Package ‘wavClusteR’

May 22, 2024

Type Package
Title Sensitive and highly resolved identification of RNA-protein interaction sites in PAR-CLIP data
Version 2.38.0
Date 2015-05-07
Depends R (>= 3.2), GenomicRanges (>= 1.31.8), Rsamtools
Imports methods, BiocGenerics, S4Vectors (>= 0.17.25), IRanges (>= 2.13.12), Biostrings (>= 2.47.6), foreach, GenomicFeatures (>= 1.31.3), ggplot2, Hmisc, mclust, rtracklayer (>= 1.39.7), seqinr, stringr
Suggests BiocStyle, knitr, rmarkdown, BSgenome.Hsapiens.UCSC.hg19
Enhances doMC
VignetteBuilder knitr
Author Federico Comoglio and Cem Sievers
Maintainer Federico Comoglio <federico.comoglio@gmail.com>
Description The package provides an integrated pipeline for the analysis of PAR-CLIP data. PAR-CLIP-induced transitions are first discriminated from sequencing errors, SNPs and additional non-experimental sources by a non-parametric mixture model. The protein binding sites (clusters) are then resolved at high resolution and cluster statistics are estimated using a rigorous Bayesian framework. Post-processing of the results, data export for UCSC genome browser visualization and motif search analysis are provided. In addition, the package allows to integrate RNA-Seq data to estimate the False Discovery Rate of cluster detection. Key functions support parallel multicore computing. Note: while wavClusteR was designed for PAR-CLIP data analysis, it can be applied to the analysis of other NGS data obtained from experimental procedures that induce nucleotide substitutions (e.g. BisSeq).
License GPL-2
biocViews ImmunoOncology, Sequencing, Technology, RIPSeq, RNASeq, Bayesian
LazyLoad yes
RoxygenNote 7.0.2
Contents

wavClusteR-package ......................................................... 2
annotateClusters ................................................................. 4
estimateFDR ................................................................. 6
exportClusters ............................................................... 7
exportCoverage ............................................................. 8
exportHighConfSub .......................................................... 8
exportSequences ............................................................ 9
filterClusters ............................................................... 10
fitMixtureModel ............................................................ 12
getAllSub ................................................................. 13
gleClusters ............................................................... 14
getExpInterval ............................................................. 16
getHighConfSub .......................................................... 17
getMetaCoverage ......................................................... 18
getMetaGene .............................................................. 20
getMetaTSS ............................................................... 22
model ................................................................. 24
plotSizeDistribution ..................................................... 24
plotStatistics .......................................................... 25
plotSubstitutions ....................................................... 27
readSortedBam ........................................................... 28

Index 29
A comprehensive pipeline for the analysis of PAR-CLIP data. PAR-CLIP-induced transitions are first discriminated from sequencing errors, SNPs and additional non-experimental sources by a non-parametric mixture model. The protein binding sites (clusters) are then resolved at high resolution and cluster statistics are estimated using a rigorous Bayesian framework. Post-processing of the results, data export for UCSC genome browser visualization and motif search analysis are provided. In addition, the package allows to integrate RNA-Seq data to estimate the False Discovery Rate of cluster detection. Key functions support parallel multicore computing. Note: while wavClusteR was designed for PAR-CLIP data analysis, it can be applied to the analysis of other NGS data obtained from experimental procedures that induce nucleotide substitutions (e.g. BisSeq).

Description

Package: wavClusteR
Type: Package
Version: 2.5.0
Date: 2016-01-09
License: GPL-2

Author(s)

Federico Comoglio and Cem Sievers
Epigenomics Group
Department of Biosystems Science and Engineering (D-BSSE)
ETH Zurich, Switzerland
Maintainer: Federico Comoglio
<federico.comoglio@gmail.com>

References


annotateClusters  

Annotate clusters with respect to transcript features

Description

Carries out strand-specific annotation of clusters with respect to distinct transcript features, particularly introns, coding sequences, 3'-UTRs, 5'-UTRs. Mapping to multiple features and to those outside the above mentioned ones are reported. Unmapped clusters are then further analyzed and annotated with respect to features localizing on the anti-sense strand. Results can be plotted as dotchart and annotations are returned as clusters metadata.

