Package 'RJMCMCNucleosomes'

October 24, 2025

Version 1.33.2

Date 2025-07-28

Title Bayesian hierarchical model for genome-wide nucleosome positioning with high-throughput short-read data (MNase-Seq)

Description This package does nucleosome positioning using informative Multinomial-Dirichlet prior in a t-mixture with reversible jump estimation of nucleosome positions for genome-wide profiling.

Depends R (>= 3.5), IRanges, GenomicRanges

Imports Rcpp (>= 0.12.5), consensusSeekeR, BiocGenerics, Seqinfo, S4Vectors (>= 0.23.10), BiocParallel, stats, graphics, methods, grDevices

Suggests BiocStyle, knitr, rmarkdown, nucleoSim, RUnit

LinkingTo Rcpp

SystemRequirements Rcpp

Encoding UTF-8

License Artistic-2.0

URL https://github.com/ArnaudDroitLab/RJMCMCNucleosomes

BugReports https://github.com/ArnaudDroitLab/RJMCMCNucleosomes/issues

VignetteBuilder knitr

biocViews BiologicalQuestion, ChIPSeq, NucleosomePositioning, Software, StatisticalMethod, Bayesian, Sequencing, Coverage

RoxygenNote 6.0.1

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

git_branch devel

git_last_commit 74d317d

git_last_commit_date 2025-07-28

Repository Bioconductor 3.23

Date/Publication 2025-10-24

Author Pascal Belleau [aut],
Rawane Samb [aut],
Astrid Deschênes [cre, aut],
Khader Khadraoui [aut],
Lajmi Lakhal-Chaieb [aut],
Arnaud Droit [aut]

Maintainer Astrid Deschênes <adeschen@hotmail.com>

Contents

`RJMCMCNucleosomes-package`

RJMCMCNucleosomes: Bayesian hierarchical model for genome-wide nucleosome positioning with high-throughput short-read data (MNase-Seq)

Description

This package does nucleosome positioning using informative Multinomial-Dirichlet prior in a t-mixture with reversible jump estimation of nucleosome positions for genome-wide profiling.

Author(s)

Pascal Belleau, Rawane Samb, Astrid Deschênes, Khader Khadraoui, Lajmi Lakhal and Arnaud Droit

Maintainer: Astrid Deschenes <adeschen@hotmail.com>

See Also

- `rjmcmc` for profiling of nucleosome positions for a segment
- `rjmcmcCHR` for profiling of nucleosome positions for a large region. The function will take care of spliting and merging.
- `segmentation` for spliting a `GRanges` containing reads in a list of smaller segments for the `rjmcmc` function.
- `postTreatment` for merging closely positioned nucleosomes
- `mergeRDSFiles` for merging nucleosome information from selected RDS files.
- `plotNucleosomes` for generating a graph containing the nucleosome positions and the read coverage.

`mergeAllRDSFiles` *Merge nucleosome information*

Description

Merge nucleosome information present in multiple RDS files.

Usage

```
mergeAllRDSFiles(arrayOfFiles)
```

Arguments

`arrayOfFiles`	a `array`, the name of each file that must be used to merge nucleosome information.

Value

a `list` of `class` "rjmcmcNucleosomesMerge" containing:

- k a `integer`, the number of nucleosomes.
- `mu` a `GRanges` containing the positions of the nucleosomes.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Loading two files containing nucleosomes informations for two sections of
## the same chromosome
file_1 <- dir(system.file("extdata", package = "RJMCMCNucleosomes"),
                pattern = "RJMCMC_seg_01.RDS",
                full.names = TRUE)

file_2 <- dir(system.file("extdata", package = "RJMCMCNucleosomes"),
                pattern = "RJMCMC_seg_02.RDS",
                full.names = TRUE)

## Merging nucleosomes informations from the two files
result <- RJMCMCNucleosomes:::mergeAllRDSFiles(c(file_1, file_2))
```

`mergeAllRDSFilesFromDirectory`

Merge nucleosome information from all RDS files present in a same directory. Beware that only nucleosome information from same chromosome should be merged together.

Description

Merge nucleosome information, from all RDS files present in a same directory, into one object of `class` "rjmcmcNucleosomesMerge".

Usage

```
mergeAllRDSFilesFromDirectory(directory)
```

Arguments

`directory`	a `character`, the name of the directory (relative or absolute path) containing RDS files. The RDS files must contain R object of `class` "rjmcmcNucleosomes" or "rjmcmcNucleosomesMerge".

Value

a `list` of `class` "rjmcmcNucleosomesMerge" containing:

- k a `integer`, the number of nucleosomes.
- `mu` a `GRanges` containing the positions of the nucleosomes.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Use a directory present in the RJMCMC package
directoryWithRDSFiles <- system.file("extdata",
package = "RJMCMCNucleosomes")

## Merge nucleosomes info from RDS files present in directory
## It is assumed that all files present in the directory are nucleosomes
## result for the same chromosome
result <- mergeAllRDSFilesFromDirectory(directoryWithRDSFiles)

## Print the number and the position of the nucleosomes
result$k
result$mu

## Class of the output object
class(result)
```

---

mergeRDSFiles *Merge nucleosome information from selected RDS files.*

Description

Merge nucleosome information present in RDS files into one object of class "rjmcmcNucleosomesMerge".

Usage

```
mergeRDSFiles(RDSFiles)
```

Arguments

RDSFiles	a array, the names of all RDS used to merge nucleosome information. The files must contain R object of class "rjmcmcNucleosomes" or "rjmcmcNucleosomesMerge".

