segmentSeq

April 20, 2011

`alignmentData-class`
Class "alignmentData"

Description

The `alignmentData` class records information about a set of alignments of high-throughput sequencing data to a genome. Details include the alignments themselves, details on the chromosomes of the genome to which the data are aligned, and information on 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 the `processTags` function.

Slots

`alignments`**:** Object of class `"data.frame"`. Stores information about the alignments. 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.

`chrs`**:** Object of class `"character"`. The chromosome names. Must be given as `"character"` to accomodate non-standard chromosome names (such as "X"

`chrlens`**:** Object of class `"numeric"`. The lengths of each of the chromosomes defined by slot `chrs`.

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

Details

The `alignments` slot is the key element of this class. This is a `"data.frame"` object that contains the columns 'chr', 'start', 'end', 'duplicated', 'tag', 'count', 'sampleNumber' and 'replicate'. Columns 'chr', 'start' and 'end' define the chromosome, start and end point of the tag. 'duplicated' indicates whether or not the tag uniquely matches this location (`FALSE`) or whether the tag matches some other location on the genome (`TRUE`). The 'tag' column gives the sequence of the tag as a factor. The 'count' column gives the number of times the tag appears in the library. Which library is involved is specified by the 'sampleNumber' column, and the 'replicate' column gives the replicate group that this library is associated with.

The library sizes, defined in the `libsizes` slot, provide some scaling factor for the observed number of counts of a tag in different samples. One method of calculating this, for example, would be to take the number of sequences read from the high-throughput sequencing machine that align to the reference genome.

The `replicates` slot should take the form of a vector of integers such that if and only if the ith sample is a replicate of the jth sample then `@replicates[i] == @replicates[j]`. In addition, values in the replicates slot should take values from `1:n` where `n` is the number of replicate groups.

Methods

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

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

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

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

Note

Methods 'new', 'dim', '[' and 'show' have been defined for these classes.

Author(s)

Thomas J. Hardcastle

See Also

`processTags`, which will produce a `'alignmentData'` object from appropriately formatted tab-delimited 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("data", 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 <- processTags(libfiles, dir = datadir, replicates, libnames, chrlens, chrs = c
```

`filterSegments` *Filters an set of segments (given an ordering on the segments) such that no segments overlap.*

Description

This function takes a set of segments, plus an ordering on that set, and filters the set such that no segments overlap, preferentially keeping the segments first in ordering.

Usage

```
filterSegments(segs, orderOn, ...)
```

Arguments

`segs`	A `'data.frame'` containing columns 'chr', 'start' and 'end'.
`orderOn`	An vector of some statistic that can be used to create an ordering on the 'segs' data.frame.
`...`	Additional parameters that can be passed to `order` when ordering the segments using the 'orderOn' parameter.

Details

This function takes the set of segments defined by the `data.frame` 'segs', together with some statistic (e.g., likelihood of similarity with background) defined by the 'orderOn' vector. Additional options can be passed to the 'order' function (for example, relating to the direction of the ordering) through the '...' parameter.

The function takes the segment first in the ordering and discards any segments that overlap with it. It then proceeds to the next remaining segment in the ordering and discards any segments that overlap with this. This process continues until we have a set of non-overlapping segments.

This function can be used to create a random sample of non-overlapping segments by providing a randomly chosen set of values for the 'orderOn' vector.

Value

A vector giving the rows of the `data.frame` object 'segs' which form a non-overlapping set.

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("data", package = "segmentSeq")
libfiles <- dir(datadir, pattern = ".txt", full.names = TRUE)

# Establish the library names and replicate structure.

datadir <- system.file("data", 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 <- processTags(libfiles, dir = datadir, replicates, libnames, chrlens, chrs = c

# Process the alignmentData object to produce a 'segData' object.

sD <- processAD(alignData, maxgaplen = 500, cl = NULL)

# Create random sampling of non-overlapping segments for chromosome 1 of sD object.

filterSegments(subset(sD@segInfo, select = c(chr, start, end)), runif(nrow(sD)))
```

