

Package ‘ChIPanalyser’

April 11, 2018

Type Package

Title ChIPanalyser: Predicting Transcription Factor Binding Sites

Version 1.0.0

Date 2017-09-01

Author Patrick C.N.Martin & Nicolae Radu Zabet

Maintainer Patrick C.N. Martin <pm16057@essex.ac.uk>

Description Based on a statistical thermodynamic framework, ChIPanalyser tries to produce ChIP-seq like profile.

The model relies on four consideration:

TF binding sites can be scored using a Position weight Matrix,

DNA accessibility plays a role in Transcription Factor binding,

binding profiles are dependant on the number of transcription

factors bound to DNA and finally binding energy

(another way of describing PWM's) or binding specificity should be

modulated (hence the introduction of a binding specificity modulator).

The end result of ChIPanalyser is to produce profiles simulating

real ChIP-seq profile and provide accuracy measurements

of these predicted profiles after being compared to real ChIP-seq data.

The ultimate goal is to produce ChIP-seq like profiles predicting ChIP-seq like profile to circumvent the need to produce costly ChIP-seq experiments.

License GPL-3

Collate 2AllS4Class_ProfileParameters.R 3AllGenerics.R 4AllMethods.R

AllInitialize.R AllShowMethods.R computeChIPProfile.R

computeOccupancy.R computeOptimal.R computePWMScore.R

genomeWidePWM.R GenomicProfileGenericFunctions.R

plotOccupancyProfile.R plotOptimalHeatMaps.R

profileAccuracyEstimate.R

Depends R (>= 3.4.1), GenomicRanges, Biostrings, BSgenome, RcppRoll

Imports methods, IRanges, S4Vectors, grDevices, graphics, stats, utils

Suggests BSgenome.Dmelandogaster, UCSC.dm3, knitr, RUnit, BiocGenerics

Encoding UTF-8

LazyData true

biocViews Software, BiologicalQuestion, WorkflowStep, Transcription, Sequencing, ChipOnChip, Coverage, Alignment, ChIPSeq, SequenceMatching, DataImport, PeakDetection

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

ChIPAnalyser-package	3
AllSitesAboveThreshold	7
averageExpPWMScore	8
backgroundSignal	9
backgroundSignal<-	10
boundMolecules	11
boundMolecules<-	12
BPFrequency	13
BPFrequency<-	14
ChIPAnalyserData	15
chipMean	16
chipMean<-	17
chipSd	18
chipSd<-	19
chipSmooth	20
chipSmooth<-	21
computeChipProfile	22
computeGenomeWidePWMScore	24
computeOccupancy	25
computeOptimal	27
computePWMScore	29
DNASequenceLength	31
DNASequenceLength<-	32
genomicProfileParameters	33
genomicProfileParameters-class	36
GRList-class	39
maxPWMScore	39
maxSignal	41
maxSignal<-	42
minPWMScore	43
naturalLog	44
naturalLog<-	45
NoAccess	46
noOfSites	47
noOfSites<-	48
occupancyProfileParameters	49
occupancyProfileParameters-class	51
PFMFormat	53
PFMFormat<-	54
ploidy	55
ploidy<-	56
plotOccupancyProfile	57
plotOptimalHeatMaps	59
PositionFrequencyMatrix	60
PositionFrequencyMatrix<-	61
PositionWeightMatrix	62
PositionWeightMatrix<-	63
profileAccuracyEstimate	64
PWMpseudocount	66
PWMpseudocount<-	67

PWMThreshold	68
PWMThreshold<-	69
removeBackground	70
removeBackground<-	71
ScalingFactorPWM	72
ScalingFactorPWM<-	73
searchSites	74
stepSize	75
stepSize<-	76
strandRule	77
strandRule<-	78
thetaThreshold	79
thetaThreshold<-	80
whichstrand	81
whichstrand<-	82

Index	83
--------------	-----------

ChIPAnalyser-package *ChIPAnalyser: Predicting Transcription Factor Binding Sites*

