Package 'CSSQ'

November 26, 2024

Title Chip-seq Signal Quantifier Pipeline

Version 1.19.0

Author Ashwath Kumar [aut], Michael Y Hu [aut], Yajun Mei [aut], Yuhong Fan [aut]

Maintainer Fan Lab at Georgia Institute of Technology <yuhong.fan@biology.gatech.edu>

Description This package is desgined to perform statistical analysis to identify statistically significant differentially bound regions between multiple groups of ChIP-seq dataset.

License Artistic-2.0

Encoding UTF-8

biocViews ChIPSeq, DifferentialPeakCalling, Sequencing, Normalization

Depends SummarizedExperiment, GenomicRanges, IRanges, S4Vectors, rtracklayer

Imports GenomicAlignments, GenomicFeatures, Rsamtools, ggplot2, grDevices, stats, utils

Suggests BiocStyle, knitr, rmarkdown, markdown

VignetteBuilder knitr

LazyData true

RoxygenNote 7.0.0

git_url https://git.bioconductor.org/packages/CSSQ

git_branch devel

git_last_commit 9eae002

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-11-26

Contents

`ansTransform` *Perform quantification and normalization of count data*

Description

This function quantifies each each region for a sample and performs background correction and normalization as instructed. Returns a vector of count information for the input regions.

Usage

```
ansTransform(countData, noNeg = TRUE, plotDataToPDF = FALSE)
```

Arguments

`countData`	A `RangedSummarizedExperiment-class` object from `getRegionCounts` with count data.
`noNeg`	A Logical parameter indicating how to deal with negative values. When TRUE (default), all negative values will be moved to 0 before transforming. When FALSE, the signs will be maintained while the transformation will be applied to the absolute value. (default: TRUE)
`plotDataToPDF`	A logical parameter indicating whether to make plots of the data distribution to a separate PDF file for each sample. When TRUE, a histogram will be plotted for the data before and after transformation. When FALSE, no plots will be made. (default: FALSE)

Value

A `RangedSummarizedExperiment-class` object containing the anscombe transformed count data as the assay.

Examples

```
exRange <- GRanges(seqnames=c("chr1","chr2","chr3","chr4"),
ranges=IRanges(start=c(1000,2000,3000,4000),end=c(1500,2500,3500,4500)))
sampleInfo <- read.table(system.file("extdata", "sample_info.txt",
package="CSSQ",mustWork = TRUE),sep="\t",header=TRUE)
exCount <- matrix(c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),nrow=4,ncol=4)
exData <- SummarizedExperiment(assays = list(countData=exCount),
rowRanges=exRange,colData=sampleInfo)
ansExData <- ansTransform(exData)
assays(ansExData)$ansCount
```

`calculateFC` *Calculates Fold change values for the given label and comparison*

Description

This calculates the Fold change for the given comparison.

Usage

```
calculateFC(preprocessedData, label, comparison, numSamples)
```

Arguments

`preprocessedData`	A `RangedSummarizedExperiment-class` object from `preprocessData`.
`label`	A vector containing the labels to use for the samples in preprocessedData.
`comparison`	A vector containing the comparison to be made. Names here need to correspond to the sample groups in the sample file (Eg. c("G1",G2") means the comparison G1/G2).
`numSamples`	Number of samples in the dataset.

Value

A vector that is the Fold change for the comparison and labels given.

See Also

`DBAnalyze` which calls this function

`calculatePvalue` *Calculates P-value for the regions*

Description

This calculates the adjusted P-values for the regions using column permutation and Benjamini Hochberg correction methods.

Usage

```
calculatePvalue(trueTstat, compare_tstats)
```

Arguments

`trueTstat`	The T-statistics value calculated using `calculateTvalue` for the intended comparison.
`compare_tstats`	The T-statistics value calculated using `calculateTvalue` for all other comparisons possible.

Value

A vector that is the adjusted P-value for the intended comparison.

See Also

`DBAnalyze` which calls this function

`calculateTvalue` *Calculates modified T-statistics values for the given label and comparison*

Description

This calculates the modified T-statistics for the given comparison.

Usage

```
calculateTvalue(preprocessedData, label, comparison, numSamples)
```

Arguments

`preprocessedData`	A `RangedSummarizedExperiment-class` object from `preprocessData`.
`label`	A vector containing the labels to use for the samples in preprocessedData.
`comparison`	A vector containing the comparison to be made. Names here need to correspond to the sample groups in the sample file (Eg. c("G1",G2") means the comparison G1/G2).
`numSamples`	Number of samples in the dataset.

Value

A vector that is the modified T-statistics for the comparison and labels given.