Usage

annotateClusters(clusters, txDB = NULL, genome = "hg19", tablename = "ensGene", plot = TRUE, verbose = TRUE)

Arguments

- **clusters**: GRanges object containing individual clusters as identified by the `getClusters` function
- **txDB**: TranscriptDb object obtained through a call to the `makeTxDbFromUCSC` function in the GenomicFeatures package. Default is NULL, namely the object will be fetched internally
- **genome**: A character specifying the genome abbreviation used by UCSC. Available abbreviations are returned by a call to `ucscGenomes()[,"db"]`. Default is "hg19" (human genome)
- **tablename**: A character specifying the name of the UCSC table containing the transcript annotations to retrieve. Available table names are returned by a call to `supportedUCSCtables()`. Default is "ensGene", namely ensembl gene annotations
- **plot**: Logical, if TRUE a dotchart with cluster annotations is produced
- **verbose**: Logical, if TRUE processing steps are printed

Value

Same as the input GRanges object, with an additional metadata column containing the following character encoding of the genomic feature each cluster maps to:

- "CDS ss" Coding Sequence Sense Strand
- "Introns ss" Intron Sense Strand
- "3' UTR ss" 3' UTR Sense Strand
- "5' UTR ss" 5' UTR Sense Strand
- "Multiple" More than one of the above
- "CDS as" Coding Sequence Antisense Strand
- "Introns as" Intron Antisense Strand
annotateClusters

“3’ UTR as”  3’ UTR Antisense Strand
“5’ UTR as”  5’ UTR Antisense Strand
"Other"  None of the above

If plot=TRUE, a dotchart is produced in addition.

Author(s)
Federico Comoglio

References
M. Carlson and H. Pages and P. Aboyoun and S. Falcon and M. Morgan and D. Sarkar and M. Lawrence, GenomicFeatures: Tools for making and manipulating transcript centric annotations, R package version 1.12.4

See Also
getClusters

Examples

```r
require(BSgenome.Hsapiens.UCSC.hg19)

data( model, package = "wavClusteR" )

filename <- system.file( "extdata", "example.bam", package = "wavClusteR" )
exmaple <- readSortedBam( filename = filename )
countTable <- getAllSub( example, minCov = 10, cores = 1 )
highConfSub <- getHighConfSub( countTable, supportStart = 0.2, supportEnd = 0.7, substitution = "TC" )
coverage <- coverage( example )
clusters <- getClusters( highConfSub = highConfSub,
  coverage = coverage,
  sortedBam = example,
  cores = 1,
  threshold = 2 )

fclusters <- filterClusters( clusters = clusters,
  highConfSub = highConfSub,
  coverage = coverage,
  model = model,
  genome = Hsapiens,
  refBase = 'T',
  minWidth = 12 )

## Not run: fclusters <- annotateClusters( clusters = fclusters )
```
**estimateFDR**

*Estimate False Discovery Rate within the relative substitution frequency support by integrating PAR-CLIP data and RNA-Seq data*

**Description**

Estimate upper and lower bounds for the False Discovery Rate within the relative substitution frequency (RSF) support by integrating PAR-CLIP data and RNA-Seq data (current version makes use of unstranded RNA-Seq).

**Usage**

```r
estimateFDR(countTable, RNASeq, substitution = 'TC', minCov = 20, span = 0.1, cores = 1, plot = TRUE, verbose = TRUE, ...)
```

**Arguments**

- `countTable`: A GRanges object, corresponding to a count table as returned by the `getAllSub` function.
- `RNASeq`: GRanges object containing aligned RNA-Seq reads as returned by `readSortedBam`.
- `substitution`: A character indicating which substitution is induced by the experimental procedure (e.g. 4-SU treatment - a standard in PAR-CLIP experiments - induces T to C transitions and hence substitution = 'TC' in this case.)
- `minCov`: An integer defining the minimum coverage required at a genomic position exhibiting a substitution. Genomic positions of coverage less than `minCov` are discarded. Default is 20 (see Details).
- `span`: A numeric indicating the width of RSF intervals to be considered for FDR computation. Defaults is 0.1 (i.e. 10 intervals are considered spanning the RSF support (0,1]).
- `cores`: An integer defining the number of cores to be used for parallel processing, if available. Default is 1.
- `plot`: Logical, if TRUE a dotchart with cluster annotations is produced.
- `verbose`: Logical, if TRUE processing steps are printed.
- `...`: Additional parameters to be passed to the `plot` function.

**Details**

For details on the FDR computation, please see Comoglio, Sievers and Paro.

**Value**

A list with three slots, containing upper and lower FDR bounds, and the total number of positive instances each RSF interval. If `plot`, these three vectors are depicted as a line plot.
**Note**

The approach used to compute the upper bound for the FDR is very conservative. See supplementary information in Comoglio et al. for details.

**Author(s)**

Federico Comoglio and Cem Sievers

**See Also**

`readSortedBam`, `getAllSub`


---

**exportClusters**

*Export clusters as BED track*

**Description**

Export clusters as BED track, compatible with the UCSC genome browser

**Usage**

```r
exportClusters(clusters, filename = 'wavClusters.bed', trackname = 'wavClusters', description = 'wavClusters')
```

**Arguments**

- `clusters`  
  - GRanges object containing individual clusters as identified by the `filterClusters` function
- `filename`  
  - A character defining the BED file name. Default to "wavClusters.bed"
- `trackname`  
  - A character defining the track.name of the BED file. Default to "wavClusters"
- `description`  
  - A character defining the description of the BED file. Default to "wavClusters"

**Value**

A BED file of the exported GRanges object

**Note**

Clusters are color coded according to their strand information (red for the plus strand, blue for the minus strand).

