

Package ‘bsseq’

October 15, 2018

Version 1.16.1

Encoding UTF-8

Title Analyze, manage and store bisulfite sequencing data

Description A collection of tools for analyzing and visualizing bisulfite sequencing data.

Depends R (>= 3.3), methods, BiocGenerics, GenomicRanges (>= 1.29.14), SummarizedExperiment (>= 1.9.18), parallel

Imports IRanges (>= 2.11.16), GenomeInfoDb, scales, stats, graphics, Biobase, locfit, gtools, data.table, S4Vectors, R.utils (>= 2.0.0), DelayedMatrixStats (>= 1.1.12), permute, limma, DelayedArray (>= 0.5.34), HDF5Array

Suggests RUnit, bsseqData, BiocStyle, rmarkdown, knitr,

Collate utils.R hasGRanges.R BSseq_class.R BSseqTstat_class.R
BSseq_utils.R combine.R read.bsmooth.R read.bismark.R BSmooth.R
BSmooth.tstat.R dmrFinder.R gof_stats.R plotting.R fisher.R
permutations.R BSmooth.fstat.R BSseqStat_class.R getStats.R
hdf5_utils.R DelayedArray_utils.R

License Artistic-2.0

VignetteBuilder knitr

URL <https://github.com/kasperdanielhansen/bsseq>

LazyData yes

LazyDataCompression xz

BugReports <https://github.com/kasperdanielhansen/bsseq/issues>

biocViews DNAMethylation

git_url <https://git.bioconductor.org/packages/bsseq>

git_branch RELEASE_3_7

git_last_commit 8494740

git_last_commit_date 2018-06-12

Date/Publication 2018-10-15

Author Kasper Daniel Hansen [aut, cre],
Peter Hickey [aut]

Maintainer Kasper Daniel Hansen <kasperdanielhansen@gmail.com>

R topics documented:

BS.chr22	2
BSmooth	3
BSmooth.tstat	4
BSseq	6
BSseq-class	8
BSseqStat-class	10
BSseqTstat-class	12
data.frame2GRanges	13
dmrFinder	14
fisherTests	15
getCoverage	17
getMeth	18
getStats	19
GoodnessOfFit	20
hasGRanges-class	22
plotRegion	23
read.bismark	25
read.bsmooth	27
read.umtab	28
Index	30

BS.chr22

*Whole-genome bisulfite sequencing for chromosome 22 from Lister et al.***Description**

This dataset represents chromosome 22 from the IMR90 cell line sequenced in Lister et al. Only CpG methylation are included (there were very few non-CpG loci). The two samples are two different extractions from the same cell line (ie. technical replicates), and are pooled in the analysis in the original paper.

Usage

```
data(BS.chr22)
```

Format

An object of class BSseq.

Details

All coordinates are in hg18.

Source

Obtained from http://neomorph.salk.edu/human_methylome/data.html specifically the files [mc_imr90_r1.tar.gz](#) and [mc_imr90_r2.tar.gz](#). A script which downloads these files and constructs the BS.chr22 object may be found in 'inst/scripts/get_BS.chr22.R', see the example.

References

R Lister et al. *Human DNA methylomes at base resolution show widespread epigenomic differences.* Nature (2009) 462, 315-322.

Examples

```
data(BS.chr22)
BS.chr22

script <- system.file("scripts", "get_BS.chr22.R", package = "bsseq")
script
readLines(script)
```

BSmooth	<i>BSmooth, smoothing bisulfite sequence data</i>
---------	---

Description

This implements the BSsmooth smoothing algorithm for bisulfite sequencing data.

Usage

```
BSmooth(BSseq, ns = 70, h = 1000, maxGap = 10^8,
        parallelBy = c("sample", "chromosome"), mc.preschedule = FALSE,
        mc.cores = 1, keep.se = FALSE, verbose = TRUE)
```

Arguments

BSseq	An object of class BSseq.
ns	The minimum number of methylation loci in a smoothing window.
h	The minimum smoothing window, in bases.
maxGap	The maximum gap between two methylation loci, before the smoothing is broken across the gap. The default smoothes each chromosome separately.
parallelBy	Should the computation be parallel by chromosome or sample, see details.
mc.preschedule	Passed to mclapply (should the tasks be prescheduled).
mc.cores	Passed to mclapply (the number of cores used). Note that setting mc.cores to a value greater than 1 is not supported on MS Windows, see the help page for mclapply.
keep.se	Should the estimated standard errors from the smoothing algorithm be kept. This will make the return object roughly 30 percent bigger and may not be used for anything.
verbose	Should the function be verbose.

Details

ns and h are passed to the locfit function. The bandwidth used is the maximum (in genomic distance) of the h and a width big enough to contain ns number of methylation loci.

The function uses the parallel package to do parallel computations. In order to use this, make sure your system have enough RAM, these are typically big objects. The computation can either be split by chromosome or by sample, which is better depends on the number of samples and how many concurrent smoothings may be done.

Value

An object of class BSseq, containing smoothed values and optionally standard errors for these.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

References

KD Hansen, B Langmead, and RA Irizarry. *BSmooth: from whole genome bisulfite sequencing reads to differentially methylated regions*. Genome Biology (2012) 13:R83. doi:[10.1186/gb-2012-13-10-r83](https://doi.org/10.1186/gb-2012-13-10-r83).

See Also

[locfit](#) in the locfit package, as well as [BSseq](#).

Examples

```
## Not run:
data(BS.chr22)
BS.fit <- BSmooth(BS.chr22, verbose = TRUE)
BS.fit

#-----
# An example using a HDF5Array-backed BSseq object
#
library(HDF5Array)
# See ?SummarizedExperiment::saveHDF5SummarizedExperiment for details
hdf5_BS.chr22 <- saveHDF5SummarizedExperiment(x = BS.chr22,
                                              dir = tempfile())
hdf5_BS.fit <- BSmooth(hdf5_BS.chr22, verbose = TRUE)
hdf5_BS.fit

## End(Not run)
```

BSmooth.tstat

Compute t-statistics based on smoothed whole-genome bisulfite sequencing data.

Description

Compute t-statistics based on smoothed whole-genome bisulfite sequencing data.

Usage

```
BSmooth.tstat(BSseq, group1, group2,
              estimate.var = c("same", "paired", "group2"), local.correct = TRUE,
              maxGap = NULL, qSd = 0.75, k = 101, mc.cores = 1, verbose = TRUE)
```

Arguments

BSseq	An object of class BSseq.
group1	A vector of sample names or indexes for the 'treatment' group.
group2	A vector of sample names or indexes for the 'control' group.
estimate.var	How is the variance estimated, see details.
local.correct	A logical; should local correction be used, see details.
maxGap	A scalar greater than 0, see details.
qSd	A scalar between 0 and 1, see details.
k	A positive scalar, see details.
mc.cores	The number of cores used. Note that setting mc.cores to a value greater than 1 is not supported on MS Windows, see the help page for mclapply.
verbose	Should the function be verbose?

Details

T-statistics are formed as the difference in means between group 1 and group 2 divided by an estimate of the standard deviation, assuming that the variance in the two groups are the same (same), that we have paired samples (paired) or only estimate the variance based on group 2 (group2). The standard deviation estimates are then smoothed (using a running mean with a width of k) and thresholded (using qSd which sets the minimum standard deviation to be the qSd-quantile). Optionally, the t-statistics are corrected for low-frequency patterns.