Description

Based on a statistical thermodynamic framework, ChIPAnalyser tries to produce ChIP-seq like profile. The model relies on four consideration: TF binding sites can be scored using a Position weight Matrix, DNA accessibility plays a role in Transcription Factor binding, binding profiles are dependant on the number of transcription factors bound to DNA and finally binding energy (another way of describing PWM's) or binding specificity should be modulated (hence the introduction of a binding specificity modulator). The end result of ChIPAnalyser is to produce profiles simulating real ChIP-seq profile and provide accuracy measurements of these predicted profiles after being compared to real ChIP-seq data. The ultimate goal is to produce ChIP-seq like profiles predicting ChIP-seq like profile to circumvent the need to produce costly ChIP-seq experiments.

Details

The DESCRIPTION file:

```

Package:      ChIPAnalyser
Type:         Package
Title:        ChIPAnalyser: Predicting Transcription Factor Binding Sites
Version:      1.0.0
Date:         2017-09-01
Author:       Patrick C.N.Martin & Nicolae Radu Zabet
Maintainer:  Patrick C.N. Martin <pm16057@essex.ac.uk>
Description:  Based on a statistical thermodynamic framework, ChIPAnalyser tries to produce ChIP-seq like profile.
License:      GPL-3
Collate:     2AllS4Class_ProfileParameters.R 3AllGenerics.R 4AllMethods.R AllInitialize.R AllShowMethods.R
Depends:     R (>= 3.4.1), GenomicRanges, Biostrings, BSgenome, RcppRoll
Imports:     methods, IRanges, S4Vectors, grDevices, graphics, stats, utils
Suggests:    BSgenome.Dmelanogaster, UCSC.dm3, knitr, RUnit, BiocGenerics
Encoding:    UTF-8
LazyData:    true

```

biocViews: Software, BiologicalQuestion, WorkflowStep, Transcription, Sequencing, ChipOnChip, Coverage, Ali
 VignetteBuilder: knitr

Index of help topics:

AllSitesAboveThreshold	Accessor for 'AllSitesAboveThreshold' slot in a 'genomicProfileParameters' object.
BPFrequency	Accessor method for 'BPFrequency' slot in a 'genomicProfileParameters' object.
BPFrequency<-	Setter method for 'BPFrequency' slot in a 'genomicProfileParameters' object.
ChIPAnalyser	ChIPAnalyser: Predicting Transcription Factor Binding Sites
ChIPAnalyserData	ChIPAnalyserData
DNASequenceLength	Accessor method for 'DNASequenceLength' slot in a 'genomicProfileParameters'
DNASequenceLength<-	Setter Method for 'DNASequenceLength' slot in a 'genomicProfileParameters' object.
GRList-class	Class "'GRList'"
NoAccess	Accessor Method for the 'NoAccess' slot in a 'genomicProfileParameters' object.
PFMFormat	Accesor method for the 'PFMFormat' slot in a 'genomicProfileParameters' object
PFMFormat<-	Setter method for the 'PFMFormat' slot in a 'genomicProfileParameters' object
PWMThreshold	Accessor method for the 'PWMThreshold' slot in a 'genomicProfileParameters' object
PWMThreshold<-	Setter Method for the 'PWMThreshold' slot in a 'genomicProfileParameters' object
PWMpseudocount	Accessor Method for a 'PWMpseudocount' slot in a 'genomicProfileParameters'
PWMpseudocount<-	Setter Method for the 'pseudocount' slot in a 'genomicProfileParameters' object
PositionFrequencyMatrix	Accessor method for the 'PFM' slot in a 'genomicProfileParameters' object
PositionFrequencyMatrix<-	Setter method for the 'PFM' slot in a 'genomicProfileParameters' object
PositionWeightMatrix	Accessor Method for the 'PWM' slot in a 'genomicProfileParameters' object
PositionWeightMatrix<-	Setter Method for the 'PositionWeightMatrix' slot in a 'genomicProfileParameters' object
ScalingFactorPWM	Accessor method for the 'lambda' slot in a 'genomicProfileParameters' object
ScalingFactorPWM<-	Setter method for the 'lambda' slot in a 'genomicProfileParameters' object
averageExpPWMScore	Accessor for 'averageExpPWMScore' slot in a 'genomicProfileParameters' object.


```

ploidy                Accessor method for the 'ploidy' slot in a
                       'occupancyProfileParameters' object
ploidy<-              Setter Method for the 'ploidy' slot in an
                       'occupancyProfileParameters' object
plotOccupancyProfile Plot Occupancy Profiles
plotOptimalHeatMaps  Heat Map of optimal Parameters
profileAccuracyEstimate Estimating Accuracy of predicted Profiles
removeBackground      Accessor Method for the 'removeBackground' slot
                       in a 'occupancyProfileParameters' object
removeBackground<-   Setter Method for the 'removeBackground' slot
                       in a 'occupancyProfileParameters' object
searchSites           Searching function for Sites above threshold
                       and predicted ChIP-seq Profiles
stepSize              Accessor method of the 'stepSize' slot in
                       'occupancyProfileParameters' object
stepSize<-           Setter Method for the 'stepSize' slot in a
                       'occupancyProfileParameters'
strandRule            Accessor Method for the 'strandRule' slot in a
                       'genomicProfileParameters' object
strandRule<-         Setter method for the 'strandRule' slot in a
                       'genomicProfileParameters' object.
thetaThreshold        Accessor method for the 'thetaThreshold' slot
                       in a 'occupancyProfileParameters' object
thetaThreshold<-     Setter method the 'thetaThreshold' slot in a
                       'occupancyProfileParameters' object
whichstrand           Accessor method for the 'whichstrand' slot in a
                       'genomicProfileParameters' object
whichstrand<-        Setter method for the 'whichstrand' slot in a
                       'genomicProfileParameters' object