See Also

DBAnalyze which calls this function

DBAnalyze *Performs differential binding analysis*

Description

This is a wrapper function that performs the different parts of differential binding analysis. Returns a GRanges-class with a calculated P-value and Fold change for each region.

Usage

```
DBAnalyze(preprocessedData, comparison)
```

Arguments

preprocessedData	A RangedSummarizedExperiment-class object from preprocessData.
comparison	A vector containing the comparison to be made. Names here need to correspond to the sample groups in the sample file (Eg. c("G1",G2") means the comparison G1/G2).

Value

A GRanges-class object containing the regions along with their P-values and Fold change for the comparison.

Examples

```
exRange <- GRanges(seqnames=c("chr1","chr2","chr3","chr4"),
ranges=IRanges(start=c(1000,2000,3000,4000),end=c(1500,2500,3500,4500)))
sampleInfo <- read.table(system.file("extdata", "sample_info.txt",
package="CSSQ",mustWork = TRUE),sep="\t",header=TRUE)
exCount <- matrix(c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),nrow=4,ncol=4)
exData <- SummarizedExperiment(assays = list(ansCount=exCount),
rowRanges=exRange,colData=sampleInfo)
normExData <- normalizeData(exData,numClusters=2)
res <- DBAnalyze(normExData,comparison=c("HSMM","HESC"))
res
```

getBgSubVal *Perform quantification and normalization of count data*

Description

This function quantifies each each region for a sample and performs background correction and normalization as instructed. Returns a vector of count information for the input regions.

Usage

```
getBgSubVal(
  analysisInfo,
  sampleIndex,
  normalizeReadDepth = TRUE,
  normalizeLength = FALSE,
  backgroundSubtract = TRUE,
  countMode = "Union",
  ignore.strand = TRUE,
  inter.feature = FALSE
)
```

Arguments

analysisInfo	A RangedSummarizedExperiment-class object with regions and sample information from within getRegionCounts.
sampleIndex	Index of the sample to process.
normalizeReadDepth	Logical indicating if count data should be normalized for library sequencing depth. When TRUE (default), counts will be normalized for sequencing depth for each library. When FALSE, no such normalization will be performed and raw counts will be used. (default: TRUE)
normalizeLength	Logical indicating if count data should be normalized to the length of the regions. When TRUE, count data will be normalized for the length of the region being analyzed. When FALSE (default), no such normalization will be performed. (default: FALSE)
backgroundSubtract	Logical indicating if background correction should be performed. When TRUE (default), background subtraction will be performed after length and depth normalization if applicable. When FALSE, no background subtraction will be performed. (default: TRUE)
countMode	Count method passed on to summarizeOverlaps from GenomicAlignments package. ("Union", "IntersectionNotEmpty", "IntersectionStrict" or "user supplied function"). (default: "Union")
ignore.strand	A logical indicating if strand should be considered when matching. (default: TRUE) Passed on to summarizeOverlaps from GenomicAlignments package.

`inter.feature` A logical indicating if the ‘r countMode‘ should be aware of overlapping features. When TRUE, reads mapping to multiple features are dropped (i.e., not counted). When FALSE (default), these reads are retained and a count is assigned to each feature they map to. Passed on to summarizeOverlaps from GenomicAlignments package. (default: FALSE)

Value

A vector containing the counts for all the regions.

See Also

`getRegionCounts` which calls this function

Examples

```
regionBed <- read.table(system.file("extdata", "chr19_regions.bed",
package="CSSQ",mustWork = TRUE))
sampleInfo <- read.table(system.file("extdata", "sample_info.txt",
package="CSSQ",mustWork = TRUE),sep="\t",header=TRUE)
sampleInfo[,3] <- sapply(sampleInfo[,3],
function(x) system.file("extdata", x, package="CSSQ"))
sampleInfo[,5] <- sapply(sampleInfo[,5],
function(x) system.file("extdata", x, package="CSSQ"))
regionRange <- GRanges(seqnames=regionBed$V1,
ranges=IRanges(start=regionBed$V2,end=regionBed$V3))
analysisInfo <- SummarizedExperiment(rowRanges=regionRange,
colData=sampleInfo)
NormbgSubCounts <- data.frame(sapply(c(1:nrow(colData(analysisInfo))),
function(x) getBgSubVal(analysisInfo,sampleIndex = x,backgroundSubtract=TRUE,
normalizeReadDepth=TRUE,normalizeLength=FALSE,countMode="Union",
ignore.strand=TRUE,inter.feature=FALSE)))
NormbgSubCounts
```

---

getComparisons *Identify possible combinations*

Description

This function creates a data frame of all possbile combinations of sample labels. This information is utilized by `calculatePvalue` to calculate the P-value using column permutation method.