**Author(s)**

Federico Comoglio
**See Also**

- `filterClusters`

---

**Export coverage as BigWig track**

**Description**

Export coverage as BigWig track, compatible with the UCSC genome browser

**Usage**

```r
eexportCoverage(coverage, filename = 'wavClusters.BigWig')
```

**Arguments**

- `coverage`: An Rle object containing the coverage at each genomic position as returned by a call to `coverage`
- `filename`: A character defining the BED file name. Default to "wavClusters.BigWig"

**Value**

A BigWig file of the exported Rle object

**Author(s)**

Federico Comoglio

---

**Export high-confidence substitutions as BED track**

**Description**

Export high-confidence substitutions as BED track, compatible with the UCSC genome browser

**Usage**

```r
eexportHighConfSub(highConfSub, filename = 'highConfSub.bed', trackname = 'highConfSub', description = 'highConfSub')
```

**Arguments**

- `highConfSub`: GRanges object containing high-confidence substitution sites as returned by the `getHighConfSub` function
- `filename`: A character defining the BED file name. Default to "wavClusters.bed"
- `trackname`: A character defining the track.name of the BED file. Default to "wavClusters"
- `description`: A character defining the description of the BED file. Default to "wavClusters"
**Description**
Export cluster sequences for motif search analysis (FASTA format), e.g. using MEME-ChIP

**Usage**
```
exportSequences(clusters, filename = 'wavClusters.fasta')
```

**Arguments**

- **clusters**: GRanges object containing individual clusters as identified by the `filterClusters` function
- **filename**: A character defining the BED file name. Default to "wavClusters.fasta"

**Value**
A FASTA file containing the cluster sequences

**Author(s)**
Federico Comoglio

**See Also**
`filterClusters`
filterClusters

*Merge clusters and compute all relevant cluster statistics*

**Description**

If clusters have been identified using the mini-rank norm algorithm, cluster statistics are computed. In contrast, if the CWT-based cluster identification algorithm was used, clusters are first filtered to retain only those instances containing a wavelet peak and a high-confidence substitution site within their cluster boundaries.

**Usage**

```r
filterClusters(clusters, highConfSub, coverage, model, genome, refBase = 'T', minWidth = 12, verbose = TRUE)
```

**Arguments**

- `clusters`: GRanges object containing individual clusters as identified by the `getClusters` function
- `highConfSub`: GRanges object containing high-confidence substitution sites as returned by the `getHighConfSub` function
- `coverage`: An Rle object containing the coverage at each genomic position as returned by a call to `coverage`
- `model`: List of 5 items containing the estimated mixture model as returned by the `fitMixtureModel` function
- `genome`: BSgenome object of the relevant reference genome (e.g. Hsapiens for the human genome hg19)
- `refBase`: A character specifying the base in the reference genome for which transitions are experimentally induced (e.g. 4-SU treatment - a standard in PAR-CLIP experiments - induces T to C transitions and hence `refBase = "T"` in this case). Default is "T"
- `minWidth`: An integer corresponding to the minimum width of reported clusters. Shorter clusters are extended to `minWidth` starting from the cluster center
- `verbose`: Logical, if TRUE processing steps are printed

**Value**

GRanges object containing the transcriptome-wide identified clusters, having metadata:

- `Ntransitions`: The number of high-confidence transitions within the cluster
- `MeanCov`: The mean coverage within the cluster
- `NbasesInRef`: The number of genomic positions within the cluster corresponding to `refBase`
- `CrossLinkEff`: The crosslinking efficiency within the cluster, estimated as the ratio between the number of high-confidence transitions within the cluster and the total number of genomic positions therein corresponding to `refBase`
Sequence  The genomic sequence underlying the cluster (plus strand)
SumLogOdds The sum of the log-odd values within the cluster
RelLogOdds The sum of the log-odds divided by the number of high-confidence transitions within the cluster. This variable can be regarded as a proxy for statistical significance and can be therefore used to rank clusters. See Comoglio, Sievers and Paro for details.

Note

1) This function calls the appropriate processing function according to the method used to compute clusters. This information is stored in the metadata(ranges(clusters)) slot as an object of type list.

2) Notice that genome corresponds to the according reference genome matching the organism in which experiments have been carried out. For example genome = Hsapiens is used for the human reference genome (assembly 19), where Hsapiens is provided by BSgenome.Hsapiens.UCSC.hg19.

