Package ‘groHMM’

Version 1.8.0
Date 2016-02-11
Title GRO-seq Analysis Pipeline
Author Charles G. Danko, Minho Chae, Andre Martins, W. Lee Kraus
Maintainer Anusha Nagari <anusha.nagari@utsouthwestern.edu>, Venkat Malladi <venkat.malladi@utsouthwestern.edu>, Tulip Nandu <tulip.nandu@utsouthwestern.edu>, W. Lee Kraus <lee.kraus@utsouthwestern.edu>
Depends R (>= 3.0.2), MASS, parallel, S4Vectors (>= 0.9.25), IRanges (>= 2.5.27), GenomeInfoDb, GenomicRanges (>= 1.23.16), GenomicAlignments, rtracklayer
Suggests BiocStyle, GenomicFeatures, edgeR, org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg19.knownGene
Description A pipeline for the analysis of GRO-seq data.
URL https://github.com/Kraus-Lab/groHMM
BugReports https://github.com/Kraus-Lab/groHMM/issues
License GPL-3
biocViews Sequencing, Software
LazyLoad yes
NeedsCompilation yes

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>groHMM-package</td>
<td>2</td>
</tr>
<tr>
<td>averagePlot</td>
<td>3</td>
</tr>
<tr>
<td>breakTranscriptsOnGenes</td>
<td>4</td>
</tr>
<tr>
<td>combineTranscripts</td>
<td>5</td>
</tr>
<tr>
<td>countMappableReadsInInterval</td>
<td>5</td>
</tr>
<tr>
<td>detectTranscripts</td>
<td>6</td>
</tr>
<tr>
<td>evaluateHMMInAnnotations</td>
<td>7</td>
</tr>
<tr>
<td>expressedGenes</td>
<td>8</td>
</tr>
<tr>
<td>getCores</td>
<td>9</td>
</tr>
<tr>
<td>getTxDensity</td>
<td>9</td>
</tr>
<tr>
<td>limitToXkb</td>
<td>10</td>
</tr>
<tr>
<td>makeConsensusAnnotations</td>
<td>11</td>
</tr>
<tr>
<td>metaGene</td>
<td>12</td>
</tr>
<tr>
<td>metaGeneMatrix</td>
<td>13</td>
</tr>
</tbody>
</table>
groHMM-package

Description

groHMM was developed for analysis of GRO-seq data, which provides a genome wide ‘map’ of the position and orientation of all transcriptionally active RNA polymerases. groHMM predicts the boundaries of transcriptional activity across the genome de novo using a two-state hidden Markov model (HMM). The model essentially divides the genome into ‘transcribed’ and ‘non-transcribed’ regions in a strand specific manner.

We also use HMMs to identify the leading edge of Pol II at genes activated by a stimulus in GRO-seq time course data. This approach allows the genome-wide interrogation of transcription rates in cells.

In addition to these advanced features, groHMM provides wrapper functions for counting raw reads, generating wiggle files for visualization, and creating metagene (averaging) plots. Although groHMM is tailored towards GRO-seq data, the same functions and analytical methodologies can, in principal, be applied to a wide variety of other short read data sets.

Details

Package: groHMM
Type: Package
Version: 0.99.0
Date: 2014-04-02
License: GPL (>=3)
LazyLoad: yes
Depends: R (>= 2.14.0), MASS, GenomicRanges, rtracklayer, parallel

Author(s)

Charles G. Danko, Minho Chae, Andre Martins
averagePlot

Maintainer: Minho Chae<minho.chae@gmail.com>

References


averagePlot

Returns the average profile of tiling array probe intensity values or wiggle-like count data centered on a set of genomic positions (specified by 'Peaks').

Description

Supports parallel processing using mclapply in the 'parallel' package. To change the number of processors, use the argument 'mc.cores'.