Value

a list of class "rjmcmcNucleosomesMerge" containing:

- k a integer, the number of nucleosomes.
- mu a GRanges containing the positions of the nucleosomes.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Use RDS files present in the RJMCMC package
RDSFiles <- dir(system.file("extdata", package = "RJMCMCNucleosomes"),
full.names = TRUE, pattern = "*RDS")

## Merge nucleosomes info from RDS files present in directory
result <- mergeRDSFiles(RDSFiles)

## Print the number and the position of the nucleosomes
result$k
result$mu

## Class of the output object
class(result)
```

---

plotNucleosomes *Generate a graph of nucleosome positions with read coverage*

Description

Generate a graph for a GRanges or a GRangesList of nucleosome positions. In presence of only one prediction (with multiples nucleosome positions), a GRanges is used. In presence of more thant one predictions (as example, before and after post-treatment or results from different software), a GRangesList with one entry per prediction is used. All predictions must have been obtained using the same reads.

Usage

```
plotNucleosomes(nucleosomePositions, reads, seqName = NULL,
  xlab = "position", ylab = "coverage", names = NULL)
```

Arguments

nucleosomePositions	a GRanges or a GRangesList containing the nucleosome positions for one or multiples predictions obtained using the same reads. In presence of only one prediction (with multiples nucleosome positions), a GRanges is used. In presence of more thant one predictions (as example, before and after post-treatment or results from different software), a GRangesList with one entry per prediction is used.
reads	a GRanges containing forward and reverse reads. The GRanges should contain at least one read.

seqName	a character string containing the label of the chromosome, present in the GRanges object, that will be used. The NULL value is accepted when only one seqname is present in the GRanges; the only seqname present will be used. Default: NULL.
xlab	a character string containing the label of the x-axis.
ylab	a character string containing the label of the y-axis.
names	a vector of a character string containing the label of each prediction set. The vector must be the same length of the nucleosomePositions list or 1 in presence of a vector. When NULL, the name of the elements of the list are used or the string "Nucleosome" for a vector are used. Default: NULL.

Value

a graph containing the nucleosome positions and the read coverage

Author(s)

Astrid Deschenes

Examples

```
## Load reads dataset
data(reads_demo_01)

## Run RJMCMC method
result <- rjmcmc(reads = reads_demo_01,
            seqName = "chr_SYNTHETIC",
            nbrIterations = 4000, lambda = 2, kMax = 30,
            minInterval = 146, maxInterval = 292, minReads = 5,
            vSeed = 10213)

## Create graph using the synthetic map
plotNucleosomes(nucleosomePositions = result$mu, seqName = "chr_SYNTHETIC",
            reads = reads_demo_01)
```

---

postMerge *A internal post treatment function to merge closely positioned nucleosomes, from the same chromosome, identified by the rjmcmc function*

Description

A internal helper function which merges closely positioned nucleosomes to rectify the over splitting and provide a more conservative approach. Beware that each chromosome must be treated separatly.

The function uses the Bioconductor package consensusSeeker to group closely positioned nucleosomes.

Usage

```
postMerge(reads, resultRJMCMC, extendingSize, chrLength, minReads = 5,
  seqName = NULL)
```

Arguments

reads	a GRanges containing all forward and reverse reads.The start positions of both reads are going to be used for the analysis. Beware that the start position of a reverse read is always higher that the end positition.
resultRJMCMC	an object of class 'rjmcmcNucleosomes' or 'rjmcmcNucleosomesMerge' containing informations about nucleosomes.
extendingSize	a positive numeric or a positive integer indicating the size of the consensus region used to group closeley positioned nucleosomes.The minimum size of the consensus region is equal to twice the value of the extendingSize parameter. The numeric will be treated as an integer.
chrLength	a positive numeric or a positive integer indicating the lenght of the current chromosome. The length of the chromosome is used to ensure that the consensus positions are all located inside the chromosome.
minReads	a positive integer or numeric, the minimum number of reads in a potential canditate region. Non-integer values of minReads will be casted to integer and truncated towards zero. Default: 5.

Value

a array of numeric, the updated values of the nucleosome positions. When no nucleosome is present, NULL is returned.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Loading dataset
data(RJMCMC_result)
data(reads_demo_02)

## Results before post-treatment
RJMCMC_result$mu

## Post-treatment function which merged closely positioned nucleosomes
postResult <- RJMCMCNucleosomes:::postMerge(reads = reads_demo_02,
resultRJMCMC = RJMCMC_result, extendingSize = 80, chrLength = 73500)

## Results after post-treatment
postResult
```

postTreatment *A post-treatment function to merge closely positioned nucleosomes, from the same chromosome, identified by the* rjmcmc *function.*

Description

A helper function which merges closely positioned nucleosomes to rectify the over splitting and provide a more conservative approach. Beware that each chromosome must be treated separatly.

Usage

```
postTreatment(reads, seqName = NULL, resultRJMCMC, extendingSize = 74L,
  chrLength)
```

Arguments

reads	a GRanges containing forward and reverse reads. Beware that the start position of a reverse read is always higher that the end positition.
seqName	a character string containing the label of the chromosome, present in the GRanges object, that will be used. The NULL value is accepted when only one seqname is present in the GRanges; the only seqname present will be used. Default: NULL.
resultRJMCMC	an object of class "rjmcmcNucleosomes" or "rjmcmcNucleosomesMerge", the information about nucleosome positioning for an entire chromosome or a region that must be treated as one unit.
extendingSize	a positive numeric or a positive integer indicating the size of the consensus region used to group closeley positioned nucleosomes.The minimum size of the consensus region is equal to twice the value of the extendingSize parameter. The numeric will be treated as an integer. Default: 74.
chrLength	a positive numeric or a positive integer indicating the length of the current chromosome. The length of the chromosome is used to ensure that the consensus positions are all located inside the chromosome.