It is sometimes useful to use local.correct even if no large scale changes in methylation have been found; it makes the marginal distribution of the t-statistics more symmetric.

Additional details in the reference.

Value

An object of class BSseqTstat.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

References

KD Hansen, B Langmead, and RA Irizarry. *BSmooth: from whole genome bisulfite sequencing reads to differentially methylated regions*. Genome Biology (2012) 13:R83. doi:[10.1186/gb-2012-13-10-r83](https://doi.org/10.1186/gb-2012-13-10-r83).

See Also

[BSmooth](#) for the input object and [BSseq](#) for its class. [BSseqTstat](#) describes the return class. This function is likely to be followed by the use of [dmrFinder](#). And finally, see the package vignette(s) for more information on how to use it.

Examples

```

if(require(bsseqData)) {
  data(keepLoci.ex)
  data(BS.cancer.ex.fit)
  BS.cancer.ex.fit <- updateObject(BS.cancer.ex.fit)
  ## Remember to subset the BSseq object, see vignette for explanation
  BS.tstat <- BSmooth.tstat(BS.cancer.ex.fit[keepLoci.ex,],
                           group1 = c("C1", "C2", "C3"),
                           group2 = c("N1", "N2", "N3"),
                           estimate.var = "group2")

  BS.tstat
  ## This object is also stored as BS.cancer.ex.tstat in the
  ## bsseqData package

  #-----
  # An example using a HDF5Array-backed BSseq object
  #

  library(HDF5Array)
  # See ?SummarizedExperiment::saveHDF5SummarizedExperiment for details
  hdf5_BS.cancer.ex.fit <- saveHDF5SummarizedExperiment(
    x = BS.cancer.ex.fit[keepLoci.ex, ],
    dir = tempfile())
  hdf5_BS.tstat <- BSmooth.tstat(hdf5_BS.cancer.ex.fit,
                                group1 = c("C1", "C2", "C3"),
                                group2 = c("N1", "N2", "N3"),
                                estimate.var = "group2")

}

```

BSseq

*The constructor function for BSseq objects.***Description**

The constructor function for BSseq objects.

Usage

```

BSseq(M = NULL, Cov = NULL, coef = NULL, se.coef = NULL,
      trans = NULL, parameters = NULL, pData = NULL, gr = NULL,
      pos = NULL, chr = NULL, sampleNames = NULL, rmZeroCov = FALSE)

```

Arguments

M	A matrix-like object of methylation evidence (see 'Details' below).
Cov	A matrix-like object of coverage (see 'Details' below).
coef	A matrix-like object of smoothing estimates (see 'Details' below).
se.coef	A matrix-like object of smoothing standard errors (see 'Details' below).
trans	A smoothing transformation.
parameters	A list of smoothing parameters.

pData	An data.frame or DataFrame .
sampleNames	A vector of sample names.
gr	An object of type GRanges .
pos	A vector of locations.
chr	A vector of chromosomes.
rmZeroCov	Should genomic locations with zero coverage in all samples be removed.

Details

The 'M', 'Cov', 'coef', and 'se.coef' matrix-like objects will be coerced to [DelayedMatrix](#) objects; see `?DelayedArray::DelayedMatrix` for the full list of supported matrix-like objects. We recommend using [matrix](#) objects for in-memory storage of data and [HDF5Matrix](#) for on-disk storage of data.

Genomic locations are specified either through `gr` or through `chr` and `pos` but not both. There should be the same number of genomic locations as there are rows in the M and Cov matrix.

The argument `rmZeroCov` may be useful in order to reduce the size of the return object by removing methylation loci with zero coverage.

In case one or more methylation loci appears multiple times, the M and Cov matrices are summed over rows linked to the same methylation loci. See the example below.

Users should never have to specify `coef`, `se.coef`, `trans`, and `parameters`, this is for internal use (they are added by `BSmooth`).

`phenoData` is a way to specify pheno data (as known from the `ExpressionSet` and `eSet` classes), at a minimum `sampleNames` should be given (if they are not present, the function uses `col.names(M)`).

Value

An object of class `BSseq`.

Author(s)

Kasper Daniel Hansen <khansen@jhspsh.edu>

See Also

[BSseq](#)

Examples

```
M <- matrix(0:8, 3, 3)
Cov <- matrix(1:9, 3, 3)
BS1 <- BSseq(chr = c("chr1", "chr2", "chr1"), pos = c(1,2,3),
             M = M, Cov = Cov, sampleNames = c("A", "B", "C"))
BS1
BS2 <- BSseq(chr = c("chr1", "chr1", "chr1"), pos = c(1,1,1),
             M = M, Cov = Cov, sampleNames = c("A", "B", "C"))
BS2
```

```
#-----
# An example using a HDF5Array-backed BSseq object
#
library(HDF5Array)
```

```

# See ?HDF5Array::writeHDF5Array for details
hdf5_M <- writeHDF5Array(M)
hdf5_Cov <- writeHDF5Array(Cov)
hdf5_BS1 <- BSseq(chr = c("chr1", "chr2", "chr1"),
                 pos = c(1, 2, 3),
                 M = hdf5_M,
                 Cov = hdf5_Cov,
                 sampleNames = c("A", "B", "C"))

hdf5_BS1
hdf5_BS2 <- BSseq(chr = c("chr1", "chr1", "chr1"),
                 pos = c(1, 1, 1),
                 M = hdf5_M,
                 Cov = hdf5_Cov,
                 sampleNames = c("A", "B", "C"))

hdf5_BS2

```

BSseq-class

Class BSseq

Description

A class for representing whole-genome or capture bisulfite sequencing data.

Objects from the Class

An object from the class links together several pieces of information. (1) genomic locations stored as a GRanges object, a location by samples matrix of M values, a location by samples matrix of Cov (coverage) values and phenodata information. In addition, there are slots for representing smoothed data. This class is an extension of [RangedSummarizedExperiment](#).

Slots

trans: Object of class `function`. This function transforms the `coef` slot from the scale the smoothing was done to the 0-1 methylation scale.

parameters: Object of class `list`. A list of parameters representing for example how the data was smoothed.

Methods

[`signature(x = "BSseq")`]: Subsetting by location (using integer indices) or sample (using integers or sample names).

length Unlike for `RangedSummarizedExperiment`, `length()` is the number of methylation loci (equal to `length(granges(x))`).

sampleNames, sampleNames<- Sample names and its replacement function for the object. This is an alias for `colnames`.

pData, pData<- Obtain and replace the `pData` slot of the `phenoData` slot. This is an alias for `colData`.

show The show method.

combine This function combines two `BSseq` objects. The genomic locations of the new object is the union of the genomic locations of the individual objects. In addition, the methylation data matrices are placed next to each other (as appropriate wrt. the new genomic locations) and zeros are entered into the matrices as needed.

Utilities

This class extends [RangedSummarizedExperiment](#) and therefore inherits a number of useful GRanges methods that operate on the rowRanges slot, used for accessing and setting the genomic locations and also do subsetByOverlaps.

There are a number of almost methods-like functions for operating on objects of class BSseq, including getBSseq, collapseBSseq, and orderBSseq. They are detailed below.

`collapseBSseq(BSseq, columns)` is used to collapse an object of class BSseq. By collapsing we simply mean that certain columns (samples) are merge together by summing up the methylation evidence and coverage. This is a useful function if you start by reading in a dataset based on say flowcells and you (after QC) want to simply add a number of flowcells into one sample. The argument `columns` specify which samples are to be merged, in the following way: it is a character vector of new sample names, and the names of the column vector indicates which samples in the BSseq object are to be collapsed. If `columns` have the same length as the number of rows of BSseq (and has no names) it is assumed that the ordering corresponds to the sample ordering in BSseq.

