

# Package ‘segmentSeq’

April 5, 2014

**Type** Package

**Title** Methods for identifying small RNA loci from high-throughput sequencing data

**Version** 1.14.0

**Date** 2010-01-20

**Author** Thomas J. Hardcastle

**Maintainer** Thomas J. Hardcastle <tjh48@cam.ac.uk>

## Description

High-throughput sequencing technologies allow the production of large volumes of short sequences, which can be aligned to the genome to create a set of matches to the genome. By looking for regions of the genome which to which there are high densities of matches, we can infer a segmentation of the genome into regions of biological significance. The methods in this package allow the simultaneous segmentation of data from multiple samples, taking into account replicate data, in order to create a consensus segmentation. This has obvious applications in a number of classes of sequencing experiments, particularly in the discovery of small RNA loci and novel mRNA transcriptome discovery.

**License** GPL-3

**LazyLoad** yes

**Depends** R (>= 2.3.0), methods, baySeq (>= 1.15.4), ShortRead, GenomicRanges, IRanges

**Suggests** snow

**Imports** baySeq, graphics, grDevices, IRanges, methods, utils, GenomicRanges

**Collate** AllClasses.R alignmentClass-accessors.R segClass-accessors.R  
segData-accessors.R alignmentData-accessors.R  
lociData-accessors.R alignmentMeth-accessors.R  
methData-accessors.R segMeth-accessors.R getCounts.R  
plotGenome.R processAD.R processTags.R getOverlaps.R  
heuristicSeg.R classifySeg.R lociLikelihoods.R  
processPosteriors.R findChunks.R utilityFunctions.R  
constructNulls.R mergeSegs.R plotGenomicDistributions.R

**biocViews** Bioinformatics, HighThroughputSequencing, MultipleComparisons

**R topics documented:**

segmentSeq-package	2
alignmentClass-class	4
alignmentData-class	5
alignmentMeth-class	6
averageMethylationRegions	8
classifySeg	9
findChunks	11
getCounts	12
getOverlaps	14
heuristicSeg	16
lociData-class	18
lociLikelihoods	19
mergeMethSegs	21
methData-class	22
plotGenome	23
plotMeth	24
plotMethDistribution	25
processAD	26
readMethods	28
readMeths	31
segClass-class	32
segData-class	33
segMeth-class	34
SL	35
<b>Index</b>	<b>37</b>

---

segmentSeq-package	<i>Segmentation of the genome based on multiple samples of high-throughput sequencing data.</i>
--------------------	---

---

**Description**

The segmentSeq package is intended to take multiple samples of high-throughput data (together with replicate information) and identify regions of the genome which have a (reproducibly) high density of tags aligning to them. The package was developed for use in identifying small RNA precursors from small RNA sequencing data, but may also be useful in some mRNA-Seq and chIP-Seq applications.

**Details**

Package:	segmentSeq
Type:	Package
Version:	0.0.2
Date:	2010-01-20
License:	GPL-3

LazyLoad: yes  
Depends: baySeq, ShortRead

To use the package, we construct an `alignmentData` object from sets of alignment files using either the `readGeneric` function to read text files or the `readBAM` function to read from BAM format files.

We then use the `processAD` function to identify all potential subsegments of the data and the number of tags that align to these subsegments. We then use either a heuristic or empirical Bayesian approach to segment the genome into ‘loci’ and ‘null’ regions. We can then acquire posterior likelihoods for each set of replicates which tell us whether a region is likely to be a locus or a null in that replicate group.

The segmentation is designed to be usable by the `baySeq` package to allow differential expression analyses to be carried out on the discovered loci.

The package (optionally) makes use of the ‘snow’ package for parallelisation of computationally intensive functions. This is highly recommended for large data sets.

See the vignette for more details.

### Author(s)

Thomas J. Hardcastle

Maintainer: Thomas J. Hardcastle <tjh48@cam.ac.uk>

### References

Hardcastle T.J., Kelly, K.A. and Balcombe D.C. (2011). Identifying small RNA loci from high-throughput sequencing data. In press.

### See Also

[baySeq](#)

### Examples

```
# Define the chromosome lengths for the genome of interest.

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.
```

```
alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
chrlens)

# Process the alignmentData object to produce a segData object.

sD <- processAD(alignData, gap = 100, cl = NULL)
```

---

alignmentClass-class    *Class "alignmentClass"*

---

### Description

The alignmentClass class records information about a set of alignments of high-throughput sequencing data to a genome.

### Slots

**alignments:** Object of class "GRanges". Stores information about the alignments. See Details.

**libnames:** Object of class "character". The names of the libraries for which alignment data exists.

**replicates:** Object of class "factor". Replicate information for each of the libraries. See Details.

### Details

The alignments slot is a GRanges object defining the location of aligned objects to a reference genome.

The replicates slot is a vector of factors such that the *i*th sample is a replicate of the *j*th sample if and only if `@replicates[i] == @replicates[j]`.

The libnames slot is a vector defining the names of the libraries described by the object.

### Methods

```
[ signature(x = "alignmentClass"): ...
dim signature(x = "alignmentClass"): ...
initialize signature(.Object = "alignmentClass"): ...
show signature(object = "alignmentClass"): ...
```

### Author(s)

Thomas J. Hardcastle

### See Also

[alignmentData](#) [alignmentMeth](#)

---

alignmentData-class    *Class "alignmentData"*

---

### Description

The alignmentData class inherits from the alignmentClass class and records information about a set of alignments of high-throughput sequencing data to a genome. Details include the alignments themselves, the chromosomes of the genome to which the data are aligned, and counts of the aligned tags from each of the libraries from which the data come.

### Objects from the class

Objects can be created by calls of the form `new("alignmentData", ...)`, but more usually by using one of `readBAM` or `readGeneric` functions to generate the object from a set of alignment files.

### Slots

**alignments:** Object of class "GRanges". Stores information about the alignments. See Details.

**replicates:** Object of class "factor". Replicate information for each of the libraries. See Details.

**data:** Object of class "matrix". For each alignment described in the alignments slot, contains the number of times the alignment is seen in each sample.

**libnames:** Object of class "character". The names of the libraries for which alignment data exists.

**libsizes:** Object of class "numeric". The library sizes (see Details) for each of the libraries.

### Details

The alignments slot is the key element of this class. This is a GRanges object that, in addition to the usual elements defining the location of aligned objects to a reference genome, also describes the values 'tag', giving the sequence of the tag aligning to the location, 'matches', indicating in how many places that tag matches to the genome, 'chunk', an identifier for the sets of tags that align close enough together to form a potential locus, and 'chunkDup', indicating whether that tag matches to multiple places within the chunk.

The library sizes, defined in the libsizes slot, provide some scaling factor for the observed number of counts of a tag in different samples.

The replicates slot is a vector of factors such that the *i*th sample is a replicate of the *j*th sample if and only if `@replicates[i] == @replicates[j]`.

### Methods

[ signature(x = "alignmentData"): ...