Value

a GRanges, the updated nucleosome positions. When no nucleosome is present, NULL is returned.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Loading dataset
data(reads_demo_02)

## Nucleosome positioning, running both merge and split functions
```

```
result <- rjmcmc(reads = reads_demo_02,
            seqName = "chr_SYNTHETIC", nbrIterations = 1000,
            lambda = 2, kMax = 30, minInterval = 146,
            maxInterval = 490, minReads = 3, vSeed = 11)

## Before post-treatment
result

##Post-treatment function which merged closely positioned nucleosomes
postResult <- postTreatment(reads = reads_demo_02,
                    seqName = "chr_SYNTHETIC", result, 100, 73500)

## After post-treatment
postResult
```

`print.rjmcmcNucleosomes` *Formated output of predicted nucleosomes*

Description

Generated a formated output of a list marked as an `rjmcmcNucleosomes` class

Usage

```
## S3 method for class 'rjmcmcNucleosomes'
print(x, ...)
```

Arguments

`x`	the output object from `rjmcmc` function to be printed
`...`	arguments passed to or from other methods

Value

An object of class `rjmcmcNucleosomes`

Author(s)

Astrid Deschenes

Examples

```
## Loading dataset
data(RJMCMC_result)

print(RJMCMC_result)
```

print.rjmcmcNucleosomesBeforeAndAfterPostTreatment
Formated output of predicted nucleosomes

Description

Generated a formated output of a list marked as an rjmcmcNucleosomesBeforeAndAfterPostTreatment class

Usage

```
## S3 method for class 'rjmcmcNucleosomesBeforeAndAfterPostTreatment'
print(x, ...)
```

Arguments

x	the output object from rjmcmcCHR function to be printed
...	arguments passed to or from other methods

Value

an object of class rjmcmcNucleosomesBeforeAndAfterPostTreatment

Author(s)

Astrid Deschenes

Examples

```
## Load synthetic dataset of reads
data(syntheticNucleosomeReads)

## Use dataset of reads to create GRanges object
sampleGRanges <- GRanges(syntheticNucleosomeReads$dataIP)

## Run nucleosome detection on the entire sample
## Not run: result <- rjmcmcCHR(reads = sampleGRanges, zeta = 147, delta=50,
maxLength=1200, nbrIterations = 1000, lambda = 3, kMax = 30,
minInterval = 146, maxInterval = 292, minReads = 5, vSeed = 10113,
nbCores = 2, saveAsRDS = FALSE)
## End(Not run)

## Print result
## Not run: print(result)
```

print.rjmcmcNucleosomesMerge *Formated output of predicted nucleosomes*

Description

Generated a formated output of a list marked as an rjmcmcNucleosomesMerge class

Usage

```
## S3 method for class 'rjmcmcNucleosomesMerge'
print(x, ...)
```

Arguments

x	the output object from mergeAllRDSFilesFromDirectory function to be printed
...	arguments passed to or from other methods

Value

an object of class mergeAllRDSFilesFromDirectory

Author(s)

Astrid Deschenes

Examples

```
## Use a directory present in the RJMCMC package
directoryWithRDSFiles <- system.file("extdata",
package = "RJMCMCNucleosomes")

## Merge nucleosomes info from RDS files present in directory
## It is assumed that all files present in the directory are nucleosomes
## result for the same chromosome
result <- mergeAllRDSFilesFromDirectory(directoryWithRDSFiles)

## Show resulting nucleosomes
print(result)

## or simply
result
```

reads_demo_01 *Forward reads and reverse reads in* GRanges *format (for demo purpose).*

Description

A group of forward and reverse reads, in a GRanges, that can be used to test the rjmcmc function.

Usage

```
data(reads_demo_01)
```

Format

A GRanges containing forward and reverse reads.

Value

A GRanges containing forward and reverse reads.

See Also

- rjmcmc for profiling of nucleosome positions

Examples

```
## Loading dataset
data(reads_demo_01)

## Nucleosome positioning
rjmcmc(reads = reads_demo_01, nbrIterations = 100, lambda = 3, kMax = 30,
            minInterval = 146, maxInterval = 292, minReads = 5)
```

reads_demo_02 *Forward reads and reverse reads in* GRanges *format (for demo purpose).*

Description

A group of forward and reverse reads that can be used to test the rjmcmc function.

Usage

```
data(reads_demo_02)
```

Format

A GRanges containing forward and reverse reads.

Value

A GRanges containing forward and reverse reads.

See Also

- rjmcmc for profiling of nucleosome positions
- rjmcmcCHR for profiling of nucleosome positions for a large region. The function will take care of spliting and merging.
- segmentation for spliting a GRanges containing reads in a list of smaller segments for the rjmcmc function.
- postTreatment for merging closely positioned nucleosomes
- mergeRDSFiles for merging nucleosome information from selected RDS files.
- plotNucleosomes for generating a graph containing the nucleosome positions and the read coverage.

Examples

```
## Loading dataset
data(reads_demo_02)

## Nucleosome positioning
## Since there is only one chromosome present in reads_demo_02, the name
## of the chromosome does not need to be specified
rjmcmc(reads = reads_demo_02, nbrIterations = 150, lambda = 3, kMax = 30,
            minInterval = 144, maxInterval = 290, minReads = 6)
```

---

rjmcmc *Nucleosome positioning mapping on a segment*

Description

Use of a fully Bayesian hierarchical model for chromosome-wide profiling of nucleosome positions based on high-throughput short-read data (MNase-Seq data). Beware that for a genome-wide profiling, each chromosome must be treated separatly. This function is optimized to run on segments that are smaller sections of the chromosome.

Usage

