in `x` from `sequenceReference`.

Usage

```
extractSeqContext(x, sequenceContextWidth, sequenceReference)
```

Arguments

- x** A [GRanges](#) object defining the regions of interest. Alternatively, a [RangedSummarizedExperiment](#) object from which regions can be extracted using [rowRanges](#). The extracted sequences will correspond to the regions defined as `resize(x, width = sequenceContextWidth, fix = "center")`.
- sequenceContextWidth** A numeric scalar giving the width of the sequence context to be extracted from the reference (`sequenceReference` argument). This must be an odd number so that the sequence can be centered on the modified base. If `sequenceContextWidth = 0` (the default), no sequence context will be extracted.
- sequenceReference** A [DNASTringSet](#) object, a [BSgenome](#) object, or a character scalar giving the path to a fasta formatted file with reference sequences. If a [BSgenome](#) object is provided, it will be internally converted to a [DNASTringSet](#) object, since the latter allows for faster sequence retrieval.

Value

A [DNASTringSet](#) object of the same length as `x` with extracted sequence context. All elements are guaranteed to have identical length (if the sequence context extends to before the start or beyond the end of a reference sequence, it will be padded with 'N' bases).

Author(s)

Michael Stadler

See Also

[resize](#), [DNASTringSet](#)

Examples

```
# file with sequence in fasta format of length 6957060
reffile <- system.file("extdata", "reference.fa.gz",
                      package = "SingleMoleculeGenomicsIO")

# define some regions at the end of the reference sequence
regions <- GenomicRanges::GRanges(
  "chr1", IRanges::IRanges(start = 6957060 - c(4, 2, 0),
    width = 1, names = c("a", "b", "c")))

# extract sequence context (note the padding with N's)
extractSeqContext(regions, 7, reffile)
```

filterPositions	<i>Filter positions</i>
-----------------	-------------------------

Description

Filter positions based on any combination of sequence context, coverage, repetition of the same position, and presence of non-NA values. Filters are applied in the order they are specified to the `filters` argument. Any filter type can be repeated an arbitrary number of times.

Usage

```
filterPositions(
  se,
  filters = c("sequenceContext", "coverage", "all.na"),
  sequenceContext = NULL,
  assayNameCov = "Nvalid",
  minCov = 1,
  minNbrSamples = NULL,
  regions = NULL,
  seqinfo = NULL,
  assayNameAmbig = "Nvalid",
  assayNameNA = "mod_prob"
)
```

Arguments

<code>se</code>	A <code>SummarizedExperiment</code> object.
<code>filters</code>	A character vector. All values must be one of "sequenceContext", "coverage", "repeated.positions", "regions" and "all.na". Filters are applied in the order specified by this vector.
<code>sequenceContext</code>	A character vector with sequence contexts to retain. To apply this filter, the "sequenceContext" column must be present in <code>rowData(se)</code> (see <code>addSeqContext</code>).
<code>assayNameCov</code>	A character scalar indicating the assay to use to define the coverage. If this is a read-level assay, coverage is first calculated using <code>flattenReadLevelAssay(..., statistics = "Nvalid")</code> .
<code>minCov</code>	A numeric scalar indicating the lowest acceptable coverage in order to keep a position.
<code>minNbrSamples</code>	A numeric scalar, or <code>NULL</code> . If <code>NULL</code> (default), the row sum of <code>assayNameCov</code> (i.e., the total coverage across all samples) is used for the coverage filtering. If not <code>NULL</code> , a position is required to have at least <code>minCov</code> coverage in at least <code>minNbrSamples</code> to be retained.
<code>regions</code>	A GRanges object specifying the genomic regions to restrict the output to. Alternatively, regions can be specified as a character vector (e.g. "chr1:1200-1300", "chr2:-6000" or "chrM") that will be coerced into a <code>GRanges</code> object. If the end

	coordinate is not provided (for example in "chr1:10-", which means "to the end of chr1"), it will be obtained from seqinfo or default to a large value if seqinfo is not provided.
seqinfo	NULL or a Seqinfo object containing information about the set of genomic sequences (chromosomes). Alternatively, a named numeric vector with genomic sequence names and lengths.
assayNameAmbig	A character scalar indicating the assay to use to decide which row to retain if multiple rows represent the same genomic position (on different strands). The row with the largest row sum in this assay is retained.
assayNameNA	A character scalar indicating the assay to use as the basis for filtering out positions with NA values across all reads. This should be a read level assay.

Value

A filtered SummarizedExperiment.

Author(s)

Charlotte Soneson

Examples

```
modbamfiles <- system.file("extdata", c("6mA_1_10reads.bam", "6mA_2_10reads.bam"),
                           package = "SingleMoleculeGenomicsIO")
reffile <- system.file("extdata", "reference.fa.gz", package = "SingleMoleculeGenomicsIO")

se <- readModBam(bamfiles = modbamfiles, regions = "chr1:6920000-6940000",
                modbase = "a", verbose = FALSE,
                BPPARAM = BiocParallel::SerialParam())
se <- flattenReadLevelAssay(se)
se <- addSeqContext(se, sequenceContextWidth = 3, sequenceReference = reffile)
sefilt <- filterPositions(se, c("sequenceContext", "coverage", "all.na"),
                        minCov = 5, sequenceContext = "TAG")
```

filterReads	<i>Filter reads</i>
-------------	---------------------

Description

Filter reads

Usage

```

filterReads(
  se,
  assayName = "mod_prob",
  readInfoCol = "readInfo",
  qcCol = "QC",
  minQscore = 0,
  maxEntropy = Inf,
  minSNR = -Inf,
  maxFracLowConf = 1,
  minReadLength = 0,
  minAlignedLength = 0,
  minAlignedFraction = 0,
  minCoveredFraction = 0,
  region = NULL,
  prune = TRUE,
  onlyStats = FALSE,
  removeAllNapos = TRUE
)

```

Arguments

se	A SummarizedExperiment object.
assayName	A character scalar providing the name of a read-level assay in se. This assay will be used to extract read names, as well as to filter out any read that is not overlapping any of the positions in the object.
readInfoCol	A character scalar providing the name of the column in colData that contains read info. Can be NULL if no such column exists.
qcCol	A character scalar providing the name of the column in colData that contains quality metrics (calculated by calcReadStats). Can be NULL if no such column exists.
minQscore	A numeric scalar representing the smallest acceptable read-level Qscore. Reads with Qscore below this value will be filtered out.
maxEntropy	A numeric scalar representing the largest acceptable read-level entropy. Reads without modified-base calls or with entropy above this value will be filtered out.
minSNR	A numeric scalar representing the smallest acceptable read Signal to Noise ratio (SNR). Reads with SNR below this value will be filtered out.
maxFracLowConf	A numeric scalar representing the maximally acceptable fraction of low-confidence modified base calls in a read. Reads without modified-base calls or with a fraction of low confidence calls greater than this value will be filtered out.
minReadLength	A numeric scalar representing the smallest acceptable read length. Reads that are shorter than this value will be filtered out.
minAlignedLength	A numeric scalar representing the smallest acceptable aligned length. Reads with aligned length shorter than this value will be filtered out.

