Package ‘ChIPseqSpikeInFree’

May 24, 2019

Title A Spike-in Free ChIP-
Seq Normalization Approach for Detecting Global Changes in Histone Modifications

Version 1.2.2

Date 2019-05-22

Description The detection of global histone modification changes can be addressed using exogenous reference Spike-in controls. However, many ChIP-seq data sets available in public depositories nowadays were done without including Spike-in procedure. In order to do quantitative comparisons between these data, researchers have to regenerate whole data set using spike-in ChIP-seq protocols – this is an infeasible solution sometime. A basic scaling factor calculation for these scenarios remains a challenge. We present ChIPseqSpikeInFree , a novel ChIP-seq normalization method to effectively determine scaling factors for samples across different conditions or treatments, which doesn't rely on exogenous spike-in chromatin or peak detection to reveal global changes in histone modification occupancy. It can reveal similar magnitude of global changes compared to spike-In method.

Depends GenomicAlignments,
GenomicRanges,
IRanges,
R (>= 3.3.0),
Rsamtools

License GPL-3

Encoding UTF-8

LazyData TRUE

RoxygenNote 6.1.1

R topics documented:

BoxplotSF 1
CalculateSF 2
ChIPseqSpikeInFree 3
CountRawReads 4
GenerateBins 5
ParseReadCounts 5
ReadMeta 6

`BoxplotSF`	*This function generates boxplot using sacaling factor table. It's been included in the last step of ChIPseqSpikeInFree().*

Description

This function generates boxplot using sacaling factor table. It's been included in the last step of ChIPseqSpikeInFree().

Usage

```
BoxplotSF(input, prefix = "test")
```

Arguments

`input`	a file/data.frame generated/returned by CalculateSF() or ChIPseqSpikeInFree(). It looks like metadata file but has extra columns SF and COLOR.
`prefix`	prefix of output filename.

Value

A filename of generated boxplot

Examples

```
## 1. re-generate boxplot of ChIPseqSpikeInFree scaling factors
## After you run ChIPseqSipkeFree(), a sacaling factor table
## (for example, test_SF.txt) will be generated.

# BoxplotSF(input="test_SF.txt",prefix="test")
```

`CalculateSF`	*calculate scaling factors and save results*

Description

This function allows you to plot curves, caculate SF(scaling factor) per antibody based on parsedMatrix. In addtion return a data.frame of updated metadata. If you run ChIPseqSpikeInFree() seperately for two batches, the scaling factors will be not comparable between two batches. The correct way is to combine bamFiles parameter and create a new metadata file to include all bam files. Then re-run ChIPseqSpikeInFree().

Usage

```
CalculateSF(data, metaFile = "sample_meta.txt", prefix = "test")
```

Arguments

`data`	a data.frame generated by function ParseReadCounts() or a file name of parsed matrix
`metaFile`	a data.frame of metadata by ReadMeta(); or a filename of metadata file.
`prefix`	prefix of output filename to save the scaling factor values.

Value

A data.frame of the updated metaFile with scaling factors

Examples

```
## 1. start from a parsedMatrix file

# parsedMatrixFile <- "your/path/test_parsedMatrix.txt"
# metaFile <- "your/path/sample_meta.txt"
# parsedDF <- read.table(parsedMatrixFile, sep="\t",header=TRUE,fill=TRUE,
# quote="",row.names=NULL ,check.names=F)
# res <- CalculateSF (data=parsedDF,metaFile=metaFile, prefix="your/path/test")

## 2. start from a rawCount file

# metaFile <- "your/path/sample_meta.txt"
# parsedDF <- ParseReadCounts(data="your/path/test_rawCounts.txt", metaFile=metaFile,
#     prefix="your/path/test_parsedMatrix.txt")
# res <- CalculateSF (data=parsedDF,metaFile=metaFile,
#     prefix="your/path/test")
```

`ChIPseqSpikeInFree` *wrapper function - perform ChIP-seq spike-free normalization in one step.*

Description

This function wraps all steps. If you run ChIPseqSpikeInFree() seperately for two batches, the scaling factors will be not comparable between two batches. The correct way is to combine bamFiles parameter and create a new metadata file to include all bam files. Then re-run ChIPseqSpikeInFree().