```
rjmcmc(reads, seqName = NULL, nbrIterations, kMax, lambda = 3, minInterval,
  maxInterval, minReads = 5, adaptIterationsToReads = TRUE, vSeed = -1,
  saveAsRDS = FALSE)
```

Arguments

reads	a GRanges containing forward and reverse reads. Beware that the start position of a reverse read is always higher that the end positition.
seqName	a character string containing the label of the chromosome, present in the GRanges object, that will be used. The NULL value is accepted when only one seqname is present in the GRanges; the only seqname present will be used. Default: NULL.
nbrIterations	a positive integer or numeric, the number of iterations. Non-integer values of nbrIterations will be casted to integer and truncated towards zero.
kMax	a positive integer or numeric, the maximum number of degrees of freedom per region. Non-integer values of kMax will be casted to integer and truncated towards zero.
lambda	a positive numeric, the theorical mean of the Poisson distribution. Default: 3.
minInterval	a numeric, the minimum distance between two nucleosomes.
maxInterval	a numeric, the maximum distance between two nucleosomes.
minReads	a positive integer or numeric, the minimum number of reads in a potential canditate region. Non-integer values of minReads will be casted to integer and truncated towards zero. Default: 5.
adaptIterationsToReads	a logical indicating if the number of iterations must be modified in function of the number of reads. Default: TRUE.
vSeed	a integer. A seed used when reproducible results are needed. When a value inferior or equal to zero is given, a random integer is used. Default: -1.
saveAsRDS	a logical. When TRUE, a RDS file containing the complete output of the c++ rjmcmc() function is created. Default : FALSE.

Value

a list of class "rjmcmcNucleosomes" containing:

- call the matched call.
- k a integer, the final estimation of the number of nucleosomes. 0 when no nucleosome is detected.
- mu a GRanges containing the positions of the nucleosomes and '*' as strand. The seqnames of the GRanges correspond to the seqName input value. NA when no nucleosome is detected.
- k_max a integer, the maximum number of nucleosomes obtained during the iteration process. NA when no nucleosome is detected.

Author(s)

Rawane Samb, Pascal Belleau, Astrid Deschenes

Examples

```
## Loading dataset
data(reads_demo_01)

## Nucleosome positioning, running both merge and split functions
result <- rjmcmc(reads = reads_demo_01, seqName = "chr_SYNTHETIC",
            nbrIterations = 1000, lambda = 2, kMax = 30,
            minInterval = 146, maxInterval = 292, minReads = 5,
            vSeed = 10113, saveAsRDS = FALSE)

## Print the final estimation of the number of nucleosomes
result$k

## Print the position of nucleosomes
result$mu

## Print the maximum number of nucleosomes obtained during the iteration
## process
result$k_max
```

`rjmcmcCHR` *Nucleosome positioning mapping on a large segment, up to a chromosome*

Description

Use of a fully Bayesian hierarchical model for chromosome-wide profiling of nucleosome positions based on high-throughput short-read data (MNase-Seq data). Beware that for a genome-wide profiling, each chromosome must be treated separatly. This function is optimized to run on an entire chromosome.

The function will process by splittingg the `GRanges` of reads (as example, the reads from a chromosome) in a `list` of smaller `GRanges` segments that can be run by the `rjmcmc` function. All those steps are done automatically.

Usage

```
rjmcmcCHR(reads, seqName = NULL, zeta = 147, delta, maxLength,
  nbrIterations, kMax, lambda = 3, minInterval, maxInterval, minReads = 5,
  adaptIterationsToReads = TRUE, vSeed = -1, nbCores = 1,
  dirOut = "out", saveAsRDS = FALSE, saveSEG = TRUE)
```

Arguments

`reads`	a `GRanges`, the forward and reverse reads that need to be segmented.
`seqName`	a `character` string containing the label of the chromosome, present in the `GRanges` object, that will be used. The `NULL` value is accepted when only one seqname is present in the `GRanges`; the only seqname present will be used. Default: `NULL`.

zeta	a positive integer or numeric, the length of the nucleosomes. Default: 147.
delta	a positive integer or numeric, the accepted range of overlapping section between segments. The overlapping section being zeta + delta.
maxLength	a positive integer or numeric, the length of each segment.
nbrIterations	a positive integer or numeric, the number of iterations. Non-integer values of nbrIterations will be casted to integer and truncated towards zero.
kMax	a positive integer or numeric, the maximum number of degrees of freedom per region. Non-integer values of kMax will be casted to integer and truncated towards zero.
lambda	a positive numeric, the theorical mean of the Poisson distribution. Default: 3.
minInterval	a numeric, the minimum distance between two nucleosomes.
maxInterval	a numeric, the maximum distance between two nucleosomes.
minReads	a positive integer or numeric, the minimum number of reads in a potential canditate region. Non-integer values of minReads will be casted to integer and truncated towards zero. Default: 5.
adaptIterationsToReads	a logical indicating if the number of iterations must be modified in function of the number of reads. Default: TRUE.
vSeed	a integer. A seed used when reproducible results are needed. When a value inferior or equal to zero is given, a random integer is used. Default: -1.
nbCores	a positive integer, the number of cores used to run in parallel. Default: 1.
dirOut	a character string. The name of the directory where 2 directories are created (if they don't already exists). The directory "dirOut/results" contents the rjmcmc results for each segment. The directory "dirOut/done" contents file a log file for each segment in RData format. If the log file for a segment is in the directory, the program considers that it is has been processed and run the next segment. Default: "out".
saveAsRDS	a logical. When TRUE, a RDS file containing the complete output of the rjmcmc function is created. Default: FALSE.
saveSEG	a logical. When TRUE, a RDS file containing the segments generated by segmentation function is saved in directory named from paramter dirOut. Default: FALSE.

Value

a list of class "rjmcmcNucleosomesBeforeAndAfterPostTreatment" containing:

- k a integer, the number of nucleosomes.
- mu a GRanges containing the positions of the nucleosomes.
- kPost a integer, the number of nucleosomes after post-treatment and '*' as strand. The seqnames of the GRanges correspond to the seqName input value. NA when no nucleosome is detected.
- muPost a GRanges containing the positions of the nucleosomes after post-treament and '*' as strand. The seqnames of the GRanges correspond to the seqName input value. NA when no nucleosome is detected.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Load synthetic dataset of reads
data(syntheticNucleosomeReads)

## Use dataset of reads to create GRanges object
sampleGRanges <- GRanges(syntheticNucleosomeReads$dataIP)

## Run nucleosome detection on the entire sample
## Not run: result <- rjmcmcCHR(reads = sampleGRanges, zeta = 147, delta=50,
maxLength=1200, nbrIterations = 1000, lambda = 3, kMax = 30,
minInterval = 146, maxInterval = 292, minReads = 5, vSeed = 10113,
nbCores = 2, saveAsRDS = FALSE)
## End(Not run)
```

