Package 'biscuiteer'

May 15, 2025

Type Package

Title Convenience Functions for Biscuit

Description A test harness for bsseq loading of Biscuit output, summarization of WGBS data over defined regions and in mappable samples, with or without imputation, dropping of mostly-NA rows, age estimates, etc.

Version 1.23.0

Date 2024-03-01

URL https://github.com/trichelab/biscuiteer

BugReports https://github.com/trichelab/biscuiteer/issues

License GPL-3

Depends R (>= 4.1.0), biscuiteerData, bsseq

Imports readr, qualV, Matrix, impute, HDF5Array, S4Vectors, Rsamtools, data.table, Biobase, GenomicRanges, IRanges, BiocGenerics, VariantAnnotation, DelayedMatrixStats, SummarizedExperiment, GenomeInfoDb, Mus.musculus, Homo.sapiens, matrixStats, rtracklayer, QDNAseq, dmrseq, methods, utils, R.utils, gtools, BiocParallel

Suggests DSS, covr, knitr, rmarkdown, markdown, rlang, scmeth, pkgdown, roxygen2, testthat, QDNAseq.hg19, QDNAseq.mm10, BiocStyle

biocViews DataImport, MethylSeq, DNAMethylation

Encoding UTF-8

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

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

git_branch devel

git_last_commit 700b6be

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-05-15

Author Tim Triche [aut],
Wanding Zhou [aut],
Benjamin Johnson [aut],
Jacob Morrison [aut, cre],
Lyong Heo [aut],
James Eapen [aut]

Maintainer Jacob Morrison <jacob.morrison@vai.org>

Contents

`biscuiteer-package` *Convenience Functions for Biscuit*

Description

A test harness for bsseq loading of Biscuit output, summarization of WGBS data over defined regions and in mappable samples (with or without imputation, dropping mostly-NA rows, age estimates, etc.)

Author(s)

Timothy J Triche Jr <Tim.Triche@vai.org>, Wanding Zhou <Wanding.Zhou@vai.org>, Ben Johnson <Ben.Johnson@vai.org>, Jacob Morrison <Jacob.Morrison@vai.org>, Lyong Heo <Lyong.Heo@vai.org>

See Also

Useful links:

- https://github.com/trichelab/biscuiteer
- Report bugs at https://github.com/trichelab/biscuiteer/issues

Examples

```
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                    merged = FALSE)
```

`atRegions` *Summarize a bsseq dataset over defined regions*

Description

Calls summarizeBsSeqOver to summarize a bsseq object over provided DNA regions. Useful for exploring genomic data using cBioPortal.

Usage

```
atRegions(bsseq, regions, mappings = NULL, nm = "POETICname", ...)
```

Arguments

bsseq	A bsseq object
regions	A GRanges or GRangesList of regions
mappings	A mapping table with rownames(mappings) == colnames(bsseq) (DEFAULT: NULL)
nm	Column of the mapping table to map to (DEFAULT: "POETICname")
...	Other arguments to pass to summarizeBsSeqOver

Value

```
GRanges with summarized information about the bsseq object
        for the given DNA regions
```

Examples

```
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                    merged = FALSE)

reg <- GRanges(seqnames = rep("chr11",5),
               strand = rep("*",5),
               ranges = IRanges(start = c(0,2.8e6,1.17e7,1.38e7,1.69e7),
                                end= c(2.8e6,1.17e7,1.38e7,1.69e7,2.2e7))
             )
regions <- atRegions(bsseq = bisc, regions = reg)
```

---

binCoverage *Bin CpG or CpH coverage to simplify and improve CNA "sketching"*

Description

Example usage for E-M

Usage

```
binCoverage(
  bsseq,
  bins,
  which = NULL,
  QDNAseq = TRUE,
  readLen = 100,
  paired = TRUE
)
```

Arguments

bsseq	A bsseq object - supplied to getCoverage()
bins	Bins to summarize over - from tileGenome or QDNAseq.xxYY
which	Limit to specific regions? - functions as an import() (DEFAULT: NULL)
QDNAseq	Return a QDNAseqReadCounts? - if FALSE, returns a GRanges (DEFAULT: TRUE)
readLen	Correction factor for coverage - read length in bp (DEFAULT: 100)
paired	Whether the data are from paired-end sequencing (DEFAULT: TRUE)

Details

NOTE: As of early Sept 2019, QDNAseq did not have hg38 capabilities. If you desire to use the hg38 genome, biscuiteer suggests you use a GRanges object to define your bins.

NOTE: As of late July 2020, biscuiteer has started implemented support for hg38, hg19, mm10, and mm9 for bisulfite-specific features, including adaptive GC-content computation and SV integration for adjusting CNV ends.

Value

Binned read counts

Examples

```
bins <- GRanges(seqnames = rep("chr11",10),
                strand = rep("*",10),
                ranges = IRanges(start=100000*0:9, width=100000)
               )

reg <- GRanges(seqnames = rep("chr11",5),
               strand = rep("*",5),
               ranges = IRanges(start = c(0,2.8e6,1.17e7,1.38e7,1.69e7),
                                end= c(2.8e6,1.17e7,1.38e7,1.69e7,2.2e7))
               )

orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                    merged = FALSE)

bc <- binCoverage(bsseq = bisc, bins = bins, which = reg, QDNAseq = FALSE)
```

---

biscuiteer-methods *bsseq class methods (VCF-centric) added by biscuiteer*

Description

See biscuiteer manpage for package description

Usage

```
## S4 method for signature 'BSseq'
samples(object)

## S4 method for signature 'BSseq'
header(x)

## S4 method for signature 'BSseq'
meta(x)

## S4 method for signature 'BSseq'
fixed(x)

## S4 method for signature 'BSseq'
info(x)

## S4 method for signature 'BSseq,ANY'
geno(x)
```

Arguments

object	A bsseq object, preferably with !is.null(metadata(x)$vcfHeader)
x	A bsseq object, preferably with !is.null(metadata(x)$vcfHeader)

Details

biscuiteer adds VariantAnnotation methods to BSseq objects with VCF headers: samples,header,meta,fixed,info,geno

Due to inherited method signatures, the argument (singular) to the method may be named x or it may be named object. Either way, it is a BSseq object.