**dim** signature(x = "alignmentData"): ...

**initialize** signature(.Object = "alignmentData"): ...

**show** signature(object = "alignmentData"): ...

**Author(s)**

Thomas J. Hardcastle

**See Also**

[alignmentClass](#), the class from which alignmentData inherits. [readGeneric](#), which will produce a alignmentData object from appropriately formatted tab-delimited files. [readBAM](#), which will produce a alignmentData object from BAM files. [processAD](#), which will convert an alignmentData object into a 'segData' object for segmentation.

**Examples**

```
# Define the chromosome lengths for the genome of interest.

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
chrlens)
```

---

alignmentMeth-class    *Class "alignmentMeth"*

---

**Description**

The alignmentMeth class inherits from the alignmentClass class and records information about a set of alignments of high-throughput sequencing data to a genome. Details include the alignments themselves, the chromosomes of the genome to which the data are aligned, and counts of the aligned tags from each of the libraries from which the data come.

**Objects from the Class**

Objects can be created by calls of the form `new("alignmentMeth", ...)`, but more usually by using one of [readBAM](#) or [readGeneric](#) functions to generate the object from a set of alignment files.

**Slots**

- alignments:** Object of class "GRanges". Defines the location of sequenced cytosines, amongst other data. See Details.
- libnames:** Object of class "character". The names of the libraries for which alignment data exists.
- replicates:** Object of class "factor". Replicate information for each of the libraries. See Details.
- Cs:** Object of class "matrix". For each cytosine described in the alignments slot, contains the number of times the cytosine is sequenced as a 'C', and is thus methylated.
- Ts:** Object of class "matrix". For each cytosine described in the alignments slot, contains the number of times the cytosine is sequenced as a 'T', and is thus unmethylated.
- nonconversion:** Object of class "numeric". The (estimated) nonconversion rate (see Details) for each of the libraries.

**Details**

The nonconversion slot is an estimate of the rate (for each library) at which an unmethylated cytosine has failed to be converted by sodium bisulphite treatment into thymine, and is thus recorded (incorrectly) as methylated. In some cases, this can be estimated from considering observed methylation rates on regions known to be unmethylated (e.g., chloroplasts) or by introducing unmethylated control sequences.

The replicates slot is a vector of factors such that the *i*th sample is a replicate of the *j*th sample if and only if `@replicates[i] == @replicates[j]`.

**Methods**

```
[ signature(x = "alignmentMeth"): ...
dim signature(x = "alignmentMeth"): ...
initialize signature(.Object = "alignmentMeth"): ...
show signature(object = "alignmentMeth"): ...
```

**Author(s)**

Thomas J. Hardcastle

**See Also**

[alignmentClass](#), the class from which alignmentMeth inherits. [readMeths](#), which will produce a alignmentMeth object from files generated by the YAMA aligner. [processAD](#), which will convert an alignmentMeth object into a 'segData' object for segmentation.

---

averageMethylationRegions

*Plots the average distribution of methylation (taken from an alignmentMeth object) over a set of coordinates (and optionally a surrounding region).*

---

## Description

Given an alignmentMeth object and a set of coordinates, plots the average distribution of methylation over those coordinates. The distributions can be split up into different sample groups by the user.

## Usage

```
averageMethylationRegions(mD, samples, coordinates, cuts, bw = 0.01, surrounding = 0, add = FALSE, col,
```

## Arguments

mD	The <a href="#">alignmentMeth</a> object object defining the methylation at each sequenced base
samples	A factor or list defining the different groups of samples for which to plot different distributions. If a list, each member should consist of integer values defining the column numbers of the 'mD' object. If missing, will use the mD@replicates value.
coordinates	A <a href="#">GRanges</a> object defining the coordinates of interest (e.g. genic regions).
cuts	Optionally, the number of subdivisions to cut the coordinates in when calculating the average methylation density.
bw	If 'cuts' is missing, this factor, multiplied by the length of the 'coordinates' vector, defines the number of cuts (minimum twenty cuts).
surrounding	If non-zero, the size of the region up- and down-stream of the given coordinates for which average methylation should also be calculated.
add	If TRUE, the plotted distribution will be added to the current plot.
col	If given, a vector of colours for each of the groups defined in 'samples'. Defaults to 'rainbow(length(samples))'.
...	Additional arguments to be passed to the 'plot' or 'lines' functions.

## Value

Invisibly, a list containing the coordinates of the lines plotted.

## Author(s)

Thomas J. Hardcastle



---

classifySeg	<i>A method for defining a genome segment map by an empirical Bayesian classification method</i>
-------------	--

---

### Description

This function acquires empirical distributions of sequence tag density from an already existing (or heuristically defined) segment map. It uses these to classify potential segments as either segments or nulls in order to define a new (and improved) segment map.

### Usage

```
classifySeg(sD, cD, aD, lociCutoff = 0.9, nullCutoff = 0.9, subRegion =
NULL, getLikes = TRUE, lR = FALSE, samplesize = 1e5, largeness = 1e8,
tempDir = NULL, cl)
```

### Arguments

sD	A <a href="#">segData</a> object derived from the ‘aD’ object.
cD	A <a href="#">lociData</a> object containing an already existing segmentation map, or NULL.
aD	An <a href="#">alignmentData</a> object.
lociCutoff	The minimum posterior likelihood of being a locus for a region to be treated as a locus.
nullCutoff	The minimum posterior likelihood of being a null for a region to be treated as a null.
subRegion	A <code>data.frame</code> object defining the subregions of the genome to be segmented. If NULL (default), the whole genome is segmented.
getLikes	Should posterior likelihoods for the new segmented genome (loci and nulls) be assessed?
lR	If TRUE, locus and null calls are made on the basis of likelihood ratios rather than posterior likelihoods. Not recommended.
samplesize	The sample size to be used when estimating the prior distribution of the data with the <a href="#">getPriors.NB</a> function.
largeness	The maximum size for a split analysis.
tempDir	A directory for storing temporary files produced during the segmentation.
cl	A SNOW cluster object, or NULL. See Details.

### Details

This function acquires empirical distributions of sequence tag density from the segmentation map defined by the ‘cD’ argument (if ‘cD’ is NULL or missing, then the [heuristicSeg](#) function is used to define a segmentation map. It uses these empirical distributions to acquire posterior likelihoods on each potential segment being either a true segment or a null region. These posterior likelihoods are then used to define the segment map.

**Value**

A `lociData` object, containing the segmentation map discovered.

**Author(s)**

Thomas J. Hardcastle

**References**

Hardcastle T.J., Kelly, K.A. and Balcombe D.C. (2011). Identifying small RNA loci from high-throughput sequencing data. In press.

**See Also**

`heuristicSeg` a fast heuristic alternative to this function. `plotGenome`, a function for plotting the alignment of tags to the genome (together with the segments defined by this function). `baySeq`, a package for discovering differential expression in `lociData` objects.

