

# Package ‘Motif2Site’

October 17, 2024

**Type** Package

**Title** Detect binding sites from motifs and ChIP-seq experiments, and compare binding sites across conditions

**Version** 1.8.0

**Depends** R (>= 4.1)

**Description**

Detect binding sites using motifs IUPAC sequence or bed coordinates and ChIP-seq experiments in bed or bam format. Combine/compare binding sites across experiments, tissues, or conditions. All normalization and differential steps are done using TMM-GLM method. Signal decomposition is done by setting motifs as the centers of the mixture of normal distribution curves.

**BugReports** <https://github.com/ManchesterBioinference/Motif2Site/issues>

**License** GPL-2

**LazyData** false

**Encoding** UTF-8

**Imports** S4Vectors, stats, utils, methods, grDevices, graphics, BiocGenerics, BSgenome, GenomeInfoDb, MASS, IRanges, GenomicRanges, Biostrings, GenomicAlignments, edgeR, mixtools

**Suggests** BiocStyle, rmarkdown, knitr, BSgenome.Hsapiens.UCSC.hg38, BSgenome.Mmusculus.UCSC.mm10, BSgenome.Scerevisiae.UCSC.sacCer3, BSgenome.Ecoli.NCBI.20080805

**biocViews** Software, Sequencing, ChIPSeq, DifferentialPeakCalling, Epigenetics, SequenceMatching

**RoxygenNote** 7.1.2

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/Motif2Site>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** 0e322d9

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-10-16

**Author** Peyman Zarrineh [cre, aut] (<<https://orcid.org/0000-0003-4820-4101>>)

**Maintainer** Peyman Zarrineh <peyman.zarrineh@manchester.ac.uk>

## Contents

Bed2Granges . . . . .	2
combine2Table . . . . .	3
combineMotifFiles . . . . .	4
combineTestResults . . . . .	4
compareBedFiless2UserProvidedRegions . . . . .	5
CompareBeds2GivenRegions . . . . .	6
CompareMotifs2GivenRegions . . . . .	7
compareMotifs2UserProvidedRegions . . . . .	7
computeFoldEnrichment . . . . .	9
data . . . . .	9
decomposeBindingSignal . . . . .	11
DeleteMultipleFiles . . . . .	12
deriveHeuristicBindingDistribution . . . . .	12
DetectBindingSites . . . . .	13
DetectBindingSitesBed . . . . .	14
DetectBindingSitesMotif . . . . .	15
DetectFdrCutoffBH . . . . .	17
findMotifs . . . . .	18
fitKernelDensity . . . . .	19
generateIntBedAlignment . . . . .	19
Motif2Site . . . . .	20
motifBindingNegativeBinomialCount . . . . .	21
motifChipCount . . . . .	22
motifCount . . . . .	22
motifTablePreProcess . . . . .	23
NegativeBinomialTestWithReplicate . . . . .	23
pairwisDifferential . . . . .	24
quiet . . . . .	25
recenterBindingSitesAcrossExperiments . . . . .	26
removeNonBellShapedMotifs . . . . .	27
strongestMotif . . . . .	28
<b>Index</b>	<b>29</b>

---

Bed2Granges

*Read a bed file as Genomic Ranges*

---

### Description

Read a bed file as Genomic Ranges.

**Usage**

```
Bed2Granges(fileName)
```

**Arguments**

fileName      A table delimited file in bed format

**Value**

granges format of given coordinates

**Examples**

```
yeastExampleFile=system.file("extdata", "YeastSampleMotif.bed",  
                             package="Motif2Site")  
ex <- Bed2Granges(yeastExampleFile)  
ex
```

---

combine2Table	<i>Combine all IP and Input count table files</i>
---------------	---

---

**Description**

Open raw counts IP and Inut files and with given total counts calculate Fold Enrichment values, and combine them into one file

**Usage**

```
combine2Table(outputName, replicateNumber, currentDir)
```

**Arguments**

outputName      Name of the output table  
replicateNumber      Number of the replicates  
currentDir      Directory for I/O operations

**Value**

No return value

---

combineMotifFiles      *Combine motif bed files into a combined ranges*

---

**Description**

Get motif file names and combine them into a matrix, and keep the indices of original motifs in the combined file.

If the motif type is string the bed files are deleted after being combined to one matrix.

**Usage**

```
combineMotifFiles(motifFileNames, motifType = "BioString")
```

**Arguments**

motifFileNames    a vector motif file names  
motifType          Type of motif string or give bed

**Value**

No return value

---

combineTestResults      *Combine count Table and statistics table*

---

**Description**

Combine count table and pvalue FE statistics into one file for motifs and regions separately.