`orderBSseq(BSseq, seqOrder = NULL)` simply orders an object of class BSseq according to (increasing) genomic locations. The `seqOrder` vector is a character vector of `seqnames(BSseq)` describing the order of the chromosomes. This is useful for ordering chr1 before chr10.

`chrSelectBSseq(BSseq, seqnames = NULL, order = FALSE)` subsets and optionally reorders an object of class BSseq. The `seqnames` vector is a character vector of `seqnames(BSseq)` describing which chromosomes should be retained. If `order` is TRUE, the chromosomes are also re-ordered using `orderBSseq`.

`getBSseq(BSseq, type = c("Cov", "M", "gr", "coef", "se.coef", "trans", "parameters"))` is a general accessor: is used to obtain a specific slot of an object of class BSseq. It is primarily intended for internal use in the package, for users we recommend `granges` to get the genomic locations, `getCoverage` to get the coverage slots and `getMeth` to get the smoothed values (if they exist).

`hasBeenSmoothed(BSseq)` This function returns a logical depending on whether or not the BSseq object has been smoothed using `BSmooth`.

`combineList(list, BACKEND = NULL)` This function function is a faster way of using `combine` on multiple BSseq objects. The input is a list, with each component an object of class BSseq. The (slower) alternative is to use `Reduce(combine, list)`.

The `BACKEND` argument determines which backend should be used for the 'M' and 'Cov' matrices and, if present, the 'coef' and 'se.coef' matrices (the latter two can only be combined if all objects have the same rowRanges). The default, `BACKEND = NULL`, corresponds to using `matrix` objects. See `?DelayedArray::setRealizationBackend` for alternative backends.

`strandCollapse(BSseq, shift = TRUE)` This function operates on a BSseq objects which has stranded loci (ie. loci where the strand is one of '+' or '-'). It will collapse the methylation and coverage information across the two strands, into one position. The argument `shift` indicates whether the positions for the loci on the reverse strand should be shifted one (ie. the positions for these loci are the positions of the 'G' in the 'CpG'; this is the case for Bismark output for example).

Coercion

Package versions 1.5.2 and 1.11.1 introduced a new version of representing 'BSseq' objects. You can update old serialized (saved) objects by invoking `x <- updateObject(x)`.

Assays

This class overrides the default implementation of assays to make it faster. Per default, no names are added to the returned data matrices.

Assay names can conveniently be obtained by the function `assayNames(x)`

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

The package vignette. [BSseq](#) for the constructor function. [RangedSummarizedExperiment](#) for the underlying class. [getBSseq](#), [getCoverage](#), and [getMeth](#) for accessing the data stored in the object and finally [BSmooth](#) for smoothing the bisulfite sequence data.

Examples

```
M <- matrix(1:9, 3,3)
colnames(M) <- c("A1", "A2", "A3")
BStest <- BSseq(pos = 1:3, chr = c("chr1", "chr2", "chr1"), M = M, Cov = M + 2)
chrSelectBSseq(BStest, seqnames = "chr1", order = TRUE)
collapseBSseq(BStest, columns = c("A1" = "A", "A2" = "A", "A3" = "B"))

#-----
# An example using a HDF5-backed BSseq object
#
library(HDF5Array)
# See ?HDF5Array::writeHDF5Array for details
hdf5_M <- writeHDF5Array(M)
# NOTE: HDF5Array::writeHDF5Array() doesn't preserve dimnames, so have to add
#       these manually
dimnames(hdf5_M) <- dimnames(M)
hdf5_BStest <- BSseq(pos = 1:3,
                    chr = c("chr1", "chr2", "chr1"),
                    M = hdf5_M,
                    Cov = hdf5_M + 2)
chrSelectBSseq(hdf5_BStest, seqnames = "chr1", order = TRUE)
collapseBSseq(hdf5_BStest, columns = c("A1" = "A", "A2" = "A", "A3" = "B"))
```

BSseqStat-class

Class BSseqStat

Description

A class for representing statistics for smoothed whole-genome bisulfite sequencing data.

Usage

```
BSseqStat(gr = NULL, stats = NULL, parameters = NULL)
```

Arguments

<code>gr</code>	The genomic locations as an object of class <code>GRanges</code> .
<code>stats</code>	The statistics, as a list of matrix-like objects (see 'Details' below).
<code>parameters</code>	A list of parameters.

Details

The matrix-like elements of the list in the 'stats' slot will be coerced to [DelayedMatrix](#) objects; see `?DelayedArray::DelayedMatrix` for the full list of supported matrix-like objects. We recommend using [matrix](#) objects for in-memory storage of data and [HDF5Matrix](#) for on-disk storage of data.

Objects from the Class

Objects can be created by calls of the form `BSseqStat(...)`. However, usually objects are returned by `BSmooth.fstat(...)` and not constructed by the user.

Slots

stats: This is a list of [DelayedMatrix](#) objects with list elements representing various statistics for methylation loci along the genome.

parameters: Object of class `list`. A list of parameters representing how the statistics were computed.

gr: Object of class `GRanges` giving genomic locations.

Extends

Class [hasGRanges](#), directly.

Methods

[The subsetting operator; one may only subset in one dimension, corresponding to methylation loci.

show The show method.

Utilities

This class extends [hasGRanges](#) and therefore inherits a number of useful `GRanges` methods that operate on the `gr` slot, used for accessing and setting the genomic locations and also do `subsetByOverlaps`.

Coercion

Package version 1.11.1 introduced a new version of representing 'BSseqStat' objects. You can update old serialized (saved) objects by invoking `x <- updateObject(x)`.

Author(s)

Kasper Daniel Hansen <khansen@jhsp.h.edu>

See Also

[hasGRanges](#) for accessing the genomic locations. [BSmooth.fstat](#) for a function that returns objects of class `BSseqStat`, and [smoothSds](#), [computeStat](#) and [dmrFinder](#) for functions that operate based on these statistics. Also see the more specialised [BSseqTstat](#).

BSseqTstat-class *Class BSseqTstat*

Description

A class for representing t-statistics for smoothed whole-genome bisulfite sequencing data.

Usage

```
BSseqTstat(gr = NULL, stats = NULL, parameters = NULL)
```

Arguments

<code>gr</code>	The genomic locations as an object of class <code>GRanges</code> .
<code>stats</code>	The statistics, as a matrix-like object (see 'Details' below).
<code>parameters</code>	A list of parameters.

Details

The 'stats' matrix-like object will be coerced to a [DelayedMatrix](#) object; see `?DelayedArray::DelayedMatrix` for the full list of supported matrix-like objects. We recommend using [matrix](#) objects for in-memory storage of data and [HDF5Matrix](#) for on-disk storage of data.

Objects from the Class

Objects can be created by calls of the form `BSseqTstat(...)`. However, usually objects are returned by `BSmooth.tstat(...)` and not constructed by the user..

Slots

stats: This is a [DelayedMatrix](#) object with columns representing various statistics for methylation loci along the genome.

parameters: Object of class `list`. A list of parameters representing how the t-statistics were computed.

gr: Object of class `GRanges` giving genomic locations.

Extends

Class [hasGRanges](#), directly.

Methods

[The subsetting operator; one may only subset in one dimension, corresponding to methylation loci.

show The show method.