**Examples**

```
# Define the chromosome lengths for the genome of interest.

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
chrlens, gap = 100)

# Process the alignmentData object to produce a segData object.

sD <- processAD(alignData, gap = 100, cl = NULL)

# Use the classifySeg function on the segData object to produce a lociData object.

pS <- classifySeg(aD = alignData, sD = sD, subRegion = data.frame(chr = ">Chr1", start = 1, end = 1e5), getLikes = TR
```

---

findChunks	<i>Identifies 'chunks' of data within a set of aligned reads.</i>
------------	---

---

### Description

This function identifies chunks of data within a set of aligned reads by looking for gaps within the alignments; regions where no reads align. If we assume that a locus should not contain a gap of sufficient length, then we can separate the analysis of the data into chunks defined by these gaps, reducing the complexity of the problem of segmentation.

### Usage

```
findChunks(alignments, gap, checkDuplication = TRUE)
```

### Arguments

alignments	A <a href="#">GRanges</a> object defining a set of aligned reads.
gap	The minimum length of a gap across which it is assumed that no locus can exist.
checkDuplication	Should we check whether or not reads are duplicated within a chunk? Defaults to TRUE.

### Details

This function is called by the [readGeneric](#) and [readBAM](#) functions but may usefully be called again if filtering of an `linkS4class{alignmentData}` object has altered the data present, or to increase the computational effort required for subsequent analysis. The lower the 'gap' parameter used to define the chunks, the faster (though potentially less accurate) any subsequent analyses will be.

### Value

A modified [GRanges](#) object, now containing columns 'chunk' and 'chunkDup' (if 'checkDuplication' is TRUE), identifying the chunk to which the alignment belongs and whether the alignment of the tag is duplicated within the chunk respectively.

### Author(s)

Thomas J. Hardcastle

### Examples

```
# Define the chromosome lengths for the genome of interest.  
  
chrlens <- c(2e6, 1e6)  
  
# Define the files containing sample information.  
  
datadir <- system.file("extdata", package = "segmentSeq")
```

```

libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Read the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
chrlens, gap = 100)

# Filter the data on number of matches of each tag to the genome

alignData <- alignData[values(alignData@alignments)$matches < 5,]

# Redefine the chunking structure of the data.

alignData <- findChunks(alignData@alignments, gap = 100)

```

---

getCounts

*Gets counts from alignment data from a set of genome segments.*


---

### Description

A function for extracting count data from an alignmentData object given a set of segments defined on the genome.

### Usage

```

getCounts(segments, aD, preFiltered = FALSE, adjustMultireads = TRUE,
          useChunk = FALSE, c1)

```

### Arguments

segments	A GRanges object which defines a set of segments for which counts are required.
aD	An <a href="#">alignmentData</a> object.
preFiltered	The function internally cleans the data; however, this may not be needed and omitting these steps may save computational time. See Details.
adjustMultireads	If working with methylation data, this option toggles an adjustment for reads that align to multiple locations on the genome. Defaults to TRUE.
useChunk	If all segments are within defined ‘chunks’ of the alignmentData object, speed increases if this is set to TRUE. Otherwise, counts may be inaccurate. Defaults to FALSE.
c1	A SNOW cluster object, or NULL. See Details.

## Details

The function extracts count data from alignmentData object 'aD' given a set of segments. The non-trivial aspect of this function is that at a segment which contains a tag that matches to multiple places in that segment (and thus appears multiple times in the alignmentData object) should count it only once.

If `preFiltered = FALSE` then the function allows for missing (NA) data in the segments, unordered segments and duplicated segments. If the segment list has no missing data, is already ordered, and contains no duplications, then computational time can be saved by setting `preFiltered = TRUE`.

A cluster object (package: snow) is recommended for parallelisation of this function when using large data sets. Passing NULL to this variable will cause the function to run in non-parallel mode.

In general, this function will probably not be accessed by the user as the `processAD` function includes a call to `getCounts` as part of the standard processing of an alignmentData object into a segData object.

## Value

If 'as.matrix', a matrix, each column of which corresponds to a library in the alignmentData object 'aD' and each row to the segment defined by the corresponding row in 'segments'. Otherwise an equivalent `DataFrame` object.

## Author(s)

Thomas J. Hardcastle

## See Also

[processAD](#)

## Examples

```
# Define the chromosome lengths for the genome of interest.

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
```

```

chrlens, gap = 100)

# Get count data for three arbitrarily chosen segments on chromosome 1.

getCounts(segments = GRanges(seqnames = c(">Chr1"),
                             IRanges(start = c(1,100,2000), end = c(40,3000,5000))),
          aD = alignData, cl = NULL)

```

---

getOverlaps	<i>Identifies overlaps between two sets of genomic coordinates</i>
-------------	--

---

### Description

This function identifies which of a set of genomic segments overlaps with another set of coordinates; either with partial overlap or with the segments completely contained within the coordinates. The function is used within the ‘segmentSeq’ package for various methods of constructing a segmentation map, but may also be useful in downstream analysis (e.g. annotation analyses).

### Usage

```

getOverlaps(coordinates, segments, overlapType = "overlapping",
            whichOverlaps = TRUE, ignoreStrand = FALSE, cl)

```

### Arguments

coordinates	A GRanges object defining the set of coordinates with which the segments may overlap.
segments	A GRanges object defining the set of segments which may overlap within the coordinates.
overlapType	Which kind of overlaps are being sought? Can be one of ‘overlapping’, ‘contains’ or ‘within’. See Details.
whichOverlaps	If TRUE, returns the ‘segments’ overlapping with the ‘coordinates’. If FALSE, returns a boolean vector specifying which of the ‘coordinates’ overlap with the ‘segments’.
ignoreStrand	If TRUE, a segment may overlap a set of coordinates regardless of the strand the two are on. If FALSE, overlaps will only be identified if both are on the same strand (or if either has no strand specified). Defaults to FALSE.
cl	A SNOW cluster object, or NULL. See Details.

### Details

If `overlapType = "overlapping"` then any overlap between the ‘coordinates’ and the ‘segments’ is sufficient. If `overlapType = "contains"` then a region defined in ‘coordinates’ must completely contain at least one of the ‘segments’ to count as an overlap. If `overlapType = "within"` then a region defined in ‘coordinates’ must be completely contained by at least one of the ‘segments’ to count as an overlap.

A cluster object (package: snow) may usefully be used for parallelisation of this function when examining large data sets. Passing NULL to this variable will cause the function to run in non-parallel mode.

### Value

If whichOverlaps = TRUE, then the function returns a list object with length equal to the number of rows of the 'coordinates' argument. The 'i'th member of the list will be a numeric vector giving the row numbers of the 'segments' object which overlap with the 'i'th row of the 'coordinates' object, or NA if no segments overlap with this coordinate region.

If whichOverlaps = FALSE, then the function returns a boolean vector with length equal to the number of rows of the 'coordinates' argument, indicating which of the regions defined in coordinates have the correct type of overlap with the 'segments'.

### Author(s)

Thomas J. Hardcastle

### Examples

```
# Define the chromosome lengths for the genome of interest.

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
  replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
  chrlens, gap = 100)

# Find which tags overlap with an arbitrary set of coordinates.

getOverlaps(coordinates = GRanges(seqnames = c(">Chr1"),
  IRanges(start = c(1,100,2000), end = c(40,3000,5000))),
  segments = alignData@alignments, overlapType = "overlapping",
  whichOverlaps = TRUE, cl = NULL)
```

---

 heuristicSeg

*A (fast) heuristic method for creation of a genome segment map.*


---

### Description

This method identifies by heuristic methods a set of loci from a `segData` or `segMeth` object. It does this by identifying within replicate groups regions of the genome that satisfy the criteria for being a locus and have no region within them that satisfies the criteria for being a null. These criteria can be defined by the user or inferred from the data.