**Usage**

```
combineTestResults(  
  motifFile,  
  acceptedMotifsOutputFile,  
  acceptedRegionsOutputFile,  
  countTableFile,  
  testTableFile,  
  fdrCutoff,  
  windowSize  
)
```

**Arguments**

motifFile	File contains motifs
acceptedMotifsOutputFile	File name of accepted motif table information
acceptedRegionsOutputFile	File name of accepted region information
countTableFile	Table of count values file name
testTableFile	negative binomial test table file name
fdrCutoff	Pvalue cut-off related to the used FDR
windowSize	Window size around binding site. The total region would be 2*windowSize+1

**Value**

The average binding intensity for each ChIP-seq

---

compareBedFileless2UserProvidedRegions

*Compare a set of bed files to a user provided regions set*

---

**Description**

This function gets user provided bedfiles and compare them with a user provided region. It returns this comparison to given user binding regions in terms of precision/recall.

**Usage**

```
compareBedFileless2UserProvidedRegions(bedfiles, motifnames, givenRegion)
```

**Arguments**

bedfiles	a vector of bed files
motifnames	a vector of the names related to bed files
givenRegion	granges of user provided binding regions

**Value**

A dataframe which includes precision recall values for each bed file

**See Also**

[compareMotifs2UserProvidedRegions](#)

## Examples

```
yeastExampleFile=system.file("extdata", "YeastSampleMotif.bed",
                             package="Motif2Site")
YeastRegionsChIPseq <- Bed2Granges(yeastExampleFile)
bed1 <- system.file("extdata", "YeastBedFile1.bed", package="Motif2Site")
bed2 <- system.file("extdata", "YeastBedFile2.bed", package="Motif2Site")
BedFilesVector <- c(bed1, bed2)
SequenceComparison <- compareBedFiless2UserProvidedRegions(
  givenRegion=YeastRegionsChIPseq,
  bedfiles=BedFilesVector,
  motifnames=c("YeastBed1", "YeastBed2")
)
SequenceComparison
```

---

CompareBeds2GivenRegions

*Compare a set of bed files to a provided regions set*

---

## Description

Get combined ranges of bed files and compare them to given binding regions in terms of precision/recall.

## Usage

```
CompareBeds2GivenRegions(motifName, bindingRegions)
```

## Arguments

motifName      a vector of motif names  
bindingRegions  granges of provided binding regions

## Value

A dataframe which includes precision recall values for each motif

---

`CompareMotifs2GivenRegions`*Comparison motifs locations to a given regions set*

---

**Description**

Comparison of motifs locations to user provided binding regions.

It returns this comparison to given user binding regions in terms of precision/recall.

**Usage**

```
CompareMotifs2GivenRegions(motifs, mismatchNumbers, bindingRegions)
```

**Arguments**

`motifs` a vector of motif characters in nucleotide IUPAC format

`mismatchNumbers` a vector Number of mismatches allowed to match with motifs

`bindingRegions` granges of user provided binding regions

**Value**

A dataframe which includes precision recall values for each motif

---

`compareMotifs2UserProvidedRegions`*Compare a set of motifs to a user provided regions set*

---

**Description**

This function gets user provided motifs and related mismatch numbers, it detects motifs and compare them with a user provided region.

It returns this comparison to given user binding regions in terms of precision/recall.

The genome and build information should be provided and relevant BS genomes packages such as `BSgenome.Mmusculus.UCSC.mm10` or `BSgenome.Hsapiens.UCSC.hg38` must be installed for the used genome and builds.

**Usage**

```
compareMotifs2UserProvidedRegions(
  motifs,
  mismatchNumbers,
  genome,
  genomeBuild,
  DB = "UCSC",
  givenRegion,
  mainCHRs = TRUE
)
```

**Arguments**

motifs	a vector of motif characters in nucleotide IUPAC format
mismatchNumbers	a vector Number of mismatches allowed to match with motifs
genome	The genome name such as "Hsapiens", "Mmusculus", "Dmelanogaster"
genomeBuild	The genome build such as "hg38", "hg19", "mm10", "dm3"
DB	The database of genome build. default: "UCSC"
givenRegion	granges of user provided binding regions
mainCHRs	If true only the major chromosome are considered, if FALSE Random, Uncharacterised, and Mitochondrial chromosomes are also considered

**Value**

A dataframe which includes precision recall values for each motif

**See Also**

[compareBedFileless2UserProvidedRegions](#)

**Examples**

```
# Artificial example in Yeast
# install BSgenome.Scerevisiae.UCSC.sacCer3 prior to run this code
yeastExampleFile=system.file("extdata", "YeastSampleMotif.bed",
                             package="Motif2Site")
YeastRegionsChIPseq <- Bed2Granges(yeastExampleFile)
SequenceComparison <- compareMotifs2UserProvidedRegions(
  givenRegion=YeastRegionsChIPseq,
  motifs=c("TGATTSCAGGANT", "TGATTCAGGANT", "TGATWSCAGGANT"),
  mismatchNumbers=c(1,0,2),
  genome="Scerevisiae",
  genomeBuild="sacCer3"
)
SequenceComparison
```

---

computeFoldEnrichment *compute fold enrichment values for an experiment*

---

### Description

Open raw counts IP and Inut files and with given total counts calculate Fold Enrichment values for the motifs

### Usage

```
computeFoldEnrichment(
  ipCountFile,
  inputCountFile,
  ipTotalCount,
  inputTotalCount,
  outputName
)
```

### Arguments

ipCountFile	File contains motifs count values for IP experiment
inputCountFile	File contains motifs count values for Input experiment
ipTotalCount	Total short reads number in IP experiment
inputTotalCount	Total short reads number in Input experiment
outputName	Name of the output table

### Value

No return value

---

data *Synthetic datasets used in the package*

---

### Description

Comparison Yeast synthetic motifs and binding sites: Two synthetic motif files in bed format are created to compare them with a synthetic binding site set in terms of precision and recall.