```

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>
 And
 Nicolae Radu Zabet <nzabet@essex.ac.uk>
 Maintainer: Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```

#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){

```

```

source("https://bioconductor.org/biocLite.R")
biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,BPFrequency=DNASequenceSet)
OPP <- occupancyProfileParameters()

# Computing Genome Wide
GenomeWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
    genomicProfileParameters = GPP)

#Compute PWM Scores
PWMScores <- computePWMScore(DNASequenceSet = DNASequenceSet,
    genomicProfileParameters = GenomeWide,
    setSequence = eveLocus, DNAAccessibility = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(AllSitesPWMScore = PWMScores,
    occupancyProfileParameters = OPP)

#Compute ChIP profiles
chipProfile <- computeChipProfile(setSequence = eveLocus,
    occupancy = Occupancy, occupancyProfileParameters = OPP)
#Estimating accuracy estimate
AccuracyEstimate <- profileAccuracyEstimate(LocusProfile = eveLocusChip,
    predictedProfile = chipProfile, occupancyProfileParameters = OPP)

```

AllSitesAboveThreshold

Accessor for AllSitesAboveThreshold slot in a genomicProfileParameters object.

Description

Extract or Access AllSitesAboveThreshold slot in [genomicProfileParameters](#) object.

Usage

```
AllSitesAboveThreshold(object)
```

Arguments

object [genomicProfileParameters](#) object.

Details

As a general rule, AllSitesAboveThreshold is computed and updated internally in [computePWMScore](#), [computeOccupancy](#), [computeChipProfile](#). Ideally, this slot should not be updated by user. There are three different occurrences of this slot. As a [GRanges](#), AllSitesAboveThreshold represents the sites of *one* locus that are above [PWMThreshold](#). As a [GRangesList](#), AllSitesAboveThreshold represents the sites of *multiple* loci that are above [PWMThreshold](#). As a list, AllSitesAboveThreshold

represents a list of [GRanges](#) or [GRangesList](#) (as described above). When computing occupancy [computeOccupancy](#), it is possible to use multiple values of [ScalingFactorPWM](#) and [boundMolecules](#). Each element in the list represent a combination of [ScalingFactorPWM](#) and [boundMolecules](#). The specific combination of each element can be found by using [names](#) of [AllSitesAboveThreshold](#) or by using [ScalingFactorPWM](#) and [boundMolecules](#).

Value

Returns a [GRanges](#), [GRangesList](#) or list containing containing sites with a Position Weight Matrix Score above a [PWMThreshold](#).

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Accessing Data
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata", package="ChIPAnalyser"), "BCDSLx.pfm")
# Building genomicProfileParameters object
GPP <- genomicProfileParameters(PFM=PFM)
# Extracting AllSitesAboveThreshold slot
AllSitesAboveThreshold(GPP)

## Note this slot is now empty as nothing has yet been computed
```

