Package 'NADfinder'

October 18, 2017

Type Package

Title Call wide peaks for sequencing data

Version 1.0.1

Date 2017-07-24

Author Jianhong Ou, Jun Yu, Anastassiia Vertii, Paul Kaufman, Lihua Julie Zhu

Maintainer Jianhong Ou <jianhong.ou@umassmed.edu>,
Lihua Julie Zhu <julie.zhu@umassmed.edu>

Description Nucleolus is an important structure inside the nucleus in eukaryotic cells. It is the site for transcribing rDNA into rRNA and for assembling ribosomes, aka ribosome biogenesis. In addition, nucleoli are dynamic hubs through which numerous proteins shuttle and contact specific non-rDNA genomic loci. Deep sequencing analyses of DNA associated with isolated nucleoli (NAD-seq) have shown that specific loci, termed nucleolar-associated domains (NADs) form frequent three-dimensional associations with nucleoli. NAD-seq has been used to study the biological functions of NAD and the dynamics of NAD distribution during embryonic stem cell (ESC) differentiation.
Here, we developed a Bioconductor package NADfinder for bioinformatic analysis of the NAD-seq data, including normalization, smoothing, peak calling, peak trimming and annotation.

License GPL (>= 2)

Depends R (>= 3.4), BiocGenerics, IRanges, GenomicRanges, S4Vectors

Imports graphics, methods, baseline, signal, GenomicAlignments, GenomeInfoDb, rtracklayer, SummarizedExperiment, limma, trackViewer, stats, utils

Suggests RUnit, BiocStyle, knitr, BSgenome.Mmusculus.UCSC.mm10, testthat

biocViews Sequencing, DNASeq, GeneRegulation, PeakDetection

LazyData TRUE

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

R topics documented:

`NADfinder-package` *Call peaks for nucleolar-associated domains (NADs) sequencing data*

Description

Call peaks for two purified nucleoli samples: target and control. It will count the reads for tiles of the genome and NADs, then convert it to ratios. The ratios will be corrected and smoothed. The z-scores is calculated for each counting windows over the background. The peaks will be detected based on z-scores.

`backgroundCorrection` *Correct ratios for background*

Description

Background correct ratios of read counts for each window.

Usage

```
backgroundCorrection(ratios)
```

Arguments

`ratios`	A vector of numeric. It is the ratios of counts for each window.

Details

This function implements the backgound correction methods of algorithm for polynomial fitting. See details via `baseline.modpolyfit`. This function expects the trendency of decreasing of the ratios from 5' end to 3' end.

Value

A vector of numeric. It is the background corrected ratios.

Examples

```
x <- runif(200)
background <- rep(c(20:1)/100, each=10)
backgroundCorrection(x)
```

`butterFilter` *Low pass filter on ratios by butterworth filter*

Description

The Butterworth filter is a type of signal processing filter designed to have as flat a frequency response as possible in the passband.

Usage

```
butterFilter(ratios, N = ceiling(length(ratios)/200))
```

Arguments

`ratios`	A vector of numeric. It is the ratios of counts in each window.
`N`	numeric(1) or integer(1). Critical frequencies of the low pass filter will be set as 1/N. 1/N is a cutoff at 1/N-th of the Nyquist frequency. Default suppose there are about 200 peaks in the inputs.

Value

A vector of numeric with same length of input ratios. The vector indicates smoothed ratios.