These add to the existing methods defined in package bsseq for class BSseq: [,length,sampleNames,sampleNames<-,pDa

Those add to the methods BSseq inherits from SummarizedExperiment, such as: colData,rowRanges,metadata,subset,

Most of the biscuiteer methods operate on the VCF header, which readBiscuit likes to stuff into the metadata slot of BSseq objects it produces. Some may be handy for populating a BSseq object with QC stats, or querying those.

Value

Depends on the method - usually a List-like object of some sort

See Also

RangedSummarizedExperiment

VCFHeader-class

BSseq-class

BSseq

biscuitMetadata *Biscuit metadata from VCF header*

Description

Returns metadata from a Biscuit run using either a supplied VCF file or the vcfHeader metadata element from the bsseq object

Usage

```
biscuitMetadata(bsseq = NULL, VCF = NULL)

getBiscuitMetadata(bsseq = NULL, VCF = NULL)
```

Arguments

bsseq	A bsseq object with a vcfHeader element (DEFAULT: NULL)
VCF	A tabix'ed VCF file (can just be the header information) from which the bsseq vcfHeader element is derived (DEFAULT: NULL)

Value

Information regarding the Biscuit run

Functions

- getBiscuitMetadata(): Alias for biscuitMetadata

Examples

```
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                    merged = FALSE)

meta <- biscuitMetadata(bisc)
```

byChromArm *A simple parallization step*

Description

This function splits an object by chromosome arm, which tends to make parallelization much easier, as cross-arm dependencies are unusual. Therefore, the larger chromosomes can be split across processes or machines without worrying much about data starvation for processes on smaller chromosomes.

Usage

```
byChromArm(x, arms = NULL)
```

Arguments

x	Any object with a GRanges in it: bsseq, SummarizedExperiment...
arms	Another GRanges, but specifying chromosome arms (DEFAULT: NULL)

Value

```
A list, List, or *list, with pieces of x by chromosome arm
```

Examples

```
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                    merged = FALSE)

reg <- GRanges(seqnames = rep("chr11",5),
               strand = rep("*",5),
               ranges = IRanges(start = c(0,2.8e6,1.17e7,1.38e7,1.69e7),
                                end= c(2.8e6,1.17e7,1.38e7,1.69e7,2.2e7))
               )
names(reg) <- as.character(reg)

arms <- byChromArm(bisc, arms = reg)
```

---

byExtremality *Choose loci or features by extremality*

Description

This function finds the k most extremal features (features above a certain fraction of the Bernoulli variance) in 'bsseq' and returns their values.

Usage

```
byExtremality(bsseq, r = NULL, k = 500)
```

Arguments

bsseq	A bsseq object
r	Regions to consider - NULL covers all loci (DEFAULT: NULL)
k	How many rows/regions to return (DEFAULT: 500)

Details

For DNA methylation, particularly when summarized across regions, we can do better (a lot better) than MAD. Since we know: max(SD(X_j)) if X_j ~ Beta(a, b) < max(SD(X_j)) if X_j ~ Bernoulli(a/(a+b)) for X with a known mean and standard deviation (SD), then we can solve for (a+b) by MoM. We can then define the extremality by: extremality = sd(X_j) / bernoulliSD(mean(X_j))

Value

A GRanges object with methylation values sorted by extremality

Examples

```
shuf_bed <- system.file("extdata", "MCF7_Cunha_chr11p15_shuffled.bed.gz",
                        package="biscuiteer")
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
shuf_vcf <- system.file("extdata",
                        "MCF7_Cunha_shuffled_header_only.vcf.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata",
                        "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc1 <- readBiscuit(BEDfile = shuf_bed, VCFfile = shuf_vcf,
                     merged = FALSE)
bisc2 <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                     merged = FALSE)

reg <- GRanges(seqnames = rep("chr11",5),
               strand = rep("*",5),
               ranges = IRanges(start = c(0,2.8e6,1.17e7,1.38e7,1.69e7),
                                end= c(2.8e6,1.17e7,1.38e7,1.69e7,2.2e7))
               )

comb <- unionize(bisc1, bisc2)

ext <- byExtremality(comb, r = reg)
```

---

checkBiscuitBED *Inspect Biscuit VCF and BED files*

Description

A BED checker for Biscuit CpG/CpH output (BED-like format with 2 or 3 columns per sample). By default, files with more than 50 million loci will be processed iteratively, since data.table tends to run into problems with gzipped joint CpH files.

Usage

```
checkBiscuitBED(
  BEDfile,
  VCFfile,
  merged,
```

```
  sampleNames = NULL,
  chunkSize = 5e+07,
  hdf5 = FALSE,
  sparse = TRUE,
  how = c("data.table", "readr"),
  chr = NULL
)
```

Arguments

BEDfile	A BED-like file - must be compressed and tabix'ed
VCFfile	A VCF file - must be compressed and tabix'ed. Only the header information is needed.
merged	Is this merged CpG data?
sampleNames	Names of samples - NULL: create names, vector: assign names, data.frame: make pData (DEFAULT: NULL)
chunkSize	For files longer than yieldSize number of lines long, chunk the file (DEFAULT: 5e7)
hdf5	Use HDF5 arrays for backing the data? Using HDF5-backed arrays stores the data in a HDF5 file on disk, rather than loading entire object into memory. This allows for analyses to be done on memory-limited systems at the small cost of slightly reduced return times. (DEFAULT: FALSE)
sparse	Use sparse Matrix objects for the data? If TRUE, use a Matrix object for sparse matrices (matrices with many zeroes in them) (DEFAULT: TRUE)
how	How to load the data - "data.table" or "readr"? (DEFAULT: data.table)
chr	Load a specific chromosome to rbind() later? (DEFAULT: NULL)

Details

Input BED and VCF files must be tabix'ed. No exceptions!