<code>minAlignedFraction</code>	A numeric scalar representing the smallest acceptable aligned fraction of a read. Reads where the aligned fraction is smaller than this value will be filtered out.
<code>minCoveredFraction</code>	A numeric scalar giving the minimal fraction of region that a read alignment needs to cover to be retained.
<code>region</code>	A GRanges object with a single region to be used for the <code>minCoveredFraction</code> filter. Alternatively, the region can be specified as a character scalar (e.g. "chr1:1200-1300") that can be coerced into a <code>GRanges</code> object. If <code>NULL</code> (the default), all reads are retained.
<code>prune</code>	A logical scalar. If <code>TRUE</code> (the default), samples for which the filtering retains none of the reads will be completely removed from the returned <code>SummarizedExperiment</code> (also from <code>colData</code> and from assays that do not store read-level data). If <code>FALSE</code> , such samples are retained (in the assays with read-level data as a zero-column <code>SparseMatrix</code>).
<code>onlyStats</code>	A logical scalar. If <code>FALSE</code> (the default), the <code>SummarizedExperiment</code> object will be filtered according to the provided thresholds. If <code>TRUE</code> , the filter statistics are calculated and returned, but the object is not subset.
<code>removeAllNApos</code>	A logical scalar. If <code>TRUE</code> , remove all positions for which all values in <code>assayName</code> are <code>NA</code> after the read filtering.

Value

If `onlyStats` is `FALSE`, a filtered `SummarizedExperiment` object. The metadata of this object contains a slot named `filteredOutReads`, which tabulate all reads that are filtered out, together with the reason(s) for exclusion. If `onlyStats` is `TRUE`, only this table is returned.

Author(s)

Charlotte Soneson, Michael Stadler

Examples

```
library(SummarizedExperiment)
modbamfile <- system.file("extdata", "6mA_1_10reads.bam",
                          package = "SingleMoleculeGenomicsIO")
se <- readModBam(bamfile = modbamfile, regions = "chr1:6920000-6995000",
                modbase = "a", verbose = TRUE,
                BPPARAM = BiocParallel::SerialParam())
se <- addReadStats(se, name = "QC",
                  BPPARAM = BiocParallel::SerialParam())

## Filter se
## ... by quality score and aligned length
sefilt <- filterReads(se, minQscore = 14, minAlignedLength = 10000)
## ... by covered fraction of the specified region
sefilt <- filterReads(se, minCoveredFraction = 1, region = "chr1:6930000-6935000")

## Only calculate filter stats
```

```

filtstats <- filterReads(se, minQscore = 14, minAlignedLength = 10000,
                        onlyStats = TRUE)
filtstats

## Visualize filter stats in UpSet plot, e.g. with ComplexUpset
if (require(SimpleUpset)) {
  SimpleUpset::simpleUpSet(as.data.frame(filtstats$s1))
}

```

filterReadsModBam	<i>Write bam records from infile to outfile if they pass filter criteria.</i>
-------------------	---

Description

For each bam file in infiles, parse alignments, calculate read statistics and write the alignment to the corresponding output file from outfiles if the read passes all criteria defined by the filtering arguments. Filters are processed hierarchically: If a read does not pass a given filter, the remaining filters will not be examined and the processing continues with the next read. The filters are examined in this order: keepUnmapped, keepSecondary, keepSupplementary, minReadLength, minAlignedLength, minAlignedFraction, minQscore, minSNR, maxFracLowConf, maxEntropy.

Usage

```

filterReadsModBam(
  infiles,
  outfiles,
  modbase,
  indexOutfiles = TRUE,
  overwriteOutfiles = FALSE,
  keepUnmapped = TRUE,
  keepSecondary = TRUE,
  keepSupplementary = TRUE,
  minReadLength = 0,
  minAlignedLength = 0,
  minAlignedFraction = 0,
  minQscore = 0,
  minSNR = -Inf,
  maxFracLowConf = 1,
  maxEntropy = Inf,
  noiseCoef = c(NA_real_, NA_real_),
  LowConf = 0.7,
  BPPARAM = MulticoreParam(4L, RNGseed = 42L),
  verbose = FALSE
)

```

Arguments

<code>infiles</code>	Character vector with name(s) of the input bam file(s).
<code>outfiles</code>	Character vector with name(s) of the output bam file(s). Needs to have the same length as <code>infiles</code> .
<code>modbase</code>	Character scalar defining the modified base to analyze (used by <code>maxEntropy</code> and <code>maxFracLowConf</code>).
<code>indexOutfiles</code>	Logical scalar. If TRUE (the default) create a bam index file (.bai file) for each of the generated outfiles.
<code>overwriteOutfiles</code>	Logical scalar. If FALSE (the default), existing outfiles will not be overwritten and the function will abort with an error message.
<code>keepUnmapped, keepSecondary, keepSupplementary</code>	Logical scalars indicating whether to keep unmapped, secondary or supplementary alignments.
<code>minReadLength</code>	A numeric scalar representing the smallest acceptable read length. Reads that are shorter than this value will be filtered out.
<code>minAlignedLength</code>	A numeric scalar representing the smallest acceptable aligned length. Reads with aligned length shorter than this value will be filtered out.
<code>minAlignedFraction</code>	A numeric scalar representing the smallest acceptable aligned fraction of a read. Reads where the aligned fraction is smaller than this value will be filtered out.
<code>minQscore</code>	A numeric scalar representing the smallest acceptable read-level Qscore. Reads with Qscore below this value will be filtered out.
<code>minSNR</code>	Numeric scalar. Minimum acceptable read Signal-to-Noise Ratio (SNR). Reads with SNR below this value are filtered out. Set to <code>-Inf</code> to disable SNR filtering. The SNR is computed per read as $\log_2(\text{SignalVar}/\text{NoiseVar})$ where $\text{NoiseVar} \approx 0.5 \text{Var}(\Delta x)$ using adjacent methylation differences that can jump up to $k = 2$ gaps, and $\text{SignalVar} = \max(\text{Var}(x) - \text{NoiseVar}, \varepsilon)$ with $\varepsilon = 10^{-3}$.
<code>maxFracLowConf</code>	A numeric scalar representing the maximally acceptable fraction of low-confidence modified base calls in a read. Reads without modified-base calls or with a fraction of low confidence calls greater than this value will be filtered out.
<code>maxEntropy</code>	A numeric scalar representing the largest acceptable read-level entropy. Reads without modified-base calls or with entropy above this value will be filtered out. A value of <code>Inf</code> deactivates the entropy filter.
<code>noiseCoef</code>	Numeric vector of length 2 giving the background noise model coefficients (<code>b0</code> , <code>b1</code>). If provided, these are used to impose a floor on the estimated per-read noise variance (see also calcReadStats).
<code>LowConf</code>	A numeric scalar with the minimum call confidence below which calls are considered "low confidence".
<code>BPPARAM</code>	A BiocParallelParam object that controls the number of parallel CPU threads to use for some of the steps in <code>filterReadsModBam()</code> . The default value is (MulticoreParam(4L, RNGseed = 42L)) .
<code>verbose</code>	Logical scalar. If TRUE, report on progress.

Value

filterReadsModBam is called for its side effect of generating new modBam files containing the subset of bam records from input modBam files that pass all filtering criteria. In addition, it returns a data.frame with one row per infiles giving the numbers of bam records that were read in total, that were retained in the outfiles and that were filtered-out by reason of exclusion.