Fur binding sites detection in E. coli build NC\_000913: Synthetic Fur ChIP-seq in E. coli was generated using real peaks published in Seo et al 2014. The ChIP-seq data are provided in bed format in fe and dpd condition and both contains two replicates. Synthetic Input ChIP-seq datasets were generated by randomly distributing short reads in E. coli genome. User provided candidate binding sites in bed format was generated by combining instances of "GWWTGANAA" motif with 1-mismatch and "GWWTGAGAAT" with 2-mismatches in E. coli genome.

## Format

Three bed files to compare user provided motifs and binding sites in Yeast. Seven bed files to compare Fur ChIP-seq binding sites in E.coli.

**YeastBedFile1.bed** The first synthetic motif set

**YeastBedFile2.bed** The second synthetic motif set

**YeastSampleMotif.bed** The synthetic binding region

**FurMotifs.bed** User provided Fur motif set in E. coli

**FUR\_fe1.bed** Synthetic Fur ChIP-seq short reads in fe condition rep1

**FUR\_fe2.bed** Synthetic Fur ChIP-seq short reads in fe condition rep2

**FUR\_dpd1.bed** Synthetic Fur ChIP-seq short reads in dpd condition rep1

**FUR\_dpd2.bed** Synthetic Fur ChIP-seq short reads in dpd condition rep2

**Input1.bed** The synthetic background Input ChIP-seq rep1

**Input2.bed** The synthetic background Input ChIP-seq rep2

## Examples

```
## Data for example to compare user provided motifs and binding sites in Yeast
```

```
yeastExampleFile=system.file("extdata", "YeastSampleMotif.bed",
                             package="Motif2Site")
YeastRegionsChIPseq <- Bed2Granges(yeastExampleFile)
bed1 <- system.file("extdata", "YeastBedFile1.bed", package="Motif2Site")
bed2 <- system.file("extdata", "YeastBedFile2.bed", package="Motif2Site")
```

```
## Data for examples of binding site detection in E. coli
```

```
# Fur candidate motifs in NC_000913 E. coli
FurMotifs = system.file("extdata", "FurMotifs.bed", package="Motif2Site")
```

```
# ChIP-seq Fur fe datasets binding sites from user provided bed file
# ChIP-seq datasets in bed single end format
```

```
IPFe <- c(system.file("extdata", "FUR_fe1.bed", package="Motif2Site"),
          system.file("extdata", "FUR_fe2.bed", package="Motif2Site"))
```

```
# ChIP-seq Fur dpd datasets binding sites from user provided bed file
# ChIP-seq datasets in bed single end format
```

```
IPDpd <- c(system.file("extdata", "FUR_dpd1.bed", package="Motif2Site"),
           system.file("extdata", "FUR_dpd2.bed", package="Motif2Site"))
```

```
# ChIP-seq background
```

```
Inputs <- c(system.file("extdata", "Input1.bed", package="Motif2Site"),
```

```
system.file("extdata", "Input2.bed", package="Motif2Site"))
```

---

decomposeBindingSignal

*Decompose binding signal among accepted motifs*

---

### Description

Gets motif locations and related short reads and select the motifs which are non-skewed:  $\text{abs}(\text{skewness}) < 0.3$  and more short reads binds closer to site, and show strong binding after decomposition.

Decomposition is performed by using mixtools normalmixEM command fixing mu as motif locations.

### Usage

```
decomposeBindingSignal(  
  windowSize,  
  replicateNumber,  
  acceptedRegionsOutputFile = "BindingRegions",  
  acceptedMotifsOutputFile = "BindingMotifsTable",  
  currentDir  
)
```

### Arguments

windowSize	Window size around binding site. The total region would be $2 * \text{windowSize} + 1$
replicateNumber	experiment replicate number
acceptedRegionsOutputFile	File name contains binding regions coordinates and related motifs
acceptedMotifsOutputFile	File name contains motifs coordinates and related information, Pvalue, FE, etc
currentDir	Directory for I/O operations

### Value

motifStatistics Ratio of accepted motifs, rejected motifs due to skewnewss, and rejected motifs after decomposition

---

DeleteMultipleFiles     *Delete a vector of files*

---

**Description**

Delete multiple give files as a vector of characters

**Usage**

```
DeleteMultipleFiles(files)
```

**Arguments**

files                    a vector of files

**Value**

No return value

---

deriveHeuristicBindingDistribution  
*build heurisitc distribution around the binding sites*

---

**Description**

This function generates heuristic distribution of short reads around binding sites which do not need to deconvolve, total number of short reads and window size as number of neucleotid around binding sites.