averageExpPWMScore *Accessor for averageExpPWMScore slot in a genomicProfileParameters object.*

Description

Extract or Access averageExpPWMScore slot in a [genomicProfileParameters](#)

Usage

```
averageExpPWMScore(object)
```

Arguments

object object is a [genomicProfileParameters](#)

Details

As a general rule, `averageExpPWMScore` is computed and updated internally by `computeGenomeWidePWMScore`. Ideally, this slot should not be updated by user. The `averageExpPWMScore` is the sum of the exponential of every PWM score for a given DNA sequence and divided by the length of the said DNA sequence (`DNASequenceLength`). This can either be the full length sequence or only the accessible sequence (see `computeGenomeWidePWMScore`).

Value

Returns the `averageExpPWMScore` of a `genomicProfileParameters` when computed.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Accessing Data
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata", package="ChIPAnalyser"), "BCDS1x.pfm")
# Building genomicProfileParameters object
GPP <- genomicProfileParameters(PFM=PFM)
# Extracting AllSitesAboveThreshold slot
averageExpPWMScore(GPP)

## Note this slot is now empty as nothing has yet been computed
```

`backgroundSignal` *Accessor method for the `backgroundSignal` slot in a `occupancyProfileParameters` object.*

Description

Extract or access the `backgroundSignal` slot in a `occupancyProfileParameters` object.

Usage

```
backgroundSignal(object)
```

Arguments

`object` `object` is an `occupancyProfileParameters`

Details

Default value: 0. When computing `computeOccupancy`, a ChIP-seq background signal is used to scale Occupancy by considering both a `backgroundSignal` and a `maxSignal`. The `backgroundSignal` is also used to normalise occupancies to `maxOccupancy`. The `backgroundSignal` usually comes from experimental data and is provided by user. As a general rule, if ChIP-seq data is available and will be used in `computeChipProfile`, `profileAccuracyEstimate` or `plotOccupancyProfile`, it is advised to use the `backgroundSignal` from this data. We strongly encourage to set values when building a `occupancyProfileParameters` object.

Value

Returns a `occupancyProfileParameters` object with a new value assigned to the `averageExpPWMScore` slot.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
# Setting new value for backgroundSignal
backgroundSignal(OPP) <- 0.2
# Viewing whole object with new updated value
OPP
#Viewing single value in object
backgroundSignal(OPP)
```

`boundMolecules` *Accessor methods for boundMolecules slot in*
occupancyProfileParameters object.

Description

Extract or Access `boundMolecules` slot in `occupancyProfileParameters` object.

Usage

```
boundMolecules(object)
```

Arguments

`object` `object` is a `occupancyProfileParameters` object.

Details

Default value: 1000

When computing occupancy ([computeOccupancy](#)), a value for the number of bound Molecules to DNA is needed. This value can be updated and set in a [occupancyProfileParameters](#) object. If the number of molecules is unknown, it is possible to infer this value with [computeOptimal](#). We strongly encourage to set values when building a [occupancyProfileParameters](#) object.