`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, cl)
```

Arguments

`segments`	A `'data.frame'` object which defines a set of segments for which counts are required.
`aD`	An `alignmentData` object.
`cl`	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.

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

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 the data.frame 'segments'.

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("data", 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 <- processTags(libfiles, dir = datadir, replicates, libnames, chrlens, chrs = c

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

getCounts(segments = data.frame(chr = ">Chr1", start = c(1,100,2000), end =
c(40, 3000, 5000)), aD = alignData, cl = NULL)
```

`getPriors` *Finds a set of priors for a 'segData' object.*

Description

This function creates a random selection of non-overlapping segments that can be used to estimate prior parameters for the 'segData' object.

Usage

```
getPriors(sD, type = "Pois", verbose = TRUE, ...)
```

Arguments

`sD`	A `segData` object.
`type`	A string describing the type of priors to be estimated. Currently only "Pois" (Poisson-Gamma priors) and "NB" (Negative Binomial) are supported.
`verbose`	Should processing information be displayed? Defaults to TRUE.
`...`	Additional arguments to be passed to one of the `getPriors` functions. See Details.

Details

This function takes a random sample of non-overlapping potential subsegments from the 'segData' object and uses these to construct a `countData` object which is then passed to one of the `getPriors` functions belonging to the 'baySeq' package. Which function is specified depends on the string given in the `priorType` argument; currently only `priorType = "Pois"` and `priorType = "NB"` are supported. Additional arguments can be passed to whichever function is being used via the `'...'` argument.

Value

A `segData` object.

Author(s)

Thomas J. Hardcastle

References

Hardcastle T.J., and Kelly, K.A. (2010). Genome Segmentation from High-Throughput Sequencing Data. In submission.

See Also

`segData`, `getPriors`

Examples

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

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("data", package = "segmentSeq")
libfiles <- dir(datadir, pattern = ".txt", full.names = TRUE)

# Establish the library names and replicate structure.

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

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

datadir <- system.file("data", 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 <- processTags(libfiles, dir = datadir, replicates, libnames, chrlens, chrs = c

# Process the alignmentData object to produce a 'segData' object.

sD <- processAD(alignData, maxgaplen = 500, cl = NULL)

# Estimate prior parameters for the segData object.

sDP <- getPriors(sD, type = "Pois", samplesize = 100, perSE = 0.1, maxit
= 1000, cl = NULL)
```

`plotGenome` *Plots the alignment of sequence tags on the genome given an 'alignmentData' object and (optionally) a set of segments found.*

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 `countData` object containing segment information.

Usage

```
plotGenome(aD, sD, chr = 1, limits = c(0, 10^4), samples = NULL,
plotType = "pileup", ...)
```

Arguments

aD	An alignmentData object.
sD	A countData object (produced by the similaritySeg function and therefore) containing appropriate annotation information. Can be omitted if this annotation is not known/required.
chr	The name of the chromosome (translated into 'character' type if given in any other form) 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 supported.
...	Any additional graphical parameters for passing to plot.

Value

Plotting function.

Author(s)

Thomas J. Hardcastle

See Also

alignmentData, similaritySeg

Examples

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

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("data", 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 <- processTags(libfiles, dir = datadir, replicates, libnames, chrlens, chrs = c

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

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

`processAD` *Processes an 'alignmentData' object into a 'segData' object for segmentation.*

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, maxgaplen = 500, maxloclen = NULL, verbose = TRUE, cl = cl)
```

Arguments

`aD`	An `alignmentData` object.
`maxgaplen`	The maximum gap between aligned tags that should be allowed in constructing potential segments. See Details.
`maxloclen`	The maximum length that any potential segment may be. If NULL (recommended) no such limit exits. See Details.
`verbose`	Should processing information be displayed? Defaults to TRUE.
`cl`	A SNOW cluster object, or NULL. See Details.