Value

```
Parameters to be supplied to makeBSseq
```

See Also

readBiscuit

Examples

```
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
params <- checkBiscuitBED(BEDfile = orig_bed, VCFfile = orig_vcf,
                          merged = FALSE)
```

clocks *clocks*

Description

Epigenetic clock data

Usage

```
data(clocks, package="biscuiteer")
```

Details

Source: See inst/scripts/clocks.R for how the clocks data object was generated. For more information about sources, see the descriptions in ?getClock and ?WGBSage. Return type: data.frame

condenseSampleNames *Simplify sample names for a bsseq object*

Description

Utility function for extracting sample names from tabix'ed sample columns, assuming a VCF-naming scheme (such as Sample_1.foo, Sample_1.bar or Sample1_foo, Sample1_bar).

Usage

```
condenseSampleNames(tbx, stride, trailing = "\\.$")
```

Arguments

tbx	A TabixFile instance to parse
stride	How many columns per sample
trailing	Trailing character to trim (DEFAULT: "\.$")

Value

```
A character vector of sample names (longest common
          substrings)
```

Examples

```
library(Rsamtools)
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
if (length(headerTabix(orig_bed)$header) > 0) {
  condenseSampleNames(orig_bed, 2)
}
```

CpGindex *Measure methylation status for PRCs or PMDs*

Description

WARNING: This function will be deprecated in the next Bioconductor release

Usage

```
CpGindex(bsseq, CGIs = NULL, PRCs = NULL, WCGW = NULL, PMDs = NULL)
```

Arguments

bsseq	A BSseq object
CGIs	A GRanges of CpG island regions - HMM CGIs if NULL (DEFAULT: NULL)
PRCs	A GRanges of Polycomb targets - H9 state 23 low-meth if NULL (DEFAULT: NULL)
WCGW	A GRanges of solo-WCGW sites - PMD WCGWs if NULL (DEFAULT: NULL)
PMDs	A GRanges of hypomethylating regions - PMDs if NULL (DEFAULT: NULL)

Details

Measures hypermethylation at PRCs in CGIs or hypomethylation at WCGWs in PMDs

At some point in some conference call somewhere, a collaborator suggested that a simple index of Polycomb repressor complex (PRC) binding site hyper- methylation and CpG-poor "partially methylated domain" (PMD) hypomethylation would be a handy yardstick for both deterministic and stochastic changes associated with proliferation, aging, and cancer. This function provides such an index by compiling measures of aberrant hyper- and hypo-methylation along with the ratio of hyper- to hypo-methylation. (The logic for this is that while the phenomena tend to occur together, there are many exceptions) The resulting measures can provide a high-level summary of proliferation-, aging-, and/or disease-associated changes in DNA methylation across samples.

The choice of defaults is fairly straightforward: in 2006, three independent groups reported recurrent hypermethylation in cancer at sites marked by both H3K4me3 (activating) and H3K27me3 (repressive) histone marks in embryonic stem cells; these became known as "bivalent" sites. The Roadmap Epigenome project performed ChIP-seq on hundreds of normal primary tissues and cell line results from the ENCODE project to generate a systematic catalog of "chromatin states" alongside dozens of whole-genome bisulfite sequencing experiments in the same tissues. We used both to generate a default atlas of bivalent (Polycomb-associated and transcriptionally-poised) sites from H9 human embryonic stem cells which retain low DNA methylation across normal (non-placental) REMC tissues. In 2018, Zhou and Dinh (Nature Genetics) found isolated (AT)CG(AT) sites, or "solo-WCGW" motifs, in common PMDs as the most universal barometer of proliferation- and aging-associated methylation loss in mammalian cells, so we use their solo-WCGW sites in common PMDs as the default measure for hypomethylation. The resulting CpGindex is a vector of length 3 for each sample: hypermethylation, hypomethylation, and their ratio.

We suggest fitting a model for the composition of bulk samples (tumor/normal, tissue1/tissue2, or whatever is most appropriate) prior to drawing any firm conclusions from the results of this function. For example, a mixture of two-thirds normal tissue and one-third tumor tissue may produce the same or lower degree of hyper/hypomethylation than high-tumor-content cell-free DNA samples from the blood plasma of the same patient. Intuition is simply not a reliable guide in such situations,

which occur with some regularity. If orthogonal estimates of purity/composition are available (flow cytometry, ploidy, yield of filtered cfDNA), it is a Very Good Idea to include them.

The default for this function is to use the HMM-defined CpG islands from Hao Wu's paper (Wu, Caffo, Jaffee, Irizarry & Feinberg, Biostatistics 2010) as generic "hypermethylation" targets inside of "bivalent" (H3K27me3+H3K4me3) sites (identified in H9 embryonic stem cells & unmethylated across normals), and the solo-WCGW sites within common partially methylated domains from Wanding Zhou and Huy Dinh's paper (Zhou, Dinh, et al, Nat Genetics 2018) as genetic "hypomethylation" targets (as above, obvious caveats about tissue specificity and user-supplied possibilities exist, but the defaults are sane for many purposes, and can be exchanged for whatever targets a user wishes).

The function returns all three components of the "CpG index", comprised of hyperCGI and hypoPMD (i.e. hyper, hypo, and their ratio). The PMD "score" is a base-coverage-weighted average of losses to solo-WCGW bases within PMDs; the PRC score is similarly base-coverage-weighted but across HMM CGI CpGs, within polycomb repressor complex sites (by default, the subset of state 23 segments in the 25-state, 12-mark ChromImpute model for H9 which have less than 10 percent CpG methylation across the CpG-island-overlapping segment in all normal primary tissues and cells from the Reference Epigenome project). By providing different targets and/or regions, users can customize as needed.

The return value is a CpGindex object, which is really just a DataFrame that knows about the regions at which it was summarized, and reminds the user of this when they implicitly call the show method on it.