It fits a kernel to the distribution and return the distribution as output. The total sum of returned values is equal to one. It plots this kernel.

Also it calculates FRiPs (Fraction of Reads in Peaks) for each

ChIP-seq and returns it. FRiPs and kernel distributions are measures of goodness of ChIP-seq experiments and selected motifs.

**Usage**

```
deriveHeuristicBindingDistribution(  
  chipSeq,  
  averageBindings,  
  windowSize,  
  acceptedRegionsOutputFile = "BindingRegions",  
  currentDir  
)
```

**Arguments**

chipSeq	ChIP-seq aligned 1nt short reads
averageBindings	expected short reads number aligned to a random location of genes of given size
windowSize	Window size around binding site. The total region would be 2*windowSize+1
acceptedRegionsOutputFile	Accepted binding regions
currentDir	Directory for I/O operations

**Value**

FRiPs Fraction of Reads in Peaks

---

DetectBindingSites     *Detect binding sites from motif*

---

**Description**

DETECT Binding sites with given motif and mismatch number as well genome/build, False Discovery Rate for a given experiment name.

This function is called by both [DetectBindingSitesBed](#) and [DetectBindingSitesMotif](#) with different input.

**Usage**

```
DetectBindingSites(
  From,
  BedFile,
  motif,
  mismatchNumber,
  chipSeq,
  genome,
  genomeBuild,
  DB = "UCSC",
  fdrValue = 0.05,
  windowSize = 100,
  GivenRegion = NA,
  currentDir
)
```

**Arguments**

From	Type of motif dataset either "Motif" or "Bed"
BedFile	Motif locations in bed format file
motif	motif characters in nucleotide IUPAC format

mismatchNumber	Number of mismatches allowed to match with motifs
chipSeq	ChIP-seq alignment both IP and background in 1nt bed format files
genome	The genome name such as "Hsapiens", "Mmusculus", "Dmelanogaster"
genomeBuild	The genome build such as "hg38", "hg19", "mm10", "dm3"
DB	The database of genome build. default: "UCSC"
fdrValue	FDR value cut-off
windowSize	Window size around binding site. The total region would be 2*windowSize+1
GivenRegion	granges of user provided binding regions
currentDir	Directory for I/O operations

**Value**

A list of FRiPs, sequence statistics, and Motif statistics

---

DetectBindingSitesBed *Detect binding sites from bed motif input*

---

**Description**

Takes user provided bed regions, and check for validity of them. Read bam or bed alignment files and convert to 1 nt bed and call detect binding site from 1nt bed.

**Usage**

```
DetectBindingSitesBed(
  BedFile,
  IPfiles,
  BackgroundFiles,
  genome,
  genomeBuild,
  DB = "UCSC",
  fdrValue = 0.05,
  expName = "Motif_Centric_Peaks",
  windowSize = 100,
  format = ""
)
```

**Arguments**

BedFile	Motif locations in bed format file
IPfiles	IP ChIP-seq alignment files
BackgroundFiles	Background ChIP-seq alignment files. Can be Input experiment, DNA whole extract, etc.

genome	The genome name such as "Hsapiens", "Mmusculus", "Dmelanogaster"
genomeBuild	The genome build such as "hg38", "hg19", "mm10", "dm3"
DB	The database of genome build. default: "UCSC"
fdrValue	FDR value cut-off
expName	The name of the output table
windowSize	Window size around binding site. The total region would be 2*windowSize+1
format	alignment format and should be one of these: "BAMPE", "BAMSE", "BEDPE", "BEDSE"

**Value**

peakCallingStatistics A list FRiPs, sequence statistics, and Motif statistics

**See Also**

[DetectBindingSitesMotif](#)

**Examples**

```
# FUR candidate motifs in NC_000913 E. coli
FurMotifs=system.file("extdata", "FurMotifs.bed", package="Motif2Site")

# ChIP-seq datasets in bed single end format
IPFe <- c(system.file("extdata", "FUR_fe1.bed", package="Motif2Site"),
          system.file("extdata", "FUR_fe2.bed", package="Motif2Site"))
Inputs <- c(system.file("extdata", "Input1.bed", package="Motif2Site"),
            system.file("extdata", "Input2.bed", package="Motif2Site"))
FURfeBedInputStats <-
  DetectBindingSitesBed(BedFile=FurMotifs,
                       IPfiles=IPFe,
                       BackgroundFiles=Inputs,
                       genome="Ecoli",
                       genomeBuild="20080805",
                       DB="NCBI",
                       expName="FUR_Fe_BedInput",
                       format="BEDSE"
  )
```

---

DetectBindingSitesMotif

*Detect binding sites from sequence motif sequence and mismatchNumber*

---

**Description**

DETECT Binding sites with given motif and mismatch number as well genome/build, False Discovery Rate for a given experiment name. Read bam or bed alignment files and convert to 1 nt bed and detect binding site among motifs from 1nt bed alignment.

