Package 'htSeqTools'

March 26, 2013

Version 1.4.0

Date 2012-03-19

Title Quality Control, Visualization and Processing for High-Throughput Sequencing data

Author Evarist Planet, Camille Stephan-Otto, Oscar Reina, Oscar Flores, David Rossell

Maintainer Oscar Reina <oscar.reina@irbbarcelona.org>

Depends R (>= 2.12.2), methods, BiocGenerics (>= 0.1.0), Biobase,IRanges, methods, MASS, BSgenome, GenomicRanges

Enhances multicore

Description We provide efficient, easy-to-use tools for High-Throughput Sequencing (ChIP-seq, RNAseq etc.). These include MDS plots (analogues to PCA), detecting inefficient immunoprecipitation or over-amplification artifacts, tools to identify and test for genomic regions with large accumulation of reads, and visualization of coverage profiles.

LazyLoad yes

Collate ClassDefinitions.R GenericDefs.R alignPeaks.R extendRanges.R
filterDuplReads.R mergeRegions.R stdPeakLocation.R
rowLogRegLRT.R enrichedRegions.R enrichedPeaks.R listOverlap.R
tabDuplReads.R cmdsFit-class.R cmds.R fdrEnrichedCounts.R
islandCounts.R countRepeats.R ssdCoverage.R giniCoverage.R
regionsCoverage.R gridCover-methods.R enrichedChrRegions.R
plotChrRegions.R coverageDiff.R plotMeanCoverage.R

biocViews HighThroughputSequencing,QualityControl

R topics documented:

alignPeaks *Align peaks in a ChIP-Seq experiment by removing the strand specific bias.*

Description

Align peaks in a ChIP-Seq experiment by removing the shift between reads aligned to the plus and the minus strands.

Usage

```
alignPeaks(x, strand, npeaks = 1000, bandwidth = 150, mc.cores=1)
```

Arguments

x	A RangedDataList, RangedData or an IRangesList object containing the aligned reads in each chromosome.
strand	Strand that each read was aligned to. If x is of class RangedDataList, strand can be a character vector of length 1 indicating the name of the field in x indicating the strand, i.e. x[[1]][[strand]] contains the strand information.
npeaks	Number of peaks to be used to estimate the shift size.
bandwidth	Only reads with distance less than bandwidth between them and their closest gene are used to estimate the shift size.
mc.cores	Number of cores to be used for parallel computing (passed on to mclapply). Only used if x is of class RangedDataList.

Details

The procedure detects the npeaks highest peaks (using reads from both strands simultaneously). Then it selects reads which are less than bandwidth base pairs away from any of the peaks. Then it computes (a) the average distance between reads on the plus strand and the closest peak, (b) the same distance for reads on the minus strand. The mean difference between (a) and (b) is the estimated shift size. Reads on the plus strand are shifted to the right, whereas reads on the minus strands are shifted to the left.

Value

A CompressedIRangesList object with all reads shifted so that the strand specific bias is no longer present.

Methods

signature(x = "IRangesList", strand = "list") Each element in x corresponds to a chromosome, and each range gives the start/end of a sequence. strand indicates the strand for the ranges in x.

signature(x = "RangedData", strand = "character") x gives read start and end positions, and strand gives the name of the variable in values(x) containing the strand information.

signature(x = "RangedDataList", strand = "character") The method for RangedData is applied to each element in x separately, as each element may have a different strand-specific bias.

Examples

```
#Generate 1000 reads containing strand-specific bias
st <- runif(1000,1,250)
strand <- rep(c('+','-'),each=500)
st[strand=='-'] <- st[strand=='-'] + runif(500,50,100)
x <- RangedData(IRanges(st,st+38),strand=strand)
#Estimate and remove the bias
xalign <- alignPeaks(x, strand='strand', npeaks=1)
```

---

cmds *Classical Multi-Dimensional Scaling*

Description

cmds obtain the coordinates of the elements in x in a k dimensional space which best approximate the distances between objects. For high-throughput sequencing data we define the distance between two samples as 1 - correlation between their respective coverages. This provides PCA analog for sequencing data.

Usage

```
cmds(x, k=2, logscale=TRUE, mc.cores=1, cor.method='pearson')
```

Arguments

x	A RangedDataList object, e.g. each element containing the output of a sequencing run.
k	Dimensionality of the reconstructed space, typically set to 2 or 3.
logscale	If set to TRUE correlations are computed for log(x+1).
mc.cores	Number of cores. Setting mc.cores>1 allows running computations in parallel. Setting mc.cores to too large a value may require a lot of memory.
cor.method	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.

Value

The function returns a mdsFit object, with slots points containing the coordinates, d with the distances between elements, dapprox with the distances between objects in the approximated space, and R.square indicating the percentage of variability in d accounted for by dapprox.

Since the coverage distribution is typically highly asymetric, setting logscale=TRUE reduces the influence of the highest coverage regions in the distance computation, as this is based on the Pearson correlation coefficient.

Methods

signature(x = "RangedDataList") Use Classical Multi-Dimensional Scaling to plot each element of the RangedDataList object in a k-dimensional space. The coverage is computed for each element in x, and the pairwise correlations between elements is used to define distances.

Examples

```
data(htSample)
cmds1 <- cmds(htSample)

cmds1
plot(cmds1)
```

cmdsFit — *Classical Multi-Dimensional Scaling for a distance matrix*

Description

cmdsFit obtains coordinates in a k dimensional space which best approximate the given distances between objects.

Usage

```
cmdsFit(d, k=2, type='classic', add=FALSE, cor.method='pearson')
```

Arguments

d	Distances between objects
k	Dimensionality of the reconstructed space, typically set to 2 or 3.
type	Set to "classic" to perform classical MDS (uses function cmdscale from package stats). Set to "isoMDS" to use Kruskal's non-metric MDS (uses function isoMDS from package MASS).
add	Logical indicating if an additive constant c* should be computed, and added to the non-diagonal dissimilarities such that all n-1 eigenvalues are non-negative in cmdscale
cor.method	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.

Value

The function returns a cmdsFit object. See help("cmdsFit-class") for details.

Methods

signature(d = "matrix") Use Classical Multi-Dimensional Scaling to represent points in a k-dimensional space.

Examples

```
### Not run
#d <- matrix(c(0,5,10,5,0,15,10,15,0),byrow=TRUE,ncol=3)
#cmdsFit(d,add=TRUE)
```

cmdsFit-class *Class "cmdsFit"*

Description

Classical Multi-Dimensional Scaling Fit. Function cmds creates object of this class.

Objects from the Class

Objects can be created by calls of the form new("cmdsFit", ...).

