Package 'CSSP'

October 16, 2019

Type Package

Title ChIP-Seq Statistical Power

Version 1.22.0

Date 2015-02-07

Author Chandler Zuo, Sunduz Keles

Maintainer Chandler Zuo<zuo@stat.wisc.edu>

Description Power computation for ChIP-Seq data based on Bayesian estimation for local poisson counting process.

License GPL-2

Imports methods, splines, stats, utils

Suggests testthat

biocViews ChIPSeq, Sequencing, QualityControl, Bayesian

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

git_branch RELEASE_3_9

git_last_commit 42f99dc

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

R topics documented:

`bin.data` *An artificially constructed BinData-class class object.*

Description

This data set contains a typical example for a BinData class object,.

Format

a BinData-class class object.

Author(s)

Chandler Zuo zuo@stat.wisc.edu

`BinData-class` *An S-4 class containing the model fit information for a CSSP model.*

Description

chrID The chromosome ID.

coord The genome coordinates for the starting positions of each bin.

tagCount The number of ChIP reads mapped to each bin.

mappability The mappability score of each bin.

gcContent The gc-content score of each bin.

input The number of input reads mapped to each bin.

dataType Either "unique" or "multi'.

`bindata.chr1` *An artificially constructed data.frame object that can be used by cssp.fit function.*

Description

This data set contains a typical example for a data.frame object that can be imported by cssp.fit.

Format

a data.frame class object.

Author(s)

Chandler Zuo zuo@stat.wisc.edu

`bindcount` *Compute the number of reads overlapping the specified positions for the whole genome.*

Description

Compute the number of reads overlapping the specified positions for the whole genome.

Usage

```
bindcount(chipdat, inputdat, bindpos, fragL = 200, whs = 250)
```

Arguments

`chipdat`	A list of the starting coordinates for aligned reads for all chromosomes, with positive numbers representing the 5' strand and negative numbers representing the 3' strand.
`inputdat`	A list of the starting coordinates for aligned reads for the input sample for all chromosomes, with positive numbers representing the 5' strand and negative numbers representing the 3' strand.
`bindpos`	A list of genome coordinates for each chromosome whose numbers of covering tags are computed.
`fragL`	A numeric value for the fragment length of the aligned reads. Default: 200.
`whs`	A numeric value for the half window size around the binding position. All tags overlapping this region are counted. Default: 250.

Value

A list of the number of overlapping tags for all position. Each list is a data.frame corresponding to a single chromosome, containing:

chip	The number of ChIP sample reads overlapping each position.
input	The number of input sample reads overlapping each position.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( tagdat_input )
data( tagdat_chip )
data( bindpos )
bindcount( tagdat_chip, tagdat_input, bindpos, fragL = 100, whs = 300 )
```

`bindcount.chr` *Compute the number of reads overlapping the specified positions for a single chromosome.*

Description

Compute the number of reads overlapping the specified positions for a single chromosome.

Usage

```
bindcount.chr(tagdat, bindpos, fragL = 200, whs = 250)
```

Arguments

`tagdat`	A numeric vector of the genome coordinates for the starting positions of the aligned reads, with positive numbers representing the 5' strand and negative numbers representing the 3' strand.
`bindpos`	A numeric vector of the genome coordinates whose numbers of covering tags are computed.
`fragL`	A numeric value for the fragment length of the sequencing reads. Default: 200.
`whs`	A numeric value for the half window size around the binding position. All tags overlapping this region are counted. Default: 250.

Value

A numeric vector of the numbers of reads overlapping each position corresponding to "bindpos".

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( tagdat_chip )
data( bindpos )
bindcount.chr( tagdat_chip[[1]], bindpos[[1]], fragL = 100, whs = 300 )
```

`bindpos` *An artificially constructed dataset containing enrichment positions on 5 chromosomes.*

Description

This data set contains artificially generated nucleotide-leve enrichment positions on a genome of 5 chromosomes.

Usage

```
example
```

Format

A list containing the genome coordinates for enrichment sites on each of the 5 chromosomes.

Author(s)

Chandler Zuo zuo@stat.wisc.edu

`callpeak` *Call enriched bins based on the CSSP model.*

Description

Call enriched bins based on the CSSP model.