Value

A CpGindex (DataFrame w/cols `hyper`, `hypo`, `ratio` + 2 GRs)

`ENSR_subset.hg19` *ENSR_subset data from hg19 genome*

Description

Subset of ENSEMBL regulatory build regions for hg19 genome

Usage

```
data(ENSR_subset.hg19, package="biscuiteer")
```

Details

Source URL: homo_sapiens.GRCh37.Regulatory_Build.regulatory_features.20161117.gff.gz (regions that overlap Inifinium annotation manifests - described at http://zwdzwd.github.io/InfiniumAnnotation - are selected for final GRanges) Source type: GFF Return type: GRanges

ENSR_subset.hg38 *ENSR_subset data from hg38 genome*

Description

Subset of ENSEMBL regulatory build regions for hg19 genome

Usage

```
data(ENSR_subset.hg38, package="biscuiteer")
```

Details

Source URL: homo_sapiens.GRCh38.Regulatory_Build.regulatory_features.20161111.gff.gz (regions that overlap Inifinium annotation manifests - described at http://zwdzwd.github.io/InfiniumAnnotation - are selected for final GRanges) Source type: GFF Return type: GRanges

extremality *Compute fraction of a Bernoulli variance*

Description

Works efficiently on matrices and DelayedMatrix objects. Note that it is possible for "raw" extremality to be greater than 1, so this function does a second pass to correct for this.

Usage

```
extremality(x, raw = FALSE)
```

Arguments

x	A rectangular object with proportions in it
raw	Skip the correction pass? (DEFAULT: FALSE)

Value

The extremality of each row (if more than one) of the object

Examples

```
x <- rnorm(100, mean=0.5, sd=0.15)
x <- matrix(x, nrow=50, ncol=2)

ext <- extremality(x, raw=TRUE)
```

fexpit *Helper function: expanded expit*

Description

Helper function: expanded expit

Usage

```
fexpit(x, sqz = 1e-06)
```

Arguments

x	a vector of values between -Inf and +Inf
sqz	the amount by which we 'squoze', default is .000001

Value

```
a vector of values between 0 and 1 inclusive
```

Examples

```
set.seed(1234)
x <- rnorm(n=1000)
summary(x)

sqz <- 1 / (10**6)
p <- fexpit(x, sqz=sqz)
summary(p)

all( (abs(x - flogit(p)) / x) < sqz )
all( abs(x - flogit(fexpit(x))) < sqz )
```

filterLoci *Filter loci with zero coverage*

Description

Function potentially used to be a part of dmrseq. Included here to avoid dmrseq failing due to any number of reasons related to lack of coverage.

Usage

```
filterLoci(bsseq, testCovariate)
```

Arguments

bsseq	A bsseq object for filtering
testCovariate	The name of the pData column dmrseq will test on

Details

The code is adapted from the precheck loop of dmrseq::dmrseq

Value

A bsseq object ready for dmrseq to use

See Also

dmrseq

WGBSeq

RRBSeq

Examples

```
shuf_bed <- system.file("extdata", "MCF7_Cunha_chr11p15_shuffled.bed.gz",
                        package="biscuiteer")
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
shuf_vcf <- system.file("extdata",
                        "MCF7_Cunha_shuffled_header_only.vcf.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata",
                        "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc1 <- readBiscuit(BEDfile = shuf_bed, VCFfile = shuf_vcf,
                     merged = FALSE)
bisc2 <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                     merged = FALSE)

comb <- unionize(bisc1, bisc2)

filt <- filterLoci(comb, "sampleNames")
```

---

fixAge *Turn 'epigenetic clock' into actual age*

Description

Uses Horvath-type 'epigenetic clock' raw output to project into actual ages

Usage

```
fixAge(x, adult = 21)
```

Arguments

x	Untransformed or raw prediction(s)
adult	Age of adulthood (DEFAULT: 21)

Details

The 'Epigenetic Clock' (Horvath 2012) and similar schemes use a number of CpG loci (or regions, or perhaps CpH loci – it doesn't really matter what) to estimate the chronological/biological age of samples from DNA methylation with pre-trained feature weights (coefficients) for each region/locus.

All of these type of clocks use a nonlinear output transformation which switches from an exponential growth model for children into a linear model for adults, where `adult` is an arbitrary number (by default and custom, that number is 21; elsewhere it can sometimes be seen as 20, but all known epi-age transformation functions quietly add 1 to the constant internally).

This function implements the above standard output transformation step.

Value

`Transformed prediction(s)`

Examples

```
clock <- getClock(genome="hg38")
score <- clock$gr$score

age <- fixAge(score)
```

`fixNAs` *Replace NAs with another value*

Description

Useful for coercing matrices into how bsseq is expecting the M matrix to be.

Usage

```
fixNAs(x, y = 0, sparseMatrix = FALSE)
```

Arguments

`x`	The matrix-like object containing NAs to fix
`y`	The value to replace the NAs with (DEFAULT: 0)
`sparseMatrix`	Make the result a Matrix object? (DEFAULT: FALSE)

Value

`x with no NAs (possibly a sparse Matrix)`

Examples

```
nom <- c(rep(c(1,4,NA,9,NA,NA,7,NA), 5))
no_nas <- fixNAs(nom)
```

flogit *Helper function: squeezed logit*

Description

Helper function: squeezed logit

Usage

```
flogit(p, sqz = 1e-06)
```

Arguments

p	a vector of values between 0 and 1 inclusive
sqz	the amount by which to 'squeeze', default is .000001

Value

```
a vector of values between -Inf and +Inf
```

Examples

```
set.seed(1234)
p <- runif(n=1000)
summary(p)

sqz <- 1 / (10**6)
x <- flogit(p, sqz=sqz)
summary(x)

all( abs(p - fexpit(x, sqz=sqz)) < sqz )
all( abs(p - fexpit(flogit(p, sqz=sqz), sqz=sqz)) < sqz )
```

getClock *Retrieve 'epigenetic clock' models*

Description

Biscuiteer supports several 'epigenetic clock' models. This function retrieves the various models.