Examples

```
ratios <- runif(20000)
butterFilter(ratios)
```

`callPeaks` *call peaks for ratios of repeats*

Description

Use limma to call peaks for ratios of repeats

Usage

```
callPeaks(se, backgroundCorrectionAssay = "bcRatio",
  normlizationMethod = "quantile", N = 100, cutoffAdjPvalue = 0.05,
  countFilter = 1000, ...)
```

Arguments

`se`	An object of RangedSummarizedExperiment with assays of raw counts, ratios, background correct ratios, smoothed ratios and z-scores. It should be an element of output of smoothRatiosByChromosome
`backgroundCorrectionAssay`	character(1). Assays names for background correction ratios
`normlizationMethod`	character(1) specifying the normalization method to be used. Choices are "none", "scale", "quantile" or "cyclicloess". See normalizeBetweenArrays for details.
`N`	numeric(1) or integer(1). Critical frequencies of the low pass filter will be set as 1/N. 1/N is a cutoff at 1/N-th of the Nyquist frequency. Default 100.
`cutoffAdjPvalue`	numeric(1). Cutoff adjust p-value.
`countFilter`	numeric(1). Cutoff value for mean of raw reads count in each window.
`...`	Parameter not used.

Value

An object of GRanges of peak list with metadata "AveSig", "P.Value", and "adj.P.Val", where "AveSig" means average signals.

Examples

```
data(triplicates.counts)
se <- triplicates.counts
gps <- c("26", "28", "29")
se <- log2se(se,
             nucleosomeCols = paste0("N", gps, ".bam"),
             genomeCols = paste0("G", gps, ".bam"))
se<- smoothRatiosByChromosome(se, chr="chr18")
peaks <- callPeaks(se[[1]][10000:15000, ],
                cutoffAdjPvalue=0.05, countFilter=1000)
```

countByOverlaps *Count overlapping genomic ranges*

Description

Count the reads in a given feature. This function does not work for parallel.

Usage

```
countByOverlaps(features, reads, ignore.strand, inter.feature)
```

Arguments

features	A object of GRanges represents the feature regions to be counted.
reads	object that represents the data to be counted. See summarizeOverlaps.
ignore.strand	logical(1). ignore strand?
inter.feature	not used. This parameter is required by summarizeOverlaps.

Value

return a vector of counts the same length as features.

cumulativePercentage *Plot the cumulative percentage tag allocation in sample*

Description

Plot the difference between the cumulative percentage tag allocation in paired samples.

Usage

```
cumulativePercentage(se, binWidth = 1e+05,
  backgroundCorrectionAssay = "bcRatio", ...)
```

Arguments

se	An object of RangedSummarizedExperiment with assays of raw counts, ratios, background correct ratios, smoothed ratios and z-scores. It should be an element of output of smoothRatiosByChromosome.
binWidth	numeric(1) or integer(1). The width of each bin.
backgroundCorrectionAssay	character(1). Assays names for background correction ratios.
...	Parameter not used.

Value

A list of data.frame with the cumulative percentages.

References

Normalization, bias correction, and peak calling for ChIP-seq Aaron Diaz, Kiyoub Park, Daniel A. Lim, Jun S. Song Stat Appl Genet Mol Biol. Author manuscript; available in PMC 2012 May 3.Published in final edited form as: Stat Appl Genet Mol Biol. 2012 Mar 31; 11(3): 10.1515/1544-6115.1750 /j/sagmb.2012.11.issue-3/1544-6115.1750/1544-6115.1750.xml. Published online 2012 Mar 31. doi: 10.1515/1544-6115.1750 PMCID: PMC3342857

Examples

```
data(triplicates.counts)
se <- triplicates.counts
gps <- c("26", "28", "29")
se <- log2se(se,
             nucleosomeCols = paste0("N", gps, ".bam"),
             genomeCols = paste0("G", gps, ".bam"))
se <- smoothRatiosByChromosome(se, chr="chr18")
cumulativePercentage(se[["chr18"]])
```

---

exportSignals *output signals to file*

Description

Output signals to bedgraph, bed, wig, etc, for track viewer

Usage

```
exportSignals(dat, assayName, colName, con, format = "bedGraph", ...)
```

Arguments

dat	An object of GRanges, or RangedSummarizedExperiment with assays of raw counts, ratios, background correct ratios, smoothed ratios and z-scores. It should be an element of output of smoothRatiosByChromosome
assayName	character(1). Assay name for RangedSummarizedExperiment
colName	character(1). Column name of metadata of dat or assay of dat for coverage weight, see coverage, RangedSummarizedExperiment.
con	The connection to which data is saved. If this is a character vector, it is assumed to be a filename and a corresponding file connection is created and then closed after exporting the object. If missing, a SimpleRleList will be returned.
format	The format of the output. see export.
...	Parameters to pass to export

Value

If con is missing, a SimpleRleList will be returned. Otherwise, nothing is returned.