Usage

```
callpeak(fit, chip, fold = 1.8, min.count = 0, qval = 0.05, method = "",
  depth = fit@lambday)

## S4 method for signature 'CSSPFit'
callpeak(fit, chip, fold = 1.8, min.count = 0,
  qval = 0.05, method = "", depth = fit@lambday)
```

Arguments

`fit`	A CSSPFit-class object containing the fitted CSSP model.
`chip`	A numeric vector containing the bin counts for the ChIP sample.
`fold`	A numeric value for the fold change threshold for peak calling.
`min.count`	A numeric value for the minimum ChIP count threshold for peak calling.
`qval`	A numeric value for the false-discovery rate to be controlled. Default: 0.05.
`method`	A character value. By default, "min.count" is used to threshold the ChIP bin counts. If 'method=="post"', "min.count" is used to threshold the posterior bin-level poisson intensities.
`depth`	A numeric value for the sequencing depth corresponding to the ChIP sample of the "chip" argument. If not provided, sequencing depth of "fit" is used.

Value

A numeric vector of locations for binding bins.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( sampleFit )
data( bin.data )
callpeak( sampleFit, chip = bin.data@tagCount, fold = 1, min.count = 0 )
```

`createBinData` *Create a BinData object by merging lists of ChIP and input bin data with external M and GC text files.*

Description

This function create a BinData object by merging ChIP and input bin-level counts with external M/GC/N text files.

Usage

```
createBinData(dat.chip, dat.input, mfile, gcfile, nfile, m.suffix = NULL,
  gc.suffix = NULL, n.suffix = NULL, chrlist = NULL,
  dataType = "unique")
```

Arguments

`dat.chip`	Either a list of the ChIP bin level data for each chromosome, or a character string of the file name including the ChIP bin level data. If the ChIP bin level file name is provided, the file must contain at least two columns, where the chromosome information is in the first column, and the bin level counts are in the last column.
`dat.input`	A list of the input bin level data for each chromosome, or a character string for the input bin level data counts. The structure is the same as "dat.chip".
`mfile`	A character value. If "m.suffix=NULL", this is the file name of the genome-wide M file. Otherwise, this is the common prefix (including relative path) for all chromosome-level M files.
`gcfile`	A character value. If "gc.suffix=NULL", this is the file name of the genome-wide GC file. Otherwise, this is the common prefix (including relative path) for all chromosome-level GC files.
`nfile`	A character value. If "n.suffix=NULL", this is the file name of the genome-wide N file. Otherwise, this is the common prefix (including relative path) for all chromosome-level N files.
`m.suffix`	A character value. If not NULL, this is the suffix of the chromosome-wise M files. The chromosome-level file has to be named "chrX_m.suffix".
`gc.suffix`	A character value. If not NULL, this is the suffix of the chromosome-wise GC files. The chromosome-level file has to be named "chrX_gc.suffix".

n.suffix	A character value. If not NULL, this is the suffix of the chromosome-wise N files. The chromosome-level file has to be named "chrX_n.suffix".
chrlist	A list of the chromosomes that is imported. If "NULL", all chromosomes specified by "name(dat.chip)" are imported.
dataType	A character value of either "unique" or "multi".

Value

A BinData-class object.

Note

When .suffix is null, the corresponding genome-wise file must have three columns, with the first column being the chromosome names, the second column being the genome coordinates, and the third column being the corresponding scores. In contrast, when .suffix is not null, then each chromosome-level M/GC/N file should only contain two columns, with the first column being the genome coordinates and the second column being the scores.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data(tagdat_chip)
data(tagdat_input)
dat_chip <- tag2bin(tagdat_chip,binS=100,fragL=100)
dat_input <- tag2bin(tagdat_input,binS=100,fragL=100)

numBins <- as.integer(runif(5,190,220))
mapdat <- gcdat <- ndat <- list(1:5)
allmapdat <- allgcdat <- allndat <- NULL
for(i in 1:5){
  mapdat[[i]] <- data.frame(
                            pos=(0:(numBins[i]-1))*100,
                            M=runif(numBins[i],0.9,1)
                            )
  gcdat[[i]] <- data.frame(
                           pos=(0:(numBins[i]-1))*100,
                           GC=runif(numBins[i],0.5,1)
                           )
  ndat[[i]] <- data.frame(
                          pos=(0:(numBins[i]-1))*100,
                          N=rbinom(numBins[i],1,0.01)
                          )
  allmapdat <- rbind(allmapdat,
                     cbind(paste("chr",i,sep=""),mapdat[[i]]))
  allgcdat <- rbind(allgcdat,
                    cbind(paste("chr",i,sep=""),gcdat[[i]]))
  allndat <- rbind(allndat,
                   cbind(paste("chr",i,sep=""),ndat[[i]]))

  write.table( mapdat[[i]], file = paste("map_chr",i,".txt",sep=""),
              sep = "\t", row.names = FALSE, col.names = FALSE)
  write.table( gcdat[[i]], file = paste("gc_chr",i,".txt",sep=""),
```

```
              sep = "\t", row.names = FALSE, col.names = FALSE)
  write.table( ndat[[i]], file = paste("n_chr",i,".txt",sep=""),
              sep = "\t", row.names = FALSE, col.names = FALSE)
}
write.table( allmapdat, file = "allmap.txt" , sep = "\t", row.names = FALSE,
            col.names = FALSE )
write.table( allgcdat,file = "allgc.txt" , sep = "\t", row.names = FALSE,
            col.names = FALSE )
write.table( allndat,file = "alln.txt", sep = "\t", row.names = FALSE,
            col.names = FALSE )

bindata1 <- createBinData( dat_chip, dat_input, mfile = "map_",
                           gcfile = "gc_", nfile = "n_", m.suffix = ".txt",
                           gc.suffix = ".txt", n.suffix = ".txt",
                           chrlist = NULL, dataType = "unique" )
bindata2 <- createBinData( dat_chip, dat_input, mfile = "allmap.txt",
                           gcfile="gc_", nfile = "n_", m.suffix = NULL,
                           gc.suffix = ".txt", n.suffix = ".txt",
                           chrlist = NULL, dataType = "unique" )
bindata3 <- createBinData( dat_chip, dat_input, mfile = "map_",
                           gcfile = "allgc.txt", nfile="n_", m.suffix = ".txt",
                           gc.suffix = NULL, n.suffix = ".txt",
                           chrlist = NULL, dataType = "unique")
bindata4 <- createBinData( dat_chip, dat_input, mfile = "map_",
                           gcfile = "gc_", nfile = "alln.txt", m.suffix = ".txt",
                           gc.suffix = ".txt", n.suffix = NULL,
                           chrlist = NULL, dataType = "unique")

for(i in 1:5){
  for(j in c("map_","gc_","n_")){
    file.remove(paste(j,"chr",i,".txt",sep=""))
  }
}
file.remove("allmap.txt")
file.remove("alln.txt")
file.remove("allgc.txt")
```