### Usage

```
heuristicSeg(sD, aD, gap = 100, RKPM = 1000, prop = 0.2, locCutoff = 0.99, subRegion =
  NULL, largeness = 1e8, getLikes = TRUE, verbose = TRUE, cl = NULL)
```

### Arguments

<code>aD</code>	An <code>alignmentData</code> or <code>methData</code> object.
<code>sD</code>	A <code>segData</code> or <code>segMeth</code> object derived from the ‘aD’ object.
<code>gap</code>	What is the minimum length of a null region?
<code>RKPM</code>	For analysis of a <code>segData</code> object, what RKPM (reads per kilobase per million reads) distinguishes between a locus and a null region?
<code>prop</code>	For analysis of a <code>segMeth</code> object, what proportion of methylated cytosines distinguishes between a locus and a null region?. Defaults to 0.2.
<code>locCutoff</code>	For analysis of a <code>segMeth</code> object, with what likelihood must the proportion of methylated cytosines exceed the ‘prop’ option? Defaults to 0.99.
<code>subRegion</code>	A <code>data.frame</code> object defining the subregions of the genome to be segmented. If <code>NULL</code> (default), the whole genome is segmented.
<code>largeness</code>	The maximum size for a split analysis.
<code>getLikes</code>	Should posterior likelihoods for the new segmented genome (loci and nulls) be assessed?
<code>verbose</code>	Should the function be verbose? Defaults to <code>TRUE</code> .
<code>cl</code>	A <code>SNOW</code> cluster object, or <code>NULL</code> . Defaults to <code>NULL</code> . See Details.

### Details

A cluster object (package: `snow`) may be used for parallelisation of parts of this function when examining large data sets. Passing `NULL` to this variable will cause the function to run in non-parallel mode.

### Value

A `lociData` object, containing count information on all the segments discovered.



**Author(s)**

Thomas J. Hardcastle

**References**

Hardcastle T.J., Kelly, K.A. and Balcombe D.C. (2011). Identifying small RNA loci from high-throughput sequencing data. In press.

**See Also**

[classifySeg](#), an alternative approach to this problem using an empirical Bayes approach to classify segments. [plotGenome](#), a function for plotting the alignment of tags to the genome (together with the segments defined by this function). [baySeq](#), a package for discovering differential expression in [lociData](#) objects.

**Examples**

```
# Define the chromosome lengths for the genome of interest.

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
chrlens)

# Process the alignmentData object to produce a segData object.

sD <- processAD(alignData, gap = 100, cl = NULL)

# Use the segData object to produce a segmentation of the genome.

segD <- heuristicSeg(sD = sD, aD = alignData,
subRegion = data.frame(chr = ">Chr1", start = 1, end = 1e5),
cl = NULL)
```

---

lociData-class	Class "lociData"
----------------	------------------

---

## Description

The lociData class is based on the [countData](#) class defined in the 'baySeq' package, but includes a 'coordinates' slot giving the coordinates of genomic loci and a 'locLikelihoods' slot which contains the estimated likelihoods that each annotated region is a locus in each replicate group and a coordinates structure giving the locations of the loci.

## Slots

**locLikelihoods:** Object of class "matrix" describing estimated likelihoods that each region defined in 'coordinates' is a locus in each replicate group.

**coordinates:** Object of class "GRanges" defining the coordinates of the genomic loci.

**data:** Object of class "matrix" defining count data for each locus defined in 'coordinates'

**replicates:** Object of class "factor" defining the replicate structure of the data.

**libsizes:** Object of class "numeric" describing the library size (scaling factor) for each sample.

**groups:** Object of class "list" defining the group (model) structure of the data (see the [baySeq](#) package).

**annotation:** Object of class "data.frame" giving any additional annotation information for each locus.

**priorType:** Object of class "character" describing the type of prior information available in slot priors.

**priors:** Object of class "list" defining the prior parameter information. Calculated by the [baySeq](#) package.

**posteriors:** Object of class "matrix" giving the estimated posterior likelihoods for each replicate group. Calculated by the [baySeq](#) package.

**nullPosts:** Object of class "numeric" which, if calculated, defines the posterior likelihoods for the data having no true expression of any kind. Calculated by the [baySeq](#) package.

**estProps:** Object of class "numeric" giving the estimated proportion of tags belonging to each group. Calculated by the [baySeq](#) package.

**segLens:** Object of class "matrix" defining the lengths of each segment containing the counts described in the data slot. May be initialised with a vector, or ignored altogether.

## Extends

Class "[countData](#)", directly.

**Details**

The `segLens` slot describes, for each row of the data object, the length of the segment that contains the number of counts described by that row. For example, if we are looking at the number of hits matching genes, the `segLens` object would consist of transcript lengths. Exceptionally, we may want to use different segment lengths for different samples and so the slot takes the form of a matrix. If the matrix has only one column, it is duplicated for all samples. Otherwise, it should have the same number of columns as the `'data'` slot. If the slot is the empty matrix, then it is assumed that all segments have the same length.

**Methods**

Methods `'new'`, `'dim'`, `'['` and `'show'` have been defined for this class.

**Author(s)**

Thomas J. Hardcastle

---

lociLikelihoods	<i>Evaluates the posterior likelihoods of each region defined by a segmentation map as a locus.</i>
-----------------	---

---

**Description**

An empirical Bayesian approach that takes a segmentation map and uses this to bootstrap posterior likelihoods on each region being a locus for each replicate group.

**Usage**

```
lociLikelihoods(cD, aD, newCounts = FALSE, bootStraps = 3,
                inferNulls = TRUE, nasZero = FALSE, usePosteriors =
                TRUE, c1)
```

**Arguments**

<code>cD</code>	A <a href="#">lociData</a> object that defines a segmentation map.
<code>aD</code>	An <a href="#">alignmentData</a> object.
<code>newCounts</code>	Should new counts be evaluated for the segmentation map in <code>'cD'</code> before calculating loci likelihoods? Defaults to FALSE
<code>bootStraps</code>	What level of bootstrapping should be carried out on the inference of posterior likelihoods? See the <code>baySeq</code> function <a href="#">getLikelihoods.NB</a> for a discussion of bootstrapping.
<code>inferNulls</code>	Should null regions be inferred from the gaps between segments defined by the <code>'cD'</code> object?
<code>nasZero</code>	If FALSE, any locus with a posterior likelihood <code>'NA'</code> in the existing segmentation map is treated as a null region for the first bootstrap; If TRUE, it is ignored for the first bootstrap.

`usePosteriors` If TRUE, the function uses the existing likelihoods to weight the prior estimation of parameters. Defaults to TRUE.

`cl` A SNOW cluster object, or NULL. See Details.