Value

Returns boundMolecules slot in [occupancyProfileParameters](#) object.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
#Checking single value by slot accessor
boundMolecules(OPP)
```

`boundMolecules<-` *Setter method for the boundMolecules slot in a [occupancyProfileParameters](#) object.*

Description

Setter method for the boundMolecules slot in a [occupancyProfileParameters](#) object.

Usage

```
boundMolecules(object)<-value
```

Arguments

object	object is a occupancyProfileParameters object.
value	value is a positive integer or vector of positive integers describing the number of molecules bound to DNA. Default value is 1000.

Details

Default value: 1000 When computing occupancy ([computeOccupancy](#)), a value for the number of bound Molecules to DNA is needed. This value can be updated and set in a [occupancyProfileParameters](#) object. If the number of molecules is unknown, it is possible to infer this value with [computeOptimal](#). We strongly encourage to set values when building a [occupancyProfileParameters](#) object.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
data(ChIPanalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata", package="ChIPanalyser"), "BCDSLx.pfm")
# Building genomicProfileParameters object
GPP <- genomicProfileParameters(PFM=PFM)
#Extracting BPFrequency slot
BPFrequency(GPP)
```

BPFrequency<- *Setter method for BPFrequency slot in a genomicProfileParameters object.*

Description

Setter method for BPFrequency slot in a [genomicProfileParameters](#) object. If base pair frequency is unknown, BPFrequency will compute base pair frequency from a DNA sequence.

Usage

```
BPFrequency(object)<-value
```

Arguments

object object is a [genomicProfileParameters](#) object.

value value can three different objects:

- A vector of length 4 containing the probability of occurrence of each base pair (A,C,G,T) in order. Default value is `c(0.25,0.25,0.25,0.25)`.
- A `BSgenome` of the organism of interest. The base pair frequency will automatically be computed and updated in [genomicProfileParameters](#).
- A `DNAStrngSet` of the organisme of interest. The base pair frequency will automatically be computed and updated in [genomicProfileParameters](#) (Preferred method).

Details

Default value is `c(0.25,0.25,0.25,0.25)` When generating a Postion Weight Matrix from a Position Frequency Matrix, the probability of occurrence of each base pair (Base Pair Frequency) is necessary (as originally described by Gary Stormo). It is possible to set custom values for BPFrequency with a vector of length 4 containing the probability of occurrence of each base pair (A,C,G,T) in order. If Base pair frequency is unknown, BPFrequency will compute base pair frequency from a DNA sequence when building a [genomicProfileParameters](#) object. The nature of this sequence can be a `BSgenome` object or a `DNAStrngSet`. In order to decrease run time, it is advised to use [DNAStrngSet](#).

Value

Returns a [genomicProfileParameters](#) object with an updated value for BPFrequency.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata", package="ChIPAnalyser"), "BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
# Building genomicProfileParameters object
GPP <- genomicProfileParameters(PFM=PFM, BPFrequency=DNASequenceSet)
# Updating BPFrequency
## !! Note!! BPFrequency is used to compute PWM from PFM
## IF updated after building GPP, then it will not influence PWM
## Advised to build with BPFrequency directly
BPFrequency(GPP) <- DNASequenceSet
BPFrequency(GPP) <- c(0.25,0.25,0.25,0.25)
```

ChIPAnalyserData

ChIPAnalyserData

Description

ChIPAnalyserData is derived from real biological data. The source organism is *Drosophila melanogaster*. The data can be described as genomic data as it contains DNA sequences, loci, genetic information, DNA accessibility data and ChIP-seq data.

Usage

```
data(ChIPAnalyserData)
```

Format

1. Accessis [GRanges](#) containing DNA Accesibility data for the sequences described above.
2. eveLocusis [GRanges](#) containing a locus of interest. In this case *eve strip Locus* on chromosome 2R in *Drosophila melanogaster*
3. eveLocusChIPis a named list containing real ChIP-seq data (normalised to each base pair) of the eve strip locus in *Drosophila melanogaster*.
4. geneRefis a [GRanges](#) containing genetic information (exon, intron, 3'UTR, 5'UTR) for the sequence described above.

Value

Returns a set of Rdata objects as described above.

Source

Transcription Factor PFM: Berkeley Drosophila Transcription Network Project (bdtnp.lbl.gov)

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
data(ChIPanalyserData)
```

chipMean	<i>Accessor method for chipMean slot in a occupancyProfileParameters object.</i>
----------	--

Description

Accessor method for [chipMean](#) slot in a [occupancyProfileParameters](#) object.