`cssp.fit` *Fit the CSSP Model.*

Description

Fit the CSSP Model.

Usage

```
cssp.fit(dat, method = "mde", p1 = 0.5, p2 = 0.99, beta.init = NULL,
  e0.init = 0.9, e0.lb = 0.5, ngc = 9, nite = 50, tol = 0.01,
  useGrid = FALSE, nsize = NULL, ncomp = 2, nonpa = FALSE,
  zeroinfl = FALSE, seed = NULL)

## S4 method for signature 'data.frame'
cssp.fit(dat, method = "mde", p1 = 0.5, p2 = 0.99,
  beta.init = NULL, e0.init = 0.9, e0.lb = 0.5, ngc = 9, nite = 50,
```

```
  tol = 0.01, useGrid = FALSE, nsize = NULL, ncomp = 2, nonpa = FALSE,
  zeroinfl = FALSE, seed = NULL)

## S4 method for signature 'BinData'
cssp.fit(dat, method = "mde", p1 = 0.5, p2 = 0.99,
  beta.init = NULL, e0.init = 0.9, e0.lb = 0.5, ngc = 9, nite = 50,
  tol = 0.01, useGrid = FALSE, nsize = NULL, ncomp = 2, nonpa = FALSE,
  zeroinfl = FALSE, seed = NULL)
```