Author(s)

Federico Comoglio and Cem Sievers

References

Herve Pages, BSgenome: Infrastructure for Biostrings-based genome data packages


See Also
getClusters, getHighConfSub, fitMixtureModel

Examples

```r
require(BSgenome.Hsapiens.UCSC.hg19)
data( model, package = "wavClusterR" )
filename <- system.file( "extdata", "example.bam", package = "wavClusterR" )
example <- readSortedBam( filename = filename )
countTable <- getAllSub( example, minCov = 10, cores = 1 )
highConfSub <- getHighConfSub( countTable, supportStart = 0.2, supportEnd = 0.7, substitution = "TC" )
coverage <- coverage( example )
clusters <- getClusters( highConfSub = highConfSub, coverage = coverage, sortedBam = example, cores = 1, threshold = 2 )
```
fclusters <- filterClusters( clusters = clusters, 
highConfSub = highConfSub, 
coverage = coverage, 
model = model, 
genome = Hsapiens, 
refBase = 'T', 
minWidth = 12 )
fclusters

fitMixtureModel  

Fit a non-parametric mixture model from all identified substitutions

Description
Estimates the two-component mixture model consisting of the mixing coefficients and the density functions.

Usage
fitMixtureModel(countTable, substitution = "TC")

Arguments
countTable   A GRanges object, corresponding to a count table as returned by the getAllSub function
substitution A character indicating which substitution is induced by the experimental procedure (e.g. 4-SU treatment - a standard in PAR-CLIP experiments - induces T to C transitions and hence substitution = 'TC' in this case.)

Value
A list containing:

11     The first mixing coefficient
12     The second mixing coefficient
p      The mixture model
p1     The first component of the mixture
p2     The second component of the mixture

Author(s)
Federico Comoglio and Cem Sievers

See Also
getAllSub getExpInterval
getAllSub

Examples

```r
## Not run:
filename <- system.file( "extdata", "example.bam", package = "wavClusteR" )
example <- readSortedBam(filename = filename)
countTable <- getAllSub( example, minCov = 10, cores = 1 )

fitMixtureModel( countTable, substitution = "TC" )

## End(Not run)

#load and inspect the model
data( model )
str( model )

#getExpInterval( model, bayes = TRUE, plot = TRUE )
```

getAllSub

Identify all substitutions observed across genomic positions exhibiting a specified minimum coverage

Description

All substitutions observed across genomic positions exhibiting user-defined minimum coverage are extracted and a count table is returned. This function supports parallel computing.

Usage

ggetAllSub(sortedBam, minCov = 20, cores = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sortedBam</td>
<td>GRanges object containing aligned reads as returned by <code>readSortedBam</code></td>
</tr>
<tr>
<td>minCov</td>
<td>An integer defining the minimum coverage required at a genomic position exhibiting a substitution. Genomic positions of coverage less than minCov are discarded. Default is 20 (see Details).</td>
</tr>
<tr>
<td>cores</td>
<td>An integer defining the number of cores to be used for parallel processing, if available. Default is 1.</td>
</tr>
</tbody>
</table>

Details

The choice of the minimum coverage influences the variance of the relative substitution frequency estimates, which in turn affect the mixture model fit. A conservative value depending on the library size is recommended for a first analysis. Values smaller than 10 have not been tested and are therefore not recommended.
getClusters

Value

A GRanges object containing a count table, where each range correspond to a substitution. The metadata correspond to the following information:

- **substitutions**: observed substitution, e.g. AT, i.e. A in the reference sequence and T in the mapped read.
- **coverage**: strand-specific coverage.
- **count**: number of strand-specific substitutions.

Author(s)

Federico Comoglio and Cem Sievers, with contributions from Martin Morgan

See Also

readSortedBam

Examples

```r
filename <- system.file( "extdata", "example.bam", package = "wavClusteR" )
example <- readSortedBam(filename = filename)
countTable <- getAllSub( example, minCov = 10, cores = 1 )
countTable
```

---

getClusters  Identify clusters containing high-confidence substitutions and resolve boundaries at high resolution

Description

Identifies clusters using the mini-rank norm (MRN) algorithm, which employs thresholding of background coverage differences and finds the optimal cluster boundaries by exhaustively evaluating all putative clusters using a rank-based approach. This method has higher sensitivity and an approximately 10-fold faster running time than the CWT-based cluster identification algorithm.

Usage

```r
getClusters(highConfSub, coverage, sortedBam, cores = 1, threshold)
```
getClusters

Arguments

- **highConfSub**: GRanges object containing high-confidence substitution sites as returned by the `getHighConfSub` function.
- **coverage**: An Rle object containing the coverage at each genomic position as returned by a call to `coverage`.
- **sortedBam**: a GRanges object containing all aligned reads, including read sequence (qseq) and MD tag (MD), as returned by the `readSortedBam` function.
- **cores**: integer, the number of cores to be used for parallel evaluation. Default is 1.
- **threshold**: numeric, the difference in coverage to be considered noise. If not specified, a Gaussian mixture model is used to learn a threshold from the data. Empirically, 10% of the minimum coverage required at substitutions (see argument `minCov` in the `getHighConfSub` function) might suffice to provide highly resolved clusters. However, if `minCov` is much lower than the median strand-specific coverage at substitutions $m$, which can be computed using `summary(elementMetadata(highConfSub)[, 'coverage'])['Median'],`, 10% of $m$ might represent an optimal choice.