Slots

points: Object of class "matrix" with (x,y) coordinates in the approximated space.

d: Object of class "matrix" with original distances between individuals.

dapprox: Object of class "matrix" with distances between individuals in the approximated space.

R.square: Percentage of variability in d explained by dapprox (object of class "numeric")

Methods

There are show and plot methods defined for this class.

Author(s)

David Rossell

See Also

cmdscale from package base.

Examples

```
showClass("cmdsFit")
```

---

countHitsWindow *Compute number of hits in a moving window along the chromosome.*

Description

Computes a smoothed number of hits along the chromosome by using moving windows of user specified size.

Usage

```
countHitsWindow(x, chrLength, windowSize = 10^4 - 1)
```

Arguments

x	Object containing hits (start, end and chromosome). Currently only RangedData objects are accepted.
chrLength	Named vector indicating the length of each chromosome in base pairs.
windowSize	Size of the window used to smooth the hit count.

Methods

signature(x = "RangedData") x contains chromosome, start and end positions for each hit.

Examples

```
set.seed(1)
st <- round(rnorm(1000,500,100))
st[st>=10000] <- 10000
strand <- rep(c('+','-'),each=500)
space <- rep('chr1',length(st))
x <- RangedData(IRanges(st,st+38),strand=strand,space=space)
countHitsWindow(x, chrLength=c(chr1=10000), windowSize=99)
```

coverageDiff *Compute the difference in coverage between two objects*

Description

Computes coverage of sample1 minus coverage of sample2, taking into account that the chromosomes in sample1 and sample2 are not necessarily the same.

Usage

```
coverageDiff(sample1, sample2, chrLength)
```

Arguments

sample1	Object with reads from sample 1. Typically, a RangedData object.
sample2	Object with reads from sample 2. Typically, a RangedData object.
chrLength	Named vector with chromosome lengths. This can be obtained from the Bioconductor annotation packages, e.g. BSgenome.Dmelanogaster.UCSC.dm3 for drosophila melanogaster, etc.

Details

Computation is restricted to chromosomes in names(chrLength).

Value

SimpleRleList with differences in coverage.

Examples

```
sample1 <- RangedData(IRanges(1:10,11:20),space='chr1')
sample2 <- RangedData(IRanges(1:10,11:20),space=rep(c('chr1','chr2'),each=5))
chrLength <- c(50,25); names(chrLength) <- c('chr1','chr2')
coverageDiff(sample1,sample2,chrLength)
```

enrichedChrRegions *Find chromosomal regions with a high concentration of hits.*

Description

This function looks for chromosomal regions where there is a large accumulation of hits, e.g. significant peaks in a chip-seq experiment or differentially expressed genes in an rna-seq or microarray experiment. Regions are found by computing number of hits in a moving window and selecting regions based on a FDR cutoff.

Usage

```
enrichedChrRegions(hits1, hits2, chrLength, windowSize=10^4-1, fdr=0.05, nSims=10, mc.cores=1)
```

Arguments

hits1	Object containing hits (start, end and chromosome). Currently only RangedData objects are accepted.
hits2	Optionally, another object containing hits. If specified, regions will be defined by comparing hits1 vs hits2.
chrLength	Named vector indicating the length of each chromosome in base pairs
windowSize	Size of the window used to smooth the hit count (see details)
fdr	Desired FDR level (see details)
nSims	Number of simulations to be used to estimate the FDR
mc.cores	Number of processors to be used in parallel computations (passed on to mclapply)

Details

A smoothed number of hits is computed by counting the number of hits in a moving window of size windowSize. Notice that only the mid-point of each hit in hits1 (and hits2 if specified) is used. That is, hits are not treated as intervals but as being located at a single base pair.

If hits2 is missing, regions with large smoothed number of hits are selected. To assess statistical significance, we generate hits (also 1 base pair long) randomly distributed along the genome and compute the smoothed number of hits. The number of simulated hits is set equal to nrow(hits1). The process is repeated nSims times, resulting in several independent simulations. To estimate the FDR, several thresholds to define enriched chromosomal regions are considered. For each threshold, we count the number of regions above the threshold in the observed data and in the simulations. For each threshold t, the FDR is estimated as the average number of regions with score >=t in the simulations over the number of regions with score >=t in the observed data.

If hits2 is not missing, the difference in smoothed proportion of hits (i.e. the number of hits in the window divided by the overall number of hits) between the two groups is used as a test statistic. To assess statistical significance, we generate randomly scramble hits between sample 1 and sample 2 (maintaining the original number of hits in each sample), and we re-compute the test statistic. The FDR for a given threshold t is estimated as the number of bases in the simulated data with test statistic>t divided by number of bases in observed data with test statistic>t.

The lowest t with estimated FDR below fdr is used to define enriched chromosomal regions.

Value

Object of class RangedData containing regions with smoothed hit count above the specified FDR level.

Methods

signature(hits1 = "RangedData", hits2 = "missing") Look for chromosome zones with a large number of hits reported in hits1.

signature(hits1 = "RangedData", hits2 = "RangedData") Look for chromosomal zones with a different density of hits in hits1 vs hits2.

Examples

```
set.seed(1)
st <- round(rnorm(100,500,100))
st[st>10000] <- 10000
```

```
strand <- rep(c('+','-'),each=50)
space <- rep('chr1',length(st))
hits1 <- RangedData(IRanges(st,st+38),strand=strand,space=space)
chrLength <- c(chr1=10000)
enrichedChrRegions(hits1,chrLength=chrLength, windowSize=99, nSims=1)
```

---

enrichedPeaks *Find peaks in sequencing experiments.*

Description

Find peaks in significantly enriched regions found via enrichedRegions.

Usage

```
enrichedPeaks(regions, sample1, sample2, minHeight=100, space, mc.cores=1)
```

Arguments

regions	RangedDataList or RangedData indicating the regions in which we wish to find peaks.
sample1	IRangesList or IRanges object containing start and end of sequences in sample 1.
sample2	Same for sample 2. May be left missing, in which case only sample1 is used to find peaks.
minHeight	If sample2 is missing, peaks are defined as regions where the coverage in sample1 is greater or equal than minHeight. If sample2 is specified, the difference of coverage in sample1 minus sample2 must be greater or equal than minHeight.
space	Character text giving the name of the space for the RangedData object. Only used if sample1 and sample2 are of class RangedData, for RangedDataList this is set up automatically.
mc.cores	If mc.cores>1 computations for each element in the IRangesList objects are performed in parallel (using the parallel function from package multicore). Notice: this option launches as many parallel processes as there are elements in x, which can place strong demands on the processor and memory.

