chipseq

October 25, 2011

`chipseqFilter` *Filtering ChIP-seq reads*

Description

Convenience for creating an `SRFilter` object appropriate for ChIP-seq data. Typically, the result is passed to `readAligned` when loading reads.

Usage

```
chipseqFilter(exclude = "[_MXY]", uniqueness = c("location", "sequence", "locati
```

Arguments

`exclude`	A regular expression for excluding chromosomes by name. Just like the parameter to `bsapply`.
`uniqueness`	The criteria used to determine whether a read is unique. A read may be unique if it maps to a unique `location`, has a unique `sequence` or both. Specifying `none` avoids this test entirely.
`hasStrand`	Whether to require that the read is mapped to a strand, which usually translates to whether the read was mapped at all.

Value

An `SRFilter` object

Author(s)

M. Lawrence

Examples

```
sp <- SolexaPath(system.file("extdata", package="ShortRead"))

filter <- chipseqFilter()
aln <- readAligned(sp, "s_2_export.txt", filter=filter)
## allow mapping to the same location (but only if sequence is different)
filter <- chipseqFilter(uniqueness = "sequence")
aln <- readAligned(sp, "s_2_export.txt", filter=filter)
```

```
## allow sex chromosomes
filter <- chipseqFilter(exclude = "[M_]")
aln <- readAligned(sp, "s_2_export.txt", filter=filter)
```

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

Description

Combines or subsamples data from multiple lanes on a per-chromosome basis. **THIS FUNCTION IS DEPRECATED**: no longer needed given current infrastructure. With a `GRanges`, simply combine with `c` and use `unique` to mimic setting `keep.unique` to `TRUE`.

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 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
cstest.sub <- laneSubsample(cstest[[1]], cstest[[2]])
## two lanes of chr10 become one
##combineLanes(cstest, "chr10") # DEPRECATED
unlist(cstest.sub) # instead (for all chromosomes)
```

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.

DEPRECATED: The user is likely better off using the GenomicFeatures package to explore the relationship between peaks and genomic annotations. See the vignette for an example.

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

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.

DEPRECATED: Instead, construct an RleViewsList using the Views function.

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)
## deprecated:
##peaks.cov <- copyIRanges(peaks, cov)
## instead:
peaks.cov <- Views(cov, peaks)
```

---

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 <- Views(cov, peaks)
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
```

`diffPeakSummary` *A function to identify and produce summary statistics for*

Description

Given two sets of peaks, this function combines them and summarizes the individual coverage vectors under the combined peak set.

Usage

```
diffPeakSummary(ranges1, ranges2,
                viewSummary = list(sums = viewSums, maxs = viewMaxs))
```

Arguments

`ranges1`	First set of peaks (typically an `RleViewsList`).
`ranges2`	Second set of peaks (typically an `RleViewsList`).
`viewSummary`	A list of the per peak summary functions.

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)
seqlevels(cstest) <- seqlevels(Mmusculus)
seqlengths(cstest) <- seqlengths(Mmusculus)
## find peaks
findPeaks <- function(reads) {
  reads.ext <- resize(reads, width = 200)
  slice(coverage(reads.ext), lower = 8)
}
peakSummary <- diffPeakSummary(findPeaks(cstest$gfp), findPeaks(cstest$ctcf))
```

`estimate.mean.fraglen` *Estimate summaries of the distribution of fragment lengths in a*

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`, typically an AlignedRead or a GRanges object. Also supported but deprecated, as they do not have formal strand information: RangedData (with a "strand" column), or a list-like object 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.) Supported (but again, deprecated) list types include: RangesList, IntegerList or an ordinary R list. Also a GenomeData where the per-chromosome elements are one of the above list types. 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 assumed length of each read for computing the coverage. Typically the read length. This is added to the shift estimated by "coverage" and "correlation" to come up with the actual fragment 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, Michael Lawrence

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. **DEPRECATED**: use the `resize` method on the `GRanges` class.

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)
## deprecated:
##extRanges1 <- gdapply(cstest, extendReads, seqLen = 200)
## instead:
extRanges1 <- endoapply(cstest, resize, width = 200)

## AlignedRead example
sp <- SolexaPath(system.file("extdata", package="ShortRead"))
aln <- readAligned(sp, "s_2_export.txt", filter=chipseqFilter())
## deprecated:
## extRanges2 <- extendReads(aln)
## instead:
resize(as(aln, "GRanges"), width = 200)
```

islandDepthPlot *Plot island depth distribution*

Description

Plots the distribution of island depths using points for the observed islands and a line for the Poisson estimate of the noise. Useful for choosing a depth corresponding to a desired FDR.

Usage

```
islandDepthPlot(x, maxDepth = 20L)
```

Arguments

x	A coverage object, e.g., RleList.
maxDepth	The maximum depth to plot (there are usually some outliers).

Author(s)

D. Sarkar, M. Lawrence

See Also

peakCutoff for calculating a cutoff value for an FDR.

Examples

```
data(cstest)
cov <- coverage(resize(cstest$ctcf, width=200))
islandDepthPlot(cov)
```

peakCutoff *Calculate a peak cutoff*

Description

Calculates a peak cutoff value given an FDR, assuming a Poisson noise distribution estimated from the frequency of singleton and doubleton islands.

Usage

```
peakCutoff(cov, fdr.cutoff = 0.001, k = 2:20)
```

Arguments

cov	The coverage object, e.g., an RleList object.
fdr.cutoff	The maximum-allowed FDR for calculating the cutoff.
k	The coverage levels at which to estimate an FDR value. The maximal value that is less than fdr.cutoff is chosen for calculating the cutoff. Usually best left to the default.

Value

A numeric value to use for calling peaks

Author(s)

D. Sarkar and M. Lawrence

See Also

`islandDepthPlot` for the graphical equivalent; the vignette for a bit more explanation.

Examples

```
data(cstest)
cov <- coverage(resize(cstest$ctcf, width=200))
peakCutoff(cov)
```

---

peakSummary-methods

Summarizing peak sets

Description

Summarizes a set of peaks into a `RangedData` object with columns of statistics like the peak maxima and integrals (sums).

Usage

```
peakSummary(x, ...)
```

Arguments

`x`	An object containing peaks, usually a `RleViewsList`.
`...`	Arguments to pass to methods

Value

A `RangedData` object of the peaks, with columns named `max`, `maxpos` (position of the maximum, centered), and `sum`.

See Also

View summary `utilities` like `viewMaxs` and `viewSums`.

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.

DEPRECATED: Given the filter support that now exists for `readAligned`, one should now create a ChIP-Seq appropriate filter with `chipseqFilter` and pass it to `readAligned`.

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 = seqlengths(x), fg.cutoff = 6, seqLen = 200,
              fdr.cutoff = 0.001, use.fdr = FALSE, 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.
use.fdr	Whether to use the FDR detected peaks when calling foreground and background.

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.

Note

This function should be considered preliminary, in that it might change significantly or simply be removed in a subsequent version. If you like it the way it is, please notify the maintainer.

Author(s)

Deepayan Sarkar, Michael Lawrence

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