chipseq

April 19, 2010

`combineLanes` *Combine or subsample short read alignment locations*

Description

Combines or subsamples data from multiple lanes on a per-chromosome basis.

Usage

```
combineLanes(x, chromList, keep.unique = FALSE)
laneSubsample(lane1, lane2, fudge = 0.05)
```

Arguments

`x`	Typically a `"GenomeDataList"` object representing multiple lanes of aligned locations or ranges. The result will combine the locations across lanes on a per-chromosome basis.
`chromList`	Character vector specifying Which chromosomes to combine. Defaults to all chromosomes in the first lane.
`keep.unique`	logical flag. If `TRUE`, only unique locations/reads will be retained.
`lane1, lane2`	Two lanes of data, each of class `"GenomeData"`.
`fudge`	A numeric fudge factor. For each chromosome, if the difference in the sizes relative to the size of the first dataset is less than `fudge`, no subsampling is done.

Value

`combineLanes` returns an object of class `"GenomeData"`.

`laneSubsample` returns a list similar to its input, but with the larger dataset subsampled to be similar to the smaller one.

Author(s)

D. Sarkar

Examples

```
data(cstest)
## subsample to compare lanes
laneSubsample(cstest[[1]], cstest[[2]])
## two lanes of chr10 become one
combineLanes(cstest, "chr10")
```

---

contextDistribution

Tabulate peak locations according to genomic context

Description

Given two sets of intervals defined on a genome, tabulates overlap of one set with the other. The first set typically represents "peak" locations, and the second represents types of genomic regions such as promoters, downstream regions, genes, etc.

Usage

```
contextDistribution(peaks, gregions, chroms, ...)
```

Arguments

`peaks`	A data frame with one row for each "peak"; the location of peaks must be defined by the columns `chromosome`, `start`, and `end`. Columns `up` and `down`, if present, must be logical, and should indicate peaks that were down or upregulated by some definition. If present, the result will include tabulations for the up and down subsets thus defined.
`gregions`	Locations of genomic regions of interest. Currently, this must be of the form produced by the function `transcripts`.
`chroms`	Which chromosomes to use. By default, all are used.
`...`	Further arguments, currently ignored.

Value

A data frame with overlap counts.

Author(s)

Deepayan Sarkar

Examples

```
data(cstest)
library(BSgenome.Mmusculus.UCSC.mm9)
mouse.chromlens <- seqlengths(Mmusculus)
## extend reads, generate peak summary
extRanges <- gdapply(cstest, extendReads, seqLen = 200)
peakSummary <-
    diffPeakSummary(extRanges$gfp, extRanges$ctcf,
                    chrom.lens = mouse.chromlens, lower = 10)
```

```
## generate transcripts using GenomicFeatures.Mmusculus.UCSC.mm9 package
library(GenomicFeatures.Mmusculus.UCSC.mm9)
gregions <- transcripts(genes = geneMouse(), proximal = 2000)
## finally, calculate context distribution for chr10
contextDistribution(peakSummary, gregions, "chr10")
```

`copyIRangesbyChr` *Associate ranges to coverage.*

Description

Associate a set of ranges, typically derived using an independent computation, to a coverage as produced by `coverage`. This then allows one to compute various summaries such as maximum coverage in each range. `copyIRangesbyChr` does this over lists of ranges and coverage objects.

Usage

```
copyIRanges(IR1, newX)
copyIRangesbyChr(IR1, newX)
```

Arguments

`IR1`	The set of ranges (an `"IRanges"` object) or a list of such objects (usually one for each chromosome of interest).
`newX`	An `"Rle"` object, usually the result of `link[IRanges:coverage]{coverage}`, or a list of such objects.

Value

A `"View"` object, or a list of such objects.

Author(s)

Deepayan Sarkar

Examples

```
cov <- Rle(c(1:10, seq(10, 1, -2), seq(1,5,2), 4:1), rep(1:2, 11))
peaks <- slice(cov, 3)
peaks.cov <- copyIRanges(peaks, cov)
```

coverageplot *Plot coverage on a small interval.*

Description

A function that plots one or two coverage vectors over a relatively small interval in the genome.

Usage

```
coverageplot(peaks1, peaks2 = NULL, i = 1,
             xlab = "Position", ylab = "Coverage",
             opposite = TRUE, ...)
```

Arguments

peaks1, peaks2	A set of peaks as described by ranges over a coverage vector.
i	Which peak to use.
xlab, ylab	Axis labels.
opposite	Logical specifying whether the two peaks should be plotted on opposite sides (appropriate for positive and negative strand peaks).
...	extra arguments.