### Details

A cluster object (package: snow) may be used for parallelisation of this function when examining large data sets. Passing NULL to this variable will cause the function to run in non-parallel mode.

### Value

A `lociData` object.

### Author(s)

Thomas J. Hardcastle

### Examples

```
# Define the chromosome lengths for the genome of interest.

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
chrlens)

# Process the alignmentData object to produce a segData object.

sD <- processAD(alignData, gap = 100, cl = NULL)

# Use the segData object to produce a segmentation of the genome, but
# without evaluating posterior likelihoods.

segD <- heuristicSeg(sD = sD, aD = alignData,
  subRegion = data.frame(chr= ">Chr1", start = 1, end = 1e5),
  getLikes = FALSE, cl = NULL)

# Use the lociData function to evaluate the posterior likelihoods directly.
```

```
lociData <- lociLikelihoods(segD, aD = alignData, bootStraps = 5,  
inferNulls = TRUE, c1 = NULL)
```

---

mergeMethSegs	<i>Merges neighbouring methylation loci with the same pattern of expression.</i>
---------------	--

---

### Description

Within a region of cytosine methylation, there may be some cytosines which show no evidence of methylation. The presence of these cytosines may lead to the region being split into multiple methylation loci. This function merges neighbouring loci if the pattern of expression is the same in each locus, and if they are not separated by too great a genomic distance.

### Usage

```
mergeMethSegs(segs, aD, gap, c1)
```

### Arguments

segs	A <a href="#">methData</a> object defining the loci to be merged.
aD	An <a href="#">alignmentMeth</a> object from which the loci have been derived.
gap	The maximum gap below which neighbouring loci may be merged.
c1	A cluster object, or NULL.

### Value

An object of type [methData](#).

### Author(s)

Thomas J. Hardcastle

### See Also

[methData](#)

---

methData-class	<i>Class "methData"</i>
----------------	-------------------------

---

### Description

The methData class is based on the [pairedData](#) class defined in the 'baySeq' package, but includes a 'coordinates' slot giving the coordinates of genomic loci and a 'locLikelihoods' slot which contains the estimated likelihoods that each annotated region is a locus in each replicate group and a coordinates structure giving the locations of the loci.

### Slots

**locLikelihoods:** Object of class "matrix" describing estimated likelihoods that each region defined in 'coordinates' is a locus in each replicate group.

**coordinates:** Object of class "GRanges" defining the coordinates of the genomic loci.

**data:** Object of class "matrix" defining the number of methylated cytosines observed for each locus defined in 'coordinates'

**data:** Object of class "matrix" defining the number of un-methylated cytosines observed for each locus defined in 'coordinates'

**replicates:** Object of class "factor" defining the replicate structure of the data.

**libsizes:** Object of class "numeric" describing the library size (scaling factor) for each sample.

**groups:** Object of class "list" defining the group (model) structure of the data (see the [baySeq](#) package).

**annotation:** Object of class "data.frame" giving any additional annotation information for each locus.

**priorType:** Object of class "character" describing the type of prior information available in slot priors.

**priors:** Object of class "list" defining the prior parameter information. Calculated by the [baySeq](#) package.

**posteriors:** Object of class "matrix" giving the estimated posterior likelihoods for each replicate group. Calculated by the [baySeq](#) package.

**nullPosts:** Object of class "numeric" which, if calculated, defines the posterior likelihoods for the data having no true expression of any kind. Calculated by the [baySeq](#) package.

**estProps:** Object of class "numeric" giving the estimated proportion of tags belonging to each group. Calculated by the [baySeq](#) package.

**seglens:** Object of class "matrix" defining the lengths of each segment containing the counts described in the data slot. May be initialised with a vector, or ignored altogether.

### Extends

Class "[pairedData](#)", directly.

### Methods

Methods 'new', 'dim', '[' and 'show' have been defined for this class.

**Author(s)**

Thomas J. Hardcastle

---

plotGenome	<i>Plots the alignment of sequence tags on the genome given an 'alignmentData' object and (optionally) a set of segments found.</i>
------------	---

---

**Description**

Plots the data from an alignmentData object for a given set of samples. Can optionally include in the plot the annotation data from a lociData object containing segment information.

**Usage**

```
plotGenome(aD, loci, chr = 1, limits = c(0, 1e4), samples = NULL,
plotType = "pileup", plotDuplicated = FALSE, density = 0, showNumber =
TRUE, logScale = FALSE, cap = Inf, ...)
```

**Arguments**

aD	An <a href="#">alignmentData</a> object.
loci	A <a href="#">lociData</a> object (produced by the <a href="#">heuristicSeg</a> or <a href="#">classifySeg</a> function and therefore) containing appropriate annotation information. Can be omitted if this annotation is not known/required.
chr	The name of the chromosome to be plotted. Should correspond to a chromosome name in the alignmentData object.
limits	The start and end point of the region to be plotted.
samples	The sample numbers of the samples to be plotted. If NULL, plots all samples.
plotType	The manner in which the plot is created. Currently only plotType = pileup is recommended.
plotDuplicated	If TRUE, then any duplicated sequence tags (i.e., sequence tags that match to multiple places in the genome) in the 'aD' object will be plotted on a negative scale for each sample. Defaults to FALSE (recommended).
density	The density of the shading lines to be used in plotting each segment.
showNumber	Should the row number of each segment be shown?
logScale	Should a log scale be used for the number of sequence tags found at each base?
cap	A numeric value defining a cap on the maximum number of reads to be plotted at any one point. Useful if a large number of reads at one location prevent a clear signal being seen elsewhere.
...	Any additional graphical parameters for passing to plot.

**Value**

Plotting function.

**Author(s)**

Thomas J. Hardcastle

**See Also**

[alignmentData](#), [heuristicSeg](#), [classifySeg](#)

**Examples**

```
# Define the chromosome lengths for the genome of interest.
chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
chrlens, gap = 100)

# Plot the alignments to the genome on chromosome 1 between bases 1 and 10000

plotGenome(alignData, chr = ">Chr1", limits = c(1, 1e5))
```

---

plotMeth

*Plots a map of cytosine methylation (and optionally, methylation loci).*

---

**Description**

This function takes an [alignmentMeth](#) object and plots the observed levels of methylation within some defined region on the genome. If a [methData](#) object is also supplied, loci of methylation will also be shown.

**Usage**

```
plotMeth(aM, loci, chr, limits, samples, showNumber = TRUE, rgb = c(1, 0, 0), angle = 45, cap, add = FALSE)
```



**Arguments**

aM	An <a href="#">alignmentMeth</a> .
loci	
chr	The name of the chromosome to be plotted. Should correspond to a chromosome name in the <a href="#">alignmentMeth</a> object.
limits	The start and end point of the region to be plotted.
samples	The sample numbers of the samples to be plotted. If NULL, plots all samples.
showNumber	Should the row number of each segment be shown?
rgb	The rgb code ( <a href="#">rgb</a> ) with which to colour the loci.
angle	The angle at which loci are shaded (see <a href="#">rect</a> ).
cap	Caps the maximum level of coverage shown on the plot; thus, if a base has been sequenced at a level greater than the cap, the data for that base will be shown as if it has a coverage of cap.
add	If TRUE, adds the plot of methylation level to the current plot. Defaults to FALSE.