**Usage**

```
DetectBindingSitesMotif(
  motif,
  mismatchNumber,
  IPfiles,
  BackgroundFiles,
  genome,
  genomeBuild,
  DB = "UCSC",
  fdrValue = 0.05,
  expName = "Motif_Centric_Peaks",
  windowSize = 100,
  format = "",
  GivenRegion = NA
)
```

**Arguments**

motif	motif characters in nucleotide IUPAC format
mismatchNumber	Number of mismatches allowed to match with motifs
IPfiles	IP ChIP-seq alignment files
BackgroundFiles	Background ChIP-seq alignment files. Can be Input experiment, DNA whole extract, etc.
genome	The genome name such as "Hsapiens", "Mmusculus", "Dmelanogaster"
genomeBuild	The genome build such as "hg38", "hg19", "mm10", "dm3"
DB	The database of genome build. default: "UCSC"
fdrValue	FDR value cut-off
expName	The name of the output table
windowSize	Window size around binding site. The total region would be 2*windowSize+1
format	alignment format and should be one of these: "BAMPE", "BAMSE", "BEDPE", "BEDSE"
GivenRegion	ranges of user provided binding regions

**Value**

A list FRiPs, sequence statistics, and Motif statistics

**See Also**[DetectBindingSitesBed](#)**Examples**

```
# ChIP-seq datasets in bed single end format
IPFe <- c(system.file("extdata", "FUR_fe1.bed", package="Motif2Site"),
          system.file("extdata", "FUR_fe2.bed", package="Motif2Site"))
Inputs <- c(system.file("extdata", "Input1.bed", package="Motif2Site"),
            system.file("extdata", "Input2.bed", package="Motif2Site"))

# Granages region for motif search
NC_000913_Coordinate <-
  GenomicRanges::GRanges(seqnames=S4Vectors::Rle("NC_000913"),
                          ranges=IRanges::IRanges(1, 4639675))

FURfeStringInputStats <-
  DetectBindingSitesMotif(motif="GWWTGAGAA",
                          mismatchNumber=1,
                          IPfiles=IPFe,
                          BackgroundFiles=Inputs,
                          genome="Ecoli",
                          genomeBuild="20080805",
                          DB="NCBI",
                          expName="FUR_Fe_StringInput",
                          format="BEDSE",
                          GivenRegion=NC_000913_Coordinate
  )
```

---

DetectFdrCutoffBH      *FDR cut-off detection Benjamini Hochberg method*

---

**Description**

Return FDR cut-off for a user provided fdrvalue using Benjamini Hochberg on main motif test data

**Usage**

```
DetectFdrCutoffBH(TestTableFile = "TestResults", fdrValue = 0.05)
```

**Arguments**

TestTableFile	test table which contains pvalues
fdrValue	FDR cut-off

**Value**

pvalue cut-off

---

 findMotifs

*Find motif instances with a certain mismatch number*


---

### Description

Find motif instances in a given genome. It gets motif strings and related allowed mismatch numbers and returns genomewide motif instances.

The genome and build information should be provided and relevant BS genomes packages such as BSgenome.Mmusculus.UCSC.mm10 or BSgenome.Hsapiens.UCSC.hg38 must be installed for the used genome and builds.

### Usage

```
findMotifs(
  motif,
  mismatchNumber,
  genome,
  genomeBuild,
  DB = "UCSC",
  mainCHRs = TRUE,
  firstCHR = FALSE,
  MotifLocationName = "Motif_Locations",
  limitedRegion = NA
)
```

### Arguments

motif	motif characters in nucleotide IUPAC format
mismatchNumber	Number of mismatch allowed to match with motif
genome	The genome name such as "Hsapiens", "Mmusculus", "Dmelanogaster"
genomeBuild	The genome build such as "hg38", "hg19", "mm10", "dm3"
DB	The database of genome build. default: "UCSC"
mainCHRs	If true only the major chromosome are considered, if FALSE Random, Uncharacterised, and Mitochondrial chromosomes are also considered
firstCHR	If true only Chr1 is used to find motifs. Default is FALSE
MotifLocationName	The name of the file of the motif locations
limitedRegion	If specified the motifs are detected in the provided granges

### Value

No return value

---

fitKernelDensity	<i>Fit a kernel density distribution to the obersever heuristic distribution</i>
------------------	--

---

**Description**

This function gets heuristic distribution of short reads around binding sites, total number of short reads and window size as number of nucleotide around binding sites.

It fits a kernel to the distribution and return the distribution as output. The total sum of returned values is equal to one.