Arguments

dat	A data.frame or BinData-class object containing bin-level chip, input, M and GC information. For the data.frame object, the columns must contain "chip", "input", "M". For BinData object, the slots must contain "tagCount", "input", "M". If "GC" is not provided, model will be fitted without using gc-Content scores.
method	A character indicating the method of fitting algorithm to be used. "mde" (Default) - minimum distance estimation; "gem" - the generalized EM method.
p1	The numeric value for the lower bound for the p-value region where the p-values are assumed to be uniformly distributed. Default: 0.5.
p2	The numeric value for the upper bound for the p-value region where the p-values are assumed to be uniformly distributed. Default: 0.99.
beta.init	The numeric value for the initializing the size parameter for the background model of the ChIP sample. If "NULL", the size parameter of the fitted input sample model is used.
e0.init	The numeric value for initializing parameter e0. Default: 0.9.
e0.lb	The numeric value for the lower bound of parameter e0. Default is 0.5. This parameter is recommended to be set according to the p-value plot.
ngc	An integer value for the number of knots used in the spline model for the gc covariate. Default: 9.
nite	An integer value for the maximum number of iterations taken. Default: 50.
tol	A numeric value for the tolerance for convergence. Default: 1e-3.
useGrid	A logical value indicating whether the gridding method is used. If TRUE, the covariate space is grided adaptively. This trims down the sample size for fitting the regression model when the data contains too many observations, and is suggested for genome-wide analysis. Default: FALSE.
nsize	A numeric value for the number of bins to be randomly chosen in estimating the normalizatiing parameters. If Null (default), all bins are used in normalization. For genome wide analysis, nsize=5000 is suggested.
ncomp	A numeric value for the number of signal components.
nonpa	A logical value indicating whether a nonparametric model for the background ChIP sample and the input sample is fitted.
zeroinfl	A logical value indicating whether a zero-inflated negative binomial model is fitted for the ChIP background.
seed	A numeric value for the seed of generating random variables. Default: NULL. Users should specify this value for generating exactly reproducible results.

Details

The current version of cssp.fit has implemented the following method.
The "method" argument specifies the method to estimate the normalization models for the ChIP background from the input data. "mde" uses minimum distance estimation, "gem" uses generalized E-M estimation.
The 'nonpa' argument specifies whether a glm model is used. If "nonpa" is FALSE, a GLM is used to fit the input data. If "nonpa" is TRUE, the mean response within each grid is taken as the predict. These two arguments enables the analysis for genome-wide data. In this case, "nsize" grids are used.
If "nonpa" is FALSE, then "useGrid" specifies whether the covariate space is grided adaptively, and the mean values within each grid is used for regression.
If "nonpa" is TRUE, "zeroinfl" specifies whether a zero-inflation model for the background is used. This is useful for low-depth ChIP data, where too many bins have zero count.

Value

CSSPFit-class A CSSPFit object.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( bin.data )
cssp.fit( bin.data )
cssp.fit( bin.data, method = "gem" )
data( bindata.chr1 )
cssp.fit( bindata.chr1 )
cssp.fit( bindata.chr1, method = "gem", ngc = 1 )
```

---

cssp.power *Compute the weighted average of bin-wise power conditioning on the fold change and minimal ChIP count requirements.*

Description

Compute the weighted average of bin-wise power conditioning on the fold change and minimal ChIP count requirements.

Usage

```
cssp.power(fit, x, ite = 100, fold = 1, min.count = 10, useC = FALSE,
  qval = 0.05)

## S4 method for signature 'CSSPFit'
cssp.power(fit, x, ite = 100, fold = 1,
  min.count = 10, useC = FALSE, qval = 0.05)
```

Arguments

fit	A CSSPFit-class object for the CSSP model.
x	A numeric value for the sequencing depth of the ChIP sample at which the power is evaluated.
ite	A integer value for the number of iterations used for Monte-Carlo evaluation.
fold	A numeric value for the fold change threshold.
min.count	A numeric value for the minimal count threshold.
useC	A logical value. Whether the function will be evaluated using C. Default: FALSE.
qval	A numeric value for the q-value for FDR control. Default: 0.05.

Value

A numeric value for the weighted average of bin power conditioning on the minimal count and fold change thresholds.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( sampleFit )
cssp.power( sampleFit, x = sampleFit@lambday*0.1, min.count = 0, fold = 2,
useC = TRUE )
```

---

cssp.sim *Simulate bin binding intensities according to the posterior distributions of the fitted CSSP model.*

Description

Simulate bin binding intensities according to the posterior distributions of the fitted CSSP model.