Usage

averagePlot(ProbeData, Peaks, size = 50, bins = seq(-1000, 1000, size))

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProbeData</td>
<td>Data.frame representing chromosome, window center, and a value.</td>
</tr>
<tr>
<td>Peaks</td>
<td>Data.frame representing chromosome, and window center.</td>
</tr>
<tr>
<td>size</td>
<td>Numeric. The size of the moving window. Default: 50 bp.</td>
</tr>
<tr>
<td>bins</td>
<td>The bins of the meta gene – i.e. the number of moving windows to break it into. Default +/- 1kb from center.</td>
</tr>
</tbody>
</table>

Value

A vector representing the 'typical' signal centered on the peaks of interest.

Author(s)

Charles G. Danko and Minho Chae
breakTranscriptsOnGenes

Description

Breaks transcripts when they are overlapped with multiple well annotated genes.

Usage

breakTranscriptsOnGenes(tx, annox, strand = "+", geneSize = 5000, threshold = 0.8, gap = 5, plot = FALSE)

Arguments

- **tx**: GRanges of transcripts.
- **annox**: GRanges of non-overlapping annotations for reference.
- **strand**: Takes "+" or "+". Default: "+"
- **geneSize**: Numeric. Minimum gene size in annox to be used as reference. Default: 5000
- **threshold**: Numeric. Ratio of overlapped region relative to a gene width. Transcripts only greater than this threshold are subjected to be broken. Default: 0.8
- **gap**: Numeric. Gap (bp) between broken transcripts. Default: 5
- **plot**: Logical. If set to TRUE, show each step in a plot. Default: FALSE

Value

Returns GRanges object of broken transcripts.

Author(s)

Minho Chae and Charles G. Danko

Examples

tx <- GRanges("chr7", IRanges(1000, 30000), strand="+")
annox <- GRanges("chr7", IRanges(start=c(1000, 20000),
width=c(10000,10000)), strand="+")
bPlus <- breakTranscriptsOnGenes(tx, annox, strand="+")
**combineTranscripts**

**Description**

Combines transcripts that are within the same gene annotation, combining smaller transcripts for genes with low regulation into a single transcript representing the gene.

**Usage**

```r
combineTranscripts(tx, annox, geneSize = 1000, threshold = 0.8, plot = FALSE)
```

**Arguments**

- `tx` GRanges of transcripts.
- `annox` GRanges of non-overlapping annotations for reference.
- `geneSize` Numeric. Minimum gene size in annotations to be used as reference. Default: 1000
- `threshold` Numeric. Ratio of overlapped region relative to transcript width. Transcripts only greater than this threshold are subjected to be combined. Default: 0.8
- `plot` Logical. If set to TRUE, show each step in a plot. Default: FALSE

**Value**

Returns GRanges object of combined transcripts.

**Author(s)**

Minho Chae and Charles G. Danko

**Examples**

```r
tx <- GRanges("chr7", IRanges(start=c(1000, 20000), width=c(10000,10000)), strand="+")
annox <- GRanges("chr7", IRanges(1000, 30000), strand="+")
combined <- combineTranscripts(tx, annox)
```

---

**countMappableReadsInInterval**

**Description**

Counts the number of mappable reads in a set of genomic features.

**Usage**

```r
countMappableReadsInInterval(interval, genome, files = NULL)
```

**Arguments**

- `interval` GRanges object
- `genome` Genome annotation
- `files` Optional. Read files.

**Value**

Cufflinks-like Cuffdiff output file

**Examples**

```r
tx <- GRanges("chr7", IRanges(start=c(1000, 20000), width=c(10000,10000)), strand="+")
annox <- GRanges("chr7", IRanges(1000, 30000), strand="+")
combined <- countMappableReadsInInterval(tx, annox)
```


detectTranscripts

detectTranscripts detects transcripts de novo using a two-state hidden Markov model (HMM).

Description

Read counts can be specified as either a GRanges object (reads), or using a fixed-step wiggle-format passed in a list (Fp and Fm). Either reads or BOTH Fp and Fm must be specified.

Usage

detectTranscripts(reads = NULL, Fp = NULL, Fm = NULL, LtProbA = -5, LtProbB = -200, UTS = 5, size = 50, threshold = 0.1, debug = TRUE, ...)