Author(s)

Michael Stadler

Examples

```
modbamfiles <- system.file("extdata", c("6mA_1_10reads.bam", "6mA_2_10reads.bam"),
  package = "SingleMoleculeGenomicsIO")
filtbamfiles <- tempfile(fileext = rep(".bam", length(modbamfiles)))
res <- filterReadsModBam(infiles = modbamfiles, outfiles = filtbamfiles,
  modbase = "a", indexOutfiles = FALSE, minReadLength = 6746,
  minAlignedLength = 6896, minAlignedFraction = 0.56, minSNR=-0.768,
  minQscore = 9.7, maxFracLowConf = 0.11, maxEntropy = 0.29,
  BPPARAM = BiocParallel::SerialParam(), verbose = TRUE)

res
unlink(filtbamfiles)
```

flattenReadLevelAssay *Summarize a read-level object to sample-level*

Description

This function will take a [SummarizedExperiment](#) object with read-level footprinting data (e.g. returned by [readModkitExtract](#) or [readModBam](#)) and summarize reads in each sample, for instance to generate modified and total counts at each position for each sample.

Usage

```
flattenReadLevelAssay(
  se,
  assayName = "mod_prob",
  statistics = c("Nmod", "Nvalid", "FracMod"),
  modProbThreshold = 0.5,
  keepReads = TRUE,
  replaceExisting = TRUE,
  verbose = FALSE
)
```

Arguments

<code>se</code>	SummarizedExperiment object with read-level footprinting data. Rows should correspond to positions and columns to samples.
<code>assayName</code>	A character scalar specifying the assay of <code>se</code> containing the read-level data to be summarized. Typically, this assay contains modification probabilities.
<code>statistics</code>	Character vector specifying the type of statistics to be computed. Currently supported values are "Nmod" (number of values per row in the <code>assayName</code> assay that are greater than or equal to <code>modProbThreshold</code>), "Nvalid" (number of valid/non-NA values per row, typically the number of overlapping reads), "Frac-Mod" (Nmod/Nvalid), "Pmod" (row-wise average values), "Mean" (equivalent to "Pmod"), "Sum" (row-wise sums of non-NA values), "AvgConf" (average confidence of (non-)modification probabilities, more precisely the row-wise averages of the largest of the observed values and 1 - the observed values).
<code>modProbThreshold</code>	A numeric scalar, indicating the modification probability threshold to use to classify a base as 'modified' or 'unmodified'.
<code>keepReads</code>	A logical scalar. If TRUE (the default), the read-level data from <code>assayName</code> will be retained in an assay of the same name.
<code>replaceExisting</code>	A logical scalar. If TRUE (the default), any existing assays with the same name as the ones requested will be overwritten. Otherwise, existing assays will be retained and the corresponding summary statistic(s) will not be recalculated.
<code>verbose</code>	If TRUE, report on progress.

Value

A [SummarizedExperiment](#) object with the same dimensions as `se` (positions in rows and samples in columns), and added assays corresponding to the requested statistics.

Author(s)

Charlotte Soneson, Michael Stadler

See Also

[SummarizedExperiment](#) for the returned object type, [readModkitExtract](#) for the function used to read the input files.

Examples

```
exfile <- system.file("extdata", "modkit_extract_rc_6mA_1.tsv.gz",
                      package = "SingleMoleculeGenomicsIO")
se <- readModkitExtract(exfile, modbase = "a",
                      BPPARAM = BiocParallel::SerialParam())
se

se_summary <- flattenReadLevelAssay(se)
se_summary
```

getBaseCoverageForBam *Get base coverage histogram for a BAM file*

Description

Get base coverage histogram for a BAM file

Usage

```
getBaseCoverageForBam(  
  bamfile,  
  regions = NULL,  
  maxDepth = 200L,  
  method = "full",  
  nThreads = 3L  
)
```

Arguments

bamfile	A character scalar with the bam file name (and path).
regions	Character vector specifying the region(s) for which to calculate coverage. Each region uses the grammar supported by the htlib function <code>sam_parse_region</code> , for example "." (all contigs), "chr" (whole contig), "chr:START", "chr:-END" or "chr:START-END". If NULL, the whole genome (".") is used by default.
maxDepth	An integer scalar defining the maximal depth to consider.
method	Character scalar with the method used for coverage calculation. "full" (the default) considers CIGAR operations, thus not counting soft-clip bases and read-inserted positions as covered. "simple" ignores the CIGAR strings and covers the whole reference span between the first and the last aligned position of each alignment (using htlib's <code>bam_endpos</code>); this is slightly faster but overestimates coverage in the soft-clipped ends and around indels.
nThreads	A numeric scalar with the number of threads used for decompressing BAM records.

Details

Secondary and supplementary alignments and unmapped reads are not included.

Value

A named numeric vector of length `maxDepth + 1`, with values at index `i` giving the number of positions that were overlapped by exactly `i-1` alignments. Positions overlapped by more than `maxDepth` alignments are also added to the value for `maxDepth` at index `maxDepth + 1`.

References

The algorithm was described in Pedersen BS and Quinlan AR. "Mosdepth: quick coverage calculation for genomes and exomes". Bioinformatics. 2018; 34(5):867-868. <https://doi.org/10.1093/bioinformatics/btx699>

Examples

```
modbamfile <- system.file("extdata", "6mA_1_10reads.bam", package = "SingleMoleculeGenomicsIO")
getBaseCoverageForBam(modbamfile, "chr1", 12L, "full")
getBaseCoverageForBam(modbamfile, "chr1:6000000-7000000", 12L, "full")
getBaseCoverageForBam(modbamfile, "chr1:6000000-7000000", 12L, "simple")
```

getReadLevelAssayNames

Get names of assays containing read-level data

Description

The names of assays designated as containing read-level data are extracted from `metadata(se)$readLevelData$assayName`

Usage

```
getReadLevelAssayNames(se)
```

Arguments

`se` A SummarizedExperiment object.

Value

A (possibly empty) character vector with the names of the assays of `se` containing read-level data.

Author(s)

Charlotte Soneson

Examples

```
modbamfiles <- system.file("extdata", c("6mA_1_10reads.bam",
                                         "6mA_2_10reads.bam"),
                           package = "SingleMoleculeGenomicsIO")
se <- readModBam(bamfiles = modbamfiles, regions = "chr1:6940000-6955000",
                modbase = "a", verbose = FALSE,
                BPPARAM = BiocParallel::SerialParam())
se <- addReadStats(se, BPPARAM = BiocParallel::SerialParam())
getReadLevelAssayNames(se)
```

getReadLevelColDataNames

Get names of colData columns containing read-level data

Description

The names of colData column designated as containing read-level annotations are extracted from `metadata(se)$readLevelData$colDataColumns`.

Usage

```
getReadLevelColDataNames(se)
```

Arguments

`se` A SummarizedExperiment object.

Value

A (possibly empty) character vector with the names of the columns of `colData(se)` containing read-level data.