**Usage**

```
fitKernelDensity(heuristicDistribution, totalShortReads, windowSize)
```

**Arguments**

heuristicDistribution	Original short distribution
totalShortReads	Total number of short reads
windowSize	Window size around binding site. The total region would be $2 * \text{windowSize} + 1$

**Value**

kernel returns fitted kernel distribution of short reads around binding sites

---

generate1ntBedAlignment	<i>Convert bam and bed files to 1 nucleotide bed</i>
-------------------------	--

---

**Description**

Take alignment files in bam or bed format and convert them to 1 nucleotide bed file

**Usage**

```
generate1ntBedAlignment(InputFile, bedFile, format = "")
```

**Arguments**

InputFile	Original alignment file name
bedFile	Name of output 1nt bed file
format	alignment format and should be one of these: "BAMPE", "BAMSE", "BEDPE", "BEDSE"

**Value**

No return value

---

Motif2Site

*Detect and Recenter binding sites from ChIP-seq experiments*

---

**Description**

Take ChIP-seq and motifs and detect Binding sites. It also combines/compares binding sites across experiments. Here is a synthetic example of differential Fur binding sites in E.coli:

**Examples**

```
# FUR candidate motifs in NC_000913 E. coli
FurMotifs=system.file("extdata", "FurMotifs.bed", package="Motif2Site")

# ChIP-seq datasets fe in bed single end format
IPFe <- c(system.file("extdata", "FUR_fe1.bed", package="Motif2Site"),
          system.file("extdata", "FUR_fe2.bed", package="Motif2Site"))
Inputs <- c(system.file("extdata", "Input1.bed", package="Motif2Site"),
            system.file("extdata", "Input2.bed", package="Motif2Site"))
FURfeBedInputStats <-
  DetectBindingSitesBed(BedFile=FurMotifs,
                        IPfiles=IPFe,
                        BackgroundFiles=Inputs,
                        genome="Ecoli",
                        genomeBuild="20080805",
                        DB="NCBI",
                        expName="FUR_Fe_BedInput",
                        format="BEDSE"
                        )

# ChIP-seq datasets dpd in bed single end format
IPDpd <- c(system.file("extdata", "FUR_dpd1.bed", package="Motif2Site"),
           system.file("extdata", "FUR_dpd2.bed", package="Motif2Site"))
FURdpdBedInputStats <-
  DetectBindingSitesBed(BedFile=FurMotifs,
                        IPfiles=IPDpd,
                        BackgroundFiles=Inputs,
                        genome="Ecoli",
                        genomeBuild="20080805",
                        DB="NCBI",
                        expName="FUR_Dpd_BedInput",
                        format="BEDSE"
                        )

# Combine all FUR binding sites into one table
corMAT <- recenterBindingSitesAcrossExperiments(
```

```

expLocations=c("FUR_Fe_BedInput", "FUR_Dpd_BedInput"),
experimentNames=c("FUR_Fe", "FUR_Dpd"),
expName="combinedFUR",
)
corMAT

# Differential binding sites across FUR conditions fe vs dpd
diffFUR <- pairwisDifferential(tableOfCountsDir="combinedFUR",
                             exp1="FUR_Fe",
                             exp2="FUR_Dpd",
                             FDRcutoff=0.05,
                             logFCcutoff=1
                             )

FeUp <- diffFUR[[1]]
DpdUp <- diffFUR[[2]]
TotalComparison <- diffFUR[[3]]
head(TotalComparison)

```

---

motifBindingNegativeBinomialCount

*Model IP and Input count values with negative Binomial*

---

## Description

Using edgeR TMM normalization and estimating dispersion as well as Adapting exact test function from edgeR to model IP vs Input counts.

To make this function memory efficient motifs into smaller sets and compute them separately and combine them at the end.

## Usage

```

motifBindingNegativeBinomialCount(
  countTableFile,
  replicateNumber,
  outputFile,
  currentDir
)

```

## Arguments

countTableFile	Table of counts which contains all IP and Input value raw counts
replicateNumber	experiment replicate number
outputFile	The name of the output file generated by this function
currentDir	Directory for I/O operations

**Value**

A dataframe includes fold enrichment, pvalue, and normalized count values

---

motifChipCount	<i>count short reads related to each motif for a given ChIPseq file</i>
----------------	---

---

**Description**

count 1nt short reads related to each motif for a given ChIPseq file.

**Usage**

```
motifChipCount(motifFile, chipFile, windowSize, outputName)
```

**Arguments**

motifFile	File contains motifs
chipFile	ChIP-seq 1nt alignment locations in bed format
windowSize	Window size around binding site. The total region would be 2*windowSize+1
outputName	Name of the output table

**Value**

Total number of short reads in motif reagrions

---

motifCount	<i>count short reads around motifs for all ChIP-seq experiments</i>
------------	---

---

**Description**

count short reads related to each motif for all ChIPseq files both IP and Input.

**Usage**

```
motifCount(motifFile, chipSeq, windowSize, outputName, currentDir)
```

**Arguments**

motifFile	File contains motifs
chipSeq	dataframe of ChIP-seq 1nt alignment location
windowSize	Window size around binding site. The total region would be 2*windowSize+1
outputName	Name of the output table
currentDir	Directory for I/O operations

**Value**

No return value

---

motifTablePreProcess *Process count data and perform negative binomial test*

---

**Description**

Remove unmapped regions, low and high binding regions and regions without fold change, and call negative binomial or nb test for the remaining regions.