Usage

```
getClock(
  model = c("horvath", "horvathshrunk", "hannum", "skinandblood"),
  padding = 15,
  genome = c("hg19", "hg38", "GRCh37", "GRCh38"),
  useENSR = FALSE,
  useHMMI = FALSE
)
```

Arguments

model	One of "horvath", "horvathshrunk", "hannum", or "skinandblood"
padding	How many base pairs (+/-) to expand a feature's footprint (DEFAULT: 15)
genome	One of "hg19", "GRCh37", "hg38", or "GRCh38" (DEFAULT: "hg19")
useENSR	Substitute ENSEMBL regulatory feature boundaries? (DEFAULT: FALSE)
useHMMI	Substitute HMM-based CpG island boundaries? (DEFAULT: FALSE)

Details

The remapped coordinates for the Horvath (2012) and Hannum (2013) clocks, along with shrunken Horvath (2012) and improved Horvath (2018) models, are provided as part of biscuiteer (visit inst/scripts/clocks.R to find out how) along with some functionality to make them more usable in RRBS/WGBS data of varying coverage along varying genomes. For example, the HMM-based CpG island model introduced by Wu (2010) can be used to assign to within-island features the methylation rate of their associated island, and ENSEMBL regulatory build features (ENSR features, for short) such as CTCF binding sites can have their coordinates substituted for the default padded boundaries of a feature.

The net result of this process is that, while the default settings simply swap in a 30-bp stretch centered on the selected clock's CpG (and/or CpH) loci, add the intercept, and ship out the model, much more flexibility is available to the user. This function provides a single point for tuning of such options in the event that defaults don't work well for a user.

The precedence of options is as follows:

1. If a feature has neither ENSR nor HMMI IDs, it is padded (only) +/- bp.
2. If it has an HMMI but not ENSR ID or ENSR==FALSE, the HMM island is used.
3. If a feature has an ENSR ID, and ENSR==TRUE, the ENSR feature is used.

If a feature has both an ENSR ID and an HMMI ID, and both options are TRUE, then the ENSR start and end coordinates will take precedence over its HMMI.

The above shenanigans produce the GRanges object returned as `gr` in a List. The `intercept` value returned with the model is its fixed (B0) coefficient. The `cleanup` function returned with the model transforms its raw output.

Value

```
a List with elements `model`, `gr`, `intercept`,
          and `cleanup`
```

Examples

```
clock <- getClock(model="horvathshrunk", genome="hg38")
```

`getLogitFracMeth` *Helper function for compartment inference*

Description

Want an object with nominally Gaussian error for compartment inference, so this function uses 'suitable' (defaults to to 3 or more reads in 2 or more samples) measurements. Using Dirichlet smoothing (adding 'k' reads to M and U), these measurements are then turned into lightly moderated, logit-transformed methylated-fraction estimates for compartment calling.

Usage

```
getLogitFracMeth(x, minCov = 3, minSamp = 2, k = 0.1, r = NULL)

getMvals(x, minCov = 3, minSamp = 2, k = 0.1, r = NULL)
```

Arguments

`x`	A bsseq object with methylated and total reads
`minCov`	Minimum read coverage for landmarking samples (DEFAULT: 3)
`minSamp`	Minimum landmark samples with at least minCov (DEFAULT: 2)
`k`	Pseudoreads for smoothing (DEFAULT: 0.1)
`r`	Regions to collapse over - if NULL, do it by CpG (DEFAULT: NULL)

Value

`Smoothed logit(M / Cov) GRanges with coordinates as row names`

Functions

- `getMvals()`: Alias for getLogitFracMeth

Examples

```
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                    merged = FALSE)

reg <- GRanges(seqnames = rep("chr11",5),
               strand = rep("*",5),
               ranges = IRanges(start = c(0,2.8e6,1.17e7,1.38e7,1.69e7),
                                end= c(2.8e6,1.17e7,1.38e7,1.69e7,2.2e7))
               )

frac <- getLogitFracMeth(bisc, minSamp = 1, r = reg)
```

GRCh37.chromArm *GRCh37.chromArm*

Description

Chromosome arm locations for GRCh37 genome

Usage

```
data(GRCh37.chromArm, package="biscuiteer")
```

Details

Source URL: https://genome.ucsc.edu/cgi-bin/hgTables (Cytogenic bands were retrieved using the UCSC Table Browser. The output was then exported to a TXT file, where the chromosome arms were combined and formed into a GRanges) Source type: TXT Return type: GRanges

GRCh38.chromArm *GRCh38.chromArm*

Description

Chromosome arm locations for GRCh38 genome

Usage

```
data(GRCh38.chromArm, package="biscuiteer")
```

Details

Source URL: https://genome.ucsc.edu/cgi-bin/hgTables (Cytogenic bands were retrieved using the UCSC Table Browser. The output was then exported to a TXT file, where the chromosome arms were combined and formed into a GRanges) Source type: TXT Return type: GRanges

grToSeg *Dump GRanges to segmented data data.frame*

Description

Output data.frame can be written to a .seg file if supplied with filename input argument

Usage

```
grToSeg(gr, filename = NULL, minAbs = NULL)
```

Arguments

gr	A GRanges or GRangesList to dump to .seg file
filename	Where to save the result - unsaved if NULL (DEFAULT: NULL)
minAbs	Minimum absolute gain/loss cutoff (DEFAULT: NULL)

Value

```
A data.frame with columns:
          (ID, chrom, loc.start, loc.end, num.mark, seg.mean)
```

See Also

segToGr

Examples

```
clock <- getClock(model="horvathshrunk", genome="hg38")
gr <- clock$gr

df <- grToSeg(gr = gr)
```

H9state23unmeth.hg19 *H9state23unmeth.hg19*

Description

Hypermethylated targets in bivalent histone sites from H9 embryonic stem cells which were unmethylated across normal cells for hg19 genome

Usage

```
data(H9state23unmeth.hg19, package="biscuiteer")
```

Details

GRanges was generated by taking the HMM-derived CpG islands (described in ?HMM_CpG_islands.hg19) and overlapping with regions that were unmethylated in normal H9 stem cells and had a ChromHMM state of 2 or 3 (see https://www.nature.com/articles/nmeth.1906#MOESM194 for a description of ChromHMM) Return type: GRanges