Author(s)

Charlotte Soneson

Examples

```
modbamfiles <- system.file("extdata", c("6mA_1_10reads.bam",
                                         "6mA_2_10reads.bam"),
                           package = "SingleMoleculeGenomicsIO")
se <- readModBam(bamfiles = modbamfiles, regions = "chr1:6940000-6955000",
               modbase = "a", verbose = FALSE,
               BPPARAM = BiocParallel::SerialParam())
se <- addReadStats(se, BPPARAM = BiocParallel::SerialParam())
getReadLevelColDataNames(se)
```

getReadNamesBySample *Get list of read names by sample from SummarizedExperiment object*

Description

Get list of read names by sample from SummarizedExperiment object

Usage

```
getReadNamesBySample(se)
```

Arguments

`se` A [SummarizedExperiment](#) object.

Value

A named list with one entry per sample, containing the read names for the respective sample.

Author(s)

Charlotte Soneson

Examples

```
extractfiles <- system.file("extdata",
                             c("modkit_extract_rc_6mA_1.tsv.gz",
                               "modkit_extract_rc_6mA_2.tsv.gz"),
                             package = "SingleMoleculeGenomicsIO")
se <- readModkitExtract(extractfiles, modbase = "a", filter = "modkit",
                        BPPARAM = BiocParallel::SerialParam())
getReadNamesBySample(se)
```

plotReadStats

Plot distribution of QC statistics

Description

This function plots histograms of the various per-read summary statistics calculated by [addReadStats](#).

Usage

```
plotReadStats(
  se,
  readInfoCol = "readInfo",
  qcCol = "QC",
  minQscore = NULL,
  maxEntropy = NULL,
  maxFracLowConf = NULL,
  minReadLength = NULL,
  minAlignedLength = NULL,
  minAlignedFraction = NULL,
  minSNR = NULL
)
```

Arguments

se	A RangedSummarizedExperiment , typically generated by <code>addReadStats</code> .
readInfoCol	A character scalar providing the name of the column in <code>colData</code> that contains read info. Can be <code>NULL</code> if no such column exists.
qcCol	A character scalar providing the name of the column in <code>colData</code> that contains quality metrics (calculated by <code>calcReadStats</code>). Can be <code>NULL</code> if no such column exists.
minQscore, maxEntropy, maxFracLowConf, minReadLength, minAlignedLength, minAlignedFraction, minSNR	Numeric scalars representing possible threshold values used in filterReads , for illustration in the plot panels.

Value

A ggplot object.

Author(s)

Panagiotis Papapasaikas, Charlotte Soneson, Michael Stadler

Examples

```
library(SummarizedExperiment)
modbamfiles <- system.file("extdata",
                           c("6mA_1_10reads.bam", "6mA_2_10reads.bam"),
                           package = "SingleMoleculeGenomicsIO")
se <- readModBam(bamfile = modbamfiles, regions = "chr1:6940000-6955000",
                modbase = "a", verbose = TRUE,
                BPPARAM = BiocParallel::SerialParam())
se <- addReadStats(se, BPPARAM = BiocParallel::SerialParam())
plotReadStats(se)
```

readBedMethyl	<i>Read collapsed single-molecule footprinting data from a bedMethyl file</i>
---------------	---

Description

This function will read collapsed single-molecule footprinting data (reads combined per genomic position) from a `bedMethyl` file.

Usage

```
readBedMethyl(
  fnames,
  modbase,
  nrows = Inf,
  sampleAnnot = NULL,
  seqinfo = NULL,
  sequenceContextWidth = 0,
  sequenceReference = NULL,
  BPPARAM = MulticoreParam(4L, RNGseed = 42L),
  verbose = FALSE
)
```

Arguments

<code>fnames</code>	Character vector with one or several paths of <code>bedMethyl</code> files, such as generated by <code>modkit</code> <code>pileup</code> . Each file will be read separately and become one of the columns in the returned SummarizedExperiment object. If <code>fnames</code> is a named vector, the names are used as column names in the returned object. Otherwise, the column names will be <code>s1</code> , ..., <code>sN</code> , where <code>N</code> is the length of <code>fnames</code> . If several elements of <code>fnames</code> have identical names, the data from the corresponding files are summed into a single column in the returned object.
<code>modbase</code>	Character vector defining the modified base for each sample. If <code>modbase</code> is a named vector, the names should correspond to the names of <code>fnames</code> . Otherwise, it will be assumed that the elements are in the same order as the files in <code>fnames</code> . If <code>modbase</code> has length 1, the same modified base will be used for all samples.
<code>nrows</code>	Only read <code>nrows</code> rows of the input file.
<code>sampleAnnot</code>	A <code>data.frame</code> (or <code>NULL</code>) providing annotations for the samples. It must contain at least one column, named "sample", which must contain all the values of <code>names(fnames)</code> . The provided annotations will be propagated to the returned <code>SummarizedExperiment</code> object.
<code>seqinfo</code>	<code>NULL</code> or a Seqinfo object containing information about the set of genomic sequences (chromosomes). Alternatively, a named numeric vector with genomic sequence names and lengths. Useful to set the sorting order of sequence names.
<code>sequenceContextWidth</code> , <code>sequenceReference</code>	Define the sequence context to be extracted around modified bases. By default (<code>sequenceContextWidth = 0</code>), no sequence context will be extracted, otherwise it will be returned in <code>rowData(x)\$sequenceContext</code> . See addSeqContext for details.
<code>BPPARAM</code>	A BiocParallelParam object that controls the number of parallel CPU threads to use for some of the steps in <code>readBedMethyl()</code> . The default value is (<code>MulticoreParam(4L, RNGseed = 42L)</code>)
<code>verbose</code>	If <code>TRUE</code> , report on progress.

Value

A [SummarizedExperiment](#) object with genomic positions in rows and samples (the unique names of fnames) in the columns. If `sequenceContextWidth != 0`, `rowData(x)$sequenceContext` will be a [DNAStringSet](#) object with the extracted sequences.

Author(s)

Michael Stadler, Charlotte Soneson

See Also

[modkit](#) software, [bedMethyl](#) format description, [SummarizedExperiment](#) for the returned object type, [fread](#) for the function used to read the input files, [addSeqContext](#) used to add the sequence context.

Examples

```
bmfile <- system.file("extdata", "modkit_pileup_5mC_1.bed.gz",
                      package = "SingleMoleculeGenomicsIO")
readBedMethyl(bmfile, modbase = "m",
              BPPARAM = BiocParallel::SerialParam())
```

readMismatchBam

Read base modifications from "C-to-T" bam file(s)

Description

Parse read-to-genome mismatches and return a [SummarizedExperiment](#) object with information on base states. Base mismatches are typically resulting from single molecule genomics experiments, representing the accessibility of individual bases: For bisulfite-sequencing experiments, C-to-T mismatches correspond to unmethylated (inaccessible) bases (`readBaseUnmod="T"`, `readBaseMod="C"`, see below), while in deaminase-treatment based experiments, C-to-T mismatches represent modified (accessible) bases (`readBaseUnmod="C"`, `readBaseMod="T"`). Secondary and supplementary alignments are ignored.