Details

This function takes an `alignmentData` object and constructs a `segData` 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 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 'maxgaplen' where no tag aligns. The `'maxgaplen'` argument thus defines the maximum gap that can exist between tags in a segment of high density of alignments. 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.

The number of potential segments may also be decreased by setting a maximum length for any potential segments by setting the argument 'maxloclen'. The use of this argument is not recommended as it appears to substatially degrade the results.

The number of potential segments created can be further reduced by setting a limit on the maximum length that any segment may be with the `'maxloclen'` argument. The use of this limit tends to have severe negative effects on the final segmentation results, however, and is therefore not recommended.

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. similaritySeg, which segments the genome based on the segData object produced. 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("data", 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 <- processTags(libfiles, dir = datadir, replicates, libnames, chrlens, chrs = c

# Process the alignmentData object to produce a 'segData' object.

sD <- processAD(alignData, maxgaplen = 500, cl = NULL)
```

---

processTags — *Convenience function for processing tab-delimited files in a certain format into an 'alignmentData' object.*

Description

This function takes files in a text format with defined columns (see Details) that describe the alignment of sequencing tags from different libraries.

Usage

```
processTags(files, dir = ".", replicates, libnames, chrs, chrlens, cols, header
TRUE, verbose = TRUE, ...)
```

Arguments

files	Filenames of the files to be read in.
dir	Directory (or directories) in which the files can be found.
replicates	Replicate information on the libraries. See Details.
libnames	Names of the libraries defined by the file names.
chrs	Chromosome names (as 'character') 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.
header	Do the input files have a header line? Defaults to TRUE. See Details.
verbose	Should processing information be displayed? Defaults to TRUE.
...	Additional parameters to be passed to read.table.

Details

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', in which case the 'cols' argument can be ommitted 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)' would cause the function to assume that the first column contains the chromosome information, the second column contained the tag information, &c. If 'cols' is specified then information in the header is ignored. If 'cols' is missing and 'header = FALSE' then it is assumed that the data takes the form described in the example above.

The 'tag' and 'count' 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 'processTags' 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.

The replicates argument should take the form of a vector of integers such that if and only if the ith library is a replicate of the jth library then @replicates[i] == @replicates[j]. In addition, values in the replicates slot should take values from 1:n where n is the number of replicate groups.

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("data", 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 <- processTags(libfiles, dir = datadir, replicates, libnames, chrlens, chrs = c
```

---

segData-class *Class "segData"*

Description

The segData class contains data about potential segments on the genome containing data about each potential subsegment.

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

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

leftData: Object of class "matrix". Contains the number of counts observed for the region to the left of the potential segment.

rightData: Object of class "matrix". Contains the number of counts observed for the region to the right of the potential segment.

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

replicates: Object of class "numeric". The replicate structure for the samples. This should be a vector of consecutive integers starting with 1.

priorType: Character string describing the type of prior information available in slot 'priors'.

`priors`: Prior parameter information, estimated from the data (or otherwise acquired). See Details.

`segInfo`: Object of class `"data.frame"`. A data.frame containing the following columns; 'chr', 'start', 'end', 'leftSpace', 'rightSpace'. See Details.

Details

The `@segInfo` slot contains information on each of the potential segments; specifically, chromosome, start and end of the segment, together with the distance from each segment to the next segment on the left and right hand sides. These data are contained in the columns 'chr', 'start', 'end', 'leftSpace', 'rightSpace' respectively. Each row of the `@segInfo` 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. The slot '@priors' should be filled by using the `getPriors` 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. `getPriors`, a function for filling the '@priors' slot of objects of this class.