**Value**

Plotting function.

**Author(s)**

Thomas J. Hardcastle

**See Also**

[alignmentMeth](#)

---

plotMethDistribution *Plots the distribution of methylation on the genome.*

---

**Description**

Plots the distribution of methylation (as defined in an [alignmentMeth](#) object upon the genome.

**Usage**

```
plotMethDistribution(meth, samples, bw = 1e-3, subtract, centromeres,  
add = FALSE, ...)
```

**Arguments**

meth	An object of class <a href="#">alignmentMeth</a> containing the methylation data.
samples	A numeric vector defining the columns of data in the ‘meth’ object from which to estimate proportions of methylation.
subtract	A numeric vector giving values to be subtracted from the density of methylation. See Details.
bw	A scaling factor for the estimation of density of methylation.
centromeres	If given, the location of the centromeres will be plotted.
add	Should the distribution curve be added to an existing plot? Defaults to FALSE.
...	Any additional parameters to be passed to <a href="#">plot</a> .

**Details**

The function returns the density of methylation calculated. This can be used in further plots as the ‘subtract’ parameter, which allows one methylation profile to be subtracted from another.

**Value**

An object of class [density](#) describing the plotted distribution.

**Author(s)**

Thomas J. Hardcastle

**See Also**

[alignmentMeth](#)

---

processAD	<i>Processes an ‘alignmentData’ or ‘alignmentMeth’ object into a ‘segData’ or ‘segMeth’ object for segmentation.</i>
-----------	--

---

**Description**

In order to discover segments of the genome with a high density of sequenced data, a ‘segData’ object must be produced. This is an object containing a set of potential segments, together with the counts for each sample in each potential segment.

**Usage**

```
processAD(aD, gap, squeeze = 0, filterProp = 0.1, strandSplit = FALSE, verbose = TRUE, cl)
```

## Arguments

aD	An <a href="#">alignmentData</a> or <a href="#">alignmentMeth</a> object.
gap	The maximum gap between aligned tags that should be allowed in constructing potential segments. See Details.
squeeze	If greater than zero, the minimum gap between aligned tags that should be allowed in constructing potential segments. See Details.
filterProp	If 'aD' is a <a href="#">alignmentMeth</a> object and this is given, the minimum proportion of methylation at a base below which the base will be filtered out before constructing potential segments (but not during counting).
strandSplit	If TRUE, the data will be split by strand and segments will be constructed separately for each strand. Defaults to FALSE.
verbose	Should processing information be displayed? Defaults to TRUE.
cl	A SNOW cluster object, or NULL. See Details.

## Details

This function takes an [alignmentData](#) or [alignmentMeth](#) object and constructs a [segData](#) or [segMeth](#) object from it. The function creates a set of potential segments by looking for all locations on the genome where the start of a region of overlapping alignments (or, if 'squeeze' is non-zero, those alignments separated by no more than 'squeeze'.) exists in the [alignmentData](#) object. A potential segment then exists from this start point to the end of all regions of overlapping alignments such that there is no region in the segment of at least length 'gap' where no tag aligns. The number of potential segments can therefore be increased by increasing this limit, or (usually more usefully) decreased by decreasing this limit in order to save computational effort.

A cluster object (package: snow) is recommended for parallelisation of this function when using large data sets. Passing NULL to this variable will cause the function to run in non-parallel mode.

## Value

A [segData](#) object.

## Author(s)

Thomas J. Hardcastle

## See Also

[getCounts](#), which produces the count data for each potential segment. [heuristicSeg](#) and [classifySeg](#), which segment the genome based on the [segData](#) object produced by this function [segData alignmentData](#)

## Examples

```
# Define the chromosome lengths for the genome of interest.  
chrlens <- c(2e6, 1e6)
```

```

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
chrlens, gap = 100)

# Process the alignmentData object to produce a segData object.

sD <- processAD(alignData, gap = 100, cl = NULL)

```

---

readMethods	<i>Functions for processing files of various formats into an 'alignment-Data' object.</i>
-------------	---

---

## Description

These functions take alignment files of various formats to produce an object (see Details) describing the alignment of sequencing tags from different libraries. At present, BAM and text files are supported.

## Usage

```

readGeneric(files, dir = ".", replicates, libnames, chrs, chrlens, cols,
            header = TRUE, minlen = 15, maxlen = 29,
            multireads = 1000, polyLength, estimationType = "quantile",
            verbose = TRUE, filterReport = NULL, ...)

readBAM(files, dir = ".", replicates, libnames, chrs, chrlens, countID = NULL,
        minlen = 15, maxlen = 29, multireads = 1000,
        polyLength, estimationType = "quantile", verbose = TRUE,
        filterReport)

```

## Arguments

files	Filenames of the files to be read in.
dir	Directory (or directories) in which the files can be found.

replicates	A vector defining the replicate structure if the group. If and only if the <i>i</i> th library is a replicate of the <i>j</i> th library then @replicates[ <i>i</i> ] == @replicates[ <i>j</i> ]. This argument may be given in any form but will be stored as a factor.
libnames	Names of the libraries defined by the file names.
chrs	A character vector defining (a selection of) the chromosome names used in the alignment files.
chrlens	Lengths of the chromosomes to which the alignments were made.
cols	A named character vector which describes which column of the input files contains which data. See Details.
countID	A (two-character) string used by the BAM file to identify the ‘counts’ of individual sequenced reads; that is, how many times a given read appears in the sequenced library. If NULL, it is assumed that the data are redundant (see Details).
header	Do the input files have a header line? Defaults to TRUE. See Details.
minlen	Minimum length for reads.
maxlen	Maximum length for reads.
multireads	The functions will discard any read that aligns to the genome in more locations than given by this value. Set to Inf to keep everything. Defaults to 1000.
polyLength	If given, an integer value <i>N</i> defining the length of (approximate) homopolymers which will be removed from the data. If a tag contains a sequence of <i>N</i> +1 reads consisting of at least <i>N</i> identical bases, it will be removed. If not given, all data is used.
estimationType	The estimationType that will be used by the ‘baySeq’ function <a href="#">getLibsizes</a> to infer the library sizes of the samples.
verbose	Should processing information be displayed? Defaults to TRUE.
filterReport	If not NULL, this should be a string defining a file to which will be written those data filtered on the basis of chromosome choices, widths of sequences, multireads or polyBase.
...	Additional parameters to be passed to <a href="#">read.table</a> . In particular, the ‘sep’ and ‘skip’ arguments may be useful.

## Details

**readBAM:** This function takes a set of BAM files and generates the alignmentData object from these. If a character string for ‘countID’ is given, the function assumes the data are non-redundant and that ‘countID’ identifies the count data (i.e., how many times each read appears in the sequenced library) in each BAM file. If ‘countID’ is NULL, then it is assumed that the data are redundant, and the count data are inferred from the file.