Usage

```
readMismatchBam(
  bamfiles,
  bamFormat = "QuasR",
  regions = NULL,
  sequenceContext = "GCH",
  readBaseUnmod = "T",
  readBaseMod = "C",
  level = "read",
  overlapAggregation = "maxQscore",
```

```

sampleAnnot = NULL,
nAlnsToSample = 0,
seqnamesToSampleFrom = character(0),
seqinfo = NULL,
sequenceReference = NULL,
variantPositions = NULL,
trim = FALSE,
maxCoverage = NULL,
BPPARAM = MulticoreParam(4L, RNGseed = 42L),
verbose = FALSE
)

```

Arguments

- | | |
|----------------------------|--|
| bamfiles | Character vector with one or several paths of BAM files, containing alignments with specific (e.g., C-to-T) mismatches. If bamfiles is a named vector, the names are used as sample names and prefixes for read names. Otherwise, the prefixes will be s1, ..., sN, where N is the length of bamfiles. All bamfiles must have an index. |
| bamFormat | A character scalar giving the format of BAM files. Currently supported are BAM files that have been created with "QuasR" (using qAlign in bisulfite mode) or "Bismark". |
| regions | A GRanges object specifying which genomic regions to extract the reads from. Alternatively, regions can be specified as a character vector (e.g. "chr1:1200-1300", "chr2:-6000" or "chrM") that will be coerced into a GRanges object. If the end coordinate is not provided (for example in "chr1:10-", which means "to the end of chr1"), it will be obtained from seqinfo or default to a large value if seqinfo is not provided. Note that unless trim=TRUE, the reads are not trimmed to the boundaries of the specified ranges. As a result, returned positions will typically extend out of the specified regions. If nAlnsToSample is set to a non-zero value, regions is ignored. |
| sequenceContext | A character scalar with an odd number of characters, specifying the genomic context on the plus strand, for which to report (mis-)matches. The string may contain IUPAC ambiguity codes, e.g. sequenceContext="GCH" would correspond to all GpC dinucleotides, excluding C's that are in CpG dinucleotides. The call will be made by comparing the genomic base in the middle of sequenceContext to the aligned base in the read and interpreted according to readBaseUnmod and readBaseMod. The genomic sequence context will be returned in rowData(x)\$sequenceContext. |
| readBaseUnmod, readBaseMod | Character scalars defining the read bases that are interpreted as unmodified or modified, respectively, when aligned to the middle base of sequenceContext. |
| level | Character scalar specifying the level of returned modification data. Supported values are:

"read" : Extracts modification probabilities for individual reads into an assay called "mod_prob", with each column (sample) consisting of a position-by-read NaMatrix . This is the default. |

"quickread" : Like "read", but using pileup-based data rather than parsing individual alignments. This mode does not support variant labels or read sampling.

"summary" : Counts the total and modified bases for each position and strand and returns them in assays named "Nvalid" and "Nmod", respectively, as well as the Nmod/Nvalid ratio in the assay "FracMod".

overlapAggregation

Character scalar indicating how to aggregate modification probabilities for positions that are covered by both mates in a paired-end bam file. Only used if level = "read". In other modes, aggregation corresponding to overlapAggregation = "maxQscore" is performed automatically. Currently supported values are "maxQscore".

sampleAnnot

A data.frame (or NULL) providing annotations for the samples. It must contain at least one column, named "sample", which must contain all the values of names(bamfiles). The provided annotations will be propagated to the returned SummarizedExperiment object.

nAlnsToSample

A numeric scalar. If non-zero, regions is ignored and approximately nAlnsToSample randomly selected alignments on seqnamesToSampleFrom are read from each of the bamfiles. In order to make the results reproducible, make sure to set the RNGseed argument in the provided BPPARAM object (see below). Please note that secondary and supplementary alignments in bamfiles contribute to the total number of alignments but will not be sampled, thus the number of returned alignments may be lower than nAlnsToSample. Also, sampled reads that do not overlap a site with the indicated sequence context will not be returned, which may further reduce the number of returned alignments. Note that for paired-end bam files, individual reads are sampled and pairs will not be complete.

seqnamesToSampleFrom

A character vector with one or several sequence names (chromosomes) from which to sample alignments from (only used if nAlnsToSample is greater than zero). If set to character(0), all chromosomes present in any of the bam files will be used.

seqinfo

NULL or a Seqinfo object containing information about the set of genomic sequences (chromosomes). Alternatively, a named numeric vector with genomic sequence names and lengths. Useful to set the sorting order of sequence names.

sequenceReference

A BSgenome object, or a character scalar giving the path to a fasta formatted file with reference sequences, or a DNASTringSet object. The sequence context (see sequenceContextWidth argument) will be extracted from these sequences.

variantPositions

An optional GPos object with seqnames and coordinates of single nucleotide variant positions, to be used to construct read labels for allele-specific analysis. Ignored if NULL or nAlnsToSample > 0 (sampling-mode).

trim

A logical scalar. If TRUE, the returned SummarizedExperiment object will only contain the positions overlapping the specified regions. If FALSE (default), the object will be extended to all positions covered by the reads overlapping regions. In both cases, only reads overlapping the specified regions are included.

maxCoverage	Integer scalar used to increase the maximal coverage for which samtools pileup will allocate cache memory. Ignored if level="read". A value of NULL will use the samtools default (at the time of writing 8000). Large values will increase memory consumption. If the actual coverage is larger than this value, alignments may be silently ignored.
BPPARAM	A BiocParallelParam object that controls the number of parallel CPU threads to use for some of the steps in readModBam(). The default value is (MulticoreParam (4L, RNGseed = 42L)). If randomly sampling reads (nAlnsToSample > 0), make sure to set the RNGseed argument when constructing the BPPARAM object for reproducible results (see also vignette("Random_Numbers", package = "BiocParallel")).
verbose	Logical scalar. If TRUE, report on progress.

Value

A [SummarizedExperiment](#) object with genomic positions in rows and samples in columns. The assays depend on the value of the level argument (see above).

Author(s)

Michael Stadler, Charlotte Soneson

Examples

```
bamfile <- system.file("extdata", "BisSeq_quasr_single.bam",
  package = "SingleMoleculeGenomicsIO")
reffile <- system.file("extdata", "reference.fa.gz",
  package = "SingleMoleculeGenomicsIO")
se <- readMismatchBam(bamfiles = bamfile, regions = "chr1:6940000-6955000",
  sequenceReference = reffile, sequenceContext = "NCG",
  readBaseUnmod = "T", readBaseMod = "C",
  verbose = TRUE, BPPARAM = BiocParallel::SerialParam())
```

readModBam	<i>Read base modifications from bam file(s)</i>
------------	---

Description

Parse ML and MM tags (see <https://samtools.github.io/hts-specs/SAMtags.pdf>, section 1.7) and return a [SummarizedExperiment](#) object with information on modified bases. Implicitly called bases will get a modification probability of zero. Secondary and supplementary alignments are ignored.