Value

GRanges object containing the identified cluster boundaries.

Note

Clusters returned by this function need to be further merged by the function `filterClusters`, which also computes all relevant cluster statistics.

Author(s)

Federico Comoglio and Cem Sievers

References


See Also

`getHighConfSub`, `filterClusters`

Examples

```r
filename <- system.file( "extdata", "example.bam", package = "wavClusteR" )
example <- readSortedBam( filename = filename )
countTable <- getAllSub( example, minCov = 10, cores = 1 )
highConfSub <- getHighConfSub( countTable, supportStart = 0.2, supportEnd = 0.7, substitution = "TC" )
coverage <- coverage( example )
```
clusters <- getClusters( highConfSub = highConfSub, coverage = coverage, sortedBam = example, cores = 1, threshold = 2 )

getExpInterval  Identify the interval of relative substitution frequencies dominated by experimental induction.

Description

Identifies the interval/support of relative substitution frequencies (RSFs) dominated by the second model component, i.e. by the probability of being induced by the experimental procedure. In addition, this function can be used to generate diagnostic plots of the model fit, representing (i) model densities and log odds ratio (ii) the posterior class probability, i.e. the probability of a given observation being generated by experimental induction.

Usage

getExpInterval(model, bayes = TRUE, leftProb, rightProb, plot = TRUE)

Arguments

model A list containing the model as returned by the function fitMixtureModel
bayes Logical, if TRUE the Bayes classifier (cutoff at posterior class probabilities >= 0.5) is applied. If FALSE, custom cutoff values should be provided through leftProb and rightProb. Default is TRUE.
leftProb Numeric, the posterior probability corresponding to the left boundary (start) of the high confidence RSF interval.
rightProb Numeric, the posterior probability corresponding to the right boundary (end) of the high confidence RSF interval.
plot Logical, if TRUE diagnostics plot showing the model components, log odds and the computed posterior with highlighted identified RSF interval are returned.

Value

A list with two numeric slots, corresponding to the extremes of the RSF interval (RSF support).

supportStart start of the high confidence RSF interval
supportEnd end of the high confidence RSF interval

Author(s)

Federico Comoglio and Cem Sievers
getHighConfSub

References


See Also

fitMixtureModel getHighConfSub estimateFDR

Examples

data( model )

#default
support <- getExpInterval( model = model, bayes = TRUE, plot = TRUE )
support

#custom interval (based, e.g. on visual inspection of posterior class probability
# or evaluation of FDR using the estimateFDRF function)
support <- getExpInterval( model = model, leftProb = 0.2, rightProb = 0.7, plot = TRUE )
support

getHighConfSub-countTable

Classify substitutions based on identified RSF interval and return high confidence transitions

Description

Classify genomic positions exhibiting a substitution based on the relative substitution frequency (RSF) interval. The latter is returned by the getExpInterval function, but can be user-specified through visual inspection of the posterior class probability returned by the same function.

Usage

getHighConfSub(countTable, support, supportStart = NA, supportEnd = NA, substitution = "TC")

Arguments

countTable A GRanges object, corresponding to a count table as returned by the getAllSub function

support List, consisting of two numeric slots defining the left and right boundaries (start and end values, respectively) of the RSF interval, as returned by the getExpInterval function.
getMetaCoverage

supportStart Numeric, if support not provided, the RSF value determining the left boundary (start) of the RSF interval. Use this argument to specify a user-defined RSF interval.

supportEnd Numeric, if support not provided, the RSF value determining the right boundary (end) of the RSF interval. Use this argument to specify a user-defined RSF interval.

substitution A character indicating which substitution is induced by the experimental procedure (e.g. 4-SU treatment - a standard in PAR-CLIP experiments - induces T to C transitions and hence substitution = 'TC' in this case.)

Value

a GRanges object containing high confidence substitutions, with strand-specific coverage, counts and RSF values as metadata.

Note

In the example below, left and right boundaries were arbitrarily chosen as showcase.

Author(s)

Federico Comoglio and Cem Sievers

See Also

getAllSub, getExpInterval

Examples

```r
filename <- system.file( "extdata", "example.bam", package = "wavClusteR" )
example <- readSortedBam( filename = filename )
countTable <- getAllSub( example, minCov = 10, cores = 1 )
highConfSub <- getHighConfSub( countTable, supportStart = 0.2, supportEnd = 0.7, substitution = "TC" )
highConfSub
```

getMetaCoverage

Compute and plot distribution of average coverage or relative log-odds as metagene profile using identified clusters

Description

Transcriptome-wide identified clusters are used to generate a metagene profile by binning gene bodies. Within each bin, the distribution of the average cluster coverage or of the relative log-odds is computed.
getMetaCoverage

Usage

getMetaCoverage(clusters, txDB = NULL, upstream = 1e3, downstream =
1e3, nBins = 40, nBinsUD = 10, minLength = 1, genome = 'hg19', tablename =
'ensGene', odds = FALSE, plot = TRUE, verbose = TRUE, ...)