---

rjmcmcNucleo *Interface for the RJMCMC nucleosome mapping method in C++*

Description

Function that calls the core of the nucleosome positioning mapping function that is implemented in C++.

Usage

```
rjmcmcNucleo(startPosForwardReads, startPosReverseReads, nbrIterations, kMax,
  lambda, minInterval, maxInterval, minReads = 5L,
  adaptIterationsToReads = TRUE, vSeed = -1)
```

Arguments

`startPosForwardReads`	a `vector` of `numeric`, the start position of all the forward reads.
`startPosReverseReads`	a `vector` of `numeric`, the start position of all the reverse reads. Beware that the start position of a reverse read is always higher that the end positition.
`nbrIterations`	a positive `integer` or `numeric`, the number of iterations. Non-integer values of `nbrIterations` will be casted to `integer` and truncated towards zero.
`kMax`	a positive `integer` or `numeric`, the maximum number of nucleosomes per region. Non-integer values of `kMax` will be casted to `integer` and truncated towards zero.
`lambda`	a positive `numeric`, the theorical mean of the Poisson distribution. Default: 3.
`minInterval`	a `numeric`, the minimum distance between two nucleosomes.

maxInterval — a numeric, the maximum distance between two nucleosomes.

minReads — a positive integer or numeric, the minimum number of reads in a potential canditate region. Non-integer values of minReads will be casted to integer and truncated towards zero. Default: 5.

adaptIterationsToReads — a logical indicating if the number of iterations must be modified in function of the number of reads. Default: TRUE.

vSeed — a integer. A seed used when reproducible results are needed. When a value inferior or equal to zero is given, a random integer is used. Default: -1.

Value

a list containing:

- k a integer, the number of nucleosomes.
- k_max a integer, the maximum number of nucleosomes obtained during the iteration process.
- it a vector of integer of length k, the variance of the forward reads for each nucleosome.
- nbState a integer, the number of changes of state.
- mu a matrix of numeric with k_max columns and nbState row containing, in each row, the mu values associated the the state identified by the row number.
- muHat a matrix of numeric with k_max columns and k_max rows containing, in each row, the mean mu values associated the number of nucleosomes detected. The row number corresponds to the number of nucleosomes detected.
- nbK a vector of length k_max containing integer, the number of iterations which detected a specific number of nucleosomes. The position in the vector correspond to the number of nucleosomes.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
data(reads_demo_01)

forward <- start(reads_demo_01[strand(reads_demo_01) == "+"])
reverse <- end(reads_demo_01[strand(reads_demo_01) == "-"])

## Run nucleosome positioning
result <- RJMCMCNucleosomes:::rjmcmcNucleo(
            startPosForwardReads = forward,
            startPosReverseReads = reverse,
            nbrIterations = 1000, lambda = 2, kMax = 30,
            minInterval = 146, maxInterval = 292, minReads = 5,
            adaptIterationsToReads = TRUE, vSeed = -1)

## Print the final estimation of the number of nucleosomes
```

```
result$k

## Print the position of nucleosomes
result$mu
```

RJMCMC_result *Nucleosomes obtained by running RJMCMC function using reads from reads_demo_02 dataset (for demo purpose).*

Description

A `list` of `class` "rjmcmcNucleosomes" which contains the information about the detected nucleosomes.

Usage

```
data(RJMCMC_result)
```

Format

A `list` of `class` "rjmcmcNucleosomes" containing:

- `call` the matched call.
- `k` a `integer`, the final estimation of the number of nucleosomes. `0` when no nucleosome is detected.
- `mu` a `vector` of `numeric` of length `k`, the positions of the nucleosomes. `NA` when no nucleosome is detected.
- `k_max` a `integer`, the maximum number of nucleosomes obtained during the iteration process. `NA` when no nucleosome is detected.

Value

A `list` of `class` "rjmcmcNucleosomes" containing:

- `call` the matched call.
- `k` a `integer`, the final estimation of the number of nucleosomes. `0` when no nucleosome is detected.
- `mu` a `vector` of `numeric` of length `k`, the positions of the nucleosomes. `NA` when no nucleosome is detected.
- `k_max` a `integer`, the maximum number of nucleosomes obtained during the iteration process. `NA` when no nucleosome is detected.

See Also

- rjmcmc for profiling of nucleosome positions
- rjmcmcCHR for profiling of nucleosome positions for a large region. The function will take care of spliting and merging.
- segmentation for spliting a GRanges containing reads in a list of smaller segments for the rjmcmc function.
- postTreatment for merging closely positioned nucleosomes
- mergeRDSFiles for merging nucleosome information from selected RDS files.
- plotNucleosomes for generating a graph containing the nucleosome positions and the read coverage.

Examples