Usage

```
readModBam(
  bamfiles,
  regions = NULL,
  modbase,
```

```

level = ifelse(nAlnsToSample == 0 & is.null(variantPositions), "quickread", "read"),
sampleAnnot = NULL,
nAlnsToSample = 0,
seqnamesToSampleFrom = character(0),
seqinfo = NULL,
sequenceContextWidth = 0,
sequenceReference = NULL,
variantPositions = NULL,
modProbThreshold = 0.5,
trim = FALSE,
maxCoverage = NULL,
BPPARAM = MulticoreParam(4L, RNGseed = 42L),
verbose = FALSE
)

```

Arguments

bamfiles	Character vector with one or several paths of modBAM files, containing information about base modifications in MM and ML tags. If bamfiles is a named vector, the names are used as sample names and prefixes for read names. Otherwise, the prefixes will be s1, ..., sN, where N is the length of bamfiles. All bamfiles must have an index.
regions	A GRanges object specifying which genomic regions to extract the reads from. Alternatively, regions can be specified as a character vector (e.g. "chr1:1200-1300", "chr2:-6000" or "chrM") that will be coerced into a GRanges object. If the end coordinate is not provided (for example in "chr1:10-", which means "to the end of chr1"), it will be obtained from seqinfo or default to a large value if seqinfo is not provided. Note that unless trim=TRUE, the reads are not trimmed to the boundaries of the specified ranges. As a result, returned positions will typically extend out of the specified regions. If nAlnsToSample is set to a non-zero value, regions is ignored.
modbase	Character vector defining the modified base for each sample. If modbase is a named vector, the names should correspond to the names of bamfiles. Otherwise, it will be assumed that the elements are in the same order as the files in bamfiles. If modbase has length 1, the same modified base will be used for all samples.
level	Character scalar specifying the level of returned modification data. Supported values are: "read" : Extracts modification probabilities for individual reads into an assay called "mod_prob", with each column (sample) consisting of a position-by-read NaMatrix . This is the default if nAlnsToSample is non-zero or variantPositions is not NULL. "quickread" : Like "read", but runs faster, does not support read sampling, and does not return all annotations currently provided by "read". This is the default if nAlnsToSample is zero and variantPositions is NULL. "summary" : Counts the total and modified bases for each position and strand and returns them in assays named "Nvalid" and "Nmod", respectively, as well as the Nmod/Nvalid ratio in the assay "FracMod".

sampleAnnot	A <code>data.frame</code> (or <code>NULL</code>) providing annotations for the samples. It must contain at least one column, named "sample", which must contain all the values of <code>names(bamfiles)</code> . The provided annotations will be propagated to the returned <code>SummarizedExperiment</code> object.
nAlnsToSample	A numeric scalar. If non-zero, <code>regions</code> is ignored and approximately <code>nAlnsToSample</code> randomly selected alignments on <code>seqnamesToSampleFrom</code> are read from each of the bamfiles. In order to make the results reproducible, make sure to set the <code>RNGseed</code> argument in the provided <code>BPPARAM</code> object (see below). Please note that secondary and supplementary alignments in bamfiles contribute to the total number of alignments but will not be sampled, thus the number of returned alignments may be lower than <code>nAlnsToSample</code> .
seqnamesToSampleFrom	A character vector with one or several sequence names (chromosomes) from which to sample alignments from (only used if <code>nAlnsToSample</code> is greater than zero). If set to <code>character(0)</code> , all chromosomes present in any of the bam files will be used.
seqinfo	<code>NULL</code> or a Seqinfo object containing information about the set of genomic sequences (chromosomes). Alternatively, a named numeric vector with genomic sequence names and lengths. Useful to set the sorting order of sequence names.
sequenceContextWidth, sequenceReference	Define the sequence context to be extracted around modified bases. By default (<code>sequenceContextWidth = 0</code>), no sequence context will be extracted, otherwise it will be returned in <code>rowData(x)\$sequenceContext</code> . See addSeqContext for details.
variantPositions	An optional <code>GPos</code> object with <code>seqnames</code> and coordinates of single nucleotide variant positions, to be used to construct read labels for allele-specific analysis. Ignored if <code>NULL</code> or <code>nAlnsToSample > 0</code> (sampling-mode).
modProbThreshold	A numeric scalar, indicating the modification probability threshold to use to classify a base as 'modified' or 'unmodified'. Only used if <code>level</code> is "summary".
trim	A logical scalar. If <code>TRUE</code> , the returned <code>SummarizedExperiment</code> object will only contain the positions overlapping the specified regions. If <code>FALSE</code> (default), the object will be extended to all positions covered by the reads overlapping regions. In both cases, only reads overlapping the specified regions are included.
maxCoverage	Integer scalar used to increase the maximal coverage for which samtools pileup will allocate cache memory. Ignored if <code>level="read"</code> . A value of <code>NULL</code> will use the samtools default (at the time of writing 8000). Large values will increase memory consumption. If the actual coverage is larger than this value, alignments may be silently ignored.
BPPARAM	A BiocParallelParam object that controls the number of parallel CPU threads to use for some of the steps in <code>readModBam()</code> . The default value is (<code>MulticoreParam(4L, RNGseed = 42L)</code>). If randomly sampling reads (<code>nAlnsToSample > 0</code>), make sure to set the <code>RNGseed</code> argument when constructing the <code>BPPARAM</code> object for reproducible results (see also <code>vignette("Random_Numbers", package = "BiocParallel")</code>).
verbose	Logical scalar. If <code>TRUE</code> , report on progress.

Value

A `SummarizedExperiment` object with genomic positions in rows and samples in columns. The assays depend on the value of the `level` argument (see above).

Author(s)

Michael Stadler, Charlotte Soneson

See Also

<https://samtools.github.io/hts-specs/SAMtags.pdf> describing the SAM ML and MM tags for base modifications.

Examples

```
modbamfile <- system.file("extdata", "6mA_1_10reads.bam",
                           package = "SingleMoleculeGenomicsIO")
readModBam(bamfiles = modbamfile, regions = "chr1:6940000-6955000",
           modbase = "a", verbose = TRUE,
           BPPARAM = BiocParallel::SerialParam())
```

<code>readModkitExtract</code>	<i>Read modkit extract file(s)</i>
--------------------------------	------------------------------------

Description

This function reads one or more modkit extract files and generates a single `SummarizedExperiment` object with rows representing positions and columns representing samples. The `SummarizedExperiment` object contains one assay (`mod_prob`) with modification probabilities for each position in each read. Unobserved read/position combinations are represented by NA.

Usage

```
readModkitExtract(
  fnames,
  modbase,
  filter = NULL,
  nrows = Inf,
  sampleAnnot = NULL,
  seqinfo = NULL,
  sequenceContextWidth = 0,
  sequenceReference = NULL,
  BPPARAM = MulticoreParam(4L, RNGseed = 42L),
  verbose = FALSE
)
```