Author(s)

Deepayan Sarkar

Examples

```
cov <- Rle(c(1:10, seq(10, 1, -2), seq(1,5,2), 4:1), rep(1:2, 11))
peaks <- slice(cov, 3)
peaks.cov <- copyIRanges(peaks, cov)
peaks.cov.rev <- rev(peaks.cov)
coverageplot(peaks.cov, peaks.cov.rev, ylab = "Example")
```

cstest *A test ChIP-Seq dataset*

Description

A small subset of a ChIP-Seq dataset downloaded from the Short-Read Archive.

Usage

```
data(cstest)
```

Format

The dataset is on object of class `GenomeDataList` with data from three chromosomes in two lanes representing CTCF and GFP pull-down in mouse.

The per-chromosome data is represented as a list of positive and negative strand alignment locations. The recorded locations represent the aligned position at the first cycle.

Source

Short Read Archive, GEO accession number GSM288351 http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM288351

References

Chen X., Xu H., Yuan P., Fang F., Huss M., Vega V.B., Wong E., Orlov Y.L., Zhang W., Jiang J., Loh Y.H., Yeo H.C., Yeo Z.X., Narang V., Govindarajan K.R., Leong B., Shahab A.S., Ruan Y., Bourque G., Sung W.K., Clarke N.D., Wei C.L., Ng H.H. (2008), "Integration of External Signaling Pathways with the Core Transcriptional Network in Embryonic Stem Cells". *Cell*, 133:1106-1117.

Examples

```
data(cstest)
names(cstest)
cstest$gfp
str(cstest$ctcf$chr10)
```

`diffPeakSummary` *A function to identify and produce summary statistics for differentially expressed peaks.*

Description

Given two sets of reads this function identifies all peaks in the combined data with height larger than `lower` and then uses those regions to compute summary statistics for each of the sets separately.

Usage

```
diffPeakSummary(ranges1, ranges2, chrom.lens,
                lower = 10, extend = 0,
                peak.fun = NULL, merge = 0L, islands = FALSE,
                viewSummary = list(sums = viewSums, maxs = viewMaxs))
```

Arguments

`ranges1`	First set of reads (as IRanges).
`ranges2`	Second set of reads (as IRanges).
`chrom.lens`	The lengths of the chromosomes for the organism.
`lower`	The height used to declare a peak in the combined samples.
`extend`	Currently unused. The intent is to extend peaks by this amount before summarizing.

peak.fun	Function use to use to find peaks. The default makes use of additional arguments merge and islands, which are otherwise ignored.
merge	Integer giving the amount of gaps between peaks that should be considered significant. Smaller gaps are removed by combining neighbouring peaks.
islands	Logical indicating whether or not to use islands (coverage > 0) as peaks.
viewSummary	A list of the per peak summaries.

Value

A data.frame with one row for each peak in the combined data. The chromosome, start and stop nucleotide positions (+ strand) are given as are the summary statistics requested.

Author(s)

D. Sarkar

Examples

```
data(cstest)
library(BSgenome.Mmusculus.UCSC.mm9)
mouse.chromlens <- seqlengths(Mmusculus)
## extend reads, generate peak summary
extRanges <- gdapply(cstest, extendReads, seqLen = 200)
peakSummary <-
    diffPeakSummary(extRanges$gfp, extRanges$ctcf,
                    chrom.lens = mouse.chromlens, lower = 10)
```

---

estimate.mean.fraglen

Estimate summaries of the distribution of fragment lengths in a short-read experiment. The methods are designed for ChIP-Seq experiments and may not work well in data without peaks.

Description

estimate.mean.fraglen implements three methods for estimating mean fragment length. The other functions are related helper functions implementing various methods, but may be useful by themselves for diagnostic purposes. Many of these operations are potentially slow.

sparse.density is intended to be similar to density, but returns the results in a run-length encoded form. This is useful when long stretches of the range of the data have zero density.