Usage

```
cssp.sim(fit, x = fit@lambday)

## S4 method for signature 'CSSPFit'
cssp.sim(fit, x = fit@lambday)
```

Arguments

fit	A CSSPFit-class class object describing the CSSP model.
x	A numeric value for the sequencing depth of the ChIP sample at which the new binding intensities at simulated.

Value

A list object containing

chip	A numeric vector for the binding intensities for the ChIP sample.
bind	A numeric vector for the simulated binding regions.
bind.sig	A numeric vector for the signal component for each bin.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( sampleFit )
cssp.sim( fit = sampleFit, x = sampleFit@lambday*0.1 )
```

`CSSPFit-class` *An S-4 class containing the model fit information for CSSP model.*

Description

lambdax Sequencing depth of the input sample.

lambday Sequencing depth of the ChIP sample.

e0 The normalization parameter for the ChIP sample.

pi0 The pi_0 parameter of CSSP model, denoting the proportion of bins that are enriched.

mu.chip The vector of the estimated hyper means for the background model of the ChIP sample.

mu.input The vector of the estimated hyper means for the input sample.

mean.sig The vector of the hyper means for each signal component.

size.sig The vector of the size parameters for each signal component.

a The size parameter of the input sample model.

b The size parameter of the background model for the ChIP sample.

p.sig The vector of the proportions of enrichment as each signal component across all enrichment bins.

prob.zero The vector of the prior inflated probability at 0.

post.p.sig The matrix for the posterior probability of each bin being enriched as a signal component conditioning on the event that the bin is enriched. Each column corresponds to one signal component.

post.p.bind Posterior probability of each bin being enriched.

post.p.zero Posterior probability of the inflated probability at 0.

post.shape.sig The matrix for the shape parameters for the posterior gamma distributions of bin level poisson parameters, conditioning on the event that the bins are enriched as each signal component. Each column corresponds to one signal component.

post.scale.sig The matrix for the scale parameters of the posterior gamma distributions of bin level poisson parameters, conditioning on the event that the bins are enriched as each signal component. Each column corresponds to one signal component.

post.shape.back The shape parameters for the posterior gamma distributions of bin level poisson parameters, conditioning on each bin being enriched.

post.scale.back The scale parameters for the posterior gamma distributions of bin level poisson parameters, conditioning on each bin being unenriched.

n The number of mappable bins that are fitted by the model.

k The number of signal components.

map.id The indices for the mappable bins that are fitted by the model.

pvalue The continuously corrected p-values for a subset of ChIP sample bin counts against the background model.

cum.pval The cumulative distribution for p-values for a subset of ChIP sample bin counts against the background model.

Examples

```
showClass("CSSPFit")
```

`fit.freq` *Compute the estimated frequency for ChIP counts based on the CSSP model.*

Description

Compute the estimated frequency for ChIP counts based on the CSSP model.

Usage

```
fit.freq(fit, chip)

## S4 method for signature 'CSSPFit'
fit.freq(fit, chip)
```

Arguments

`fit`	A CSSPFit-class object for the fitted CSSP model.
`chip`	A numeric vector of ChIP sample bin counts.

Value

A data.frame object containing

count The counts of each bin.\ freq The ChIP data frequency at this count value.\ freq.est The estimated frequency using th

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( sampleFit )
data( bin.data )
fit.freq( sampleFit, chip = bin.data@tagCount )
```

pBBT *Compute the cumulative probability of the bin-level poisson parameters.*

Description

Compute the cumulative probability of the bin-level poisson parameters.

Usage

```
pBBT(fit, x, depth = fit@lambday, lower = TRUE)

## S4 method for signature 'CSSPFit'
pBBT(fit, x, depth = fit@lambday, lower = TRUE)
```

Arguments

fit	A CSSPFit-class object for the CSSP model.
x	A numeric value for the percentile level of bin-level poisson parameters.
depth	A numeric value for the sequencing depth at which the probability is estimated.
lower	A logical value. If TRUE, the lower quantile is computed. If FALSE (Default), the upper quantile is computed.