Arguments

fnames	Character vector with one or several paths of <code>extract.tsv</code> files, such as generated by <code>modkit</code> . If <code>fnames</code> is a named vector, the names are used as prefixes for the column (read) names in the returned <code>SummarizedExperiment</code> object, and as column names for the object itself. Otherwise, the prefixes/names will be <code>s1, ..., sN</code> , where <code>N</code> is the length of <code>fnames</code> .
modbase	Character vector defining the modified base for each sample. If <code>modbase</code> is a named vector, the names should correspond to the names of <code>fnames</code> . Otherwise, it will be assumed that the elements are in the same order as the files in <code>fnames</code> . If <code>modbase</code> has length 1, the same modified base will be used for all samples.
filter	Either <code>NULL</code> (no filtering is performed), <code>"modkit"</code> (only records where the <code>"fail"</code> column in the extract file is <code>FALSE</code> are retained), or a named vector, providing filter thresholds (on the call probability) for the unmodified and modified bases, respectively. The names of the vector should be <code>"-"</code> and the value(s) in <code>modbase</code> .
nrows	Only read <code>nrows</code> rows of each input file.
sampleAnnot	A <code>data.frame</code> (or <code>NULL</code>) providing annotations for the samples. It must contain at least one column, named <code>"sample"</code> , which must contain all the values of <code>names(fnames)</code> . The provided annotations will be propagated to the returned <code>SummarizedExperiment</code> object.
seqinfo	<code>NULL</code> or a Seqinfo object containing information about the set of genomic sequences (chromosomes). Alternatively, a named numeric vector with genomic sequence names and lengths. Useful to set the sorting order of sequence names.
sequenceContextWidth, sequenceReference	Define the sequence context to be extracted around modified bases. By default (<code>sequenceContextWidth = 0</code>), no sequence context will be extracted, otherwise it will be returned in <code>rowData(x)\$sequenceContext</code> . See addSeqContext for details.
BPPARAM	A BiocParallelParam object that controls the number of parallel CPU threads to use for some of the steps in <code>readModkitExtract()</code> . The default value is (MulticoreParam (4L, <code>RNGseed = 42L</code>)).
verbose	If <code>TRUE</code> , report on progress.

Value

A [SummarizedExperiment](#) object with genomic positions in rows and samples in columns. The assay `"mod_prob"` contains per-read modification probabilities, with each column (sample) corresponding to a position-by-read [NaMatrix](#).

Author(s)

Charlotte Soneson, Michael Stadler

See Also

[modkit](#) software, [SummarizedExperiment](#) for the returned object type, [fread](#) for the function used to read the input files

Examples

```
extrfile <- system.file("extdata", "modkit_extract_rc_5mC_1.tsv.gz",
                        package = "SingleMoleculeGenomicsIO")

## ... no filtering
readModkitExtract(extrfile, modbase = "m", filter = NULL,
                  BPPARAM = BiocParallel::SerialParam())

## ... modkit filtering
readModkitExtract(extrfile, modbase = "m", filter = "modkit",
                  BPPARAM = BiocParallel::SerialParam())
```

refargToDNAStrngSet *Convert a supported reference sequence argument to a DNAStrngSet*

Description

This function takes a supported argument value specifying the reference sequence (e.g. genome) and returns it as a [DNAStrngSet](#) object. Currently supported argument values are:

"BSgenome:" : A BSgenome object

"DNAStrngSet" : A DNAStrngSet object

"character" : A path to a fasta file

Usage

```
refargToDNAStrngSet(sequenceReference)
```

Arguments

sequenceReference

A supported value defining the reference sequence.

Value

A [DNAStrngSet](#) object.

Author(s)

Michael Stadler, Charlotte Soneson

Examples

```
ref <- refargToDNAStrngSet(system.file("extdata", "reference.fa.gz",
                                       package = "SingleMoleculeGenomicsIO"))

ref
```

`regionStringToGRanges` *Convert character region(s) to a GRanges object*

Description

This function takes a character vector with one or several strings corresponding to genomic regions and converts them to a [GRanges](#) object. The supported forms of strings are the same as the ones supported by the `htslib` C API that is for example used in `samtools`:

"REF" or "REF:" : All of seqname REF

"REF:START" : Seqname REF from START to end of REF

"REF:-END" : Seqname REF from 1 to END

"REF:START-END" : Seqname REF from START to END

"." : All seqnames from 1 to their ends

Please note that the returned `GRanges` is generally parallel to the elements of `regions`, with the exception of `"."`: This special region is only allowed as a length-one `regions` argument and will possibly result several returned regions corresponding to the sequences in `seqinfo`.

Usage

```
regionStringToGRanges(regions, seqinfo = NULL, maxend = .Machine$integer.max)
```

Arguments

<code>regions</code>	Character vector with region strings. See details for supported forms.
<code>seqinfo</code>	One of <code>NULL</code> , an object for which a seqlengths method is available, such as a <code>BSgenome</code> , <code>Seqinfo</code> or <code>SummarizedExperiment</code> object or a named numeric vector with genomic sequence names and lengths. If not <code>NULL</code> , it will be used to obtain "END" for regions that do not specify it. Otherwise, <code>maxend</code> will be used for "END".
<code>maxend</code>	Integer scalar giving the maximal value for END in cases where it is not given in <code>regions</code> (for example "REF:START") and was also not provided through other parameters.

Value

A [GRanges](#) object.

Author(s)

Michael Stadler

Examples

```
regionStringToGRanges("chr1:6940000-6955000")
regionStringToGRanges("chr1", seqinfo = c(chr1 = 999))
regionStringToGRanges("chr1")
```

regroupReads	<i>Regroup reads</i>
--------------	----------------------

Description

Regroup reads into new "samples", defined by readGroups.

Usage

```
regroupReads(se, readGroups)

regroupReadsByColData(se, colNames, withinSample = FALSE)
```

Arguments

se	A SummarizedExperiment object, e.g. obtained by readModBam . Rows represent positions, and columns samples. The object should have at least one read-level assay.
readGroups	A named list, where each element corresponds to a new group of reads (a "sample").
colNames	A character vector corresponding to the names of annotation (colData) columns, the combination of which represent the desired grouping of the reads. The names can be either columns of colData(se) itself, or columns in a nested (read-level) annotation column of colData(se). In the latter case, the colNames should be of the form outerColName:innerColName, where outerColName is a column name in colData(se), and innerColName is a column name in each element of colData(se)[[outerColName]].
withinSample	A logical scalar, indicating whether the regrouping should be done within each current sample (column of se) or not. If FALSE (default), reads are pooled across samples before being regrouped.

Value

A [SummarizedExperiment](#) object with columns corresponding to the names of readGroups. Only the read-level assays will be returned (as summary-level assays are likely no longer consistent with the regrouped read-level assays).

Author(s)

Charlotte Soneson

Examples

```

library(SummarizedExperiment)
library(GenomicRanges)
modbamfiles <- system.file("extdata", c("6mA_1_10reads.bam",
                                         "6mA_2_10reads.bam"),
                           package = "SingleMoleculeGenomicsIO")
se <- readModBam(bamfiles = modbamfiles, regions = "chr1:6940000-6955000",
                modbase = "a", verbose = TRUE,
                variantPositions = GPos(seqnames = "chr1",
                                         pos = c(6940000, 6940500)),
                BPPARAM = BiocParallel::SerialParam())
# number of reads per sample
lapply(assay(se, "mod_prob"), ncol)
# define groups
groups <- list(g1 = c("s1-233e48a7-f379-4dcf-9270-958231125563",
                      "s2-d03efe3b-a45b-430b-9cb6-7e5882e4faf8"),
               g2 = "s1-92e906ae-cddb-4347-a114-bf9137761a8d",
               g3 = c("s2-034b625e-6230-4f8d-a713-3a32cd96c298",
                      "s1-d52a5f6a-a60a-4f85-913e-eada84bfbfb9"))
# regroup reads
sere <- regroupReads(se, readGroups = groups)
lapply(assay(sere, "mod_prob"), ncol)

# regroup by read annotation (variant label)
# ... across samples
sere <- regroupReadsByColData(se, colNames = "readInfo:variant_label",
                              withinSample = FALSE)

sere
lapply(assay(sere, "mod_prob"), ncol)
# ... within sample
sere <- regroupReadsByColData(se, colNames = "readInfo:variant_label",
                              withinSample = TRUE)

sere
lapply(assay(sere, "mod_prob"), ncol)