Utilities

This class extends `hasGRanges` and therefore inherits a number of useful `GRanges` methods that operate on the `gr` slot, used for accessing and setting the genomic locations and also do `subsetByOverlaps`.

Coercion

Package version 1.11.1 introduced a new version of representing 'BSseqTstat' objects. You can update old serialized (saved) objects by invoking `x <- updateObject(x)`.

Author(s)

Kasper Daniel Hansen <khansen@jhsp.h.edu>

See Also

The package vignette(s). [hasGRanges](#) for accessing the genomic locations. [BSmooth.tstat](#) for a function that returns objects of class BSseqTstat, and [dmrFinder](#) for a function that computes DMRs based on the t-statistics. Also see [BS.cancer.ex.tstat](#) for an example of the class in the **bsseqData** package.

data.frame2GRanges *Converts a data frame to a GRanges.*

Description

Converting a data.frame to a GRanges object. The data.frame needs columns like chr, start and end (strand is optional). Additional columns may be kept in the GRanges object.

Usage

```
data.frame2GRanges(df, keepColumns = FALSE, ignoreStrand = FALSE)
```

Arguments

df	A data.frame with columns chr or seqnames, start, end and optionally a strand column.
keepColumns	In case df has additional columns, should these columns be stored as metadata columns on the return GRanges or should they be discarded.
ignoreStrand	In case df has a strand column, should this column be ignored.

Value

An object of class GRanges

Note

In case df has rownames, they will be used as names for the return object.

Author(s)

Kasper Daniel Hansen <khansen@jhsp.h.edu>

Examples

```
df <- data.frame(chr = "chr1", start = 1:3, end = 2:4,  
                strand = c("+", "-", "+"))  
data.frame2GRanges(df, ignoreStrand = TRUE)
```

dmrFinder	<i>Finds differentially methylated regions for whole genome bisulfite sequencing data.</i>
-----------	--

Description

Finds differentially methylated regions for whole genome bisulfite sequencing data. Essentially identifies regions of the genome where all methylation loci have an associated t-statistic that is beyond a (low, high) cutoff.

Usage

```
dmrFinder(bstat, cutoff = NULL, qcutoff = c(0.025, 0.975),
          maxGap=300, stat = "tstat.corrected", verbose = TRUE)
```

Arguments

bstat	An object of class BSseqStat or BSseqTstat.
cutoff	The cutoff of the t-statistics. This should be a vector of length two giving the (low, high) cutoff. If NULL, see qcutoff.
qcutoff	In case cutoff is NULL, compute the cutoff using these quantiles of the t-statistic.
maxGap	If two methylation loci are separated by this distance, break a possible DMR. This guarantees that the return DMRs have CpGs that are this distance from each other.
stat	Which statistic should be used?
verbose	Should the function be verbose?

Details

The workhorse function is BSmooth.tstat which sets up a t-statistic for a comparison between two groups.

Note that post-processing of these DMRs are likely to be necessary, filtering for example for length (or number of loci).

Value

A data.frame with columns

start, end, width, chr	genomic locations and width.
n	The number of methylation loci.
invdensity	Average length per loci.
group1.mean	The mean methylation level across samples and loci in 'group1'.
group2.mean	The mean methylation level across samples and loci in 'group2'.
meanDiff	The mean difference in methylation level; the difference between group1.mean and group2.mean.
idxStart, idxEnd, cluster	Internal use.

areaStat	The 'area' of the t-statistic; equal to the sum of the t-statistics for the individual methylation loci.
direction	either 'hyper' or 'hypo'.
areaStat.corrected	Only present if column = "tstat.corrected", contains the area of the corrected t-statistics.

Author(s)

Kasper Daniel Hansen <khansen@jhsp.h.edu>.

References

KD Hansen, B Langmead, and RA Irizarry. *BSmooth: from whole genome bisulfite sequencing reads to differentially methylated regions*. Genome Biology (2012) 13:R83. doi:[10.1186/gb-2012-13-10-r83](https://doi.org/10.1186/gb-2012-13-10-r83).

See Also

[BSmooth.tstat](#) for the function constructing the input object, and [BSseqTstat](#) for its class. In the example below, we use [BS.cancer.ex.tstat](#) as the actual input object. Also see the package vignette(s) for a detailed example.

Examples

```
if(require(bsseqData)) {
  dmrs0 <- dmrFinder(BS.cancer.ex.tstat, cutoff = c(-4.6, 4.6), verbose = TRUE)
  dmrs <- subset(dmrs0, abs(meanDiff) > 0.1 & n >= 3)
}
```

fisherTests

Compute Fisher-tests for a BSseq object

Description

A function to compute Fisher-tests for an object of class BSseq.

Usage

```
fisherTests(BSseq, group1, group2, lookup = NULL,
  returnLookup = TRUE, mc.cores = 1, verbose = TRUE)
```

Arguments

BSseq	An object of class BSseq.
group1	A vector of sample names or indexes for the 'treatment' group.
group2	A vector of sample names or indexes for the 'control' group.
lookup	A 'lookup' object, see details.
returnLookup	Should a 'lookup' object be returned, see details.
mc.cores	The number of cores used. Note that setting mc.cores to a value greater than 1 is not supported on MS Windows, see the help page for mclapply.
verbose	Should the function be verbose.

Details

This function computes row-wise Fisher's exact tests. It uses an internal lookup table so rows which forms equivalent 2x2 tables are group together and only a single test is computed. If `returnLookup` is TRUE the return object contains the lookup table which may be feed to another call to the function using the `lookup` argument.

If `group1`, `group2` designates more than 1 sample, the samples are added together before testing.

This function can use multiple cores on the same computer.

This test cannot model biological variability.

Value

if `returnLookup` is TRUE, a list with components `results` and `lookup`, otherwise just the `results` component. The `results` (component) is a matrix with the same number of rows as the `BSseq` argument and 2 columns `p.value` (the unadjusted p-values) and `log2OR` (log2 transformation of the odds ratio).

Author(s)

Kasper Daniel Hansen <khansen@jhsp.h.edu>

See Also

[fisher.test](#) for information about Fisher's test. [mclapply](#) for the `mc.cores` argument.

Examples

```
M <- matrix(1:9, 3,3)
colnames(M) <- c("A1", "A2", "A3")
BStest <- BSseq(pos = 1:3, chr = c("chr1", "chr2", "chr1"),
               M = M, Cov = M + 2)
results <- fisherTests(BStest, group1 = "A1", group2 = "A2",
                      returnLookup = TRUE)
results

#-----
# An example using a HDF5Array-backed BSseq object
#
library(HDF5Array)
# See ?SummarizedExperiment::saveHDF5SummarizedExperiment for details
hdf5_BStest <- saveHDF5SummarizedExperiment(x = BStest,
                                           dir = tempfile())
results <- fisherTests(hdf5_BStest,
                      group1 = "A1",
                      group2 = "A2",
                      returnLookup = TRUE)
results
```

getCoverage	<i>Obtain coverage for BSseq objects.</i>
-------------	---

Description

Obtain coverage for BSseq objects.

Usage

```
getCoverage(BSseq, regions = NULL, type = c("Cov", "M"),
  what = c("perBase", "perRegionAverage", "perRegionTotal"))
```

Arguments

BSseq	An object of class BSseq.
regions	An optional data.frame or GenomicRanges object specifying a number of genomic regions.
type	This returns either coverage or the total evidence for methylation at the loci.
what	The type of return object, see details.