Arguments

clusters    GRanges object containing individual clusters as identified by the getClusters
            function

txDB         TranscriptDb object obtained through a call to the makeTxDbFromUCSC function
            in the GenomicFeatures package. Default is NULL, namely the object will be
            fetched internally

upstream     An integer corresponding to the number of bases to be considered upstream the
            gene. Default is 1000

downstream   An integer corresponding to the number of bases to be considered downstream
            the gene. Default is 1000

nBins        An integer corresponding to the number of bins to be used to partition the genes.
            Default is 40

nBinsUD      An integer corresponding to the number of bins to be used to partition upstream
            and downstream regions. Defaults is 10, i.e. the bin size is 100 bases for the
            default extension

minLength    An integer indicating the the minimum required length of a gene in order for it
            to be considered. Default is 1, i.e. all genes are considered

genome       A character specifying the genome abbreviation used by UCSC. Available abbreviations
            are returned by a call to ucscGenomes()[, "db"]. Default is "hg19" (human genome)

tablename    A character specifying the name of the UCSC table containing the transcript
            annotations to retrieve. Available table names are returned by a call to
            supportedUCSCtables(). Default is "ensGene", namely ensembl gene annotations

odds         Logical, if TRUE relative log-odds distributions are shown instead of mean
            coverage

plot         Logical, if TRUE a dotchart with cluster annotations is produced

verbose      Logical, if TRUE processing steps are printed

...          Additional parameters to be passed to the plot function

Value

Called for its effects.

Author(s)

Federico Comoglio
References

Comoglio F*, Sievers C* and Paro R, wavClusteR: an R package for PAR-CLIP data analysis, submitted

See Also

getClusters

Examples

```r
require(BSgenome.Hsapiens.UCSC.hg19)

data( model, package = "wavClusteR" )

filename <- system.file( "extdata", "example.bam", package = "wavClusteR" )
example <- readSortedBam( filename = filename )
countTable <- getAllSub( example, minCov = 10, cores = 1 )
highConfSub <- getHighConfSub( countTable, supportStart = 0.2, supportEnd = 0.7, substitution = "TC" )
coverage <- coverage( example )
clusters <- getClusters( highConfSub = highConfSub,
                         coverage = coverage,
                         sortedBam = example,
                         threshold = 2 )

fclusters <- filterClusters( clusters = clusters,
                             highConfSub = highConfSub,
                             coverage = coverage,
                             sortedBam = example,
                             model = model,
                             genome = Hsapiens,
                             refBase = 'T',
                             minWidth = 12 )

## Not run: getMetaCoverage( clusters = fclusters, odds = FALSE )
```

getMetaGene

Compute and plot metagene profile using identified clusters

Description

Transcriptome-wide identified clusters are used to generate a metagene profile by binning gene bodies, upstream and downstream regions.

Usage

```r
getMetaGene(clusters, txDB = NULL, upstream = 1e3, downstream = 1e3,
            nBins = 40, nBinsUD = 10, minLength = 1, genome = 'hg19', tablename = 'ensGene', plot = TRUE, verbose = TRUE, ...)
```
getMetaGene

Arguments

clusters  GRanges object containing individual clusters as identified by the getClusters function

txDB  TranscriptDb object obtained through a call to the makeTxDbFromUCSC function in the GenomicFeatures package. Default is NULL, namely the object will be fetched internally

upstream  An integer corresponding to the number of bases to be considered upstream the gene. Default is 1000

downstream  An integer corresponding to the number of bases to be considered downstream the gene. Default is 1000

nBins  An integer corresponding to the number of bins to be used to partition the genes. Default is 40

nBinsUD  An integer corresponding to the number of bins to be used to partition upstream and downstream regions. Default is 10, i.e. the bin size is 100 bases for the default extension

minLength  An integer indicating the minimum required length of a gene in order for it to be considered. Default is 1, i.e. all genes are considered

genome  A character specifying the genome abbreviation used by UCSC. Available abbreviations are returned by a call to ucscGenomes()[,"db"]. Default is "hg19" (human genome)

tablename  A character specifying the name of the UCSC table containing the transcript annotations to retrieve. Available table names are returned by a call to supportedUCSCtables(). Default is "ensGene", namely ensembl gene annotations

plot  Logical, if TRUE a dotchart with cluster annotations is produced

verbose  Logical, if TRUE processing steps are printed

...  Additional parameters to be passed to the plot function

Value

A numeric vector of the same length as nBins + 2 * nBinsUD containing normalized counts. If plot, the metagene profile is also depicted as a line plot.