Value

Object of class RangedData indicating peaks higher than minHeight. Only peaks overlapping with regions are reported. The maximum of the coverage in each selected peak is reported in the column height (coverage in sample1 - sample2 when sample2 is specified). The column region.pvalue returns the p-value associated to the region that the peak belongs to (i.e. it is inherited from regions). Therefore, notice that all peaks corresponding to a single region will present the same region.pvalue.

Methods

signature(regions = "RangedData", sample1 = "IRanges", sample2 = "IRanges") sample1 indicates the start/end of reads in sample 1, and similarly for sample2. Only the subset of regions indicated by the argument space will be used.

signature(regions = "RangedData", sample1 = "IRanges", sample2 = "missing") sample1 indicates the start/end of reads in sample 1, and similarly for sample2. Only the subset of regions indicated by the argument space will be used.

signature(regions = "RangedData", sample1 = "IRangesList", sample2 = "IRangesList") regions contains the regions of interest, sample1 and sample2 the reads in sample 1 and sample 2, respectively. names(sample1) and names(sample2) must correspond to the space names used in regions.

signature(regions = "RangedData", sample1 = "IRangesList", sample2 = "missing") regions contains the regions of interest, sample1 the reads in sample 1. names(sample1) must correspond to the space names used in regions.

signature(regions = "RangedData", sample1 = "RangedData", sample2 = "missing") space(sample1) indicates the chromosome, and start(sample1) and end(sample1) indicate the start/end of the reads in sample 1.

signature(regions = "RangedData", sample1 = "RangedData", sample2 = "RangedData") space(sample1) indicates the chromosome, and start(sample1) and end(sample1) indicate the start/end of the reads in sample 1. Similarly for sample2.

See Also

enrichedRegions

Examples

```
set.seed(1)
st <- round(rnorm(1000,500,100))
strand <- rep(c('+','-'),each=500)
space <- rep('chr1',length(st))
sample1 <- RangedData(IRanges(st,st+38),strand=strand,space=space)
st <- round(runif(1000,1,1000))
sample2 <- RangedData(IRanges(st,st+38),strand=strand,space=space)

#Find enriched regions and call peaks
mappedreads <- c(sample1=nrow(sample1),sample2=nrow(sample2))
regions <- enrichedRegions(sample1,sample2,mappedreads=mappedreads,minReads=50)
peaks <- enrichedPeaks(regions,sample1=sample1,sample2=sample2,minHeight=50)
peaks <- peaks[width(peaks)>10,]
peaks

#Compute coverage in peaks
cover <- coverage(sample1)
coverinpeaks <- regionsCoverage(chr=space(peaks),start=start(peaks),end=end(peaks),cover=cover)

#Evaluate coverage in regular grid and plot
#Can be helpful fo clustering of peak profiles
coveringrid <- gridCoverage(coverinpeaks)
coveringrid
plot(coveringrid)

#Standardize peak profiles dividing by max coverage
```

```
stdcoveringrid <- stdGrid(coveringrid, colname='maxCov')
stdcoveringrid
```

---

enrichedRegions *Find significantly enriched regions in sequencing experiments.*

Description

Find regions with a significant accumulation of reads in a sequencing experiment.

Usage

```
enrichedRegions(sample1, sample2, regions, minReads=10, mappedreads,
pvalFilter=0.05, exact=FALSE, p.adjust.method='none', twoTailed=FALSE,
mc.cores=1)
```

Arguments

sample1 Either start and end of sequences in sample 1 (IRangesList, RangedData or IRanges object), of RangedDataList with sequences for all samples (sample2 must be left missing in this case) .

sample2 Same for sample 2. Can be left missing.

regions If specified, the analysis is restricted to the regions indicated in regions. If not specified, the regions are automatically defined using the argument minReads.

minReads This argument is only used when regions is not specified. The regions to be tested for enrichment are those with coverage greater or equal than minReads. If sample1 is a RangedDataList, the overall coverage adding all samples is used. Otherwise, if twoTailed is FALSE, only the reads in sample 1 are counted. If twoTailed is TRUE, the sum of reads in samples 1 and 2 are counted.

mappedreads Number of mapped reads for the sample. Has to be of class integer. Will be used to compute RPKM.

pvalFilter Only regions with P-value below pvalFilter are reported as being enriched.

exact If set to TRUE, an exact test is used whenever some expected cell counts are 5 or less (chi-square test based on permutations if sample1 is a RangedDataList object, Fisher's exact test otherwise), i.e. when the asymptotic chi-square/likelihood-ratio test calculations break down. Ignored if sample2 is missing, as in this case calculations are always exact.

p.adjust.method P-value adjustment method, passed on to p.adjust.

twoTailed If set to FALSE, only regions with a higher concentration of reads in sample 1 than in sample 2 are reported. If set to TRUE, regions with higher concentration of sample 2 reads are also reported. Ignored if sample2 is missing.

mc.cores If mc.cores is greater than 1, computations are performed in parallel for each element in the IRangesList objects. Whenever possible the mclapply function is used, therefore exactly mc.cores are used. For some signatures mclapply cannot be used, in which case the parallel function from package multicore is used. Note: the latter option launches as many parallel processes as there are elements in x, which can place strong demands on the processor and memory.

Details

The calculations depend on whether sample2 is missing or not. Non-missing sample2 case. First, regions with coverage above minReads are selected. Second, the number of reads falling in the selected regions are computed for sample 1 and sample 2. Third, the counts are compared via a chi-square test (with Yates continuity correction), which takes into account the total number of sequences in each sample. Finally, statistically significant regions are selected and returned in RangedData or RangedDataList objects.

Missing sample2. First, regions with coverage above minReads are selected. Second, the number of reads in sample 1 falling in the selected regions is computed. Third, the proportion of reads in each region is tested for enrichment via a one-tailed Binomial exact test.

Value

Object of class RangedData indicating the significantly enriched regions, the number of reads in each sample for those regions, the fold changes (adjusted considering the overall number of sequences in each sample) and the chi-square test P-values.

Methods

signature(sample1 = "missing", sample2 = "missing", regions = "RangedData") ranges(regions) indicates the chromosome, start and end of genomic regions, while values{regions} should indicate the observed number of reads for each group in each region. enrichedRegions tests the null hypothesis that the proportion of reads in the region is equal across all groups via a likelihood-ratio test (or permutation-based chi-square for regions where the expected counts are below 5 for some group).

signature(sample1 = "RangedDataList", sample2 = "missing", regions = "missing") Each element in sample1 contains the read start/end of an individual sample. enrichedRegions identifies regions with high concentration of reads (across all samples) and then compares the counts across groups using a likelihood-ratio test (or permutation-based chi-square for regions where the expected counts are below 5 for some group).

signature(sample1 = "RangedData", sample2 = "RangedData", regions = "missing") space(sample1) indicates the chromosome, start(sample1) and end(sample1) the start/end position of the reads. Similarly for sample2. enrichedRegions identifies regions with high concentration of reads (across all samples) and then compares the counts across groups using a likelihood-ratio test (or permutation-based chi-square for regions where the expected counts are below 5 for some group).

signature(sample1 = "RangedData", sample2 = "missing", regions = "missing") space(sample1) indicates the chromosome, start(sample1) and end(sample1) the start/end position of the reads. enrichedRegions tests the null hypothesis that an unusually high proportion of reads has been observed in the region using an exact binomial test.