Examples

```
gr <- GRanges("chr1", IRanges(seq_len(100), 201:300), reads=rep(1, 100))
myTrackLine <- new("TrackLine", name="my track",
                   description="description of my track",
                   color=col2rgb("red")[, 1],
                   visibility="full")
exportSignals(gr, colName="reads",
              con="test.bedGraph", trackLine=myTrackLine)
data(triplicates.counts)
exportSignals(triplicates.counts, "counts",
              "G26.bam", "test.bw", format="bigWig")
```

`getCorrelations` *get correlations for replicates*

Description

Get the correlations of replicates by the coverage of peaks. The signals will be filter by the background cutoff value and the correlations will be calculated.

Usage

```
getCorrelations(se, chr = paste0("chr", seq_len(21)), ratioAssay = "ratio",
  window = 10000, cutoff = 1, method = c("spearman", "pearson",
  "kendall"), ...)
```

Arguments

`se`	A RangedSummarizedExperiment object. The output of log2se.
`chr`	A vector of character. Filter for seqnames. It should be the chromosome names to be kept.
`ratioAssay`	character(1). Column name of ratio for correlation calculation.
`window`	numeric(1) or integer(1). The window size for summary of the ratios.
`cutoff`	numeric(1). All the coverages lower than cutoff value in a given window will be filtered out.
`method`	A character string indicating which correlation coefficient is to be computed. See cor.
`...`	Parameters not used.

Value

A list of matrixes of correlation and coefficient.

Examples

```
data(triplicates.counts)
se <- triplicates.counts
gps <- c("26", "28", "29")
se <- log2se(se,
             nucleosomeCols = paste0("N", gps, ".bam"),
             genomeCols = paste0("G", gps, ".bam"))
getCorrelations(se, chr="chr18")
```

`groupZscores` *Calculate z-scores for each peak*

Description

Detect peaks and calculate z-scores for each peak

Usage

```
groupZscores(zscore)
```

Arguments

`zscore`	A vector of numeric. It is the z-scores of ratios in each window.

Value

A data.frame with column names as "zscore", "group", "grp.zscore", and "pvalue".

Examples

```
x <- seq_len(500)
a <- 2 * 2*pi/length(x)
y <- 20 * sin(x*a)
noise1 <- 20 * 1/10 * sin(x*a*10)
zscore <- y+noise1
groupZscores(zscore)
```

`log2ratio` *calculate the log2 transformed ratios*

Description

calculate the log2 transformed ratios for nucleosome vs genome. pseudo-count will be used to avoid x/0.

Usage

```
log2ratio(A, B, pseudocount)
```

Arguments

A, B	counts for nucleosome and genome. They should be numeric vectors with identical length.
pseudocount	pseudo-count will be used to aviod x/0 by x/pseudocount. If it is not set, pseudo-count will be the minimal count except 0 of inputs.

Value

A vector of numeric of log2 transformed ratios.

Examples

```
log2ratio(seq_len(10), 10:1)
```

---

log2se *calculate the log2 transformed ratios for SummarizedExperiment class*

Description

Calculate the log2 transformed ratios for nucleosome vs genome. pseudo-count will be used to avoid x/0.