Author(s)

Federico Comoglio


References

Comoglio F*, Sievers C* and Paro R, wavClusteR: an R package for PAR-CLIP data analysis, submitted

See Also

getClusters
getMetaTSS

Compute and plot read densities in genomic regions around transcription start sites

Description

Aligned reads are used to generate a metaTSS profile across genomic regions containing transcription start sites (TSSs).

Usage

getMetaTSS(sortedBam, txDB = NULL, upstream = 1e3, downstream = 1e3,
nBins = 40, genome = 'hg19', tablename = 'ensGene', unique = FALSE, plot =
TRUE, verbose = TRUE, ...)
downstream
An integer corresponding to the number of bases to be considered downstream the TSS. Default is 1000

nBins
An integer corresponding to the number of bins to be used to partition the genes. Default is 40, i.e. bin size 50 bases

geno me
A character specifying the genome abbreviation used by UCSC. Available abbreviations are returned by a call to `ucscGenomes()[, "db"]`. Default is "hg19" (human genome)

tablename
A character specifying the name of the UCSC table containing the transcript annotations to retrieve. Available table names are returned by a call to `supportedUCSCtables()`. Default is "ensGene", namely ensembl gene annotations

unique
Logical, if TRUE only non-overlapping TSSs extended by upstream/downstream are considered. Default is FALSE, i.e. all TSSs are considered

plot
Logical, if TRUE a dotchart with cluster annotations is produced

verbose
Logical, if TRUE processing steps are printed

... Additional parameters to be passed to the `plot` function

Value
A numeric vector of the same length as `nBins` containing normalized counts. If `plot`, the metaTSS profile is also depicted as a line plot.

Author(s)
Federico Comoglio

References
Comoglio F*, Sievers C* and Paro R, wavClusteR: an R package for PAR-CLIP data analysis, submitted

See Also
`readSortedBam`

Examples
```r
require(BSgenome.Hsapiens.UCSC.hg19)

filename <- system.file( "extdata", "example.bam", package = "wavClusteR" )
example <- readSortedBam( filename = filename )
## Not run: tss <- getMetaTSS( sortedBam = example )
```
**model**  
*Components of the non-parametric mixture model fitted on Ago2 PAR-CLIP data*

**Description**

The non-parametric mixture model was fit on the entire Ago2 public available PAR-CLIP dataset (Kishore et al.) using the `fitMixtureModel` function.

**Usage**

```r
data(model)
model
```

**Format**

List of 5 items containing the estimated mixing coefficients and model densities. See the help page of the `fitMixtureModel` function for a detailed description of the output.

**References**


**See Also**

`fitMixtureModel`

---

**plotSizeDistribution**  
*Plot the distribution of cluster sizes*

**Description**

Produce an histogram of cluster sizes

**Usage**

```r
plotSizeDistribution( clusters, showCov = FALSE, ... )
```

**Arguments**

- `clusters`  
  GRanges object containing individual clusters as identified by the `getClusters` function

- `showCov`  
  logical, if TRUE a scatter plot of average cluster coverage vs. cluster size is shown along with a loess fit. Default is FALSE.

- `...`  
  Additional parameters to be passed to the `hist` function
plotStatistics

Value
Called for its effect, returns a histogram.

Author(s)
Federico Comoglio

See Also
getClusters

Examples

require(BSgenome.Hsapiens.UCSC.hg19)
data( model, package = "wavClusteR" )
filename <- system.file("extdata", "example.bam", package = "wavClusteR")
example <- readSortedBam( filename = filename )
countTable <- getAllSub( example, minCov = 10, cores = 1 )
highConfSub <- getHighConfSub( countTable, supportStart = 0.2, supportEnd = 0.7, substitution = "TC" )
coverage <- coverage( example )
clusters <- getClusters( highConfSub = highConfSub,
                       coverage = coverage,
                       sortedBam = example,
                       threshold = 2 )
fclusters <- filterClusters( clusters = clusters,
                             highConfSub = highConfSub,
                             coverage = coverage,
                             model = model,
                             genome = Hsapiens,
                             refBase = 'T',
                             minWidth = 12 )
plotSizeDistribution(fclusters, breaks = 30, col = 'skyblue2')
plotStatistics

Arguments

clusters  GRanges object containing individual clusters as identified by the getClusters function

corMethod  A character defining the correlation coefficient to be computed. See the help page of the cor function for possible options. Default is "spearman". Hence, rank-based Spearman’s correlation coefficients are computed

lower  A function compatible with the lower panel argument of the pairs function

...  Additional parameters to be passed to the pairs function

Value
called for its effect

Author(s)
Federico Comoglio

See Also
getClusters

Examples

require(BSgenome.Hsapiens.UCSC.hg19)
data( model, package = "wavClusteR" )
filename <- system.file( "extdata", "example.bam", package = "wavClusteR" )
example <- readSortedBam( filename = filename )
countTable <- getAllSub( example, minCov = 10, cores = 1 )
highConfSub <- getHighConfSub( countTable, supportStart = 0.2, supportEnd = 0.7, substitution = "TC" )
coverage <- coverage( example )
clusters <- getClusters( highConfSub = highConfSub, coverage = coverage,
                        sortedBam = example,
                        threshold = 2 )
fclusters <- filterClusters( clusters = clusters,
                           highConfSub = highConfSub,
                           coverage = coverage,
                           model = model,
                           genome = Hsapiens,
                           refBase = 'T',
                           minWidth = 12 )
plotStatistics( clusters = fclusters )
plotSubstitutions

Barplot visualization of the number of genomic positions exhibiting a given substitution and, if model provided, additional diagnostic plots.