Arguments

reads A GRanges object representing a set of mapped reads.

Fp Wiggle-formatted read counts on "+" strand. Optionally, Fp and Fm represent list() filled with a vector of counts for each chromosome. Can detect transcripts starting from a fixed-step wiggle.

Fm Wiggle-formatted read counts on "-" strand.

LtProbA Log probability of t... . Default: -5. One of these is just an initialization, and the final value is set by EM. The other is a holdout parameter.

LtProbB Log probability of t... . Default: -200.

UTS Variance in read counts of the untranscribed sequence. Default: 5.

size Log probability of t... . Default: -5.

threshold Threshold change in total likelihood, below which EM exits.

depend debug If set to TRUE, provides additional print options. Default: FALSE

...

Extra argument passed to mclapply
evaluateHMMInAnnotations

Details

Supports parallel processing using mclapply in the ’parallel’ package. To change the number of processors set the option ’mc.cores’.


Value

Returns a list of emisParams, trnasParams, viterbiStates, and transcripts. The transcript element is a GRanges object representing the predicted genomic coordinates of transcripts on both the + and - strand.

Author(s)

Charles G. Danko and Minho Chae

Examples

S0mR1 <- as(readGAlignments(system.file("extdata", "S0mR1.bam", package="groHMM")), "GRanges")
## Not run:
# hmmResult <- detectTranscripts(S0mR1, LtProbB=-200, UTS=5, threshold=1)
# txHMM <- hmmResult$transcripts

evaluateHMMInAnnotations

Description

Evaluates HMM calling of transcripts compared to known annotations.

Usage

evaluateHMMInAnnotations(tx, annox)

Arguments

tx GRanges of transcripts predicted by HMM.
annox GRanges of non-overlapping annotations.

Value

a list of error information; merged annotations, dissociated annotation, total, and rate.

Author(s)

Minho Chae
expressedGenes

Function identifies expressed features using the methods introduced in
Core, Waterfall, Lis; Science, Dec. 2008.

Description

Supports parallel processing using mclapply in the ‘parallel’ package. To change the number of
processors use the argument ‘mc.cores’.

Usage

expressedGenes(features, reads, Lambda = NULL, UnMap = NULL,
                   debug = FALSE, ...)

Arguments

features    A GRanges object representing a set of genomic coordinates. The meta-plot will
             be centered on the start position. There can be optional “ID” column for gene
             ids.
reads       A GRanges object representing a set of mapped reads.
Lambda      Measurement of assay noise. Default: 0.04 reads/kb in a library of 10,751,533
UnMap       List object representing the position of un-mappable reads. Default: not used.
debug       If set to true, returns the number of positions. Default: FALSE.
...          Extra argument passed to mclapply

Value

Returns a data.frame representing the expression p.values for features of interest.

Author(s)

Charles G. Danko
getCores

Returns the number of cores.

**Description**

Returns the number of cores.

**Usage**

`getCores(cores)`

**Arguments**

- **cores**
  
  the number of cores, it is 1 in windows platform.

**Examples**

```r
cores <- getCores(2L)
```

getTxDensity

Calculates transcript density.

**Description**

Calculates transcript density for transcripts which overlaps with annotations. For 'run genes together' or 'broken up a single annotation' errors, best overlapped transcripts or annotations are used.

**Usage**

`getTxDensity(tx, annox, plot = TRUE, scale = 1000L, nSampling = 0L, samplingRatio = 0.1, ...)`

**Arguments**

- **tx**
  
  GRanges of transcripts.

- **annox**
  
  GRanges of non-overlapping annotations.

- **plot**
  
  Logical. If TRUE, plot transcript density. Default: TRUE

- **scale**
  
  Numeric. Scaled size of a gene for transcript density calculation. Default: 1000L

- **nSampling**
  
  Numeric. Number of subsampling. Default: 0L

- **samplingRatio**
  
  Numeric. Ratio of sampling for annotations. Default: 0.1

- **...**
  
  Extra argument passed to mclapply.