```
## Loading dataset
data(RJMCMC_result)
data(reads_demo_02)

## Results before post-treatment
RJMCMC_result$mu

## Post-treatment function which merged closely positioned nucleosomes
postResult <- postTreatment(reads = reads_demo_02,
    extendingSize = 60, chrLength = 100000, resultRJMCMC = RJMCMC_result)

## Results after post-treatment
postResult
```

---

runCHR *Run rjmcmc on multiples segments and merge results.*

Description

Run rjmcmc on a segment that is contained in a list of segments. Files generated by the function are all saved in a directory specified by user. A RData log file is created when a segment has been run while the result is saved in a RDS file.

If the same output directory is used more than once, the rjmcmc won't be called for segments that have the à corresponding RData log file.

Usage

```
runCHR(p, seg, niter, kmax, lambda, ecartmin, ecartmax, minReads,
  adaptNbrIterations, vSeed = -1, saveAsRDS = FALSE, dirOut = "out")
```

Arguments

p	a integer, the position of the segment to treat in a list of GRanges.
seg	a list a GRanges containing the segments to be process.
niter	a positive integer or numeric, the number of iterations. Non-integer values of nbrIterations will be casted to integer and truncated towards zero.
lambda	a positive numeric, the theorical mean of the Poisson distribution. Default: 3.
minReads	a positive integer or numeric, the minimum number of reads in a potential canditate region. Non-integer values of minReads will be casted to integer and truncated towards zero. Default: 5.
adaptNbrIterations	a logical indicating if the number of iterations must be modified in function of the number of reads.
vSeed	a integer. A seed used when reproducible results are needed. When a value inferior or equal to zero is given, a random integer is used. Default: -1.
saveAsRDS	a logical. When TRUE, a RDS file containing the complete output of the c++ rjmcmc() function is created. Default : FALSE.
kMax	a positive integer or numeric, the maximum number of degrees of freedom per region. Non-integer values of kMax will be casted to integer and truncated towards zero.
minInterval	a numeric, the minimum distance between two nucleosomes.
maxInterval	a numeric, the maximum distance between two nucleosomes.
maxLength	a positive integer or numeric, the length of each segment.

Value

0.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Load synthetic dataset of reads
data(syntheticNucleosomeReads)

## Use dataset of reads to create GRanges object
sampleGRanges <- GRanges(seqnames = syntheticNucleosomeReads$dataIP$chr,
    ranges = IRanges(start = syntheticNucleosomeReads$dataIP$start,
    end = syntheticNucleosomeReads$dataIP$end),
    strand = syntheticNucleosomeReads$dataIP$strand)

# Segmentation of the reads
seg <- segmentation(sampleGRanges, zeta = 147, delta = 50, maxLength = 1000)
## Not run:
dir.create("out")
dir.create("out/done")
```

```
dir.create("out/results")

runCHR(p=1, seg=seg, niter=1000, kmax=330,lambda=3,
            ecartmin=147, ecartmax=297, minReads=5)
## End(Not run)
```

---

segmentation *Split a* GRanges *containing reads in a list of smaller segments for the* rjmcmc *function.*

Description

Split a GRanges of reads (as example, the reads from a chromosome) in a list of smaller GRanges sot that the rjmcmc function can be run on each segments.

Usage

```
segmentation(reads, zeta = 147, delta, maxLength)
```

Arguments

reads	a GRanges, the reads that need to be segmented.
zeta	a positive integer or numeric, the length of the nucleosomes. Default: 147.
delta	a positive integer or numeric, the accepted range of overlapping section between segments. The overlapping section being zeta + delta.
maxLength	a positive integer or numeric, the length of each segment.

Value

a GRangesList containing all the segments.

Author(s)

Pascal Belleau, Astrid Deschenes

Examples

```
## Load synthetic dataset of reads
data(syntheticNucleosomeReads)

## Use dataset of reads to create GRanges object
sampleGRanges <- GRanges(seqnames = syntheticNucleosomeReads$dataIP$chr,
    ranges = IRanges(start = syntheticNucleosomeReads$dataIP$start,
    end = syntheticNucleosomeReads$dataIP$end),
    strand = syntheticNucleosomeReads$dataIP$strand)

# Segmentation of the reads
```

```
segmentation(reads = sampleGRanges, zeta = 147, delta = 50,
maxLength = 1000)
```

`syntheticNucleosomeReads`

Simulated dataset of reads generated by `nucleoSim` *package (for demo purpose).*

Description

A `list` of `class` "syntheticNucReads" which contains the information about synthetic reads related to nucleosomes. The datset has been created using a total of 300 well-positioned nucleosomes, 30 fuzzy nucleosomes with variance of reads following a Normal distribution.