Examples

```
set.seed(1)
st <- round(rnorm(1000,500,100))
strand <- rep(c('+','-'),each=500)
space <- rep('chr1',length(st))
sample1 <- RangedData(IRanges(st,st+38),strand=strand,space=space)
st <- round(rnorm(1000,1000,100))
sample2 <- RangedData(IRanges(st,st+38),strand=strand,space=space)
enrichedRegions(sample1,sample2,twoTailed=TRUE)
```

extendRanges — *Extend reads or sequences by a user-specified number of bases.*

Description

This function allows to extend ranges up to a user-specified length, which can be helpful in ChIP-seq analysis.

Usage

```
extendRanges(x, seqLen = 200, chrlength, mc.cores=1)
```

Arguments

x	Object containing reads.
seqLen	Desired sequence length after extension.
chrlength	Integer vector indicating the length of each chromosome. names(chrlength) must match those in x. This argument is used to ensure that no reads are extended beyond the maximum chromosome length.
mc.cores	Number of cores to use in parallel computations (passed on to mclapply).

Value

A list of IRanges objects with extended sequence length.

Methods

signature(x = "RangedData") space(x) indicates the chromosome, start(x) and end(x) the start/end positions of each read.

signature(x = "RangedDataList") Each element in x is assumed to correspond to a different sample.

Author(s)

David Rossell

Examples

```
set.seed(1)
st <- round(rnorm(1000,500,100))
st[st>2000] <- 2000
strand <- rep(c(’+’,’-’),each=500)
space <- rep(’chr1’,length(st))
sample1 <- RangedData(IRanges(st,st+38),strand=strand,space=space)
extendRanges(sample1, seqLen=200, chrlength=c(chr1=2000))
```

fdrEnrichedCounts *Posterior probability that a certain number of repeats are higher than expected by chance.*

Description

Given a vector of number of repeats (e.g. there are 100 sequences appearing once, 50 sequences appearing twice etc.) the function computes the false discovery rate that each number of repeats is unusually high.

Usage

```
fdrEnrichedCounts(counts,use=1:10,components=0,mc.cores=1)
```

Arguments

counts	vector with observed frequencies. The vector must have names. tabDuplReads function can be used for this purpose.
use	number of repeats to be used when estimating the null distribution. The number of repeats expected if no unusually high repeats are present. The first 10 are used by default.
components	number of negative binomials that will be used to fit the null distribution. The default value is 1. This value has to be between 0 and 4. If 0 is given the optimal number of negative biomials is chosen using the Bayesian information criterion (BIC)
mc.cores	number of cores to be used to compute calculations. This parameter will be passed bt to mclappply

Details

The null distribution is a combination of n negative binomials where. n is assigned through the components parameter. If components is equal to 0 the optimal number of negative binomials is choosen using the Bayesian information criterion (BIC). The parameters of the null distribution are estimated from the number of observations with as many repeats as told in the use parameter. If use is 1:10 the null distribution will be estimated using repeats that appear 1 time, 2 times, ... or 10 times.

False discovery rate for usually high number of repeats is done following an empirical Bayes scheme similar to that in Efron et al. Let f0(x) be the null distribution, f(x) be the overall distribution and (1-pi0) the proportion of unusually high repeats. We assume the two component mixture f(x)= pi0 f0(x) + (1-pi0)f1(x). Essentially, f(x) is estimated from the data (imposing that f(x) must be monotone decreasing after its mode using isoreg from packabe base, to improve the estimate in the tails). Currently pi0 is set to 1, i.e. its maximum possible value, which provides an upper bound for the FDR. The estimated false discovery rate for enrichment is 1-pi0*(1-cumsum(f0(x)))/(1-cumsum(f(x))). A monotone regression (isoreg) is applied to remove small random fluctuations in the estimated FDR and to guarantee that it decreases with x.

Value

data.frame with the following columns:

pdfH0	vector with pdf under the null hypothesis of no enrichment
pdfOverall	vector with pdf for mixture distribution
fdrEnriched	vector with false discovery rate that each count is significantly enriched

References

Ji et al. An integrated software system for analyzing ChIP-chip and ChIP-seq data. Nature Biotechnology, 2008, 26, 1293-1300.

Efron et al. Empirical Bayes Analysis of a Microarray Experiment, Journal of the American Statistical Association, 2001, 96, 1151-1160.

Examples

```
#Generate 1000 sequences repeated once, on the average
nrepeats <- c(rpois(10^4,1),rpois(10,10))
nrepeats <- nrepeats[nrepeats>0]
counts <- table(nrepeats)
barplot(counts) -> xaxis #observe bimodality around 10
fdrest <- fdrEnrichedCounts(counts,use=1:5,components=1)
cutoff <- xaxis[which(fdrest$fdrEnriched<0.95)[1]]
abline(v=cutoff,col=2)
text(cutoff,counts[1]/2,'cut-off',col=2)
head(fdrest)
```

---

filterDuplReads *Detect and filter duplicated reads/sequences.*

Description

filterDuplReads filters highly repeated sequences, i.e. with the same chromosome, start and end positions. As many such sequences are likely due to over-amplification artifacts, this can be a useful pre-processing step for ultra high-throughput sequencing data. A false discovery rate is computed for each number of repeats being unusually high. The reads with a higher false discovery rate will be removed. For more information on the false discovery rate calculation please read the fdrEnrichment manual.

tabDuplReads counts the number reads with no duplications, duplicated once, twice etc.

Usage

```
filterDuplReads(x, maxRepeats, fdrOverAmp=0.01, negBinomUse=.999,components=0, mc.cores=1)

tabDuplReads(x,  minRepeats=1, mc.cores=1)
```

Arguments

x Object containing read locations. Currently methods for RangedData and RangedDataList. Duplication is assessed based only on the space, start, end and x[['strand']], i.e. even if they are different based on other variables stored in values(x), the reads are considered duplicated and only the first appearance is returned.

maxRepeats Reads appearing maxRepeats or more times will be excluded. If not specified, this is setup automatically based on fdrOverAmp.

fdrOverAmp Reads with false discovery rate of being over-amplified greater than fdrOverAmp are excluded.

negBinomUse Number of counts that will be used to compute the null distribution. Using 1 - 1/1000 would mean that 99.9% of the reads will be used. The ones with higher number of repetitions are the excluded ones.

components number of negative binomials that will be used to fit null distribution. The default value is 1. This value hase to be between 0 and 4. If 0 is given the optimal number of negative biomials is choosen using the Bayesian information criterion (BIC)

mc.cores Number of cores to be used in parallel computing (passed on to mclapply).

minRepeats The table is only produced for reads with at least minRepeats repeats.