Value

A numeric value for the cumulative distribution of bin-level poisson parameters.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( sampleFit )
pBBT( sampleFit, x = 10 )
```

peakcount *Compute the number of aligned reads overlapping the specified peak intervals for the whole genome.*

Description

Compute the number of aligned reads overlapping the specified peak intervals for the whole genome.

Usage

```
peakcount(chipdat, inputdat, peakpos, fragL = 200, unique = FALSE)
```

Arguments

chipdat	A list of the starting positions of the ChIP sample aligned reads for each chromosome. The sign of each coordinate represents its strand direction, with a positive numbers on the 5' strand and a negative numbers on the 3' strand.
inputdat	A list of the starting positions of the input sample aligned reads for each chromosome. The sign of each coordinate represents its strand direction, with a positive numbers on the 5' strand and a negative numbers on the 3' strand.
peakpos	A list containing the genome coordinates for each peak interval on each chromosome. Each list component is a 2-column matrix containing the left and right boundary of the peak intervals on one chromosome.
fragL	A numeric value of the fragment length of the aligned reads. Default: 200.
unique	A logical value for whether only reads mapping to unique nucleotide positions are counted.

Value

A list of the numbers of reads that overlap the corresponding peak intervals.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( peakpos )
data( tagdat_input )
data( tagdat_chip )
peakcount( tagdat_chip, tagdat_input, peakpos, fragL = 100 )
```

---

peakcount.chr *Compute the number of aligned reads overlapping peaks for one chromosome.*

Description

Compute the number of aligned reads overlapping peaks for one chromosome.

Usage

```
peakcount.chr(tagdat, peakpos, fragL = 200, unique = FALSE)
```

Arguments

tagdat	A numeric vector of the genome coordinates for the starting positions of aligned reads. The signs of coordinates represent their strand direction, with positive numbers representing the 5' strand and negative numbers representing the 3' strand.
peakpos	A 2-column matrix matrix containing the left and right position of the peaks for one chromosome.
fragL	A numeric value for the fragment length of the sequencing reads. Default: 200.
unique	A logical value for whether only reads mapping to unique nucleotide positions are counted.

Value

A numeric vector of the number of overlapping tags for all peaks.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( peakpos )
data( tagdat_input )
peakcount.chr( tagdat_input[[1]], peakpos[[1]], fragL = 100 )
```

---

peakpos *An artificially generated dataset containing peak invervals on 5 chromosomes.*

Description

This data set contains the genome coordinates of artificially generated peak intervals on a genome of 5 chromosomes.

Format

a list of 2-column matrices. Each matrix contains the coordinates of the peak intervals for one chromosome.

Author(s)

Chandler Zuo zuo@stat.wisc.edu

---

qBBT *Compute the quantile estimate for the bin-level poisson parameters.*

Description

Compute the quantile estimate for the bin-level poisson parameters.

Usage

```
qBBT(fit, prob, depth = fit@lambday, lower = FALSE)

## S4 method for signature 'CSSPFit'
qBBT(fit, prob, depth = fit@lambday, lower = FALSE)
```

Arguments

fit	A CSSPFit-class object for the CSSP model.
prob	A numeric value for the percentile level of bin-level poisson parameters.
depth	A numeric value for the sequencing depth at which the quantile is evaluated.
lower	A logical value. If TRUE, the lower quantile is computed. If FALSE (Default), the upper quantile is computed.

Value

A numeric value for the percentile of bin-level poisson parameters.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( sampleFit )
qBBT( sampleFit, prob = 0.99, depth = sampleFit@lambday*0.1 )
```

---

readBinFile *Read the bin-level text files containing ChIP and input sample counts as well as M and GC scores.*

Description

Read the bin-level text files containing ChIP and input sample counts as well as M and GC scores.

Usage

```
readBinFile(type = c("chip", "input", "M", "GC"), fileName)
```

Arguments

type	A character vector indicating data types to be imported. This vector can contain "chip" (ChIP data), "input" (input data), "M" (mappability score), "GC" (GC content score). Default: c("chip","input","M","GC").
fileName	A character vector of file names, each of which matches each element of "type". "type" and "fileName". This vector should have the same length with "type" and corresponding elements in two vectors should appear in the same order.

Value

A data.frame of the processed bin files, containing ChIP, input, M and GC in different columns.