Value

NOTE: The return type of getCoverage varies depending on its arguments.

If regions are not specified (regions = NULL) a [DelayedMatrix](#) object (what = "perBase") is returned. This will either contain the per-base coverage, the average coverage, or the genome total coverage (depending on value of what).

If what = "perBase" and regions are specified, a list is returned. Each element of the list is a [DelayedMatrix](#) object corresponding to the genomic loci inside the region. It is conceptually the same as splitting the coverage by region.

If what = "perRegionAverage" or what = "perRegionTotal" and regions are specified the return value is a [DelayedMatrix](#) object. Each row of the [DelayedMatrix](#) corresponds to a region and contains either the average coverage or the total coverage in the region.

Author(s)

Kasper Daniel Hansen <khansen@jhsp.h.edu>.

See Also

[BSseq](#) for the BSseq class.

Examples

```
data(BS.chr22)
head(getCoverage(BS.chr22, type = "M"))
reg <- GRanges(seqnames = c("chr22", "chr22"),
  ranges = IRanges(start = c(1, 2*10^7), end = c(2*10^7 + 1, 4*10^7)))
getCoverage(BS.chr22, regions = reg, what = "perRegionAverage")
cList <- getCoverage(BS.chr22, regions = reg)
length(cList)
head(cList[[1]])
```

```

#-----
# An example using a HDF5Array-backed BSseq object
#

library(HDF5Array)
# See ?SummarizedExperiment::saveHDF5SummarizedExperiment for details
hdf5_BS.chr22 <- saveHDF5SummarizedExperiment(x = BS.chr22,
                                             dir = tempfile())

head(getCoverage(hdf5_BS.chr22, type = "M"))
reg <- GRanges(seqnames = c("chr22", "chr22"),
               ranges = IRanges(start = c(1, 2 * 10 ^ 7),
                                end = c(2 * 10 ^ 7 + 1, 4 * 10 ^ 7)))
getCoverage(hdf5_BS.chr22, regions = reg, what = "perRegionAverage")
hdf5_cList <- getCoverage(hdf5_BS.chr22, regions = reg)
length(hdf5_cList)
head(hdf5_cList[[1]])

```

getMeth

Obtain methylation estimates for BSseq objects.

Description

Obtain methylation estimates for BSseq objects, both smoothed and raw.

Usage

```

getMeth(BSseq, regions = NULL, type = c("smooth", "raw"),
        what = c("perBase", "perRegion"), confint = FALSE, alpha = 0.95)

```

Arguments

BSseq	An object of class BSseq.
regions	An optional data.frame or GenomicRanges object specifying a number of genomic regions.
type	This returns either smoothed or raw estimates of the methylation level.
what	The type of return object, see details.
confint	Should a confidence interval be return for the methylation estimates (see below). This is only supported if what is equal to perBase.
alpha	alpha value for the confidence interval.

Value

NOTE: The return type of getMeth varies depending on its arguments.

If region = NULL the what argument is ignored. This is also the only situation in which confint = TRUE is supported. The return value is either a [DelayedMatrix](#) (confint = FALSE or a list with three [DelayedMatrix](#) components confint = TRUE (meth, upper and lower), giving the methylation estimates and (optionally) confidence intervals.

Confidence intervals for type = "smooth" is based on standard errors from the smoothing algorithm (if present). Otherwise it is based on pointwise confidence intervals for binomial distributions described in Agresti (see below), specifically the score confidence interval.

If regions are specified, `what = "perBase"` will make the function return a list, each element of the list being a [DelayedMatrix](#) corresponding to a genomic region (and each row of the [DelayedMatrix](#) being a loci inside the region). If `what = "perRegion"` the function returns a [DelayedMatrix](#), with each row corresponding to a region and containing the average methylation level in that region.

Note

A BSseq object needs to be smoothed by the function `BSmooth` in order to support `type = "smooth"`.

Author(s)

Kasper Daniel Hansen <khansen@jhspsh.edu>.

References

A Agresti and B Coull. *Approximate Is Better than "Exact" for Interval Estimation of Binomial Proportions*. The American Statistician (1998) 52:119-126.

See Also

[BSseq](#) for the BSseq class and [BSmooth](#) for smoothing such an object.

Examples

```
data(BS.chr22)
head(getMeth(BS.chr22, type = "raw"))
reg <- GRanges(seqnames = c("chr22", "chr22"),
  ranges = IRanges(start = c(1, 2*10^7), end = c(2*10^7 + 1, 4*10^7)))
head(getMeth(BS.chr22, regions = reg, type = "raw", what = "perBase"))

#-----
# An example using a HDF5Array-backed BSseq object
#

library(HDF5Array)
# See ?SummarizedExperiment::saveHDF5SummarizedExperiment for details
hdf5_BS.chr22 <- saveHDF5SummarizedExperiment(x = BS.chr22,
  dir = tempfile())
head(getMeth(hdf5_BS.chr22, type = "raw"))
head(getMeth(hdf5_BS.chr22, regions = reg, type = "raw", what = "perBase"))
```

getStats

Obtain statistics from a BSseqTstat object

Description

Essentially an accessor function for the statistics of a BSseqTstat object.

Usage

```
getStats(bstat, regions = NULL, ...)
```

Arguments

<code>bstat</code>	An object of class <code>BSseqStat</code> or <code>BSseqTstat</code> .
<code>regions</code>	An optional <code>data.frame</code> or <code>GenomicRanges</code> object specifying a number of genomic regions.
<code>...</code>	Additional arguments passed to the different backends based on the class of <code>bstat</code> ; see <code>Details</code> .

Details

Additional argument when the `bstat` object is of class `BSseqTstat`:

stat Which statistics column should be obtained.

Value

An object of class `data.frame` possible restricted to the regions specified.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

[BSseqTstat](#) for the `BSseqTstat` class, and [getCoverage](#) and [getMeth](#) for similar functions, operating on objects of class `BSseq`.

Examples

```
if(require(bsseqData)) {
  data(BS.cancer.ex.tstat)
  head(getStats(BS.cancer.ex.tstat))
  reg <- GRanges(seqnames = c("chr22", "chr22"),
    ranges = IRanges(start = c(1, 2*10^7), end = c(2*10^7 +1, 4*10^7)))
  head(getStats(BS.cancer.ex.tstat, regions = reg))
}
```

GoodnessOfFit

Binomial and poisson goodness of fit statistics for BSseq objects

Description

Binomial and poisson goodness of fit statistics for `BSseq` objects, including plotting capability.

Usage

```
poissonGoodnessOfFit(BSseq, nQuantiles = 10^5)
binomialGoodnessOfFit(BSseq, method = c("MLE"), nQuantiles = 10^5)
## S3 method for class 'chisqGoodnessOfFit'
print(x, ...)
## S3 method for class 'chisqGoodnessOfFit'
plot(x, type = c("chisq", "pvalue"), plotCol = TRUE, qqline = TRUE,
  pch = 16, cex = 0.75, ...)
```

Arguments

BSseq	An object of class BSseq.
x	A chisqGoodnessOfFit object (as produced by poissonGoodnessOfFit or binomialGoodnessOfFit).
nQuantiles	The number of (evenly-spaced) quantiles stored in the return object.
method	How is the parameter estimated.
type	Are the chisq or the p-values being plotted.
plotCol	Should the extreme quantiles be colored.
qqline	Add a qqline.
pch, cex	Plotting symbols and size.
...	Additional arguments being passed to qqplot (for plot) or ignored (for print).