Value

filterDuplReads returns x without highly repetitive sequencesas, determined by maxRepeats or ppOverAmp.

tabDuplReads returns a table counting the number of sequences repeating 1 times, 2 times, 3 times etc.

Methods

Methods for filterDuplReads and tabDuplReads

signature(x = "RangedData") Two reads are duplicated if they have the same space, start and end position.

signature(x = "RangedDataList") The method is applied separately to each RangedData element in the list.

Author(s)

Evarist Planet, David Rossell, Oscar Flores

See Also

fdrEnrichedCounts to compute the posterior probability that a certain number of repeats is due to over-amplification.

Examples

```
set.seed(1)
st <- round(rnorm(1000,500,100))
strand <- rep(c('+','-'),each=500)
space <- sample(c('chr1','chr2'),size=length(st),replace=TRUE)
sample1 <- RangedData(IRanges(st,st+38),strand=strand,space=space)
```

```
#Add artificial repeats
st <- rep(400,20)
repeats <- RangedData(IRanges(st,st+38),strand='+',space='chr1')
sample1 <- rbind(sample1,repeats)

filterDuplReads(sample1)
```

giniCoverage *Compute Gini coefficient.*

Description

Calculate Gini coefficient of High-throughput Sequencing aligned reads. The index provides a measure of "inequality" in read coverage which can be used for quality control purposes (see details).

Usage

```
giniCoverage(sample, mc.cores = 1, mk.plot = FALSE, seqName = "missing", species="missing", chrLengths="
```

Arguments

sample A RangedData or RangedDataList object

seqName If sample is a RangedData, name of sequence to use in plots

mk.plot Logical. If TRUE, logarithm of coverage values' histogram and Lorenz Curve plot are plotted.

mc.cores If mc.cores is greater than 1, computations are performed in parallel for each element in the IRangesList object.

chrLengths An integer array with lengths of chromosomes in sample for simluations of uniformily distributed reads.

species A BSgenome species to obtain chromosome lengths for simluations of uniformily distributed reads.

numSim Number of simulations to perform in order to find the expected Gini coefficient.

Details

The Gini coefficient provides a measure of "inequality" in read coverage. This can be used in any sequencing experiment where the goal is to find peaks, i.e. unusual accumulation of reads in some genomic regions. For instance, Chip-Seq etc. Typically these experiments will consist of samples of interest (e.g. immuno-precipitated) and controls. The samples of interest should exhibit higher peaks, whereas reads in the controls should show a more uniform distribution. Since the Gini coefficient can be seen as a measure of departure from uniformity, the coefficient should present smaller values in the control samples. Since the Gini coefficient depends on the number of reads per sample, a correction is performed by substracting the Gini index from a sample with uniformily distributed reads.

Value

If mk.plot==FALSE, the Gini index and adjusted Gini index for each element in the RangedDataList or RangedData object.

If mk.plot==TRUE, a plot is produced showing the logarithm of coverage values' histogram and Lorenz Curve plot.

Methods

signature(sample = "RangedData", mc.cores = "ANY", mk.plot = "ANY", seqName = "ANY", species = "c
: Analize a single RangeData object with 'chrLengths' used for simulations ('Species' is ignored).

signature(sample = "RangedData", mc.cores = "ANY", mk.plot = "ANY", seqName = "ANY", species = "c
: Analize a single RangeData object with chromosome lengths for simulations taken from BSgenome 'species' (package must be installed).

signature(sample = "RangedData", mc.cores = "ANY", mk.plot = "ANY", seqName = "ANY", species = "
: Analize a single RangeData object with 'chrLengths' used as chromosome lengths in simulations.

signature(sample = "RangedData", mc.cores = "ANY", mk.plot = "ANY", seqName = "ANY", species = "
: Analize all RangeData objects from sample (RangedDataList) with hromosome lengths for simulations taken as the largest end position of reads in each chromosome of all samples.

signature(sample = "RangedDataList", mc.cores = "ANY", mk.plot = "ANY", seqName = "ANY", species
: Analize all RangeData objects from sample (RangedDataList) with 'chrLengths' used as chromosome lengths in simulations ('Species' is ignored).

signature(sample = "RangedDataList", mc.cores = "ANY", mk.plot = "ANY", seqName = "ANY", species
: Analize all RangeData objects from sample (RangedDataList) with chromosome lengths for simulations taken from BSgenome 'species' (package must be installed).

signature(sample = "RangedDataList", mc.cores = "ANY", mk.plot = "ANY", seqName = "ANY", species
: Analize all RangeData objects from sample (RangedDataList) with 'chrLengths' used as chromosome lengths in simulations.

signature(sample = "RangedDataList", mc.cores = "ANY", mk.plot = "ANY", seqName = "ANY", species
: Analize all RangeData objects from sample (RangedDataList) with chromosome lengths for simulations taken as the largest end position of reads in each chromosome of sample.

Author(s)

Camille Stephan-Otto

References

See the definition of the Gini coefficient and Lorenz curve at http://en.wikipedia.org/wiki/Gini_coefficient

See Also

ssdCoverage for another measure of inequality in coverage.

Examples

```
set.seed(1)
peak1 <- round(rnorm(500,100,10))
peak1 <- RangedData(IRanges(peak1,peak1+38),space='chr1')
peak2 <- round(rnorm(500,200,10))
peak2 <- RangedData(IRanges(peak2,peak2+38),space='chr1')
ip <- rbind(peak1,peak2)
bg <- runif(1000,1,300)
bg <- RangedData(IRanges(bg,bg+38),space='chr1')

rdl <- RangedDataList(ip,bg)
ssdCoverage(rdl)
giniCoverage(rdl)
```

gridCover-class *Class "gridCover"*

Description

Objects of class gridCover store coverage information evaluated on a grid on pre-specified genomic regions.

Objects from the Class

Objects of this class are returned by call to the function gridCoverage.

Slots

cover**:** Object of class "matrix" with one row for each genomic region of interest, and 500 columns. Columns 1-100 contain the coverage in the promoter region (as specified in argument promoterDistance to gridCoverage). Columns 101-500 contain the coverage between start and end as indicated to promoterDistance.

viewsInfo**:** Object of class "DataFrame" with information relative to each region (strand, mean and maximum coverage).