Usage

```
log2se(se, nucleosomeCols, genomeCols, pseudocount)
```

Arguments

se	A RangedSummarizedExperiment object. The output of tileCount.
nucleosomeCols, genomeCols	column Names of counts for nucleosome and genome. They should be the column names in the assays of se. Ratios will be calculated as log2(nucleosomeCols/genomeCols).
pseudocount	pseudo-count will be used to aviod x/0 by x/pseudocount. If it is not set, pseudo-count will be the minimal count except 0 of inputs.

Value

A RangedSummarizedExperiment object with log2 transformed ratios. Assays will be named as nucleosome, genome and ratio.

Examples

```
library(SummarizedExperiment)
se <-
 SummarizedExperiment(assays=list(counts=DataFrame(A=seq_len(3),
                                                   B=rep(1, 3))),
                      rowRanges=GRanges("chr1",
                                        IRanges(c(1, 10, 20),
                                                width=9)))
log2se(se, "A", "B")
```

peakdet *Detect peak positions*

Description

Detect the peaks positions and valley positions. The algorithm is modified from github::dgromer/peakdet

Usage

```
peakdet(y, delta = 0, silence = TRUE)
```

Arguments

y	A vector of numeric where to search peaks
delta	A numeric of length 1, defining the local threshold for peak detection. If it is set to 0, the delta will be set to 1/10 of the range of y.
silence	logical(1). If false, echo the delta value when delta is set as 0.

Value

A list with peakpos and valleypos. Both peakpos and valleypos are vectors of numeric which indicate the positions of peak or valley.

Examples

```
y <- runif(200)
peakdet(y)
```

plotSig *plot signals with ideograms*

Description

Plot signals with ideograms for GRangesList.

Usage

```
plotSig(ideo, grList, mcolName, ...)
```

Arguments

ideo	Output of loadIdeogram.
grList	A GRangesList of data to plot.
mcolName	Column name of metadata of GRangesList for plotting.
...	Parameters to pass to ideogramPlot

Value

Invisible argument list for ideogramPlot.

Examples

```
library(trackViewer)
#ideo <- loadIdeogram("mm10")
ideo <- readRDS(system.file("extdata", "ideo.mm10.rds",
                            package = "NADfinder"))
gr1 <- gr2 <- ideo
mcols(gr1) <- DataFrame(score=runif(length(gr1)))
mcols(gr2) <- DataFrame(score=runif(length(gr2)))
grList <- GRangesList(gr1, gr2)
plotSig(ideo, grList, mcolName="score", layout=list("chr1"))
```

---

single.count *counts data for single experiment of chromosome 18*

Description

counts data for single experiment of chromosome 18

---

smoothRatiosByChromosome *smooth the ratios by chromosome*

Description

Split the ratios by chromosome and do background correction and smooth.

Usage

```
smoothRatiosByChromosome(se, chr = paste0("chr", c(seq_len(21), "X", "Y")),
  ratioAssay = "ratio", backgroundCorrectionAssay = "bcRatio",
  smoothedRatioAssay = "smoothedRatio", zscoreAssay = "zscore",
  backgroundPercentage = 0.25, ...)
```

Arguments

se	An object of RangedSummarizedExperiment with scores. Output of log2se
chr	A vector of character. Filter for seqnames. It should be the chromosome names to be kept.
ratioAssay	The name of assay in se, which store the values to be smoothed.
backgroundCorrectionAssay, smoothedRatioAssay, zscoreAssay	character(1). Assays names for background correction ratios, smoothed ratios and z-score based on background correction ratios.
backgroundPercentage	numeric(1). Percentage of values for background, see zscoreOverBck. How many percent lower values will be treated as background.
...	Parameters could be passed to butterFilter.

Value

A SimpleList of RangedSummarizedExperiment with smoothed ratios.

Examples

```
data(single.count)
se <- single.count
dat <- log2se(se, nucleosomeCols="nucleosome.bam", genomeCols="genome.bam")
dat <- smoothRatiosByChromosome(dat, N=100)
```

---

tileCount *Perform overlap queries between reads and genome by windows*

Description

tileCount extends summarizeOverlaps by providing fixed window size and step to split whole genome into windows and then do queries. It will return counts in each window.