Details

These functions compute and plot goodness of fit statistics for BSseq objects. For each methylation loci, the Poisson goodness of fit statistic tests whether the coverage (at that loci) is independent and identically Poisson distributed across the samples. In a similar fashion, the Binomial goodness of fit statistic tests whether the number of reads supporting methylation are independent and identically binomial distributed across samples (with different size parameters given by the coverage vector).

These functions do not handle NA values.

Value

The plotting method is invoked for its side effect. Both poissonGoodnessOfFit and binomialGoodnessOfFit returns an object of class chisqGoodnessOfFit which is a list with components

chisq	a vector of Chisq values.
quantiles	a vector of quantiles (of the chisq values).
df	degrees of freedom

Author(s)

Kasper Daniel Hansen <khansen@jhsp.h.edu>

See Also

For the plotting method, see qqplot.

Examples

```
if(require(bsseqData)) {
  data(BS.cancer.ex)
  BS.cancer.ex <- updateObject(BS.cancer.ex)
  gof <- poissonGoodnessOfFit(BS.cancer.ex)
  plot(gof)

#-----
# An example using a HDF5Array-backed BSseq object
#

  library(HDF5Array)
```

```

# See ?SummarizedExperiment::saveHDF5SummarizedExperiment for details
hdf5_BS.cancer.ex <- saveHDF5SummarizedExperiment(x = BS.cancer.ex,
                                                dir = tempfile())
hdf5_gof <- poissonGoodnessOfFit(hdf5_BS.cancer.ex)
plot(hdf5_gof)
}

```

hasGRanges-class *Class hasGRanges*

Description

A class with a GRanges slot, used as a building block for other classes. Provides basic accessor functions etc.

Objects from the Class

Objects can be created by calls of the form `new("hasGRanges", ...)`.

Slots

gr: Object of class GRanges.

Methods

"[" Subsets a single dimension.

granges Get the GRanges object representing genomic locations.

start,start<-,end,end<-,width,width<- Start, end and width for the genomic locations of the object, also replacement functions. This accessor functions operate directly on the gr slot.

strand,strand<- Getting and setting the strand of the genomic locations (the gr slot).

seqlengths,seqlengths<- Getting and setting the seqlengths of the genomic locations (the gr slot).

seqlevels,seqlevels<- Getting and setting the seqlevels of the genomic locations (the gr slot).

seqnames,seqnames<- Getting and setting the seqnames of the genomic locations (the gr slot).

show The show method.

findOverlaps (query = "hasGRanges", subject = "hasGRanges"): finds overlaps between the granges() of the two objects.

findOverlaps (query = "GenomicRanges", subject = "hasGRanges"): finds overlaps between query and the granges() of the subject.

findOverlaps (query = "hasGRanges", subject = "GenomicRanges"): finds overlaps between the granges() of the query and the subject.

subsetByOverlaps (query = "hasGRanges", subject = "hasGRanges"): Subset the query, keeping the genomic locations that overlaps the subject.

subsetByOverlaps (query = "hasGRanges", subject = "GenomicRanges"): Subset the query, keeping the genomic locations that overlaps the subject.

subsetByOverlaps (query = "GenomicRanges", subject = "hasGRanges"): Subset the query, keeping the genomic locations that overlaps the subject.

Note

If you extend the `hasGRanges` class, you should consider writing a `subset` method (`[]`), and a `show` method. If the new class supports single index subsetting, the `subsetByOverlaps` methods will automatically extend.

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

Examples

```
showClass("hasGRanges")
```

plotRegion

Plotting BSmooth methylation estimates

Description

Functions for plotting BSmooth methylation estimates. Typically used to display differentially methylated regions.

Usage

```
plotRegion(BSseq, region = NULL, extend = 0, main = "",
  addRegions = NULL, annoTrack = NULL, cex.anno = 1,
  geneTrack = NULL, cex.gene = 1.5, col = NULL, lty = NULL,
  lwd = NULL, BSseqStat = NULL, stat = "tstat.corrected",
  stat.col = "black", stat.lwd = 1, stat.lty = 1, stat.ylim = c(-8, 8),
  mainWithWidth = TRUE, regionCol = alpha("red", 0.1), addTicks = TRUE,
  addPoints = FALSE, pointsMinCov = 5, highlightMain = FALSE)
```

```
plotManyRegions(BSseq, regions = NULL, extend = 0, main = "",
  addRegions = NULL, annoTrack = NULL, cex.anno = 1,
  geneTrack = NULL, cex.gene = 1.5, col = NULL, lty = NULL,
  lwd = NULL, BSseqStat = NULL, stat = "tstat.corrected",
  stat.col = "black", stat.lwd = 1, stat.lty = 1, stat.ylim = c(-8, 8),
  mainWithWidth = TRUE, regionCol = alpha("red", 0.1), addTicks = TRUE,
  addPoints = FALSE, pointsMinCov = 5, highlightMain = FALSE,
  verbose = TRUE)
```

Arguments

<code>BSseq</code>	An object of class <code>BSseq</code> .
<code>region</code>	A <code>data.frame</code> (with <code>start</code> , <code>end</code> and <code>chr</code> columns) with 1 row or <code>GRanges</code> of length 1. If <code>region</code> is <code>NULL</code> the entire <code>BSseq</code> argument is plotted.
<code>regions</code>	A <code>data.frame</code> (with <code>start</code> , <code>end</code> and <code>chr</code> columns) or <code>GRanges</code> .
<code>extend</code>	Describes how much the plotting region should be extended in either direction. The total width of the plot is equal to the width of the region plus twice <code>extend</code> .
<code>main</code>	The plot title. The default is to construct a title with information about which genomic region is being plotted.

addRegions	A set of additional regions to be highlighted on the plots. As the regions argument.
annoTrack	A named list of GRanges objects. Each component is a track and the names of the list are the track names. Each track will be plotted as solid bars, and we routinely display information such as CpG islands, exons, etc.
cex.anno	cex argument when plotting annoTrack.
geneTrack	EXPERIMENTAL: A data.frame with columns: chr, start, end, gene_ID, exon_number, strand, gene_name, isoforms. This interface is under active development and subject to change.
cex.gene	cex argument when plotting geneTrack.
col	The color of the methylation estimates, see details.
lty	The line type of the methylation estimates, see details.
lwd	The line width of the methylation estimates, see details.
BSseqStat	An object of class BSseqStat. If present, a new panel will be shown with the t-statistics.
stat	Which statistics will be plotted (only used if BSseqStat is not NULL.)
stat.col	color for the statistics plot.
stat.lwd	line width for the statistics plot.
stat.lty	line type for the statistics plot.
stat.ylim	y-limits for the statistics plot.
mainWithWidth	Should the default title include information about width of the plot region.
regionCol	The color used for highlighting the region.
addTicks	Should tick marks showing the location of methylation loci, be added?
addPoints	Should the individual unsmoothed methylation estimates be plotted. This usually leads to a very confusing plot, but may be useful for diagnostic purposes.
pointsMinCov	The minimum coverage a methylation loci need in order for the raw methylation estimates to be plotted. Useful for filtering out low coverage loci. Only used if addPoints = TRUE.
highlightMain	Should the plot region be highlighted?
verbose	Should the function be verbose?

Details

The correct choice of aspect ratio depends on the width of the plotting region. We tend to use width = 10, height = 5.

plotManyRegions is used to plot many regions (hundreds or thousands), and is substantially quicker than repeated calls to plotRegion.