**Details**

Supports parallel processing using mclapply in the 'parallel' package. To change the number of processors set the option 'mc.cores'.
limitToXkb

limitToXkb truncates a set of genomic intervals at a constant, maximum size.

Description

limitToXkb truncates a set of genomic intervals at a constant, maximum size.

Usage

\[
\text{limitToXkb(features, offset = 1000, size = 13000)}
\]

Arguments

- **features**: A GRanges object representing a set of genomic coordinates. The meta-plot will be centered on the start position.
- **offset**: Starts the interval from this position relative to the start of each genomic features.
- **size**: Specifies the size of the window.

Value

Returns GRanges object with new genomic coordinates.

Author(s)

Minho Chae and Charles G. Danko

Examples

\[
\begin{align*}
\text{tx &lt;- GRanges("chr7", IRanges(start=seq(1000,4000, by=1000), width=seq(1000, 1300, by=100)), strand=rep("+", 4))} \\
\text{annox &lt;- GRanges("chr7", IRanges(start=seq(1100,4100, by=1000), width=seq(900, 1200, by=100)), strand=rep("+", 4))} \\
\text{## Not run:} \\
\text{# density &lt;- getTxDensity(tx, annox)}
\end{align*}
\]
makeConsensusAnnotations

Description

Makes a non-overlapping consensus annotation. Gene annotations are often overlapping due to multiple isoforms for a gene. In consensus annotation, isoforms are first reduced so that only redundant intervals are used to represent a genomic interval for a gene, i.e., a gene id. Remaining unresolved annotations are further reduced by truncating 3’ end of annotations.

Usage

makeConsensusAnnotations(ar, minGap = 1L, minWidth = 1000L, ...)

Arguments

ar GRanges of annotations to be collapsed.
minGap Minimum gap between overlapped annotations after truncated. Default: 1L
minWidth Minimum width of consensus annotations. Default: 1000L
... Extra argument passed to mclapply.

Details

Supports parallel processing using mclapply in the `parallel` package. To change the number of processors, use the argument `mc.cores`.

Value

Returns GRanges object of annotations.

Author(s)

Minho Chae

Examples

```r
## Not run:
# library(TxDb.Hsapiens.UCSC.hg19.knownGene)
# txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
# tx <- transcripts(txdb, columns=c("gene_id", "tx_id", "tx_name"),
# # filter=list(tx_chrom="chr7"))
# tx <- tx[grep("random", as.character(seqnames(tx)), invert=TRUE),]
# ca <- makeConsensusAnnotations(tx)
```
**metaGene**

Returns a histogram of the number of reads in each section of a moving window centered on a certain feature.

---

**Description**

Supports parallel processing using `mclapply` in the `parallel` package. To change the number of processors, set the option `mc.cores`.

**Usage**

```r
metaGene(features, reads = NULL, plusCVG = NULL, minusCVG = NULL, size = 100L, up = 10000L, down = NULL, ...)
```

**Arguments**

- `features`: A GRanges object representing a set of genomic coordinates. The meta-plot will be centered on the transcription start site (TSS).
- `reads`: A GRanges object representing a set of mapped reads. Instead of `reads`, `plusCVG` and `minusCVG` can be used. Default: NULL.
- `plusCVG`: A RangesList object for reads with `'+` strand.
- `minusCVG`: A RangesList object for reads with `'-'` strand.
- `size`: The size of the moving window.
- `up`: Distance upstream of each feature to align and histogram. Default: 10 kb.
- `down`: Distance downstream of each feature to align and histogram. If NULL, same as up. Default: NULL.
- `...`: Extra argument passed to `mclapply`.

**Value**

Returns a integer-Rle representing the 'typical' signal centered on a point of interest.

**Author(s)**

Charles G. Danko and Minho Chae

**Examples**

```r
features <- GRanges("chr7", IRanges(1000, 1000), strand="+
reads <- GRanges("chr7", IRanges(start=c(1000:1004, 1100),
  width=rep(1, 6)), strand="+
mg <- metaGene(features, reads, size=4, up=10)
```
metaGeneMatrix

Returns a matrix, with rows representing read counts across a specified gene, or other features of interest.