`H9state23unmeth.hg38` *H9state23unmeth.hg38*

Description

Hypermethylated targets in bivalent histone sites from H9 embryonic stem cells which were unmethylated across normal cells for hg38 genome

Usage

```
data(H9state23unmeth.hg38, package="biscuiteer")
```

Details

GRanges was generated by taking the HMM-derived CpG islands (described in ?HMM_CpG_islands.hg38) and overlapping with regions that were unmethylated in normal H9 stem cells and had a ChromHMM state of 2 or 3 (see https://www.nature.com/articles/nmeth.1906#MOESM194 for a description of ChromHMM) Return type: GRanges

`hg19.chromArm` *hg19.chromArm*

Description

Chromosome arm locations for hg19 genome

Usage

```
data(hg19.chromArm, package="biscuiteer")
```

Details

Source URL: http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/cytoBand.txt.gz (Chromosome arms were combined to form the final GRanges) Source type: TXT Return type: GRanges

`hg38.chromArm` *hg38.chromArm*

Description

Chromosome arm locations for hg38 genome

Usage

```
data(hg38.chromArm, package="biscuiteer")
```

Details

Source URL: http://hgdownload.cse.ucsc.edu/goldenPath/hg38/database/cytoBand.txt.gz (Chromosome arms were combined to form the final GRanges) Source type: TXT Return type: GRanges

HMM_CpG_islands.hg19 *HMM_CpG_islands.hg19*

Description

Hidden Markov Model-derived CpG islands from hg19 genome

Usage

```
data(HMM_CpG_islands.hg19, package="biscuiteer")
```

Details

Source URL: https://www.ncbi.nlm.nih.gov/pubmed/20212320 (Hidden Markov Model CpG islands were produced using the method described in this paper. The hg19 genome was used for the CpG island production.) Source type: hg19 genome and procedure described in paper Return type: GRanges

HMM_CpG_islands.hg38 *HMM_CpG_islands.hg38*

Description

Hidden Markov Model-derived CpG islands from hg38 genome

Usage

```
data(HMM_CpG_islands.hg38, package="biscuiteer")
```

Details

Source URL: https://www.ncbi.nlm.nih.gov/pubmed/20212320 (Hidden Markov Model CpG islands were produced using the method described in this paper. The hg19 genome was used for the CpG island production.) Source type: hg19 genome and procedure described in paper Return type: GRanges

makeBSseq *Make an in-memory bsseq object from a biscuit BED*

Description

Beware that any reasonably large BED files may not fit into memory!

Usage

```
makeBSseq(tbl, params, simplify = FALSE, verbose = FALSE)
```

Arguments

tbl	A tibble (from read_tsv) or a data.table (from fread)
params	Parameters from checkBiscuitBED
simplify	Simplify sample names by dropping .foo.bar.hg19? (or similar) (DEFAULT: FALSE)
verbose	Print extra statements? (DEFAULT: FALSE)

Value

An in-memory bsseq object

Examples

```
library(data.table)
library(R.utils)

orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
params <- checkBiscuitBED(BEDfile = orig_bed, VCFfile = orig_vcf,
                          merged = FALSE, how = "data.table")

select <- grep("\\.context", params$colNames, invert=TRUE)
tbl <- fread(gunzip(params$tbx$path, remove = FALSE), sep="\t", sep2=",",
             fill=TRUE, na.strings=".", select=select)
unzippedName <- sub("\\.gz$", "", params$tbx$path)
if (file.exists(unzippedName)) {
  file.remove(unzippedName)
}
if (params$hasHeader == FALSE) names(tbl) <- params$colNames[select]
names(tbl) <- sub("^#", "", names(tbl))

tbl <- tbl[rowSums(is.na(tbl)) == 0, ]
bsseq <- makeBSseq(tbl = tbl, params = params)
```

`readBiscuit` *Read biscuit output into bsseq object*

Description

Takes BED-like format with 2 or 3 columns per sample. Unmerged CpG files have 2 columns (beta values and coverage), whereas merged CpG files have 3 columns (beta values, coverage, and context).

Usage

```
readBiscuit(
  BEDfile,
  VCFfile,
  merged,
  sampleNames = NULL,
  simplify = FALSE,
  genome = "hg19",
  how = c("data.table", "readr"),
  hdf5 = FALSE,
  hdf5dir = NULL,
  sparse = FALSE,
  chunkSize = 1e+06,
  chr = NULL,
  which = NULL,
  verbose = FALSE
)

loadBiscuit(
  BEDfile,
  VCFfile,
  merged,
  sampleNames = NULL,
  simplify = FALSE,
  genome = "hg19",
  how = c("data.table", "readr"),
  hdf5 = FALSE,
  hdf5dir = NULL,
  sparse = FALSE,
  chunkSize = 1e+06,
  chr = NULL,
  which = NULL,
  verbose = FALSE
)
```

Arguments

`BEDfile`	A BED-like file - must be compressed and tabix'ed
`VCFfile`	A VCF file - must be compressed and tabix'ed. Only the header information is needed.

merged	Is this merged CpG data?
sampleNames	Names of samples - NULL: create names, vector: assign names, data.frame: make pData (DEFAULT: NULL)
simplify	Simplify sample names by dropping .foo.bar.hg19? (or similar) (DEFAULT: FALSE)
genome	Genome assembly the runs were aligned against (DEFAULT: "hg19")
how	How to load data - either data.table or readr (DEFAULT: "data.table")
hdf5	Make the object HDF5-backed - CURRENTLY NOT AVAILABLE (DEFAULT: FALSE)
hdf5dir	Directory to store HDF5 files if 'hdf5' = TRUE (DEFAULT: NULL)
sparse	Use sparse Matrix objects for the data? (DEFAULT: FALSE)
chunkSize	Number of rows before readr reading becomes chunked (DEFAULT: 1e6)
chr	Load a specific chromosome? (DEFAULT: NULL)
which	A GRanges of regions to load - NULL loads them all (DEFAULT: NULL)
verbose	Print extra statements? (DEFAULT: FALSE)