This function has grown to be rather complicated over time. For custom plotting, it is sometimes useful to use the function definition as a skeleton and directly modify the code.

Value

This function is invoked for its side effect: producing a plot.

Author(s)

Kasper Daniel Hansen <khansen@jhsp.h.edu>

See Also

The package vignette has an extended example.

read.bismark	<i>Parsing output from the Bismark alignment suite.</i>
--------------	---

Description

Parsing output from the Bismark alignment suite.

Usage

```
read.bismark(files,
             sampleNames,
             rmZeroCov = FALSE,
             strandCollapse = TRUE,
             fileType = c("cov", "oldBedGraph", "cytosineReport"),
             mc.cores = 1,
             verbose = TRUE,
             BACKEND = NULL)
```

Arguments

files	Input files. Each sample is in a different file. Input files are created by running Bismark's methylation extractor; see Note for details.
sampleNames	sample names, based on the order of files.
rmZeroCov	Should methylation loci that have zero coverage in all samples be removed. This will result in a much smaller object if the data originates from (targeted) capture bisulfite sequencing.
strandCollapse	Should strand-symmetric methylation loci, e.g., CpGs, be collapsed across strands. This option is only available if fileType = "cytosineReport" since the other file types do not contain the necessary strand information.
fileType	The format of the input file; see Note for details.
mc.cores	The number of cores used. Note that setting mc.cores to a value greater than 1 is not supported on MS Windows, see the help page for mclapply.
verbose	Make the function verbose.
BACKEND	The backend used for the 'M' and 'Cov' matrices. The default, NULL, corresponds to using matrix objects. See <code>?DelayedArray::setRealizationBackend</code> for alternative backends.

Value

An object of class [BSseq](#).

Note

Input files can either be gzipped or not.

The user must specify the relevant file format via the `fileType` argument. The format of the output of the Bismark alignment suite will depend on the version of Bismark and on various user-specified options. Please consult the Bismark documentation and the Bismark RELEASE NOTES (http://www.bioinformatics.bbsrc.ac.uk/projects/bismark/RELEASE_NOTES.txt) for the definitive list of changes between versions. When possible, it is strongly recommended that you use the most recent version of Bismark.

The "cov" and "oldBedGraph" formats both have six columns ("chromosome", "position", "strand", "methylation percentage", "count methylated", "count unmethylated"). If you are using a recent version of Bismark ($v \geq 0.10.0$) then the standard file extension for this file is ".cov". If, however, you are using an older version of Bismark ($v < 0.10.0$) then the file extension will be ".bedGraph". Please note that the ".bedGraph" file created in recent versions of Bismark ($v \geq 0.10.0$) is **not** suitable for analysis with `bsseq` because it only contains the "methylation percentage" and not "count methylated" nor "count unmethylated".

The "cytosineReport" format has seven columns ("chromosome", "position", "strand", "count methylated", "count unmethylated", "C-context", "trinucleotide context"). There is no standard file extension for this file. The "C-context" and "trinucleotide context" columns are not currently used by `bsseq`.

The following is a list of some issues to be aware of when using output from Bismark's methylation extractor:

- The program to extract methylation counts was named `methylation_extractor` in older versions of Bismark ($v < 0.8.0$) and re-named `bismark_methylation_extractor` in recent versions of Bismark ($v \geq 0.8.0$). Furthermore, very old versions of Bismark ($v < 0.7.7$) required that user run a separate script (called something like `genome_methylation_bismark2bedGraph`) to create the six-column "cov"/"oldBedGraph" file.
- The `--counts` and `--bedGraph` arguments must be supplied to `methylation_extractor/bismark_methylation_extractor` in order to use the output with `bsseq::read.bismark()`.
- The genomic co-ordinates of the Bismark output file may be zero-based or one-based depending on whether the `--zero_based` argument is used. Furthermore, the default co-ordinate system varies by version of Bismark. `bsseq` makes no assumptions about the basis of the genomic co-ordinates and it is left to the user to ensure that the appropriate basis is used in the analysis of their data. Since Bioconductor packages and [GRanges](#) use one-based co-ordinates, it is recommended that your Bismark files are also one-based.

Author(s)

Peter Hickey <peter.hickey@gmail.com>

See Also

[read.bsmooth](#) for parsing output from the BSmooth alignment suite. [read.umtab](#) for parsing legacy (old) formats from the BSmooth alignment suite. [collapseBSseq](#) for collapse (merging or summing) the data in two different directories.

Examples

```
infile <- system.file("extdata/test_data.fastq_bismark.bismark.cov.gz",
  package = 'bsseq')
bismarkBSseq <- read.bismark(files = infile,
```

```

                                sampleNames = "test_data",
                                rmZeroCov = FALSE,
                                strandCollapse = FALSE,
                                fileType = "cov",
                                verbose = TRUE)

bismarkBSseq

#-----
# An example constructing a HDF5Array-backed BSseq object
#
library(HDF5Array)
# See ?DelayedArray::setRealizationBackend for details
hdf5_bismarkBSseq <- read.bismark(files = infile,
                                sampleNames = "test_data",
                                rmZeroCov = FALSE,
                                strandCollapse = FALSE,
                                fileType = "cov",
                                verbose = TRUE,
                                BACKEND = "HDF5Array")

```

read.bssmooth

Parsing output from the BSmooth alignment suite

Description

Parsing output from the BSmooth alignment suite.

Usage

```
read.bssmooth(dirs, sampleNames = NULL, seqnames = NULL,
              returnRaw = FALSE, qualityCutoff = 20, rmZeroCov = FALSE,
              verbose = TRUE)
```

Arguments

<code>dirs</code>	Input directories. Usually each sample is in a different directory, or perhaps each (sample, lane) is a different directory.
<code>sampleNames</code>	sample names, based on the order of <code>dirs</code> . If <code>NULL</code> either set to <code>basename(dirs)</code> (if unique) or <code>dirs</code> .
<code>seqnames</code>	The default is to read all BSmooth output files in <code>dirs</code> . Using this argument, it is possible to restrict this to only files with names in <code>seqnames</code> (apart from <code>.cpg.tsv</code> and optionally <code>.gz</code>).
<code>returnRaw</code>	Should the function return the complete information in the output files?
<code>qualityCutoff</code>	Only use evidence (methylated and unmethylated evidence) for a given methylation loci, if the base in the read has a quality greater than this cutoff.
<code>rmZeroCov</code>	Should methylation loci that have zero coverage in all samples be removed. This will result in a much smaller object if the data originates from (targeted) capture bisulfite sequencing.
<code>verbose</code>	Make the function verbose.

Value

Either an object of class BSseq (if returnRaw = FALSE) or a list of GRanges which each component coming from a directory.

Note

Input files can either be gzipped or not. Gzipping the input files results in much greater speed of reading (and saves space), so it is recommended.

We are working on making this function faster and less memory hungry.

Author(s)

Kasper Daniel Hansen <khansen@jhspsh.edu>

See Also

[read.umtab](#) for parsing legacy (old) formats from the BSmooth alignment suite. [collapseBSseq](#) for collapse (merging or summing) the data in two different directories.

read.umtab	<i>Parsing UM tab files (legacy output) containing output from the BSmooth aligner.</i>
------------	---

Description

Parsing UM tab files containing output from the bisulfite aligner Merman. This is two different legacy formats, which we keep around. These functions are likely to be deprecated in the future.