Usage

```
data(syntheticNucleosomeReads)
```

Format

A `list` containing:

- `call` the called that generated the dataset.
- `dataIP` a `data.frame` with the chromosome name, the starting and ending positions and the direction of all forward and reverse reads for all well-positioned and fuzzy nucleosomes. Paired-end reads are identified with an unique id.
- `wp` a `data.frame` with the positions of all the well-positioned nucleosomes, as well as the number of paired-reads associated to each one.
- `fuz` a `data.frame` with the positions of all the fuzzy nucleosomes, as well as the number of paired-reads associated to each one.
- `paired` a `data.frame` with the starting and ending positions of the reads used to generate the paired-end reads. Paired-end reads are identified with an unique id.

Value

A `list` containing:

- `call` the called that generated the dataset.
- `dataIP` a `data.frame` with the chromosome name, the starting and ending positions and the direction of all forward and reverse reads for all well-positioned and fuzzy nucleosomes. Paired-end reads are identified with an unique id.
- `wp` a `data.frame` with the positions of all the well-positioned nucleosomes, as well as the number of paired-reads associated to each one.
- `fuz` a `data.frame` with the positions of all the fuzzy nucleosomes, as well as the number of paired-reads associated to each one.
- `paired` a `data.frame` with the starting and ending positions of the reads used to generate the paired-end reads. Paired-end reads are identified with an unique id.

validateDirectoryParameters *Parameters validation for the* mergeAllRDSFilesFromDirectory *function*

Description

Validation of all parameters needed by the public mergeAllRDSFilesFromDirectory function.

Usage

```
validateDirectoryParameters(directory)
```

Arguments

directory	a character, the name of the directory (relative or absolute path) containing RDS files.

Value

0 indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes

Examples

```
## Load an existing directory
directory <- system.file("extdata", package = "RJMCMCNucleosomes")

## Testing using a real directory
RJMCMCNucleosomes:::validateDirectoryParameters(directory)
```

validatePlotNucleosomesParameters *Parameters validation for the* plotNucleosomes *function*

Description

Validation of all parameters needed by the public plotNucleosomes function.

Usage

```
validatePlotNucleosomesParameters(nucleosomePositions, reads, seqName, xlab,
  ylab, names)
```

Arguments

nucleosomePositions	a GRanges or a GRangesList containing the nucleosome positions for one or multiples predictions obtained using the same reads. In presence of only one prediction (with multiples nucleosome positions), a GRanges is used. In presence of more thant one predictions (as example, before and after post-treatment or results from different software), a GRangesList with one entry per prediction is used.
reads	a GRanges containing forward and reverse reads. The GRanges should contain at least one read.
seqName	a character string containing the label of the chromosome, present in the GRanges object, that will be used. The NULL value is accepted when only one seqname is present in the GRanges; the only seqname present will be used.
xlab	a character string containing the label of the x-axis.
ylab	a character string containing the label of the y-axis.
names	a vector of a character string containing the label of each prediction set. The vector must be the same length of the nucleosomePositions list or 1 in presence of a vector.

Value

0 indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Load GRanges dataset
data(reads_demo_01)

## Load RJMCMC result
data(RJMCMC_result)

## The function returns 0 when all parameters are valid
RJMCMCNucleosomes:::validatePlotNucleosomesParameters(nucleosomePositions =
RJMCMC_result$mu, reads = reads_demo_01, seqName = "chr_SYNTHETIC",
xlab = "position", ylab = "coverage", names = c("test"))

## The function raises an error when at least one paramater is not valid
#\dontrun{RJMCMCNucleosomes:::validatePlotNucleosomesParameters(
#nucleosomePositions = c("hi"), reads = reads,
#xlab = "position", ylab = "coverage", names = c("test"))}

#\dontrun{RJMCMCNucleosomes:::validatePlotNucleosomesParameters(
#nucleosomePositions = RJMCMC_result$mu, reads = reads_demo_01,
#seqName = "chr_SYNTHETIC", xlab = "position", ylab = "coverage",
#names = c("test_one", "test_false"))}
```

validatePrepMergeParameters
Parameters validation for the postMerge *function*

Description

Validation of all parameters needed by the public postMerge function.

Usage

```
validatePrepMergeParameters(reads, seqName, resultRJMCMC, extendingSize,
  chrLength)
```

Arguments

reads	a GRanges containing all forward and reverse reads.The start positions of both reads are going to be used for the analysis. Beware that the start position of a reverse read is always higher that the end positition. The GRanges should at least contain one read.
seqName	a character string containing the label of the chromosome, present in the GRanges object, that will be used. The NULL value is accepted when only one seqname is present in the GRanges; the only seqname present will be used.
resultRJMCMC	an object of class "rjmcmcNucleosomes" or "rjmcmcNucleosomesMerge" that contain information about nucleosome positioning for an entire chromosome.
extendingSize	a positive numeric or a positive integer indicating the size of the consensus region used to group closeley positioned nucleosomes.The minimum size of the consensus region is equal to twice the value of the extendingSize parameter. The numeric will be treated as an integer.
chrLength	a positive numeric or a positive integer indicating the lenght of the current chromosome. The length of the chromosome is used to ensure that the consensus positions are all located inside the chromosome.

Value

0 indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes

Examples

```
## Load dataset containing forward and reverse reads
data(reads_demo_01)

## Load dataset containing nucleosome information
file_002 <- dir(system.file("extdata", package = "RJMCMCNucleosomes"),
pattern = "RJMCMC_seg_02.RDS", full.names = TRUE)
nucleosome_info <- readRDS(file_002)

## The function returns 0 when all parameters are valid
RJMCMCNucleosomes:::validatePrepMergeParameters(reads = reads_demo_01,
seqName = "chr_SYNTHETIC", resultRJMCMC = nucleosome_info,
extendingSize = 74, chrLength = 10000000)