Details

NOTE: Assumes alignment against hg19 (use genome argument to override). NOTE: Requires header from VCF file to detect sample names

Value

`A bsseq::BSseq object`

Functions

- `loadBiscuit()`: Alias for readBiscuit

See Also

bsseq

checkBiscuitBED

Examples

```
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                    merged = FALSE)
```

`readEpibed` *Read in and decode the RLE representation of the epibed format out of biscuit epiread*

Description

Read in and decode the RLE representation of the epibed format out of biscuit epiread

Usage

```
readEpibed(
  epibed,
  genome = NULL,
  chr = NULL,
  start = 1,
  end = 2^28,
  fragment_level = TRUE
)
```

Arguments

`epibed`	The path to the epibed file (must be bgzip and tabix indexed)
`genome`	What genome did this come from (e.g. 'hg19') (default: NULL)
`chr`	Which chromosome to retrieve (default: NULL)
`start`	The starting position for a region of interest (default: 1)
`end`	The end position for a region of interest (default: 2^28)
`fragment_level`	Whether to collapse reads to the fragment level (default: TRUE)

Value

A GRanges object

Examples

```
epibed.nome <- system.file("extdata", "hct116.nome.epibed.gz", package="biscuiteer")
epibed.bsseq <- system.file("extdata", "hct116.bsseq.epibed.gz", package="biscuiteer")

epibed.nome.gr <- readEpibed(epibed = epibed.nome, genome = "hg19", chr = "chr1")
epibed.bsseq.gr <- readEpibed(epibed = epibed.bsseq, genome = "hg19", chr = "chr1")
```

RRBSeq *(e)RRBS settings for dmrseq*

Description

(e)RRBS settings for dmrseq

Usage

```
RRBSeq(bsseq, testCovariate, cutoff = 0.2, bpSpan = 750, ...)
```

Arguments

bsseq	A bsseq object
testCovariate	The pData column to test on
cutoff	The minimum CpG-wise difference to use (DEFAULT: 0.2)
bpSpan	Span of smoother AND max gap in DMR CpGs (DEFAULT: 750)
...	Other arguments to pass along to dmrseq

Value

```
A GRanges object (same as from dmrseq)
```

Examples

```
data(BS.chr21, package="dmrseq")
dat <- BS.chr21

rrbs <- RRBSeq(dat[1:500, ], "Rep", cutoff = 0.05, BPPARAM=BiocParallel::SerialParam())
```

segToGr *Import a segmentation file into GRanges object*

Description

Reverse of grToSeg

Usage

```
segToGr(seg, genome = "hg19", name = "ID", score = "seg.mean")
```

Arguments

seg	The .seg filename
genome	Genome against which segments were annotated (DEFAULT: "hg19")
name	.seg file column to use as $name metadata (DEFAULT: "ID")
score	.seg file column to use as $score metadata (DEFAULT: "seg.mean")

Value

A GRanges object

See Also

grToSeg

Examples

```
clock <- getClock(model="horvathshrunk", genome="hg38")
gr <- clock$gr

df <- grToSeg(gr = gr, file = "test_grToSeg.seg")
segs <- segToGr("test_grToSeg.seg", genome="hg38")

if (file.exists("test_grToSeg.seg")) file.remove("test_grToSeg.seg")
```

---

seqinfo.hg19 *seqinfo.hg19*

Description

Seqinfo for hg19 genome

Usage

```
data(seqinfo.hg19, package="biscuiteer")
```

Details

Source URL: http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/hg19.chrom.sizes (The output from this site was downloaded into a TXT file and then loaded into a sorted Seqinfo table) Source type: TXT Return type: Seqinfo

---

seqinfo.hg38 *seqinfo.hg38*

Description

Seqinfo for hg38 genome

Usage

```
data(seqinfo.hg38, package="biscuiteer")
```

Details

Source URL: http://hgdownload.cse.ucsc.edu/goldenPath/hg38/bigZips/hg38.chrom.sizes (The output from this site was downloaded into a TXT file and then loaded into a sorted Seqinfo table) Source type: TXT Return type: Seqinfo

seqinfo.mm10 *seqinfo.mm10*

Description

Seqinfo for mm10 genome

Usage

```
data(seqinfo.mm10, package="biscuiteer")
```

Details

Source URL: http://hgdownload.cse.ucsc.edu/goldenPath/mm10/bigZips/mm10.chrom.sizes (The output from this site was downloaded into a TXT file and then loaded into a sorted Seqinfo table) Source type: TXT Return type: Seqinfo

simplifySampleNames *Simplify bsseq sample names*

Description

Tries using the longest common subsequence to figure out what can be dropped. Usually used for VCF columns.

Usage

```
simplifySampleNames(x)
```

Arguments

x	A SummarizedExperiment-derived object, or a character vector

Value

The input object, but with simplified sample names

Examples

```
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                    merged = FALSE)

bisc <- simplifySampleNames(bisc)
```

summarizeBsSeqOver *Summarize methylation over provided regions*

Description

Used for bsseq objects. Mostly a local wrapp for getMeth.

Usage

```
summarizeBsSeqOver(bsseq, segs, dropNA = FALSE, impute = FALSE)
```

Arguments

bsseq	The bsseq object to summarize
segs	Regions to summarize over (GRanges object, no GRangesList yet)
dropNA	Whether to drop rows if more than half of samples are NA (DEFAULT: FALSE)
impute	Whether to impute NAs/NaNs (DEFAULT: FALSE)

Value

```
A matrix of regional methylation fractions
```

Examples

```
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata", "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                    merged = FALSE)

reg <- GRanges(seqnames = rep("chr11",5),
               strand = rep("*",5),
               ranges = IRanges(start = c(0,2.8e6,1.17e7,1.38e7,1.69e7),
                                end= c(2.8e6,1.17e7,1.38e7,1.69e7,2.2e7))
              )
summary <- summarizeBsSeqOver(bsseq = bisc, segs = reg, dropNA = TRUE)
```

tabixRetrieve *Read from tabix-indexed bed file to list objects*

Description

Read from tabix-indexed bed file to list objects

Usage

```
tabixRetrieve(
  paths,
  chr,
  start = 1,
  end = 2^28,
  sample_names = NULL,
  is.epibed = FALSE,
  BPPARAM = SerialParam()
)
```