Description

Supports parallel processing using mclapply in the `parallel` package. To change the number of processors, use the argument `mc.cores`.

Usage

```
metaGeneMatrix(features, reads, size = 50, up = 1000, down = up,
               debug = FALSE, ...)
```

Arguments

- `features`: A GRanges object representing a set of genomic coordinates.
- `reads`: A GRanges object representing a set of mapped reads.
- `size`: The size of the moving window.
- `up`: Distance upstream of each f to align and histogram Default: 1 kb.
- `down`: Distance downstream of each f to align and histogram Default: same as up.
- `debug`: If set to TRUE, provides additional print options. Default: FALSE
- `...`: Extra argument passed to mclapply

Value

Returns a vector representing the ‘typical’ signal across genes of different length.

Author(s)

Charles G. Danko and Minho Chae

---

metaGene_nL

Returns a histogram of the number of reads in each section of a moving window of # variable size across genes.

Description

Supports parallel processing using mclapply in the `parallel` package. To change the number of processors, use the argument `mc.cores`.

Usage

```
metaGene_nL(features, reads, n_windows = 1000, debug = FALSE, ...)
```
pausingIndex

Arguments

- **features**: A GRanges object representing a set of genomic coordinates.
- **reads**: A GRanges object representing a set of mapped reads.
- **n_windows**: The number of windows to break genes into.
- **debug**: If set to TRUE, provides additional print options. Default: FALSE
- **...**: Extra argument passed to mclapply

Value

Returns a vector representing the 'typical' signal across genes of different length.

Author(s)

Charles G. Danko and Minho Chae

---

pausingIndex **Returns the pausing index for different genes. TODO: DESCRIBE THE PAUSING INDEX.**

Description

Supports parallel processing using mclapply in the 'parallel' package. To change the number of processors, use the argument 'mc.cores'.

Usage

```r
pausingIndex(features, reads, size = 50, up = 1000, down = 1000,
UnMAQ = NULL, debug = FALSE, ...)
```

Arguments

- **features**: A GRanges object representing a set of genomic coordinates.
- **reads**: A GRanges object representing a set of mapped reads.
- **size**: The size of the moving window.
- **up**: Distance upstream of each f to align and histogram.
- **down**: Distance downstream of each f to align and histogram (NULL).
- **UnMAQ**: Data structure representing the coordinates of all un-mappable regions in the genome.
- **debug**: If set to TRUE, provides additional print options. Default: FALSE
- **...**: Extra argument passed to mclapply

Value

Returns a data.frame of the pausing indices for the input genes.

 Returns the pausing index for different genes.

Author(s)

Charles G. Danko and Minho Chae.
polymeraseWave

Examples

```r
features <- GRanges("chr7", IRanges(2394474,2420377), strand="+")
reads <- as(readGAlignments(system.file("extdata", "S0mR1.bam", 
package="groHMM")), "GRanges")
## Not run:
# pi <- pausingIndex(features, reads)
```

description

Given GRO-seq data, identifies the location of the polymerase 'wave' in up- or down-regulated genes.

Usage

```r
polymeraseWave(reads1, reads2, genes, approxDist, size = 50,
upstreamDist = 10000, TSmooth = NA, NonMap = NULL, prefix = NULL,
emissionDistAssumption = "gamma", finterWindowSize = 10000,
limitPCRDups = FALSE, returnVal = "simple", debug = TRUE)
```