Methods

[signature(x = "gridCover", i = "ANY", j = "ANY"): Select a subset of peaks.

plot signature(x = "gridCover", y = "ANY"): Plot the coverage.

lines signature(x = "gridCover"): Add lines to an existing plot.

show signature(object = "gridCover"): Show method.

stdGrid signature(x = "gridCover"): Standardize the coverage by dividing by either the mean or the maximum coverage in each region.

getViewsinfo signature(x = "gridCover"): Accessor for the viewsInfo slot.

getCover signature(x = "gridCover"): Accessor for the cover slot.

Author(s)

David Rossell

See Also

regionsCoverage to compute coverage on pre-specified regions, gridCoverage to compute coverage on a grid.

Examples

```
##See help(gridCoverage)
```

htSample — *Example ChIP-sequencing data with 2 replicates per group obtained in two different dates.*

Description

This RangedDataList contains a subset of drosophila melanogaster ChIP-sequencing data obtained with the Illumina sequencer. An immuno-precipitated and a control input sample were obtained at two experimental dates (details not provided as this is still unpublished data). In order to save space and let the examples run quicker, only reads mapping to the first 500kb of chr2L are included.

Usage

```
data(htSample)
```

Format

RangedDataList where each element contains reads from a different sample. names(htSample) indicate the group and batch (experimental date) that each sample corresponds to.

Details

Data was pre-processed using the Illumina pipeline and mapped to the drosophila melanogaster dm3 genome using Bowtie. Only uniquely mapping sequences with at most 2 mismatches in the first 28 bases were kept. See the package vignette for some more details on this dataset.

Examples

```
data(htSample)
htSample
```

islandCounts — *Find genomic regions with high coverage and count number of reads overlapping each region in each sample*

Description

Finds genomics regions where the coverage is above a user-specified threshold and counts the number of ranges in each sample overlapping each region.

Usage

```
islandCounts(x, minReads=10, mc.cores=1)
```

Arguments

x	RangedData or RangedDataList containing the reads. If a RangedDataList is provided, the overall coverage across all its elements is used to find the regions of interest, but individual counts are computed for each element in the list.
minReads	Only regions with coverage above minReads are considered.
mc.cores	If mc.cores>1 computations are performed in parallel, using function mclapply from package multicore.

Details

The output of islandCounts can be the input data for a number of downstream analysis methods. Although for a simple-minded analysis one could use enrichedRegions, one will usually want to use more sofisticated analyses (e.g. from packages DEseq, BayesPeak, limma etc.)

Value

Object of class RangedData indicating the regions with coverage above minReads and the number of reads overlapping each sample for those regions.

Methods

signature(x = "RangedData") x is assumed to contain the reads from a single sample. Genomic regions with high coverage will be detected and the number of reads overlapping these regions will be computed.

signature(x = "RangedDataList") x is assumed to contain the reads for several samples, one sample in each element of the list. The overall coverage across all samples is computed by adding the coverage in the individual samples, and the regions with overall coverage above the user-specified threshold are selected. Then the number of reads overlapping each region is computed.

Examples

```
set.seed(1)
st <- round(rnorm(1000,500,100))
strand <- rep(c('+','-'),each=500)
space <- rep('chr1',length(st))
sample1 <- RangedData(IRanges(st,st+38),strand=strand,space=space)
st <- round(rnorm(1000,1000,100))
sample2 <- RangedData(IRanges(st,st+38),strand=strand,space=space)

regions <- islandCounts(RangedDataList(sample1,sample2),minReads=50)
regions

#Plot coverage
plot(coverage(sample1)[[1]],type='l',xlim=c(0,2000))
lines(coverage(sample2)[[1]],col=2)
```

---

listOverlap *Assess the overlap between two or three lists.*

Description

Assess the overlap between two or three lists, e.g. ChIP-Seq peaks vs. genes selected from a microarray, or peaks obtained in different experiments.

Usage

```
listOverlap(list1, list2, list3, univ, ...)
```

Arguments

list1 Vector with elements in the first list. This can either be a character vector indicating the element names, or a named factor vector indicating some classification for the elements in the first list.

list2 Vector with elements in the second list. This should be a character vector indicating the element names.

list3 Vector with elements in the third list. This should be a character vector indicating the element names. The overlap assesment method used depends on whether this argument is specified or not. See details.

univ character vector indicating the universe of all elements from which list1 and list2 were obtained. The overlap assessment depends on whether this argument is specified or not. See details.

... Further arguments to be passed on to chisq.test in 2 list overlapping.

Details

For signature(list1='character', list2='character', list3='missing', univ='character') the overlap is assessed with respect to the universe of all possible elements univ. That is, we count the number of elements that are common to list1 and list2, those appearing only in either list1 or list2, and those not appearing in either (but appearing in univ). A typical example: list1 contains names of genes with a peak in ChIP-Seq experiment 1, list2 names of genes with a peak in ChIP-Seq experiment 2, and univ the names of all genes in the organism.

For signature(list1='character', list2='character', list='character', univ='character') the overlap is assessed by fitting and anova comparison of linear models. This is done to test whether 3-way overlap is significant with respect to the universe of all possible elements univ when compared to a model considering just the combination of 2-way overlapping. A typical example: list1, list2 and list3 contain names of genes with peaks in three different ChIP-Seq experiments, and univ the names of all genes in the organism.

For signature(list1='factor', list2='character', univ='missing') the distribution of list1 is compared between elements appearing and not appearing in list2. A typical example: list1 indicates the differential expression status for a number of genes, and list2 contains the names of the genes which had a peak in a ChIP-Seq experiment.

Value

For comparison of 2 lists, an htest object from a chi-square test that evaluates if the two lists are statistically independent from each other. This is a named list: the observed overlap is stored in observed and the P-value in p.value.

For 3 list comparison, a list object containing the occurrence and frequency tables (xtab, ftable), the fitted linear models (glm1, glm2), and the anova P-value (pvalue).

Methods

signature(list1 = "character", list2 = "character", list3 = "character", univ = "character") Studies 3-way associations.

signature(list1 = "character", list2 = "character", list3 = "missing", univ = "character") Studies bivariate associations.

signature(list1 = "factor", list2 = "character", list3 = "missing", univ = "missing") Studies bivariate associations.