Usage

```
estimate.mean.fraglen(x, method = c("SISSR", "coverage", "correlation"),
                      ...)

basesCovered(x, shift = seq(5, 300, 5), seqLen = 35, verbose = FALSE)

densityCorr(x, shift = seq(0, 500, 5), center = FALSE, width = 50, ...)

sparse.density(x, width = 50, kernel = "epanechnikov",
               experimental = TRUE, from, to)
```

Arguments

x	For `estimate.mean.fraglen`, a RangedData object, GenomeData object, AlignedRead object, RangesList object, IntegerList object or a list with elements `"+"` and `"-"` representing locations of reads aligned to positive and negative strands (the values should be integers denoting the location where the first sequenced base matched.) For `basesCovered`, and `densityCorr`, a list with elements `"+"` and `"-"` representing locations of reads aligned to positive and negative strands (the values should be integers denoting the location where the first sequenced base matched.) For `sparse.density`, a numeric or integer vector for which density is to be computed.
method	Character string giving method to be used. `method = "SISSR"` implements the method described in Jothi et al (see References below). `method = "correlation"` implements the method described in Kharchenko et al (see References below), where the idea is to compute the density of tag start positions separately for each strand, and then determine the amount of shift that maximizes the correlation between these two densities. `method = "coverage"` computes the optimal shift for which the number of bases covered by any read is minimized.
shift	Integer vector giving amount of shifts to be tried when optimizing. The current algorithm simply evaluates all supplied values and reports the one giving minimum coverage or maximum correlation.
seqLen	For the `"coverage"` method, the amount by which each read should be extended before computing the coverage. Typically the read length.
verbose	Logical specifying whether progress information should be printed during execution.
center	For the `"correlation"` method, whether the calculations should incorporate centering by the mean density. The default is not to do so; as the density is zero over most of the genome, this slightly improves efficiency at negligible loss in accuracy.
width	half-bandwidth used in the computation. This needs to be specified as an integer, data-driven rules are not supported.
kernel	A character string giving the density kernel.
experimental	logical. If `TRUE`
from, to	specifies range over which the density is to be computed.
...	Extra arguments, passed on as appropriate to other functions.

Details

These functions are typically used in conjunction with `gdapply`.

For the correlation method, the range over which densities are computed only cover the range of reads; that is, the beginning and end of chromosomes are excluded.

Value

`estimate.mean.fraglen` gives an estimate of the mean fragment length.

`basesCovered` and `densityCorr` give a vector of the corresponding objective function evaluated at the supplied values of `shift`.

`sparse.density` returns an object of class `"Rle"`.

Author(s)

Deepayan Sarkar

References

R. Jothi, S. Cuddapah, A. Barski, K. Cui, and K. Zhao. Genome-wide identification of in vivo protein-DNA binding sites from ChIP-Seq data. *Nucleic Acids Research*, 36:5221–31, 2008.

P. V. Kharchenko, M. Y. Tolstorukov, and P. J. Park. Design and analysis of ChIP experiments for DNA-binding proteins. *Nature Biotechnology*, 26:1351–1359, 2008.

See Also

gdapply

Examples

```
data(cstest)
estimate.mean.fraglen(cstest[["ctcf"]], method = "coverage")
```

---

extendReads *A function to extend short reads.*

Description

Since the short read is typically represents one end of a longer fragment there are situations where extending it to the approximate length of the fragment can be useful.

Usage

```
extendReads(reads, seqLen = 200, strand = c("+", "-"))
```

Arguments

reads	Either an AlignedReads object or a list of AlignedReads objects (or a list with aligned reads for each strand.)
seqLen	The desired length of the final sequence, assumed to be the same for all reads.
strand	Which strand + or − the read is aligned to.

Details

Read locations are presumed to be the 5' end (relative to the + strand of the chromosome). Thus reads on the plus strand are simply extended. Those that align to the minus strand, we must subtract the read length, then grow the read towards the 5' end of the + strand (3' end of the minus strand).

Value

An IRanges object with the new ranges, or a list of IRanges objects, depending on the input.

Author(s)

R. Gentleman

Examples

```
data(cstest)
extRanges1 <- gdapply(cstest, extendReads, seqLen = 200)

## AlignedRead example
sp <- SolexaPath(system.file("extdata", package="ShortRead"))
aln <- readAligned(sp, "s_2_export.txt")
extRanges2 <- extendReads(aln[!is.na(position(aln))])
```

---

genomic_regions *(Deprecated) Functions that compute genomic regions of interest.*

Description

Functions that compute genomic regions of interest such as promotor, upstream regions etc, from the genomic locations provided in data like `geneMouse`. **These functions are deprecated in favor of `transcripts`, `exons`, and `introns` in the `GenomicFeatures` package.**

Usage

```
genomic_regions(genes, proximal = 500, distal = 10000)
genomic_exons(genes)
genomic_introns(genes)
```

Arguments

`genes`	A data.frame like that provided by `geneMouse`.
`proximal`	The number of bases on either side of TSS and 3'-end for the promoter and end region, respectively.
`distal`	The number of bases on either side for upstream/downstream, i.e. enhancer/silencer regions.