**readGeneric:** The purpose of this function is to take a set of plain text files and produce an alignmentData object. The function uses [read.table](#) to read in the columns of data in the files and so by default columns are separated by any white space. Alternative separators can be used by passing the appropriate value for sep to [read.table](#).

The files may contain columns with column names chr, tag, count, start, end, strand in which case the ‘cols’ argument can be omitted and ‘header’ set to TRUE. If this is the case, there is no

requirement for all the files to have the same ordering of columns (although all must have these column names).

Alternatively, the columns of data in the input files can be specified by the 'cols' argument in the form of a named character vector (e.g; cols = c(chr = 1, tag = 2, count = 3, start = 4, end = 5, strand = 6) would cause the function to assume that the first column contains the chromosome information, the second column contained the tag information, etc. If 'cols' is specified then information in the header is ignored. If 'cols' is missing and 'header' is FALSE, then it is assumed that the data takes the form described in the example above.

The tag, count and strand columns may optionally be omitted from either the file column headers or the 'cols' argument. If the tag column is omitted, then the data will not account for duplicated sequences when estimating the number of counts in loci. If the count column is omitted, the readGeneric function will assume that the file contains the alignments of each copy of each sequence tag, rather than an aggregated alignment of each unique sequence. The unique alignments will be identified and the number of sequence tags aligning to each position will be calculated. If strand is omitted, the strand will simply be ignored.

### Value

An alignmentData object.

### Author(s)

Thomas J. Hardcastle

### See Also

[alignmentData](#)

### Examples

```
# Define the chromosome lengths for the genome of interest.

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
chrlens)
```

---

readMeths	<i>A function for reading data from the YAMA methylation aligner (or similarly parsed data) from which to identify methylation loci and/or differentially methylated regions.</i>
-----------	---

---

## Description

This function takes as input a set of files that describe the number of times a set of cytosines are observed to be methylated or unmethylated in some high-throughput sequencing data. It merges the data from these files into an object of `'alignmentMeth'` class which can then be further processed to identify methylation loci.

## Usage

```
readMeths(files, dir = ".", libnames, replicates, nonconversion)
```

## Arguments

files	A character vector defining the file names of the alignment files to be read in.
dir	The directory in which the files are located.
libnames	A character vector giving the names of the samples to be read in.
replicates	A vector defining the replicate structure of the data. The 'i'th and 'j'th libraries are treated as replicates if and only if replicates[i] == replicates[j].
nonconversion	A numeric vector (all members should lie between 0 and 1) defining the non-conversion rate of each library. See <a href="#">alignmentMeth-class</a> for details.

## Value

An object of class `alignmentMeth`.

## Author(s)

Thomas J. Hardcastle

## See Also

[alignmentMeth-class](#).

## Examples

```
datadir <- system.file("extdata", package = "segmentSeq")
files <- c("short_18B_C24_C24_trim.fastq_CG_methCalls",
"short_Sample_17A_trimmed.fastq_CG_methCalls",
"short_13_C24_col_trim.fastq_CG_methCalls",
"short_Sample_28_trimmed.fastq_CG_methCalls")

mD <- readMeths(files = files, dir = datadir,
```

```
libnames = c("A1", "A2", "B1", "B2"), replicates = c("A", "A", "B", "B"),
nonconversion = c(0.004777, 0.005903, 0.016514, 0.006134))
```

---

segClass-class	Class "segClass"
----------------	------------------

---

## Description

The `segClass` class contains data about potential segments on the genome.

## Objects from the class

Objects can be created by calls of the form `new("segClass", ..., seglens)`. However, more usually they will be created by calling the `processAD` function.

## Slots

**coordinates:** A `GRanges` object defining the coordinates of the segments.

**replicates:** Object of class "factor". The replicate structure for the samples.

**locLikelihoods:** Object of class "DataFrame" describing estimated likelihoods that each region defined in 'coordinates' is a locus in each replicate group.

## Details

The `@coordinates` slot contains information on each of the potential segments; specifically, chromosome, start and end of the segment, together. Each row of the `@coordinates` slot should correspond to the same row of the `@data` slot.

In almost all cases objects of this class should be produced by the `processAD` function.

## Methods

Methods 'new', 'dim', '[' and 'show' have been defined for this class.

## Author(s)

Thomas J. Hardcastle

## See Also

`processAD`, the function that will most often be used to create objects of this class. `segData`, which inherits from this class. `segMeth`, which inherits from this class.



---

segData-class	Class "segData"
---------------	-----------------

---

### Description

The `segData` class inherits from the `segClass` class and contains data about potential segments on the genome, together with counts for each of those segments.

### Objects from the class

Objects can be created by calls of the form `new("segData", ..., seglens)`. However, more usually they will be created by calling the [processAD](#) function.

### Slots

**coordinates:** A [GRanges](#) object defining the coordinates of the segments.

**replicates:** Object of class "factor". The replicate structure for the samples.

**locLikelihoods:** Object of class "DataFrame" describing estimated likelihoods that each region defined in 'coordinates' is a locus in each replicate group.

**data:** Object of class matrix. Contains the number of counts observed for each sample in each potential segment.

**libsizes:** Object of class "numeric". The library sizes for each sample.

### Details

The `@coordinates` slot contains information on each of the potential segments; specifically, chromosome, start and end of the segment, together. Each row of the `@coordinates` slot should correspond to the same row of the `@data` slot.

In almost all cases objects of this class should be produced by the [processAD](#) function.

### Methods

Methods 'new', 'dim', '[' and 'show' have been defined for this class.

### Author(s)

Thomas J. Hardcastle

### See Also

[processAD](#), the function that will most often be used to create objects of this class. [classifySeg](#), an empirical Bayesian method for defining a segmentation based on a `segData` object.

## Examples

```
# Define the chromosome lengths for the genome of interest.

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("extdata", package = "segmentSeq")
libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")

# Establish the library names and replicate structure.

libnames <- c("SL9", "SL10", "SL26", "SL32")
replicates <- c(1,1,2,2)

# Process the files to produce an alignmentData object.

alignData <- readGeneric(file = libfiles, dir = datadir, replicates =
replicates, libnames = libnames, chrs = c(">Chr1", ">Chr2"), chrlens =
chrlens)

# Process the alignmentData object to produce a segData object.

sD <- processAD(alignData, gap = 100, cl = NULL)
```

---

segMeth-class

*Class "segMeth"*

---

## Description

The `segMeth` class inherits from the `segClass` class and contains data about potential segments on the genome, together with counts for each of those segments.

## Objects from the class

Objects can be created by calls of the form `new("segMeth", ..., seglens)`. However, more usually they will be created by calling the `processAD` function.

## Slots

**coordinates:** A `GRanges` object defining the coordinates of the segments.

**replicates:** Object of class `"factor"`. The replicate structure for the samples.

**locLikelihoods:** Object of class `"DataFrame"` describing estimated likelihoods that each region defined in `'coordinates'` is a locus in each replicate group.

**Cs:** Object of class `matrix`. Contains the number of methylated cytosines (which are sequenced as a `'C'`) observed for each sample in each potential segment.

**Ts:** Object of class `matrix`. Contains the number of unmethylated cytosines (which are sequenced as a ‘T’) observed for each sample in each potential segment.