Arguments

read1: Mapped reads in time point 1.
read2: Mapped reads in time point 2.
genex: A set of genes in which to search for the wave.
approxDist: The approximate position of the wave. Suggest using 2000 [bp/ min] * time [min], for mammalian data.
upstreamDist: The amount of upstream sequence to include Default: 10 kb.
TSmooth: Optimally, outlying windows are set a maximum value over the inter-quantile interval, specified by TSmooth. Reasonable value: 20; Default: NA (for no smoothing). Users are encouraged to use this parameter ONLY in combination with the normal distribution assumptions.
NonMap: Optionally, un-mappable positions are treated as missing data. NonMap passes in the list() structure for un-mappable regions.
prefix: Optionally, writes out png images of each gene examined for a wave. 'Prefix' denotes the file prefix for image names written to disk. Users are encouraged to create a new directory and write in a full path.
emissionDistAssumption: Takes values "norm", "normExp", and "gamma". Specifies the functional form of the 'emission' distribution for states I and II (i.e. 5' of the gene, and inside of the wave). In our experience, "gamma" works best for highly-variable 'spikey' data, and "norm" works for smooth data. As a general rule of thumb, "gamma" is used for libraries made using the direct ligation method, and "norm" for circular ligation data. Default: "gamma".
polymeraseWave

finterWindowSize
Method returns 'quality' information for each gene to which a wave was fit. Included in these metrics are several that define a moving window. The moving window size is specified by filterWindowSize. Default: 10 kb.

limitPCRDups
If true, counts only 1 read at each position with >= 1 read. NOT recommended to set this to TRUE. Default: FALSE.

returnVal
Takes value "simple" (default) or "alldata". "simple" returns a data.frame with Pol II wave end positions. "alldata" returns all of the available data from each gene, including the full posterior distribution of the model after EM.

departure
If TRUE, prints error messages.

Details
The model computes differences in read counts between the two conditions. Differences are assumed fit a functional form which can be specified by the user (using the emissionDistAssumption argument). Currently supported functional forms include a normal distribution (good for GRO-seq data prepared using the circular ligation protocol), a gamma distribution (good for 'spikey' ligation based GRO-seq data), and a long-tailed normal+exponential distribution was implemented, but never deployed.

Initial parameter estimates are based on initial assumptions of transcription rates taken from the literature. Subsequently all parameters are fit using Baum-Welch expectation maximization.


Arguments:

Value
Returns either a data.frame with Pol II wave end positions, or a List() structure with additional data, as specified by returnVal.

Author(s)
Charles G. Danko

Examples

genes <- GRanges("chr7", IRanges(2394474,2420377), strand="+", SYMBOL="CYP2W1", ID="54905")
reads1 <- as(readGAlignments(system.file("extdata", "S0mR1.bam", package="groHMM")), "GRanges")
reads2 <- as(readGAlignments(system.file("extdata", "S40mR1.bam", package="groHMM")), "GRanges")
approxDist <- 2000*10
# Not run:
# pw <- polymeraseWave(reads1, reads2, genes, approxDist)
**readBed**

`readBed` Returns a GenomicRanges object constructed from the specified bed file.

**Description**

Bed file format is assumed to be either four column: seqnames, start, end, strand columns; or six column: seqnames, start, end, name, score, and strand. Three column format is also possible when there is no strand information.

**Usage**

```r
readBed(file, ...)  
```

**Arguments**

- `file` Path to the input file.
- `...` Extra argument passed to `read.table`

**Details**

Any additional arguments available to `read.table` can be specified.

**Value**

Returns GRanges object representing mapped reads.

**Author(s)**

Minho Chae and Charles G. Danko.

---

**RgammaMLE**

RgammaMLE fits a gamma distribution to a specified data vector using maximum likelihood.

**Description**

RgammaMLE fits a gamma distribution to a specified data vector using maximum likelihood.

**Usage**

```r
RgammaMLE(X)  
```

**Arguments**

- `X` A vector of observations, assumed to be real numbers in the interval [0,+Inf).

**Value**

Returns a list of parameters for the best-fit gamma distribution (shape and scale).
Author(s)
Charles G. Danko

Rnorm

Rnorm fits a normal distribution to a specified data vector using maximum likelihood.

Usage
Rnorm(X)

Arguments
X
A vector of observations, assumed to be real numbers in the interval (-Inf,+Inf).

Value
Returns a list of parameters for the best-fit normal distribution (mean and variance).

Author(s)
Charles G. Danko

Rnorm.exp

Rnorm.exp fits a normal+exponential distribution to a specified data vector using maximum likelihood.