Usage

```
ChIPseqSpikeInFree(bamFiles, chromFile = "hg19",
  metaFile = "sample_meta.txt", prefix = "test", binSize = 1000,
  ncores = 2)
```

Arguments

`bamFiles`	a vector of bam filenames.
`chromFile`	chrom.size file. Given "hg19","mm10","mm9" or "hg38", will load chrom.size file from package folder.

`metaFile`	a filename of metadata file. the file must have three columns: ID (bam filename without full path), ANTIBODY and GROUP
`prefix`	prefix of output filename.
`binSize`	size of bins (bp). Recommend a value bwteen 200 and 10000
`ncores`	number of cores for parallel computing.

Value

A data.frame of the updated metaFile with scaling factor

Examples

```
## 1 first You need to generate a sample_meta.txt (tab-delimited txt file).
# metaFile <- "your/path/sample_meta.txt"
# meta <- ReadMeta(metaFile)
# head(meta)
# ID ANTIBODY GROUP
# ChIPseq1.bam H3K27me3 WT
# ChIPseq2.bam H3K27me3 K27M

## 2. bam files
# bams <- c("ChIPseq1.bam","ChIPseq2.bam")
# prefix <- "test"

## 3. run ChIPseqSpikeInFree pipeline
# ChIPseqSpikeInFree(bamFiles=bams, chromFile="mm9",metaFile=metaFile,prefix="test")
```

`CountRawReads` *count raw reads for all bins*

Description

This function counts raw reads for each bin.

Usage

```
CountRawReads(bamFiles, chromFile = "hg19", prefix = "test",
  singleEnd = TRUE, binSize = 1000)
```

Arguments

`bamFiles`	a vector of bam filenames.
`chromFile`	a chrom.size file. Given "hg19","mm10","mm9" or "hg38", will load chrom.size file from package folder. Otherwise, give a /your/path/chrom.size
`prefix`	prefix of output file name
`singleEnd`	To count paired-end reads, set argument singleEnd=FALSE
`binSize`	size of bins (bp). Recommend a value bwteen 200 and 10000

Value

a data.frame of raw counts for each bin

Examples

```
## 1.count reads using mm9 bams
# bams <- c("your/path/ChIPseq1.bam","your/path/ChIPseq2.bam")
# rawCountDF <- CountRawReads(bamFiles=bams,chromFile="mm9",prefix="your/path/test",singl
```

`GenerateBins` *generate genome-wide bins for counting purpose*

Description

Given a chrom.size file, this function allows you to generate a your own sliding windows (bins).

Usage

```
GenerateBins(chromFile, binSize = 1000, overlap = 0, withChr = TRUE)
```

Arguments

`chromFile`	chrom.size. Given "hg19","mm10","mm9" or "hg38", will load chrom.size file from package folder.
`binSize`	size of bins (bp)
`overlap`	overlaps between two consecutive bins (bp)
`withChr`	chromosome names in bin File have chr if set withChr to TRUE; FALSE - no chr

Value

A data.frame of generated bins

Examples

```
## 1. generate a mm10 binFile without chr and use a binSize of 1000 bp
## and overlap of 500 bp between two consecutive bins

## "mm10" will be parsed as system.file("extdata", "mm10.chrom.sizes",
##       package = "ChIPseqSpikeInFree")
# binDF <- GenerateBins(chromFile="mm10",binSize=1000, overlap=500,
# withChr=FALSE,prefix="mm10")

## 2. generate a hg19 binFile with chr and use a binSize of 2000 bp

## "hg19" will be parsed as system.file("extdata", "hg19.chrom.sizes",
##       package = "ChIPseqSpikeInFree")
# binDF<- GenerateBins(chromFile="hg19",binSize=2000, overlap=0,
# withChr=TRUE,prefix="hg19")
```

`ParseReadCounts` *parse readCounts matrix*

Description

This function allows you to parse rawCount table (generated by CountRawReads() function) to a parsedMatrix of (cutoff, and percent of reads accumulatively passed the cutoff in each sample).