Usage

```
getComparisons(trueLabel, comparison, numSamples)
```

Arguments

trueLabel	The true labels for the samples.
comparison	A vector containing the comparison to be made. Names here need to correspond to the sample groups in the sample file (Eg. c("G1",G2") means the comparison G1/G2).
numSamples	Number of samples in the dataset.

Value

A data frame with possible combinations of samples other the true intended comparison.

See Also

DBAnalyze which calls this function and getNewLabels which this function calls

---

getNewLabels *Labels the samples according to the combinations from getComparisons*

Description

This function labels the samples according the combinations generated by getComparisons.

Usage

```
getNewLabels(trueLabel, comparison, numSamples, combns, index)
```

Arguments

trueLabel	The true labels for the samples.
comparison	A vector containing the comparison to be made. Names here need to correspond to the sample groups in the sample file (Eg. c("G1",G2") means the comparison G1/G2).
numSamples	Number of samples in the dataset.
combns	Possible combinations of sample index generated in getComparisons.
index	index of the combination to use for labeling.

Value

A vector with labels.

See Also

getComparisons which calls this function

getRegionCounts *Quantify region level count data*

Description

The input is the set of regions and the sample information. It will calculate the number of reads falling in each region for each sample. Returns a RangedSummarizedExperiment-class object with regions, sample informationa and counts for all samples.

Usage

```
getRegionCounts(
  regionBed,
  sampleInfo,
  sampleDir = ".",
  backgroundSubtract = TRUE,
  ...
)
```

Arguments

regionBed	A bed file containing the list of regions that are being analyzed.
sampleInfo	Object from preprocessData containing sample information.
sampleDir	Location of the input sample files in 'sampleInfo' file. (default: ".")
backgroundSubtract	Logical indicating if background correction should be performed. (default: TRUE)
...	Additional arguments passed on to getBgSubVal.

Value

RangedSummarizedExperiment-class containing the regions, sample information and counts for all samples.

See Also

getBgSubVal which this function calls.

Examples

```
sampleInfo <- read.table(system.file("extdata", "sample_info.txt",
package="CSSQ",mustWork = TRUE),sep="\t",header=TRUE)
countData <- getRegionCounts(system.file("extdata", "chr19_regions.bed",
package="CSSQ"),sampleInfo,
sampleDir = system.file("extdata", package="CSSQ"))
countData
head(assays(countData)$countData)
```

```
colData(countData)
rowRanges(countData)
```

`kmeansNormalize` *Perform k-means clustering, normalize anscombe data and calculate cluster variances for a sample.*

Description

This function performs normalization on the anscombe transformed data by clustering them using k-means algorithmn and utilizing the information from clusters. It returns an DataFrame object normalized counts, cluster information and the variance of that cluster for that sample.

Usage

```
kmeansNormalize(ansDataVec, numClusters = 4)
```

Arguments

`ansDataVec`	Anscombe transformed count data for a sample.
`numClusters`	A number indicating the number of clusters to use for k-means clustering. (default: 4)

Value

DataFrame containing the normalized counts, cluster information and the variance of the cluster in the sample.

See Also

`normalizeData` which iterates over this function.

Examples

```
exCount <- c(1,2,3,4,5,6,7,8,9,10)
kmeansEx <- kmeansNormalize(exCount,numClusters=2)
kmeansEx
```

loadCountData *Load count data from input file.*

Description

It converts input count file and a bed file regions into a RangedSummarizedExperiment-class object.

Usage

```
loadCountData(countFile, regionBed, sampleInfo)
```

Arguments

countFile	A path to file containing the count data for the dataset. This should be a tab separated file sample names as header.
regionBed	A bed file containing the list of regions that are being analyzed.
sampleInfo	Object from preprocessData containing sample information.

Value

RangedSummarizedExperiment-class object containing the region information, sample information and the count data.

Examples

```
countData <- loadCountData(system.file("extdata", "sample_count_data.txt",
package="CSSQ",mustWork = TRUE),system.file("extdata", "chr19_regions.bed",
package="CSSQ"),
read.table(system.file("extdata", "sample_info.txt", package="CSSQ",
mustWork = TRUE),
sep="\t",header=TRUE))
countData
```

normalizeData *Normalize anscombe transformed data*

Description

This function iterates over kmeansNormalize to perform normalization for all samples in the dataset. It returns an RangedSummarizedExperiment-class object normalized counts, cluster information and the variance of that cluster for that sample.