Examples

```
#Overlap between diff expression and chip-seq peaks
deStatus <- factor(c(0,0,0,0,1,1,1))
names(deStatus) <- paste('Gene',1:7)
peaks <- c('Gene 6','Gene 7')
ans <- listOverlap(list1=deStatus,list2=peaks)
ans$observed
ans$p.value

#Overlap between peaks obtained from two different experiments
peaks2 <- c('Gene 1','Gene 2','Gene 7')
univ <- paste('Gene',1:7)
ans <- listOverlap(list1=peaks,list2=peaks2,univ=univ)
ans$observed
ans$p.value
```

---

mergeRegions *Merge nearby chromosomal regions.*

Description

Merges regions that are less than maxDist bases apart.

Usage

```
mergeRegions(intervals, chromosome, score, annot, aggregateFUN='median', maxDist=300)
```

Arguments

intervals	Object indicating start and end of each region. It can either be a matrix, data.frame, IRanges, RangedData or an RleViews object. If a matrix or data.frame, it must have columns named start and end.
chromosome	Chromosome that the region belongs to (optional). If supplied, must be of the same length as start and end.
score	Numerical score for each interval. Scores in merged intervals are aggregated using function aggregateFUN. If intervals is of class RangedData, this should be a character vector of length 1 indicating the name of the variable in values(x) containing the score.
annot	Character indicating annotation information for each interval. Annotations in merged intervals are pasted in a single string (annotations appearing in more than one interval are only reported once in the merged interval).
aggregateFUN	Function to aggregate score.
maxDist	Regions less than maxDist apart are merged into a single region

Value

The result is returned in a data.frame indicating the start and end of each merged interval. If the arguments were provided, the information in chromosome, score and annot is provided in additional columns. If the input argument intervals was of class RangedData, the results are returned in a RangedData object.

Methods

signature(intervals = "data.frame") intervals$start and intervals$end give the interval start/end positions.

signature(intervals = "IRanges") start(intervals) and end(intervals) give the interval start/end positions.

signature(intervals = "matrix") The columns start and end in intervals give the interval start/end positions

signature(intervals = "RangedData") start(intervals) and end(intervals) give the interval start/end positions.

signature(intervals = "RleViews") start(intervals) and end(intervals) give the interval start/end positions.

Author(s)

David Rossell

Examples

```
st <- c(10,20,1000)
intervals <- RangedData(IRanges(st,st+10),space='chr1')

intervals
mergeRegions(intervals,maxDist=300)
```

---

plot-methods *Methods for Function plot in Package 'htSeqTools'*

Description

Methods for function plot in Package 'htSeqTools'

Methods

signature(x = "cmdsFit") Produces a Multi-Dimensional scaling plot. See cmds for details.

signature(x = "gridCover") Plots the average coverage for each point in the grid. See gridCover for details.

Examples

```
### Not run
#d <- matrix(c(0,5,10,5,0,15,10,15,0),byrow=TRUE,ncol=3)
#rownames(d) <- colnames(d) <- letters[1:3]
#fit1 <- cmdsFit(d,add=TRUE)
#plot(fit1)
```

plotChrRegions *Plot chromosomal regions of interest*

Description

Produces a plot with all chromosomes for a given organism, marking regions of interest in a user-specified color.

Usage

```
plotChrRegions(regions, chrLength, markColor='red', ...)
```

Arguments

regions	RangedData object with chromosome, start and end positions (chromosome must be stored in space(regions).
chrLength	Named integer vector with chromosome lengths in base pairs.
markColor	Color to be used to mark the regions in the chromosome.
...	Further parameters passed on to plot.

Value

This function produces a plot.

Examples

```
set.seed(1)
chr <- rep(c('chr1','chr2'),each=10)
chrLength <- c(chr1=10000,chr2=5000)
st <- c(runif(10,1,10000),runif(10,1,5000))
regions <- RangedData(IRanges(st,st+50),space=chr)

plotChrRegions(regions,chrLength=chrLength)
```

regionsCoverage *Compute coverage on user specified genomic regions.*

Description

regionsCoverage computes coverage for user specified genomic regions.

gridCoverage evaluates the coverage on a regular grid with the same number of points for each region (facilitating further plotting, clustering etc).

stdGrid standardized the coverage by diviging by the average or maximum coverage at each region.

Usage

```
regionsCoverage(chr, start, end, cover)

gridCoverage(cover)

stdGrid(cover, colname="maxCov")
```

Arguments

chr — Vector with chromosome names.

start — Vector with start position. start>end indicates that region is on the negative strand.

end — Vector with end position. start>end indicates that region is on the negative strand.

cover — For regionsCoverage, cover is an object of class RleList with the genome-wide coverage (typically obtained by a previous call to coverage). For gridCoverage this is the coverage evaluated at user-specified regions, as returned by regionsCoverage. For stdGrid this is the coverage evaluated on a grid, as returned by gridCoverage.

colname — Name of the column in cover@viewsInfo to be used for the standardizing. Currently only "meanCov" and "maxCov" are implemented.

Value

regionsCoverage returns a list with two components

views — RleViewsList with coverage evaluated at specified regions. Orientation is always so that start<end, i.e. For most practical purposes, regions on the reverse strand will need to be inverted.

viewsInfo — SplitDataFrameList containing information about each peak (chromosome, strand, mean and maximum coverage).

gridCoverage and stdGrid return an object of class gridCover. The slot cover is a matrix with the coverage evaluated on a grid of 500 equi-spaced points, whereas the slot viewsInfo is the same as that returned by regionsCoverage (see above). For regions between 100bp and 500bp long, a linear interpolation is used to evaluate the coverage on the 500 points grid. For regions less than 100bp long, NAs are returned.

Methods

Methods for regionsCoverage:

signature(chr = "ANY", start = "ANY", end = "ANY", cover = "RleList") Evaluates the coverage cover at the genomic positions specified by chr, start, end.

Methods for stdGrid:

signature(cover = "gridCover") Standardizes the coverage evaluated on a grid (typically, as returned by gridCoverage) by dividing by the mean or maximum coverage.

See Also

gridCover-class

Examples

```
#See help(enrichedPeaks)
```

rowLogRegLRT *Row-wise logistic regression*

Description

Row-wise logistic regressions are applied to a matrix with counts. For each row, an overall test comparing the column counts across columns is performed. Optionally, chi-square permutation tests are used when the expected counts are below 5 for some column.

Usage

```
rowLogRegLRT(counts, exact = TRUE, p.adjust.method = "none")
```

Arguments

counts	Matrix with counts
exact	If set to TRUE, an exact test is used whenever some expected cell counts are 5 or less
p.adjust.method	p-value adjustment method, passed on to p.adjust

Details

For each column, the proportion of counts in each row (with respect to the overall counts in that column) is computed. Then a statistical comparison of these proportions across groups is performed via a likelihood-ratio test (if exact==TRUE a permutation based chi-square test is used whenever the expected counts in some column is below 5).