Arguments

paths	path(s) to the bed files
chr	chromosome name
start	start coordinate of region of interest
end	end coordinate of region of interest
sample_names	sample names, just use paths if not specified
is.epibed	whether the input is epibed format
BPPARAM	how to parallelize

Value

a list object with DNA methylation level and depth

---

unionize *Combine bsseq objects together without losing information*

Description

Wrapper for the combine(bsseq1, ...) method in bsseq

Usage

```
unionize(bs1, ...)
```

Arguments

bs1	A bsseq object
...	One or more bsseq objects to combine with bs1

Details

Takes provided bsseq objects, the union of their GRanges, fills out the sites not in the union with 0M/0Cov, and returns the even-sparser bsseq holding all of them.

Value

A larger and more sparse bsseq object

Examples

```
shuf_bed <- system.file("extdata", "MCF7_Cunha_chr11p15_shuffled.bed.gz",
                        package="biscuiteer")
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
shuf_vcf <- system.file("extdata",
                        "MCF7_Cunha_shuffled_header_only.vcf.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata",
                        "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc1 <- readBiscuit(BEDfile = shuf_bed, VCFfile = shuf_vcf,
                     merged = FALSE)
bisc2 <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                     merged = FALSE)

comb <- unionize(bisc1, bisc2)
```

`WGBSage` *Guess ages using Horvath-style 'clock' models*

Description

See Horvath, Genome Biology, 2013 for more information

Usage

```
WGBSage(
  bsseq,
  model = c("horvath", "horvathshrunk", "hannum", "skinandblood"),
  padding = 15,
  useENSR = FALSE,
  useHMMI = FALSE,
  minCovg = 5,
  impute = FALSE,
  minSamp = 5,
  genome = NULL,
  dropBad = FALSE,
  ...
)
```

Arguments

`bsseq`	A bsseq object (must have assays named `M` and `Cov`)
`model`	Which model ("horvath", "horvathshrunk", "hannum", "skinandblood")
`padding`	How many bases +/- to pad the target CpG by (DEFAULT: 15)
`useENSR`	Use ENSEMBL regulatory region bounds instead of CpGs (DEFAULT: FALSE)
`useHMMI`	Use HMM CpG island boundaries instead of padded CpGs (DEFAULT: FALSE)
`minCovg`	Minimum regional read coverage desired to estimate 5mC (DEFAULT: 5)

impute	Use k-NN imputation to fill in low-coverage regions? (DEFAULT: FALSE)
minSamp	Minimum number of non-NA samples to perform imputation (DEFAULT: 5)
genome	Genome to use as reference, if no genome(bsseq) is set (DEFAULT: NULL)
dropBad	Drop rows/cols with > half missing pre-imputation? (DEFAULT: FALSE)
...	Arguments to be passed to impute.knn, such as rng.seed

Details

Note: the accuracy of the prediction will increase or decrease depending on how various hyper-parameters are set by the user. This is NOT a hands-off procedure, and the defaults are only a starting point for exploration. It will not be uncommon to tune `padding`, `minCovg`, and `minSamp` for each WGBS or RRBS experiment (and the latter may be impacted by whether dupes are removed prior to importing data). Consider yourself forewarned. In the near future we may add support for arbitrary region-coefficient inputs and result transformation functions, which of course will just make the problems worse.

Also, please cite the appropriate papers for the Epigenetic Clock(s) you use:

For the 'horvath' or 'horvathshrunk' clocks, cite Horvath, Genome Biology 2013. For the 'hannum' clock, cite Hannum et al, Molecular Cell 2013. For the 'skinandblood' clock, cite Horvath et al, Aging 2018.

Last but not least, keep track of the parameters YOU used for YOUR estimates. The `call` element in the returned list of results is for this exact purpose. If you need recover the GRanges object used to average(or impute) DNAme values for the model, try `granges(result$methcoefs)` on a result. The methylation fraction and coefficients for each region can be found in the GRanges object, result$methcoefs, where each sample has a corresponding column with the methylation fraction and the coefficients have their own column titled "coefs". Additionally, the age estimates are stored in result$age (named, in case dropBad == TRUE).

Value

```
A list with call, methylation estimates, coefs, age estimates
```

Examples

```
shuf_bed <- system.file("extdata", "MCF7_Cunha_chr11p15_shuffled.bed.gz",
                        package="biscuiteer")
orig_bed <- system.file("extdata", "MCF7_Cunha_chr11p15.bed.gz",
                        package="biscuiteer")
shuf_vcf <- system.file("extdata",
                        "MCF7_Cunha_shuffled_header_only.vcf.gz",
                        package="biscuiteer")
orig_vcf <- system.file("extdata",
                        "MCF7_Cunha_header_only.vcf.gz",
                        package="biscuiteer")
bisc1 <- readBiscuit(BEDfile = shuf_bed, VCFfile = shuf_vcf,
                     merged = FALSE)
bisc2 <- readBiscuit(BEDfile = orig_bed, VCFfile = orig_vcf,
                     merged = FALSE)

comb <- unionize(bisc1, bisc2)
ages <- WGBSage(comb, "horvath")
```

WGBSeq *Wrapper for WGBS settings for dmrseq*

Description

Wrapper for WGBS settings for dmrseq

Usage

```
WGBSeq(bsseq, testCovariate, bpSpan = 1000, ...)
```

Arguments

bsseq	A bsseq object
testCovariate	The pData column to test on
bpSpan	Span of smoother AND 2x max gap in DMR CpGs (DEFAULT: 1000)
...	Other arguments to pass along to dmrseq

Value

A GRanges object (same as from dmrseq)

Examples

```
data(BS.chr21, package="dmrseq")
dat <- BS.chr21

wgbs <- WGBSeq(dat[1:500, ], "CellType", cutoff = 0.05,
               BPPARAM=BiocParallel::SerialParam())
```

Index