Examples

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

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("data", 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 <- processTags(libfiles, dir = datadir, replicates, libnames, chrlens, chrs = c

# Process the alignmentData object to produce a 'segData' object.

sD <- processAD(alignData, maxgaplen = 500, cl = NULL)

# Estimate prior parameters for the segData object.
```

```
sDP <- getPriors(sD, type = "Pois", samplesize = 100, perSE = 0.1, maxit
= 1000, cl = NULL)

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

segD <- similaritySeg(sDP, pcut = 0.1, cl = NULL)
```

`segmentSeq-package` *Segmentation of the genome based on multiple samples of high-throughput sequencing data.*

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.

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 (either explicitly or using the `processTags` function). containing the alignment information for each sample. 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 empirically determine the prior parameters of the data using the `getPriors` function, and finally identify all segments to which a high density of tags align in at least one replicate group using the `segmentSeq` function. The output from this segmentation is designed to be usable by the `baySeq` package.

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., and Kelly, K.A. (2010). Genome Segmentation from High-Throughput Sequencing Data. In submission.

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("data", 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 <- processTags(libfiles, dir = datadir, replicates, libnames, chrlens, chrs = c

# Process the alignmentData object to produce a 'segData' object.

sD <- processAD(alignData, maxgaplen = 500, cl = NULL)

# Estimate prior parameters for the segData object.

sDP <- getPriors(sD, type = "Pois", samplesize = 100, perSE = 0.1, maxit
= 1000, cl = NULL)

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

segD <- similaritySeg(sDP, pcut = 0.1, cl = NULL)
```

`similaritySeg` *Takes the 'segData' object and uses it to define segments on the genome.*

Description

This function takes the 'segData' object and produces a set of segments on the genome based on the data contained within it.

Usage

```
similaritySeg(sDP, pcut = 0.5, priorDE = 1e-2, verbose = TRUE,
topOnly = FALSE, ..., cl)
```

Arguments

sDP	A segData object.
pcut	The maximum acceptable likelihood that a potential segment is similar to background or to the regions to either the left and right of the potential segment. See Details.
priorDE	Prior likelihood of similarity.
verbose	Should processing information be displayed? Defaults to TRUE.
topOnly	If TRUE, then only the potential segment with the lowest likelihood of similarity is returned. Defaults to FALSE.
...	Additional parameters to be passed to the getLikelihoods function of the 'baySeq' package.
cl	A SNOW cluster object, or NULL. See Details.

Details

This function takes each potential segment defined by the segData object and evaluates the likelihood that it is similar to either background or the empty region to the left and right of the potential segment in all replicate groups. See Hardcastle & Kelly (2010) for more details on how this likelihood is evaluated.

The potential segments are then ranked by increasing likelihood of similarity. Any potential segment with a likelihood of similarity greater than 'pcut' is discarded.

If 'estimatePriors = TRUE' then an attempt will be made to empirically re-estimate the prior likelihoods of similarity from the data. This may improve the segmentation slightly at some computational cost.

There is then a filtration step (filterSegments). The segment with the lowest likelihood of similarity is kept, and any segments that have overlap with this segment are discarded. The segment with the next lowest likelihood of similarity is then kept, and any segments that have overlap with this segment are discarded. This process continues until we have a set of non-overlapping segments.

Value

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

Author(s)

Thomas J. Hardcastle

References

Hardcastle T.J., and Kelly, K.A. (2010). Genome Segmentation From High-Throughput Sequencing Data. In submission.

Hardcastle T.J., and Kelly, K.A. (2010). Empirical Bayesian Methods For Identifying Patterns of Differential Expression in Count Data. In submission.

See Also

getPriors, a function for establishing prior parameters used to estimate posterior likelihoods. 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 countData objects.

Examples

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

chrlens <- c(2e6, 1e6)

# Define the files containing sample information.

datadir <- system.file("data", 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 <- processTags(libfiles, dir = datadir, replicates, libnames, chrlens, chrs = c

# Process the alignmentData object to produce a 'segData' object.

sD <- processAD(alignData, maxgaplen = 500, cl = NULL)

# Estimate prior parameters for the segData object.

sDP <- getPriors(sD, type = "Pois", samplesize = 100, perSE = 0.1, maxit
= 1000, cl = NULL)

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

segD <- similaritySeg(sDP, pcut = 0.1, cl = NULL)
```

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