Description
Distribution function defined by: alpha*Normal(mean, varience)+(1-alpha) *Exponential(lambda).

Usage
Rnorm.exp(xi, wi = rep(1, NROW(xi)), guess = c(0.5, 0, 1, 1),
          tol = sqrt(.Machine$double.eps), maxit = 10000)

Arguments
xi
A vector of observations, assumed to be real numbers in the interval (-Inf,+Inf).

wi
A vector of weights. Default: vector of repeating 1; indicating all observations are weighted equally. (Are these normalized internally? Or do they have to be [0,1]?)

guess
Initial guess for parameters. Default: c(0.5, 0, 1, 1).

tol

maxit
Maximum number of iterations. Default: 10,000.
Details

Fits nicely with data types that look normal overall, but have a long tail starting for positive values.

Value

Returns a list of parameters for the best-fit normal distribution (alpha, mean, variance, and lambda).

Author(s)

Charles G. Danko

runMetaGene  Runs metagene analysis for sense and antisense direction.

Description

Supports parallel processing using mclapply in the ‘parallel’ package. To change the number of processors, set the option ‘mc.cores’.

Usage

runMetaGene(features, reads, anchorType = "TSS", size = 100L,
            normCounts = 1L, up = 10000L, down = NULL, sampling = FALSE,
            nSampling = 1000L, samplingRatio = 0.1, ...)

Arguments

features  GRanges A GRanges object representing a set of genomic coordinates, i.e., set of genes.
reads  GRanges of reads.
anchorType  Either ‘TSS’ or ‘TTS’. Metagene will be centered on the transcription start site(TSS) or transcription termination site(TTS). Default: TSS.
size  Numeric. The size of the moving window. Default: 100L
normCounts  Numeric. Normalization vector such as average reads. Default: 1L
up  Numeric. Distance upstream of each feature to align and histogram. Default: 1 kb
down  Numeric. Distance downstream of each feature to align and histogram. If NULL, down is same as up. Default: NULL
sampling  Logical. If TRUE, subsampling of Metagene is used. Default: FALSE
nSampling  Numeric. Number of subsampling. Default: 1000L
samplingRatio  Numeric. Ratio of sampling for features. Default: 0.1
...  Extra argument passed to mclapply.

Value

A list of integer-Rle for sense and antisense.
Author(s)
Minho Chae

Examples
features <- GRanges("chr7", IRanges(start=1000:1001, width=rep(1,2)),
  strand=c("+", "-"))
reads <- GRanges("chr7", IRanges(start=c(1000:1003, 1100:1101),
  width=rep(1, 6)), strand=rep(c("+","-"), 3))
## Not run:
# mg <- runMetaGene(features, reads, size=4, up=10)

tlsDeming A 'total least squares' implementation using demming regression.

Description
A 'total least squares' implementation using demming regression.

Usage
tlsDeming(x, y, d = 1)

Arguments
x X values.
y Y values.
d Ratio of variances. Default: 1, for orthogonal regression.

Value
Parameters for the linear model.

Author(s)
Charles G. Danko

tlsLoess A 'total least squares'-like hack for LOESS. Works by rotating points 45 degrees, fitting LOESS, and rotating back.

Description
A 'total least squares'-like hack for LOESS. Works by rotating points 45 degrees, fitting LOESS, and rotating back.

Usage
tlsLoess(x, y, theta = -pi/4, span = 1)
**tlsSvd**

**Arguments**

- `x`: X values.
- `y`: Y values.
- `theta`: Amount to rotate, sets the ratio of variances that are assumed by the hack. Default: `-pi/4` radians (45 degrees) for orthogonal regression.
- `span`: The LOESS span parameter. Default: 1

**Value**

List of input values and LOESS predictions.

**Author(s)**

Charles G. Danko

---

**tlsSvd**

A 'total least squares' implementation using singular value decomposition.

**Description**

A 'total least squares' implementation using singular value decomposition.