## The function raises an error when at least one paramater is not valid
## Not run: RJMCMCNucleosomes:::validatePrepMergeParameters(
reads = c(72400, 72431, 72428, 72429, 72426),
resultRJMCMC = NA, extendingSize = 74, chrLength = 10000000)
## End(Not run)
```

---

validateRDSFilesParameters

Parameters validation for the `mergeRDSFiles` *function*

Description

Validation of all parameters needed by the public `mergeRDSFiles` function.

Usage

```
validateRDSFilesParameters(RDSFiles)
```

Arguments

`RDSFiles`	a `array`, the names of all RDS used to merge nucleosome information. The files must contain R object of `class` "rjmcmcNucleosomes" or "rjmcmcNucleosomesMerge".

Value

`0` indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes

Examples

```
## Loading a file
file_test <- dir(system.file("extdata", package = "RJMCMCNucleosomes"),
pattern = "RJMCMC_seg_02.RDS", full.names = TRUE)

## Testing using a real file
RJMCMCNucleosomes:::validateRDSFilesParameters(file_test)
```

`validateRJMCMCParameters` *Parameters validation for the* `rjmcmc` *function*

Description

Validation of all parameters needed by the public `rjmcmc` function.

Usage

```
validateRJMCMCParameters(reads, seqName, nbrIterations, kMax, lambda,
  minInterval, maxInterval, minReads, adaptIterationsToReads, vSeed)
```

Arguments

`reads`	a `GRanges` containing all forward and reverse reads.The start positions of both reads are going to be used for the analysis. Beware that the start position of a reverse read is always higher that the end posititon.
`seqName`	a `character` string containing the label of the chromosome, present in the `GRanges` object, that will be used. The `NULL` value is accepted when only one seqname is present in the `GRanges`; the only seqname present will be used.
`nbrIterations`	a positive `integer` or `numeric`, the number of iterations. Non-integer values of `nbrIterations` will be casted to `integer` and truncated towards zero.
`kMax`	a positive `integer` or `numeric`, the maximum number of nucleosomes per region. Non-integer values of `kMax` will be casted to `integer` and truncated towards zero.
`lambda`	a positive `numeric`, the theorical mean of the Poisson distribution.
`minInterval`	a `numeric`, the minimum distance between two nucleosomes.
`maxInterval`	a `numeric`, the maximum distance between two nucleosomes.
`minReads`	a positive `integer` or `numeric`, the minimum number of reads in a potential canditate region. Non-integer values of `minReads` will be casted to `integer` and truncated towards zero.
`adaptIterationsToReads`	a `logical` indicating if the number of iterations must be modified in function of the number of reads.
`vSeed`	a `integer`. A seed used when reproducible results are needed. When a value inferior or equal to zero is given, a random integer is used.

Value

`0` indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes

Examples

```
reads <- GRanges(seqnames = Rle(c("chr1"), c(10)),
    ranges = IRanges(101:110, end = 111:120, names = head(letters, 10)),
    strand = Rle(strand(c("-", "+", "-", "+", "-")), c(1, 2, 2, 3, 2)))

## The function returns 0 when all paramaters are valid
RJMCMCNucleosomes:::validateRJMCMCParameters(reads = reads,
seqName = "chr1", nbrIterations = 2, kMax = 10, lambda = 1, minReads = 1,
minInterval = 100, maxInterval = 200, adaptIterationsToReads = TRUE,
vSeed = 100)

## The function raises an error when at least one paramater is not valid
## Not run: RJMCMCNucleosomes:::validateRJMCMCParameters(
reads = NA, seqName = "chr1",
nbrIterations = 2, kMax = 10, lambda = 1, minReads = 1, minInterval = 100,
maxInterval = 200, adaptIterationsToReads = TRUE, vSeed = -1)
## End(Not run)
```

---

validateSegmentationParameters

Parameters validation for the `segmentation` *function*

Description

Validation of all parameters needed by the public `segmentation` function.

Usage

```
validateSegmentationParameters(reads, zeta = 147, delta, maxLength)
```

Arguments

`reads`	a `GRanges`, the reads that need to be segmented.
`zeta`	a positive `integer` or `numeric`, the length of the nucleosomes. Default: 147.
`delta`	a positive `integer` or `numeric`, the accepted range of overlapping section between segments. The overlapping section being `zeta` + `delta`.
`maxLength`	a positive `integer` or `numeric`, the length of each segment.

Value

`0` indicating that all parameters validations have been successful.

Author(s)

Astrid Deschenes, Pascal Belleau

Examples

```
## Load synthetic dataset of reads
data(syntheticNucleosomeReads)

## Use dataset of reads to create GRanges object
sampleGRanges <- GRanges(seqnames = syntheticNucleosomeReads$dataIP$chr,
ranges = IRanges(start = syntheticNucleosomeReads$dataIP$start,
end = syntheticNucleosomeReads$dataIP$end),
strand = syntheticNucleosomeReads$dataIP$strand)

## The function returns 0 when all parameters are valid
RJMCMCNucleosomes:::validateSegmentationParameters(reads = sampleGRanges,
zeta = 147, delta = 30, maxLength = 12000)

## The function raises an error when at least one paramater is not valid
#\dontrun{RJMCMCNucleosomes:::validateSegmentationParameters(
#reads = c(100), zeta = 147, delta = 30, maxLength = 12000)}

#\dontrun{RJMCMCNucleosomes:::validateSegmentationParameters(
#reads = sampleGRanges, zeta = "hi", delta = 30, maxLength = 12000)}
```

Index