Details

Fairly simply additions/subtractions are made. The assumption made for introns is that there must be more than one exon, and that the introns are between the end of one exon and before the start of the next exon.

Value

For `genomic_regions` a data.frame with all components computed. For `genomic_exons` a data.frame with one row per exon. For `genomic_introns` a data.frame with one row per intron.

Author(s)

M. Lawrence.

Examples

```
## use functions in GenomicFeatures
```

`readReads` *A function to read in Aligned Short Reads*

Description

This is a helper function for reading in aligned reads with a number of parameters preset at values we have found useful for analyzing ChIP-seq data.

Usage

```
readReads(srcdir, lane, ...,
          include = "chr[0-9]+$", type = "MAQMapShort",
          simplify = TRUE, minScore = 15)
```

Arguments

`srcdir`	The source directory.
`lane`	The name of the file for each lane (logical subset).
`...`	Additional parameters.
`include`	A regular expression indicating which chromosomes to retain.
`type`	The type of alignment used (MAQ, Bowtie etc).
`simplify`	Logical indicating whether the result should be reduced to a simpler `"GenomeData"` object, which only retains the locations of the alignments.
`minScore`	A minimum quality score cutoff (possibly MAQ specific).

Details

This has mainly been used for MAQ alignments. Our default parameters are to include only autosomal chromosomes (there seem to be problems with the others that will require details). We reduce to one read per start location and strand.

Value

If `simplify=FALSE`, a `"AlignedRead"` object; otherwise, a `"GenomeData"` object.

Coercion

When `simplify=TRUE` is specified, the return value is simplified to contain only alignment locations (and not associated quality information, etc.). This simplification can also be done afterwards through coercion methods:

`as.list(x)`: where `x` is an object of class `"AlignedRead"`

`as(object, "GenomeData")`: where `object` is an object of class `"AlignedRead"`

Author(s)

D. Sarkar

See Also

`readAligned`, `GenomeData`

Examples

```
## Not run:
## load reads mapped to chr10 in lane 2 from current working directory
readReads(".", "s_2_export.txt", include = "chr10")
## load all chromosomes in lane 1 from Bowtie output (20 quality cutoff)
readReads(".", "s_1_export.txt", type="Bowtie", minScore=20)

## End(Not run)
```

`subsetSummary` *Compute summaries for cumulative subsets of a short-read data set.*

Description

Divides a short-read dataset into several subsets, and computes various summaries cumulatively. The goal is to study the characteristics of the data as a function of sample size.

Usage

```
subsetSummary(x, chr, nstep, props = seq(0.1, 1, 0.1),
              chromlens, fg.cutoff = 6, seqLen = 200,
              fdr.cutoff = 0.001, resample = TRUE,
              islands = TRUE, verbose = getOption("verbose"))
```

Arguments

`x`	A `"GenomeData"` object representing alignment locations at the sample level.
`chr`	The chromosome for which the summaries are to be obtained. Must specify a valid element of `x`
`nstep`	The number of maps in each increment for the full dataset (not per-chromosome). This will be translated to a per-chromosome number proportionally.
`props`	Alternatively, an increasing sequence of proportions determining the size of each subset. Overrides `nstep`.
`chromlens`	A named vector of per-chromosome lengths, typically the result of `seqlengths`.
`fg.cutoff`	The coverage depth above which a region would be considered foreground.
`seqLen`	The number of bases to which to extend each read before computing coverage.
`resample`	Logical; whether to randomly reorder the reads before dividing them up into subsets. Useful to remove potential order effects (for example, if data from two lanes were combined to produce `x`).
`fdr.cutoff`	The maximum false discovery rate for a region that is considered to be foreground.
`islands`	Logical. If `TRUE`, the whole island would be considered foreground if the maximum depth equals or exceeds `fg.cutoff`. If `FALSE`, only the region above the cutoff would be considered foreground.
`verbose`	logical controlling whether progress information will be shown during computation (which is potentially long-running).

Value

A data frame with various per-subset summaries.

Author(s)

Deepayan Sarkar

Examples

```
data(cstest)
library(BSgenome.Mmusculus.UCSC.mm9)
## summarize lane 1, chr10 at 0.1, 0.6 and 1.0 proportions
subsetSummary(cstest[[1]], "chr10", props=seq(0.1, 1, 0.5),
              chromlens=seqlengths(Mmusculus))
```

Index