Usage

```
read.umtab(dirs, sampleNames = NULL, rmZeroCov = FALSE,
  pattern = NULL, keepU = c("U10", "U20", "U30", "U40"),
  keepM = c("M10", "M20", "M30", "M40"), verbose = TRUE)
```

```
read.umtab2(dirs, sampleNames = NULL, rmZeroCov = FALSE,
  readCycle = FALSE, keepFilt = FALSE,
  pattern = NULL, keepU, keepM, verbose = TRUE)
```

Arguments

dirs	Input directories. Usually each sample is in a different directory.
pattern	An optional pattern, see <code>list.files</code> in the base package.
sampleNames	sample names, based on the order of <code>dirs</code> .
rmZeroCov	Should methylation loci that have zero coverage in all samples be removed. This will result in a much smaller object if the data originates from (targeted) capture bisulfite sequencing.
keepU	A vector of U columns which are kept.
keepM	A vector of M columns which are kept.
readCycle	Should the cycle columns be returned?
keepFilt	Should the filter columns be returned?
verbose	Make the function verbose.

Details

read.umtab2 is newer than read.umtab and both process output from older versions of the BSsmooth alignment suite (versions older than 0.6.1). These functions are likely to be deprecated in the future. Newer output from the BSsmooth alignment suite can be parsed using read.bsmooth.

A script using this function can be found in the bsseqData package, in the file 'scripts/create_BS.cancer.R'.

Value

Both functions returns lists, the components are

BSdata	An object of class BSseq containing the methylation evidence.
GC	A vector of local GC content values.
Map	A vector of local mapability values.
Mcy	A matrix of the number of unique M cycles.
Ucy	A matrix of the number of unique U cycles.
chr	A vector of chromosome names.
pos	A vector of genomic positions.
M	A matrix representing methylation evidence.
U	A matrix representing un-methylation evidence.
csums	Description of 'comp2'

Author(s)

Kasper Daniel Hansen <khansen@jhsph.edu>

See Also

[read.bsmooth](#).

Examples

```
## Not run:
require(bsseqData)
umDir <- system.file("umtab", package = "bsseqData")
sampleNames <- list.files(umDir)
dirs <- file.path(umDir, sampleNames, "umtab")
umData <- read.umtab(dirs, sampleNames)

## End(Not run)
```

Index

*Topic **classes**

BSseq-class, 8
BSseqStat-class, 10
BSseqTstat-class, 12
hasGRanges-class, 22

*Topic **datasets**

BS.chr22, 2

[,BSseq-method (BSseq-class), 8
[,BSseqStat,ANY,ANY,ANY-method
(BSseqStat-class), 10
[,BSseqStat-method (BSseqStat-class), 10
[,BSseqTstat,ANY,ANY,ANY-method
(BSseqTstat-class), 12
[,BSseqTstat-method (BSseqTstat-class),
12
[,hasGRanges,ANY,ANY,ANY-method
(hasGRanges-class), 22
[,hasGRanges-method (hasGRanges-class),
22

assayNames,BSseq-method (BSseq-class), 8
assays,BSseq-method (BSseq-class), 8

binomialGoodnessOfFit (GoodnessOfFit),
20

BS.cancer.ex.tstat, 13, 15

BS.chr22, 2

BSmooth, 3, 5, 10, 19

BSmooth.fstat, 11

BSmooth.tstat, 4, 13, 15

BSseq, 4, 5, 6, 7, 9, 10, 17, 19, 25

BSseq-class, 8

BSseqStat (BSseqStat-class), 10

BSseqStat-class, 10

BSseqTstat, 5, 11, 15, 20

BSseqTstat (BSseqTstat-class), 12

BSseqTstat-class, 12

chisqGoodnessOfFit (GoodnessOfFit), 20

chrSelectBSseq (BSseq-class), 8

collapseBSseq, 26, 28

collapseBSseq (BSseq-class), 8

combine,BSseq,BSseq-method
(BSseq-class), 8

combineList (BSseq-class), 8
computeStat, 11

data.frame2GRanges, 13

DataFrame, 7

DelayedMatrix, 7, 11, 12, 17–19

dmrFinder, 5, 11, 13, 14

end,hasGRanges-method
(hasGRanges-class), 22

end<-,hasGRanges-method
(hasGRanges-class), 22

findOverlaps,GenomicRanges,hasGRanges-method
(hasGRanges-class), 22

findOverlaps,hasGRanges,GenomicRanges-method
(hasGRanges-class), 22

findOverlaps,hasGRanges,hasGRanges-method
(hasGRanges-class), 22

fisher.test, 16

fisherTests, 15

getBSseq, 10

getBSseq (BSseq-class), 8

getCoverage, 10, 17, 20

getMeth, 10, 18, 20

getStats, 19

GoodnessOfFit, 20

GRanges, 7, 26

granges,hasGRanges-method
(hasGRanges-class), 22

hasBeenSmoothed (BSseq-class), 8

hasGRanges, 11–13

hasGRanges-class, 22

HDF5Matrix, 7, 11, 12

length,BSseq-method (BSseq-class), 8

length,hasGRanges-method
(hasGRanges-class), 22

locfit, 4

matrix, 7, 9, 11, 12, 25

mclapply, 16

- orderBSseq (BSseq-class), 8
- pData, BSseq-method (BSseq-class), 8
- pData<- , BSseq, data.frame-method (BSseq-class), 8
- pData<- , BSseq, DataFrame-method (BSseq-class), 8
- plot.chisqGoodnessOfFit (GoodnessOfFit), 20
- plotManyRegions (plotRegion), 23
- plotRegion, 23
- poissonGoodnessOfFit (GoodnessOfFit), 20
- print.chisqGoodnessOfFit (GoodnessOfFit), 20

- RangedSummarizedExperiment, 8–10
- read.bismark, 25
- read.bsmooth, 26, 27, 29
- read.umtab, 26, 28, 28
- read.umtab2 (read.umtab), 28

- sampleNames, BSseq-method (BSseq-class), 8
- sampleNames<- , BSseq, ANY-method (BSseq-class), 8
- seqlengths, hasGRanges-method (hasGRanges-class), 22
- seqlengths<- , hasGRanges-method (hasGRanges-class), 22
- seqlevels, hasGRanges-method (hasGRanges-class), 22
- seqlevels<- , hasGRanges-method (hasGRanges-class), 22
- seqnames, hasGRanges-method (hasGRanges-class), 22
- seqnames<- , hasGRanges-method (hasGRanges-class), 22
- setRealizationBackend, 9, 25
- show, BSseq-method (BSseq-class), 8
- show, BSseqStat-method (BSseqStat-class), 10
- show, BSseqTstat-method (BSseqTstat-class), 12
- smoothSds, 11
- start, hasGRanges-method (hasGRanges-class), 22
- start<- , hasGRanges-method (hasGRanges-class), 22
- strand, hasGRanges-method (hasGRanges-class), 22
- strand<- , hasGRanges, ANY-method (hasGRanges-class), 22
- strand<- , hasGRanges-method (hasGRanges-class), 22
- strandCollapse (BSseq-class), 8
- updateObject, BSseq-method (BSseq-class), 8
- updateObject, BSseqStat-method (BSseqStat-class), 10
- updateObject, BSseqTstat-method (BSseqTstat-class), 12
- width, hasGRanges-method (hasGRanges-class), 22
- width<- , hasGRanges-method (hasGRanges-class), 22