**Usage**

```
motifTablePreProcess(countTableFile, outFile, currentDir)
```

**Arguments**

countTableFile Tabl of count values around motifs for all ChIP-seq experiments  
 outFile The name of the output file  
 currentDir Directory for I/O operations

**Value**

sequencingStatistics A dataframe consists of the ratio of non-sequenced, low-sequenced, ang high-sequenced regions.

---

NegativeBinomialTestWithReplicate

*Negative binomial test of binding using all replicates*

---

**Description**

Adapted exact test function from edgeR to compare IP vs Input with replicates. Input is a DGELIST with common and tag-wise dispersion has been already calculated by edgeR commands.

It calculates abundaces with mglnOneGroup identical to edgeR. logFE was calculated identical to edgeR. For the pvalue test negative binomial test is performed on the calculated abundance.

**Usage**

```
NegativeBinomialTestWithReplicate(object, prior.count = 0.125)
```

**Arguments**

object Table of counts which contains all IP and Input value counts, TMM normalized and contains dispersion values  
 prior.count edgeR prior value

**Value**

log fold enrichment, pvalue, and normalized count values

---

pairwisDifferential     *Detect differential motifs*

---

### Description

Take combined matrix of motif counts generated by [recenterBindingSitesAcrossExperiments](#), and experiment names. It detect differential motifs using edgeR TMM nomralizaiton with Generalized linear model

### Usage

```
pairwisDifferential(
  tableOfCountsDir = "",
  exp1,
  exp2,
  FDRcutoff = 0.05,
  logFCcutoff = 1
)
```

### Arguments

tableOfCountsDir	Directory which conatins the combined motifs and ChIP-seq count file
exp1	Experiment name which will be compared in pairwise comparison
exp2	Experiment name which will be compared in pairwise comparison
FDRcutoff	FDR cutoff applies on pvalue distribution
logFCcutoff	log fold change cutoff

### Value

A list of differential motifs, motif1 and motif2 as well as a table of total motifs and log fold changes

### See Also

[recenterBindingSitesAcrossExperiments](#)

### Examples

```
# FUR candidate motifs in NC_000913 E. coli
FurMotifs=system.file("extdata", "FurMotifs.bed", package="Motif2Site")

# ChIP-seq datasets fe in bed single end format
IPFe <- c(system.file("extdata", "FUR_fe1.bed", package="Motif2Site"),
          system.file("extdata", "FUR_fe2.bed", package="Motif2Site"))
Inputs <- c(system.file("extdata", "Input1.bed", package="Motif2Site"),
            system.file("extdata", "Input2.bed", package="Motif2Site"))
FURfeBedInputStats <-
```

```

DetectBindingSitesBed(BedFile=FurMotifs,
  IPfiles=IPFe,
  BackgroundFiles=Inputs,
  genome="Ecoli",
  genomeBuild="20080805",
  DB="NCBI",
  expName="FUR_Fe_BedInput",
  format="BEDSE"
)

# ChIP-seq datasets dpd in bed single end format
IPDpd <- c(system.file("extdata", "FUR_dpd1.bed", package="Motif2Site"),
  system.file("extdata", "FUR_dpd2.bed", package="Motif2Site"))
FURdpdBedInputStats <-
  DetectBindingSitesBed(BedFile=FurMotifs,
    IPfiles=IPDpd,
    BackgroundFiles=Inputs,
    genome="Ecoli",
    genomeBuild="20080805",
    DB="NCBI",
    expName="FUR_Dpd_BedInput",
    format="BEDSE"
  )

# Combine all FUR binding sites into one table
corMAT <- recenterBindingSitesAcrossExperiments(
  expLocations=c("FUR_Fe_BedInput", "FUR_Dpd_BedInput"),
  experimentNames=c("FUR_Fe", "FUR_Dpd"),
  expName="combinedFUR",
)

# Differential binding sites across FUR conditions fe vs dpd
diffFUR <- pairwiseDifferential(tableOfCountsDir="combinedFUR",
  exp1="FUR_Fe",
  exp2="FUR_Dpd",
  FDRcutoff=0.05,
  logFCcutoff=1
)

FeUp <- diffFUR[[1]]
DpdUp <- diffFUR[[2]]
TotalComparison <- diffFUR[[3]]
head(TotalComparison)

```

---

quiet

*Suppress messages generated by in external package*


---

## Description

mixtools and MASS::fitdistr generates warning by cat which is suppressed by this functions

**Usage**

```
quiet(func)
```

**Arguments**

func                    functional input call for which cat messages should be suppressed

**Value**

No return value

---

```
recenterBindingSitesAcrossExperiments
```

*Combine binding sites across experiments*

---

**Description**

Take experiment folder locations and experiment names and combine them into a combined matrix of motifs and ChIP-seq counts

Experiment folders must be generated either by [DetectBindingSitesBed](#) or [DetectBindingSitesMotif](#).