Notice that data from column j can be viewed as a multinomial distribution with probabilities pj, where pj is a vector of length nrow(x). rowLogRegLRT tests the null hypothesis p1[i]=...pc[i] for i=1...nrow(x), where c is ncol(x). This actually ignores the multinomial sampling model and focuses on its binomial margins, which is a reasonable approximation when the number nrow(x) is large and substantially improves computation speed.

Examples

```
#The first two rows present different counts across columns
#The last two columns do not
x <- matrix(c(70,10,10,10,35,35,10,10),ncol=2)
x
rowLogRegLRT(x)
```

ssdCoverage *Standardized SD of the genomic coverage*

Description

Compute variability of the genomic coverage, measured as standardized SD per thousand sequences (see details). For instance, this can measure how pronounced are the peaks in a ChIP-Seq experiments, which can serve as a quality control to detect inefficient immuno-precipitation.

Usage

```
ssdCoverage(x, mc.cores=1)
```

Arguments

x Object with ranges indicating the start and end of each read. Currently, x can be of class RangedDataList, RangedData and IRangesList.

mc.cores Set mc.cores to a value greater than 1 to perform computations in parallel, using package mclapply.

Details

ssdCoverage first computes the coverage for each sample and computes the standard deviation (SD) of the coverage. However, SD is not an appropriate measure of coverage unevenness, as its expected value is proportional to sqrt(n), where n is the number of reads (this can be seen with simple algebra).

ssdCoverage therefore reports 1000*SD/sqrt(n), which can be interpreted as the standardized SD per thousand sequences.

Value

Numeric vector with coefficients of variation.

Methods

signature(x = "IRangesList") A single coefficient of variation is returned, as a weighted average of the coefficients of variation for each chromosome (weighted according to the chromosome length).

signature(x = "RangedData") The method for IRangesList is used on ranges(x).

signature(x = "RangedDataList") A vector with coefficients of variation for each element in x are returned, by repeatedly calling the method for RangedData objects. Use mc.cores to speed up computations with mclapply, but be careful as this requires more memory.

Examples

```
set.seed(1)
#Simulate IP data
peak1 <- round(rnorm(500,100,10))
peak1 <- RangedData(IRanges(peak1,peak1+38),space='chr1')
peak2 <- round(rnorm(500,200,10))
peak2 <- RangedData(IRanges(peak2,peak2+38),space='chr1')
ip <- rbind(peak1,peak2)

#Generate uniform background
bg <- runif(1000,1,300)
bg <- RangedData(IRanges(bg,bg+38),space='chr1')

rdl <- RangedDataList(ip,bg)
ssdCoverage(rdl)
giniCoverage(rdl)
```

stdPeakLocation *Peak density with respect to closest gene.*

Description

stdPeakLocation plots the density of peaks with respect to the genomic feature (e.g. gene) in standardized gene coordinates so that genes with different lengths are comparable.

PeakLocation produces the same plot in non-standardized coordinates (i.e. distances are measured in base pairs).

plotMeanCoverage plots the mean coverage in a series of selected genomic regions.

Usage

```
stdPeakLocation(x, peakDistance=1000, startpos='start_position', endpos='end_position',
strand='strand', distance, main='', xlab='Distance relative to feature length', xaxt='n',
  xlim=c(-1,2), densityType="kernel", nbreaks=10, ...)

PeakLocation(x, peakDistance=1000, startpos='start_position', endpos='end_position',
strand='strand', distance, main='', xlab='Distance (bp)',
  densityType="kernel", breaks, ...)

plotMeanCoverage(cover, x, upstreambp=1000, downstreambp=5000,
startpos='start_position', endpos='end_position', normalize=FALSE,
smooth=FALSE, span=0.05, main='', xlab='(bp)', ylab='Average coverage', ...)
```

Arguments

x — A RangedData or data.frame indicating peak start and end in start and end, and start and end of the closest genomic feature (e.g. gene) in startpos and endpos.

cover — An RleList object containing the coverage, as returned by the function coverage.

peakDistance — Peaks more than peakDistance bases upstream or more than 3*peakDistance downstream of the closest feature are discarded.

startpos — Name of the variable storing the start position of the closest genomic feature.

endpos — Name of the variable storing the end position of the closest genomic feature.

strand — Name of the variable storing the strand for the closest genomic feature.

distance — Name of the variable indicating the distance between the peak and the closest genomic feature. If left missing the distance between the feature start and the mid-point of the peak is computed.

main — Graphical parameter passed on to plot.

xlab — Graphical parameter passed on to plot.

ylab — Graphical parameter passed on to plot.

xaxt — Graphical parameter passed on to plot.

xlim — In stdPeakLocation the x-axis limit is set to xlim*peakDistance.

densityType — If we eant a density plot or a histogram. Has to be one of "kernel" (for the density plot) or "hist" for the histogram.

nbreaks	Number of breaks to be used. It will not be used if densityType is different from "hist".
breaks	This parameter will be passed to the hist plotting function. It will not be used if densityType is different from "hist".
upstreambp	Number of bp upstream of the TSS where the coverage should be computed
downstreambp	Number of bp downstream of the TSS where the coverage should be computed
normalize	When set to TRUE the average coverage in each position is divided by the average across all positions. This is useful when trying to super-impose data from several experiments that had different read coverage.
smooth	If set to TRUE, the average coverage is smooth by calling loess.
span	Parameter controlling smoothing, passed on to loess. Larger values indicate more smoothing.
...	Further parameters passed on to plot.

Value

This function produces a density plot.

Methods

Methods for stdPeakLocation, PeakLocation

signature(x = "data.frame") The data frame should contain columns named start and end indicating the peak location, txStart, txEnd indicating transcription start/end of the closest gene and strand indicating the strand.

signature(x = "RangedData") start(x) and end(x) indicate the peak location. x should contain variables x[['txStart']], x[['txEnd']] indicating the transcription start/end of the closest gene and x[['strand']] indicating the strand.

Methods for plotMeanCoverage

signature(cover="RleList", x="RangedData") cover contains the coverage and x the genomic regions of interest.

Examples

```
#Generate synthetic peaks
set.seed(1)
st <- runif(100,1,1000)
en <- st+runif(length(st),25,100)
peaks <- RangedData(IRanges(st,en),space='chr1')

#Assign distance to closest gene
#(typically one would call annotatePeakInBatch
#from package ChIPpeakAnno to do this)
peaks[['start_position']] <- start(peaks) + runif(nrow(peaks),-500,1000)
peaks[['end_position']] <- peaks[['start_position']] + 500
peaks[['distance']] <- peaks[['start_position']] - start(peaks)
peaks[['strand']] <- sample(c('+','-'),nrow(peaks),replace=TRUE)
PeakLocation(peaks,peakDistance=1000)
```

Index