Description

Graphical representation of the total number of genomic positions exhibiting one or more substitutions of a given type. This information is used to estimate the mixing coefficients of the non-parametric mixture model. If the mixture model fit is provided, returns additional diagnostic plots such as the total number of reads exhibiting a given substitution and relative substitution frequency-dependent representations of the total number of genomic positions with substitutions of a given type.

Usage

plotSubstitutions(countTable, highlight = "TC", model)

Arguments

- **countTable**: A GRanges object, corresponding to a count table as returned by the `getAllSub` function.
- **highlight**: A character indicating which substitution should be highlighted in the barplot. A standard PAR-CLIP experiment employing 4-SU treatment induces T to C transitions, encoded as "TC". Default is "TC".
- **model**: A list containing the model as returned by the function `fitMixtureModel`.

Value

called for its effect

Author(s)

Federico Comoglio and Cem Sievers

See Also

getAllSub

Examples

```r
filename <- system.file( "extdata", "example.bam", package = "wavClusteR" )
example <- readSortedBam( filename = filename )
countTable <- getAllSub( example, minCov = 10, cores = 1 )
plotSubstitutions(countTable = countTable, highlight = "TC")
```
**readSortedBam**  

*Load a sorted BAM file*

**Description**

Load a sorted BAM file. Optionally, only reads mapping to a specific set of genomics coordinates are loaded. Only fields strictly necessary to run a wavClusteR analysis are loaded.

**Usage**

`readSortedBam(filename, which)`

**Arguments**

- `filename`  
  Name of the sorted BAM file, including full path to file if it is located outside the current working directory.

- `which`  
  a GRanges or IntegerRangesList specifying the regions on the reference sequence for which matches are desired. See the documentation of the Rsamtools package for details.

**Value**

a GRanges object containing aligned reads, including read sequence (qseq) and MD tag (MD)

**Note**

The input BAM file must be sorted and indexed. Alignment with bowtie or bowtie2, conversion from SAM to BAM output, sorting and indexing using SAMtools is recommended.

**Author(s)**

Federico Comoglio

**References**


**Examples**

```r
library(Rsamtools)
filename <- system.file("extdata", "example.bam", package = "wavClusteR")
sortedBam <- readSortedBam( filename = filename )
```
Index

* **Import**
  - readSortedBam, 28
* **core**
  - estimateFDR, 6
  - filterClusters, 10
  - fitMixtureModel, 12
  - getAllSub, 13
  - getClusters, 14
  - getExpInterval, 16
  - getHighConfSub, 17
* **datasets**
  - model, 24
* **graphics**
  - annotateClusters, 4
  - estimateFDR, 6
  - getExpInterval, 16
  - getMetaCoverage, 18
  - getMetaGene, 20
  - getMetaTSS, 22
  - plotSizeDistribution, 24
  - plotStatistics, 25
  - plotSubstitutions, 27
* **model**
  - fitMixtureModel, 12
* **package**
  - wavClusteR-package, 3
* **postprocessing**
  - annotateClusters, 4
  - exportClusters, 7
  - exportCoverage, 8
  - exportHighConfSub, 8
  - exportSequences, 9
  - getMetaCoverage, 18
  - getMetaGene, 20
  - getMetaTSS, 22
  - plotSizeDistribution, 24
  - plotStatistics, 25
  - readSortedBam, 6, 7, 13–15, 22, 23, 28
* coverage, 10, 15
  - estimateFDR, 6, 17
  - exportClusters, 7
  - exportCoverage, 8
  - exportHighConfSub, 8
  - exportSequences, 9
  - filterClusters, 7–9, 10, 15
  - fitMixtureModel, 10, 11, 12, 17, 24
  - getAllSub, 6, 7, 12, 13, 17, 18, 27
  - getClusters, 4, 5, 10, 11, 14, 19–21, 24–26
  - getExpInterval, 12, 16, 17, 18
  - getHighConfSub, 8–11, 15, 17, 17
  - getMetaCoverage, 18
  - getMetaGene, 20
  - getMetaTSS, 22
  - model, 24
  - plotSizeDistribution, 24
  - plotStatistics, 25
  - plotSubstitutions, 27
  - wavClusteR (wavClusteR-package), 3
  - wavClusteR-package, 2

annotateClusters, 4

29