**Usage**

```
recenterBindingSitesAcrossExperiments(  
  expLocations,  
  experimentNames,  
  expName = "combinedData",  
  fdrValue = 0.05,  
  fdrCrossExp = 0.001  
)
```

**Arguments**

expLocations    The path to the experiment folders  
 experimentNames  
                   Name of the experiment to be used in combined ChIP-seq  
 expName        Name of the combined matrix  
 fdrValue       FDR cut-off to accept binding in each ChIP-seq experiments  
 fdrCrossExp    If no experiment fullfill this cutoff, the motif is not considered

**Value**

A pairwise Pearson correlation matrix across experiments

**See Also**

[pairwisDifferential](#)

**Examples**

```

# FUR candidate motifs in NC_000913 E. coli
FurMotifs=system.file("extdata", "FurMotifs.bed", package="Motif2Site")

# ChIP-seq datasets fe in bed single end format
IPFe <- c(system.file("extdata", "FUR_fe1.bed", package="Motif2Site"),
          system.file("extdata", "FUR_fe2.bed", package="Motif2Site"))
Inputs <- c(system.file("extdata", "Input1.bed", package="Motif2Site"),
            system.file("extdata", "Input2.bed", package="Motif2Site"))
FURfeBedInputStats <-
  DetectBindingSitesBed(BedFile=FurMotifs,
                        IPfiles=IPFe,
                        BackgroundFiles=Inputs,
                        genome="Ecoli",
                        genomeBuild="20080805",
                        DB="NCBI",
                        expName="FUR_Fe_BedInput",
                        format="BEDSE"
                      )

# ChIP-seq datasets dpd in bed single end format
IPDpd <- c(system.file("extdata", "FUR_dpd1.bed", package="Motif2Site"),
           system.file("extdata", "FUR_dpd2.bed", package="Motif2Site"))
FURdpdBedInputStats <-
  DetectBindingSitesBed(BedFile=FurMotifs,
                        IPfiles=IPDpd,
                        BackgroundFiles=Inputs,
                        genome="Ecoli",
                        genomeBuild="20080805",
                        DB="NCBI",
                        expName="FUR_Dpd_BedInput",
                        format="BEDSE"
                      )

# Combine all FUR binding sites into one table
corMAT <- recenterBindingSitesAcrossExperiments(
  expLocations=c("FUR_Fe_BedInput", "FUR_Dpd_BedInput"),
  experimentNames=c("FUR_Fe", "FUR_Dpd"),
  expName="combinedFUR",
)
corMAT

```

---

removeNonBellShapedMotifs

*Remove non-bell shape motifs prior to binding signal decomposition*


---

**Description**

Gets motif locations and related short reads and returns the motifs which are non-skewed ( $\text{abs}(\text{skewness}) < 0.3$ ) and more short reads binds closer to site.

It counts around motif with interval `windowSize` and `windowSize/2`, if the smaller window is less than half of the larger one then motif is not considered as Bell-shape

**Usage**

```
removeNonBellShapedMotifs(motifLocations, readLocations, windowSize)
```

**Arguments**

`motifLocations` A vector of motif locations  
`readLocations` A vector of `Int` short reads  
`windowSize` Window size around binding site. The total region would be  $2 * \text{windowSize} + 1$

**Value**

The coordinates of accepted motifs

---

<code>strongestMotif</code>	<i>Returns the motif with the highest count</i>
-----------------------------	---

---

**Description**

Gets motif locations and related short reads and returns the motif which include the highest number of short reads around it.

**Usage**

```
strongestMotif(motifLocations, readLocations, windowSize)
```

**Arguments**

`motifLocations` A vector of motif locations  
`readLocations` A vector of `Int` short reads  
`windowSize` Window size around binding site. The total region would be  $2 * \text{windowSize} + 1$

**Value**

The strongest motif

# Index

Bed2Granges, [2](#)

combine2Table, [3](#)  
combineMotifFiles, [4](#)  
combineTestResults, [4](#)  
compareBedFiless2UserProvidedRegions,  
[5, 8](#)  
CompareBeds2GivenRegions, [6](#)  
CompareMotifs2GivenRegions, [7](#)  
compareMotifs2UserProvidedRegions, [5, 7](#)  
computeFoldEnrichment, [9](#)

data, [9](#)  
decomposeBindingSignal, [11](#)  
DeleteMultipleFiles, [12](#)  
deriveHeuristicBindingDistribution, [12](#)  
DetectBindingSites, [13](#)  
DetectBindingSitesBed, [13, 14, 17, 26](#)  
DetectBindingSitesMotif, [13, 15, 15, 26](#)  
DetectFdrCutoffBH, [17](#)

findMotifs, [18](#)  
fitKernelDensity, [19](#)

generateIntBedAlignment, [19](#)

Motif2Site, [20](#)  
motifBindingNegativeBinomialCount, [21](#)  
motifChipCount, [22](#)  
motifCount, [22](#)  
motifTablePreProcess, [23](#)

NegativeBinomialTestWithReplicate, [23](#)

pairwisDifferential, [24, 26](#)

quiet, [25](#)

recenterBindingSitesAcrossExperiments,  
[24, 26](#)  
removeNonBellShapedMotifs, [27](#)  
strongestMotif, [28](#)