Usage

```
ParseReadCounts(data, metaFile = "sample_meta.txt", by = 0.05,
  prefix = "test", binSize = 1000, ncores = 2)
```

Arguments

`data`	a data.frame returned by readRawCounts() or a file name of rawCount table
`metaFile`	a data.frame of metadata by ReadMeta(); or a filename of metadata file.
`by`	step used to define cutoffs; ParseReadCounts will cumulatively calculate the percent of reads that pass the every cutoff.
`prefix`	prefix of output filename to save the parsedMatrix of (cutoff, and percent of reads accumulatively passed the cutoff in each sample).
`binSize`	size of bins (bp). Recommend a value bwteen 200 and 10000
`ncores`	number of cores for parallel computing.

Value

A data.frame of parsed data.

Examples

```
## prerequisite step 1. count raw reads
## (if your bam files were aligned to mm9 genome with chr in reference chromosomes).
# bams <- c("your/path/ChIPseq1.bam","your/path/ChIPseq2.bam")
# rawCountDF <- CountRawReads(bamFiles=bams,chromFile="mm9",prefix="your/path/test")
## output file will be "your/path/test_rawCount.txt"
# head(rawCountDF,n=2)

# bin   ChIPseq1.bam    ChIPseq2.bam
# chr1:1-1000  0   0
# chr1:1001-2000   0   0

## prerequisite step 2: generate your sample_meta.txt.
##  A tab-delimited txt file has three required columns
# ID    ANTIBODY    GROUP
# ChIPseq1.bam  H3K27me3    WT
# ChIPseq2.bam  H3K27me3    K27M

## 1.parse readCount table using this function.
# metaFile <- "your/path/sample_meta.txt"
# dat <- ParseReadCounts(data="your/path/test_rawCount.txt",
# metaFile=metaFile, prefix="your/path/test")
## output file will be "your/path/test_parsedMatrix.txt"
```

`ReadMeta` *read in sample metadata file*

Description

This function allows you to load metadat to a R data.frame and return the object. In addtion, it validates meta_info format and adds a COLOR column if it's undefined.

Usage

```
ReadMeta(metaFile = "sample_meta.txt")
```

Arguments

`metaFile`	a metadata file name; the file must have three columns: ID (bam filename without full path), ANTIBODY and GROUP. the COLOR column is optional and will be used for plotting purpose.

Value

A data.frame of metaFile

Examples

```
## 1. load an example of metadata file

metaFile <- system.file("extdata", "sample_meta.txt", package = "ChIPseqSpikeInFree")
meta <- ReadMeta(metaFile)
head(meta, n = 1)
meta
#                                              ID ANTIBODY GROUP COLOR
#  H3K27me3-NSH.K27M.A.bam H3K27me3-NSH.K27M.A.bam H3K27me3  K27M   green
#  H3K27me3-NSH.K27M.B.bam H3K27me3-NSH.K27M.B.bam H3K27me3  K27M   green
#  H3K27me3-NSH.K27M.C.bam H3K27me3-NSH.K27M.C.bam H3K27me3  K27M   green
#  H3K27me3-NSH.WT.D.bam     H3K27me3-NSH.WT.D.bam H3K27me3    WT  grey
#  H3K27me3-NSH.WT.E.bam     H3K27me3-NSH.WT.E.bam H3K27me3    WT  grey
#  H3K27me3-NSH.WT.F.bam     H3K27me3-NSH.WT.F.bam H3K27me3    WT  grey
```

Index

BoxplotSF, 1

CalculateSF, 2
ChIPseqSpikeInFree, 3
CountRawReads, 4

GenerateBins, 5

ParseReadCounts, 5

ReadMeta, 6