**nonconversion:** Object of class `"numeric"`. The (estimated) nonconversion rate (see `Details`) for each of the libraries.

### Details

The `@coordinates` slot contains information on each of the potential segments; specifically, chromosome, start and end of the segment, together. Each row of the `@coordinates` slot should correspond to the same row of the `@C` and `@T` slots.

The `nonconversion` slot is an estimate of the rate (for each library) at which an unmethylated cytosine has failed to be converted by sodium bisulphite treatment into thymine, and is thus recorded (incorrectly) as methylated. In some cases, this can be estimated from considering observed methylation rates on regions known to be unmethylated (e.g., chloroplasts) or by introducing unmethylated control sequences.

In almost all cases objects of this class should be produced by the `processAD` function.

### Methods

Methods `'new'`, `'dim'`, `'['` and `'show'` have been defined for this class.

### Author(s)

Thomas J. Hardcastle

### See Also

`processAD`, the function that will most often be used to create objects of this class. `segClass`, from which this class inherits.

---

SL

*Example data selected from a set of Illumina sequencing experiments.*

---

### Description

Each of the files `'SL9'`, `'SL10'`, `'SL26'` and `'SL32'` represents a subset of the data from an Illumina sequencing experiment. These data consist of alignment information; the tag sequence, and the number of times that each sequence is observed.

### Usage

SL

### Format

A set of tab-delimited files containing data from four sequencing experiments.

**Source**

In-house Illumina sequencing experiments

# Index

- \*Topic **textasciitildekw2**
  - readMeths, 31
- \*Topic **classes**
  - alignmentClass-class, 4
  - alignmentData-class, 5
  - alignmentMeth-class, 6
  - lociData-class, 18
  - methData-class, 22
  - segClass-class, 32
  - segData-class, 33
  - segMeth-class, 34
- \*Topic **classif**
  - classifySeg, 9
  - heuristicSeg, 16
- \*Topic **datasets**
  - SL, 35
- \*Topic **files**
  - readMethods, 28
  - readMeths, 31
- \*Topic **hplot**
  - averageMethylationRegions, 8
  - plotGenome, 23
  - plotMeth, 24
  - plotMethDistribution, 25
- \*Topic **manip**
  - classifySeg, 9
  - findChunks, 11
  - getCounts, 12
  - getOverlaps, 14
  - heuristicSeg, 16
  - lociLikelihoods, 19
  - mergeMethSegs, 21
  - processAD, 26
- \*Topic **package**
  - segmentSeq-package, 2
- [, alignmentClass, ANY, ANY-method (alignmentClass-class), 4
- [, alignmentClass-method (alignmentClass-class), 4
- [, alignmentData, ANY, ANY-method (alignmentData-class), 5
- [, alignmentData-method (alignmentData-class), 5
- [, alignmentMeth, ANY, ANY-method (alignmentMeth-class), 6
- [, alignmentMeth-method (alignmentMeth-class), 6
- [, lociData, ANY, ANY-method (lociData-class), 18
- [, lociData-method (lociData-class), 18
- [, methData, ANY, ANY-method (methData-class), 22
- [, methData-method (methData-class), 22
- [, segClass, ANY, ANY-method (segClass-class), 32
- [, segClass-method (segClass-class), 32
- [, segData, ANY, ANY-method (segData-class), 33
- [, segData-method (segData-class), 33
- [, segMeth, ANY, ANY-method (segMeth-class), 34
- [, segMeth-method (segMeth-class), 34
- alignmentClass, 6, 7
- alignmentClass (alignmentClass-class), 4
- alignmentClass-class, 4
- alignmentData, 3, 4, 9, 12, 16, 19, 23, 24, 27, 30
- alignmentData (alignmentData-class), 5
- alignmentData-class, 5
- alignmentMeth, 4, 8, 21, 24–27, 31
- alignmentMeth (alignmentMeth-class), 6
- alignmentMeth-class, 6
- averageMethylationRegions, 8
- baySeq, 3, 10, 17, 18, 22
- cbind, alignmentClass-method (alignmentClass-class), 4

- cbind,alignmentData-method  
(alignmentData-class), 5
- cbind,alignmentMeth-method  
(alignmentMeth-class), 6
- classifySeg, 9, 17, 23, 24, 27, 33
- countData, 18
- DataFrame, 13
- density, 26
- dim,alignmentClass-method  
(alignmentClass-class), 4
- dim,alignmentData-method  
(alignmentData-class), 5
- dim,alignmentMeth-method  
(alignmentMeth-class), 6
- dim,lociData-method (lociData-class), 18
- dim,methData-method (methData-class), 22
- dim,segClass-method (segClass-class), 32
- dim,segData-method (segData-class), 33
- dim,segMeth-method (segMeth-class), 34
- findChunks, 11
- getCounts, 12, 27
- getLibsizes, 29
- getLikelihoods.NB, 19
- getOverlaps, 14
- getPriors.NB, 9
- GRanges, 8, 11, 32–34
- heuristicSeg, 9, 10, 16, 23, 24, 27
- initialize,alignmentClass-method  
(alignmentClass-class), 4
- initialize,alignmentData-method  
(alignmentData-class), 5
- initialize,alignmentMeth-method  
(alignmentMeth-class), 6
- initialize,segClass-method  
(segClass-class), 32
- initialize,segData-method  
(segData-class), 33
- initialize,segMeth-method  
(segMeth-class), 34
- lociData, 9, 10, 16, 17, 19, 20, 23
- lociData (lociData-class), 18
- lociData-class, 18
- lociLikelihoods, 19
- mergeMethSegs, 21
- methData, 16, 21, 24
- methData (methData-class), 22
- methData-class, 22
- pairedData, 22
- plot, 26
- plotGenome, 10, 17, 23
- plotMeth, 24
- plotMethDistribution, 25
- processAD, 3, 6, 7, 13, 26, 32–35
- read.table, 29
- readBAM, 3, 5, 6, 11
- readBAM (readMethods), 28
- readGeneric, 3, 5, 6, 11
- readGeneric (readMethods), 28
- readMethods, 28
- readMeths, 7, 31
- rect, 25
- rgb, 25
- segClass, 35
- segClass (segClass-class), 32
- segClass-class, 32
- segData, 9, 16, 27, 32
- segData (segData-class), 33
- segData-class, 33
- segmentSeq (segmentSeq-package), 2
- segmentSeq-package, 2
- segMeth, 16, 27, 32
- segMeth (segMeth-class), 34
- segMeth-class, 34
- show,alignmentClass-method  
(alignmentClass-class), 4
- show,alignmentData-method  
(alignmentData-class), 5
- show,alignmentMeth-method  
(alignmentMeth-class), 6
- show,lociData-method (lociData-class),  
18
- show,methData-method (methData-class),  
22
- show,segClass-method (segClass-class),  
32
- show,segData-method (segData-class), 33
- show,segMeth-method (segMeth-class), 34
- SL, 35
- SL10 (SL), 35

*INDEX*

39

SL26 (SL), [35](#)

SL32 (SL), [35](#)

SL9 (SL), [35](#)