Note

"chip","input" and "M" files are all mandatory. "GC" file is optional.

Author(s)

Chandler Zuo<zuo@stat.wisc.edu>

Examples

```
data( bindata.chr1 )
pwd <- getwd()
local({
setwd( tempdir() )
on.exit( setwd( pwd ) )
write.table( bindata.chr1[,c(1,4)], file = "chr1_map.txt", sep = "\t",
row.names = FALSE, col.names = FALSE )
write.table( bindata.chr1[,c(1,5)], file = "chr1_gc.txt", sep = "\t",
row.names = FALSE, col.names = FALSE )
write.table( bindata.chr1[,c(1,2)], file = "chr1_chip.txt", sep = "\t",
row.names = FALSE, col.names = FALSE )
write.table( bindata.chr1[,c(1,3)], file = "chr1_input.txt", sep = "\t",
row.names = FALSE, col.names = FALSE )
readBinFile( fileName = c("chr1_chip.txt", "chr1_input.txt", "chr1_map.txt",
"chr1_gc.txt" ) )
file.remove( paste( "chr1_", c( "chip", "input", "map", "gc" ), ".txt", sep = "" ) )
})
```

`sampleFit` *A "CSSPFit" class object containing the fitted CSSP model for bin.data.*

Description

A CSSPFit-class class object constructed by fitting CSSP model on bin.data.

Format

a CSSPFit-class class object.

Author(s)

Chandler Zuo zuo@stat.wisc.edu

`tag2bin` *Convert the genome coordinates of aligned reads to bin-level counts for all chromosomes.*

Description

Convert the genome coordinates of aligned reads to bin-level counts for all chromosomes.

Usage

```
tag2bin(tagdat, fragL = 200, binS = 200, prob = 1)
```

Arguments

tagdat	A list of the genome coordinates for starting positions of each read, with positive numbers representing the 5' strand and negative numbers representing the 3' strand. Each list component corresponds to a single chromosome.
fragL	A numeric value for the fragment length of reads. Default: 200.
binS	A numeric value for the bin-size for the bin-level counts to be constructed. Default: 200.
prob	A numeric value for the proportion of randomly sampled reads that will be used to create bin data. Default: 1 (use all reads).

Value

A list of the bin-level counts for each chromosome.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( tagdat_chip )
tag2bin( tagdat_chip, fragL = 100, binS = 100 )
```

---

tag2bin.chr *Convert the genome coordinates of aligned reads to bin-level counting data for a single chromosome.*

Description

Convert the genome coordinates of aligned reads to bin-level counting data for a single chromosome.

Usage

```
tag2bin.chr(tagdat, fragL = 200, binS = 200)
```

Arguments

tagdat	A numeric vector of genome coordinates for the starting positions of the aligned reads, with positive numbers representing the 5' strand and negative numbers representing the 3' strand.
fragL	A numeric value of the fragment length for the reads. Default: 200.
binS	A numeric value of the bin-size for the bin-level data to be constructed. Default: 200.

Value

A numeric vector of the counts for each bin.

Author(s)

Chandler Zuo <zuo@stat.wisc.edu>

Examples

```
data( tagdat_chip )
tag2bin.chr( tagdat_chip[[1]], fragL = 100, binS = 100 )
```

---

tagdat_chip — *An artificially constructed dataset containing genome coordinates for aligned ChIP sample reads.*

Description

This dataset contains artificially generated genome coordinates for ChIP sample reads on a genome of 5 chromosomes. The sign of each read represents the strand direction, with 5' represented by positive numbers and 3' represented by negative numbers.

Usage

```
example
```

Format

a list containing the reads coordinates on each of the 5 chromosomes.

Author(s)

Chandler Zuo zuo@stat.wisc.edu

---

tagdat_input — *An artificially constructed dataset containing genome coordinates for aligned input sample reads.*

Description

This dataset contains artificially generated genome coordinates for ChIP sample reads on a genome of 5 chromosomes. The sign of each read represents the strand direction, with 5' represented by positive numbers and 3' represented by negative numbers.

Usage

```
example
```

Format

a list containing the reads coordinates on each of the 5 chromosomes.

Author(s)

Chandler Zuo zuo@stat.wisc.edu

Index