```

sampleEntropy

Sample Entropy of Time series signal

Description

sampleEntropy returns the sample entropy of a time-series signal see also https://en.wikipedia.org/wiki/Sample_entropy

Usage

```
sampleEntropy(data, m, r, maxStarts = 1000L, nThreads = 1L)
```

Arguments

data	Numeric vector
m	Integer. The pattern (embedding) length: Larger m captures finer structure but sharply reduces the number of matches, so it requires longer data and increases variance. Common choices are $m = 2$ or 3 for physiological time series.
r	Scaling parameter for the filtering factor. The filtering factor is $r \times$ standard deviation of the signal
maxStarts	Integer specifying the maximum number of signal start positions to evaluate (default 1000). If the time series has more possible starts, an evenly spaced subset of size maxStarts is used. Use -1 to include all possible starts.
nThreads	Integer giving the number of parallel OpenMP threads to use for calculation.

Details

This function calculates the Sample Entropy of a time-series vector given as argument. Sample Entropy is used to assess the complexity of physiological time-series signals. This C++ implementation is a modified version of: <https://gist.github.com/schochastics/e3684645763e93cbc2ed7d1b70ee5fe6>

Value

The Sample Entropy value of the time-series signal

See Also

[wikipedia:Sample_entropy](#) Multiscale entropy of biological signals

Examples

```
ts <- runif(100, 0, 1)
sampleEntropy(ts, m = 2L, r = 0.2)
```

subsetReads

Subset the reads from read-level assays

Description

This function takes read names or indices and subsets them from the specified assays of [RangedSummarizedExperiment](#) with read-level data. While subsetting of samples (the columns of the SummarizedExperiment object) can be done easily (e.g. `se[, 1]`), the reads are grouped by sample in read-level assays, and this function provides a more convenient way to subset these nested reads.

Usage

```
subsetReads(
  se,
  reads = NULL,
  randomSubset = NULL,
  prune = TRUE,
  invert = FALSE,
  removeAllNApos = FALSE,
  assayNameNA = "mod_prob"
)
```

Arguments

se	A RangedSummarizedExperiment , typically generated by readModBam or readModkitExtract .
reads	Defines reads to retain (or remove, if <code>invert=TRUE</code>). Either a character vector of read identifiers, or a named list in which names are samples from <code>colnames(se)</code> and the elements are index vectors (character, integer or logical) defining the reads for each sample.
randomSubset	A numeric scalar, enabling sampling of a random set of reads for each sample. If <code>randomSubset</code> is larger than or equal to 1, it defines the number of reads that will be sampled for each sample (if it is larger than the available number of reads, all reads will be retained). If <code>randomSubset</code> is in $[0, 1)$, it is interpreted as the fraction of reads to retain for each sample. If both <code>reads</code> and <code>randomSubset</code> are specified, <code>reads</code> takes precedence and <code>randomSubset</code> will be ignored.
prune	A logical scalar. If <code>TRUE</code> (the default), samples for which the subsetting retains none of the reads will be completely removed from the returned <code>SummarizedExperiment</code> (also from <code>colData</code> and from assays that do not store read-level data). If <code>FALSE</code> , such samples are retained (in the assays with read-level data as a zero-column <code>SparseMatrix</code>).
invert	A logical scalar. If <code>FALSE</code> (the default), only the reads defined by <code>reads</code> are retained. If <code>invert=TRUE</code> , all reads except the ones in <code>reads</code> are retained.
removeAllNApos	A logical scalar. If <code>TRUE</code> , remove all positions for which all values in <code>assayNameNA</code> are NA after the read subsetting.
assayNameNA	A character scalar corresponding to the name of a read-level assay (if <code>removeAllNApos</code> is <code>TRUE</code>).

Value

A subset [RangedSummarizedExperiment](#) object.

Author(s)

Michael Stadler, Charlotte Soneson

Examples

```
library(SummarizedExperiment)
modbamfiles <- system.file("extdata",
                           c("6mA_1_10reads.bam", "6mA_2_10reads.bam"),
                           package = "SingleMoleculeGenomicsIO")
se <- readModBam(modbamfiles, "chr1:6940000-6955000", "a",
                 BPPARAM = BiocParallel::SerialParam())
lapply(assay(se, "mod_prob"), colnames)

# subset by read identifiers
seSub <- subsetReads(se, c("s1-233e48a7-f379-4dcf-9270-958231125563",
                           "s2-034b625e-6230-4f8d-a713-3a32cd96c298"))
lapply(assay(seSub, "mod_prob"), colnames)

# subset by a list of indices
seSub <- subsetReads(se, list(s1 = c(1, 3),
                              s2 = c(TRUE, FALSE)))
lapply(assay(seSub, "mod_prob"), colnames)
```

Index

addReadLevelAssay, 3
addReadStats, 32
addReadStats (calcReadStats), 6
addSeqContext, 4, 7, 34, 35, 40, 42
annotateReadSegments, 5

BiocParallelParam, 7, 12, 14, 26, 34, 38, 40, 42
BSgenome, 4, 12, 20, 37

calcReadStats, 6, 26
checkSEValidity, 9
colData, 7, 8
countAlignmentsInBam, 10
countMismatchStatePairs, 11
countStatePairs, 13

DataFrame, 3
DNAStrngSet, 4, 12, 20, 35, 37, 43

estimateNoise, 15
estimateSNR, 16
expandSEToBaseSpace, 17
extractColDataColumns, 18
extractSeqContext, 4, 19

filterPositions, 21
filterReads, 22, 33
filterReadsModBam, 25
flattenReadLevelAssay, 27
fread, 35, 42

getBaseCoverageForBam, 29
getReadLevelAssayNames, 30
getReadLevelColDataNames, 31
getReadNamesBySample, 31
GRanges, 7, 11, 14, 17, 20, 21, 24, 36, 39, 44

IRangesList, 5

MulticoreParam, 7, 12, 14, 26, 34, 38, 40, 42

NaMatrix, 3, 36, 39, 42

plotReadStats, 32

qAlign, 11, 36

RangedSummarizedExperiment, 4, 7, 17, 18, 20, 33, 47, 48
readBedMethyl, 33
readMismatchBam, 35
readModBam, 7, 27, 38, 45, 48
readModkitExtract, 7, 27, 28, 41, 48
refargToDNAStrngSet, 43
regionStrngToGRanges, 44
regroupReads, 45
regroupReadsByColData (regroupReads), 45
resize, 20
rowRanges, 4, 20

sampleEntropy, 7, 46
Seqinfo, 12, 17, 22, 34, 37, 40, 42
seqlengths, 44
subsetReads, 47
SummarizedExperiment, 3–6, 8, 19, 27, 28, 32, 34, 35, 38, 41, 42, 45