Usage

```
tileCount(reads, genome, windowSize = 100000L, step = 10000L,
  mode = countByOverlaps, dataOverSamples = FALSE, ...)
```

Arguments

reads	A GRanges, GRangesList (should be one read per list element), GAlignments, GAlignmentsList, GAlignmentPairs or BamFileList object that represents the data to be counted by summarizeOverlaps.
genome	The object from/on which to get/set the sequence information.
windowSize	numeric(1) or integer(1). Size of windows.
step	numeric(1) or integer(1). Step of windows.
mode	mode can be one of the pre-defined count methods. see summarizeOverlaps. default is countByOverlaps, alia of countOverlaps(features, reads, ignore.strand=ignore.strand)
dataOverSamples	logical(1). Data over several samples when use GRangesList as input?
...	Additional arguments passed to summarizeOverlaps.

Value

A RangedSummarizedExperiment object. The assays slot holds the counts, rowRanges holds the annotation from sliding widows of genome.

Examples

```
## Not run:
fls <- list.files(system.file("extdata", package="GenomicAlignments"),
recursive=TRUE, pattern="*bam$", full=TRUE)
names(fls) <- basename(fls)
genes <- GRanges(seqlengths = c(chr2L=7000, chr2R=10000))
se <- tileCount(fls, genes, windowSize=1000, step=500)

## End(Not run)

##
genome <- GRanges("chr1", IRanges(1, 1))
seqlengths(genome) <- c(chr1=1000)
reads <- GRanges("chr1", IRanges((seq_len(90))*10, width=10))
tileCount(reads, genome, windowSize=100, step=50)
```

`trimPeaks` *Trim peaks*

Description

Filter the peaks by pvalue and trim the range of peaks for sample without duplicates.

Usage

```
trimPeaks(se, cutoffPvalue = 0.05, backgroundPercentage = 0.25,
  countFilter = 1000, ratioAssay = "ratio",
  backgroundCorrectionAssay = "bcRatio",
  smoothedRatioAssay = "smoothedRatio", zscoreAssay = "zscore")
```

Arguments

`se`	An object of RangedSummarizedExperiment with assays of raw counts, ratios, background correct ratios, smoothed ratios and z-scores. It should be an element of output of smoothRatiosByChromosome
`cutoffPvalue`	numeric(1). Cutoff p-value.
`backgroundPercentage`	numeric(1). Cutoff value for the peaks height.
`countFilter`	numeric(1) or integer(1). Cutoff value for mean of raw reads count in each window.
`ratioAssay`	character(1). The name of assay in se, which store the values to be smoothed.
`backgroundCorrectionAssay, smoothedRatioAssay, zscoreAssay`	Assays names for background correction ratios, smoothed ratios and z-score based on background correction ratios.

Value

An object of GRanges.

Examples

```
data(single.count)
se <- single.count
## Calculate ratios for peak calling. We use signal vs input.
dat <- log2se(se, nucleosomeCols="nucleosome.bam", genomeCols="genome.bam")
## Smooth the ratios for each chromosome.
dat <- smoothRatiosByChromosome(dat, N=100)
peaks <- trimPeaks(dat[["chr18"]],
                backgroundPercentage=.25,
                cutoffPvalue=0.05, countFilter=1000)
```

---

triplicates.counts *counts data for triplicates of chromosome 18*

Description

counts data for triplicates of chromosome 18

---

zscoreOverBck *Z-scores over the background*

Description

Calculate the z-scores over the lower percentage values.

Usage

```
zscoreOverBck(ratios, backgroundPercentage = 0.25)
```

Arguments

ratios	A vector of numeric. It is the ratios of counts in each window.
backgroundPercentage	numeric(1). Percentage of value for background.

Value

A vector of numeric. Z-scores.

Examples

```
r <- runif(200)
zscoreOverBck(r)
```

Index