Usage

```
normalizeData(ansData, numClusters = 4)
```

Arguments

ansData	RangedSummarizedExperiment-class object from ansTransform.
numClusters	A number indicating the number of clusters to use for k-means clustering. (default: 4)

Value

RangedSummarizedExperiment-class containing the normalized counts, cluster information and the variance of the cluster in the sample.

See Also

kmeansNormalize which this function calls.

Examples

```
exRange <- GRanges(seqnames=c("chr1","chr2","chr3","chr4"),
ranges=IRanges(start=c(1000,2000,3000,4000),end=c(1500,2500,3500,4500)))
sampleInfo <- read.table(system.file("extdata", "sample_info.txt",
package="CSSQ",mustWork = TRUE),sep="\t",header=TRUE)
exCount <- matrix(c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),nrow=4,ncol=4)
exData <- SummarizedExperiment(assays = list(ansCount=exCount),
rowRanges=exRange,colData=sampleInfo)
normExData <- normalizeData(exData,numClusters=2)
assays(normExData)$normCount
```

---

plotDist *Plot data distribution histograms*

Description

This function is to plot data distribution histogram before and after anscombe transformation.

Usage

```
plotDist(countData, ansCount, sampleName, plotDataToPDF = FALSE)
```

Arguments

countData	A RangedSummarizedExperiment-class object from getRegionCounts with count data.
ansCount	A RangedSummarizedExperiment-class object from ansTransform with anscombe transformed data.
sampleName	Name of the sample being plotted.
plotDataToPDF	A logical parameter indicating whether to make plots of the data distribution to a separate PDF file for each sample. When TRUE, a histogram will be plotted for the data before and after transformation. When FALSE, no plots will be made. (default: FALSE)

Value

A list of the histogram of the count data before and after anscombe transformation if plotDataToPDF == FALSE. None if plotDataToPDF == TRUE.

Examples

```
exRange <- GRanges(seqnames=c("chr1","chr2","chr3","chr4"),
ranges=IRanges(start=c(1000,2000,3000,4000),end=c(1500,2500,3500,4500)))
sampleInfo <- read.table(system.file("extdata", "sample_info.txt",
package="CSSQ",mustWork = TRUE),sep="\t",header=TRUE)
exCount <- matrix(c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),nrow=4,ncol=4)
exData <- SummarizedExperiment(assays = list(countData=exCount),
rowRanges=exRange,colData=sampleInfo)
ansExData <- ansTransform(exData)
plotEx <- plotDist(exData,ansExData,"HESC_R1")
plotEx[[1]]
```

---

preprocessData *Wrapper function to preprocess the data*

Description

This is a wrapper function that calls the functions to preprocess the data. It results in a RangedSummarizedExperiment-class object normalized counts and meta data that can be used by DBAnalyze.

Usage

```
preprocessData(
  inputRegions,
  sampleInfoFile,
  sampleDir = ".",
  inputCountData,
  numClusters = 4,
  noNeg = TRUE,
  plotDataToPDF = FALSE,
  ...
)
```

Arguments

inputRegions	A bed file the regions to analyze.
sampleInfoFile	A tab separated file all sample information. The following are the columns that are present in the file. * Sample Name : Names for the samples. * Group : The group the sample belongs. * IP : The name of the sample bam file. * IP_aligned_reads : The number of aligned reads in the sample. This is used in depth normalization process. * IN : The name of the sample's control bam file. * IN_aligned_reads : The number of aligned reads in the control file. This is used in depth normalization process.

sampleDir	Location of the input sample files in 'sampleInfoFile' file. (default: ".") Name,Group/Label,IP bam location,IP number of reads,IN bam location, IN number of reads).
inputCountData	The path to the file count data. This parameter is used when directly loading count data from a file. This should be a tab separated file sample names as header.
numClusters	A numerical parameter indicating the number of clusters to use in the normalization step. Passed on to normalizeData. (default: 4)
noNeg	A logical parameter indicating how to deal negative values. It is passed to ansTransform. (default: TRUE)
plotDataToPDF	A logical parameter indicating whether to make plots of the data distribution to a separate PDF file for each sample. It is passed on to passed to ansTransform. (default: FALSE)
...	Additional arguments passed on to getRegionCounts.

Value

RangedSummarizedExperiment-class containing the normalized counts, cluster information, the variance of the cluster in the sample and metadata.

See Also

getRegionCounts, ansTransform and normalizeData which this function calls

Examples

```
processedData <- preprocessData(system.file("extdata", "chr19_regions.bed",
package="CSSQ"),system.file("extdata", "sample_info.txt", package="CSSQ"),
sampleDir = system.file("extdata", package="CSSQ"),
numClusters=4,noNeg=TRUE,plotDataToPDF=FALSE)
processedData
```

Index