**Usage**

```r
tlsSvd(x, y)
```

**Arguments**

- `x`: X values.
- `y`: Y values.

**Value**

Parameters for the linear model `Y=a*X+e`.

**Author(s)**

Charles G. Danko
windowAnalysis  

Returns a vector of integers representing the counts of reads in a moving window.

Description

Supports parallel processing using mclapply in the ‘parallel’ package. To change the number of processors, set the option ‘mc.cores’.

Usage

```
windowAnalysis(reads, strand = "*", windowSize = stepSize, 
               stepSize = windowSize, chrom = NULL, limitPCRDups = FALSE, ...)
```

Arguments

- **reads**: GenomicRanges object representing the position of reads mapping in the genome.
- **strand**: Takes values of "+", "-", or "*". "*" denotes collapsing reads on both strands. Default: "+".
- **windowSize**: Size of the moving window. Either windowSize or stepSize must be specified.
- **stepSize**: The number of bp moved with each step.
- **chrom**: Chromosome for which to return data. Default: returns all available data.
- **limitPCRDups**: Counts only one read mapping to each start site. NOTE: If set to TRUE, assumes that all reads are the same length (don’t use for paired-end data). Default: FALSE.
- **...**: Extra argument passed to mclapply

Value

Returns a list object, each element of which represents a chromosome.

Author(s)

Charles G. Danko and Minho Chae

Examples

```
S0mR1 <- as(readGAlignments(system.file("extdata", "S0mR1.bam", 
                                        package="groHMM")), "GRanges")
## Not run:
# fp <- windowAnalysis(S0mR1, strand="+", windowSize=50)
```
writeWiggle

writeWiggle writes a wiggle track or BigWig file suitable for uploading to the UCSC genome browser.

Description

writeWiggle writes a wiggle track or BigWig file suitable for uploading to the UCSC genome browser.

Usage

writeWiggle(reads, file, strand = "*", fileType = "wig", size = 50, normCounts = NULL, reverse = FALSE, seqinfo = NULL, track.type.line = FALSE, ...)

Arguments

reads  GenomicRanges object representing the position of reads mapping in the genome.
file Specifies the filename for output.
strand Takes values of "+", ",", or "*". Computes a wig for the specified strand. "*" denotes collapsing reads on both strands. Default: "+".
fileType Takes values of "wig" or "BigWig". Default: "wig".
size Size of the moving window.
normCounts A normalization factor correcting for library size or other effects. For example, total mappable read counts might be a reasonable value. Default: 1 (i.e. no normalization).
reverse If set to TRUE, multiplies values by -1. Used for reversing GRO-seq data on the negative (-) strand. Default: FALSE
seqinfo Seqinfo object for reads. Default: NULL.
track.type.line If set to TRUE, prints a header identifying the file as a wiggle. Necessary to upload a custom track to the UCSC genome browser. Default: TRUE
... Extra argument passed to mclapply.

Author(s)

Minho Chae and Charles G. Danko

Examples

S0mR1 <- as(readGAlignments(system.file("extdata", "S0mR1.bam", package="groHMM")), "GRanges")
## Not run:
# writeWiggle(reads=S0mR1, file="S0mR1_Plus.wig", fileType="wig", # strand="*", reverse=FALSE)
Index

«Topic package
  groHMM-package, 2
averagePlot, 3
breakTranscriptsOnGenes, 4
combineTranscripts, 5
countMappableReadsInInterval, 5
detectTranscripts, 6
evaluateHMMInAnnotations, 7
expressedGenes, 8
getCores, 9
getTxDensity, 9
groHMM (groHMM-package), 2
groHMM-package, 2
limitToXkb, 10
makeConsensusAnnotations, 11
metaGene, 12
metaGene_nL, 13
metaGeneMatrix, 13
pausingIndex, 14
polymeraseWave, 15
readBed, 17
RgammaMLE, 17
Rnorm, 18
Rnorm.exp, 18
runMetaGene, 19
tlsDeming, 20
tlsLoess, 20
tlsSvd, 21
windowAnalysis, 22
writeWiggle, 23