Usage

```
chipMean(object)
```

Arguments

object object is a [occupancyProfileParameters](#)

Details

Default vlaue : 150 When computing ChIP-seq like profiles ([computeChipProfile](#), the occupancy values given by [computeOccupancy](#) are transformed into ChIP-seq like profiles. The average size of a ChIP-seq peak was described by Kaplan (Kaplan et al. , 2011). It is advised to use the average width of ChIP peaks from actual ChIP-seq data. We strongly encourage to set values when building a [occupancyProfileParameters](#) object.

Value

Returns chipMean slot from a [occupancyProfileParameters](#) object.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Kaplan T., Li X.-Y., Sabo P.J., Thomas S., Stamatoyannopoulos J.A., Biggin M.D., Eisen M.B. Quantitative models of the mechanisms that control genome-wide patterns of transcription factor binding during early *Drosophila* development, *PLoS Genet.*, 2011, vol. 7 pg. e1001290

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
#Accessing chipMean slot in OPP
chipMean(OPP)
```

chipMean<-	<i>Access methods for chipMean slot in occupancyProfileParameters object.</i>
------------	---

Description

Access methods for chipMean slot in [occupancyProfileParameters](#) object.

Usage

```
chipMean(object)<-value
```

Arguments

object	object is a occupancyProfileParameters object.
value	value is a positive numeric value that will be assigned to the chipMean slot. chipMean describes the average size of a ChIP-seq peak in base pairs.

Details

Default value : 150 When computing ChIP-seq like profiles ([computeChipProfile](#), the occupancy values given by [computeOccupancy](#) are transformed into ChIP-seq like profiles. The average size of a ChIP-seq peak was described by Kaplan (Kaplan et al. , 2011). It is advised to use the average width of ChIP peaks from actual ChIP-seq data. We strongly encourage to set values when building a [occupancyProfileParameters](#) object.

Value

Returns a [occupancyProfileParameters](#) object with an updated value for chipMean slot.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Kaplan T., Li X.-Y., Sabo P.J., Thomas S., Stamatoyannopoulos J.A., Biggin M.D., Eisen M.B. Quantitative models of the mechanisms that control genome-wide patterns of transcription factor binding during early *Drosophila* development, *PLoS Genet.*, 2011, vol. 7 pg. e1001290

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
# Setting new value for slot
chipMean(OPP) <- 250
```

chipSd	<i>Accessor method for chipSd slot in a occupancyProfileParameters object.</i>
--------	--

Description

Access or Extract chipSd slot in a [occupancyProfileParameters](#) object.

Usage

```
chipSd(object)
```

Arguments

object object is a [occupancyProfileParameters](#)

Details

When computing ChIP-seq like profiles ([computeChipProfile](#), the occupancy values given by [computeOccupancy](#) are transformed into ChIP-seq like profiles. The average size of a ChIP-seq peak was described by Kaplan (Kaplan et al. , 2011). The average peak size is subject to variation. This variation is accounted for with [chipSd](#). It is advised to use the standard deviation of ChIP peak width from actual ChIP-seq data. We strongly encourage to set values when building a [occupancyProfileParameters](#) object.

Value

Returns a [occupancyProfileParameters](#) object with an updated value for [chipSd](#).

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Kaplan T, Li X.-Y., Sabo P.J., Thomas S., Stamatoyannopoulos J.A., Biggin M.D., Eisen M.B. Quantitative models of the mechanisms that control genome-wide patterns of transcription factor binding during early *Drosophila* development, *PLoS Genet.*, 2011, vol. 7 pg. e1001290

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
# Accessing chipSd slot
chipSd(OPP)
```

chipSd<- *Setter methods for chipSd slot in a [occupancyProfileParameters](#) object.*

Description

Setter methods for chipSd slot in a [occupancyProfileParameters](#) object.

Usage

```
chipSd(object)<-value
```

Arguments

object	object is occupancyProfileParameters object.
value	value is a positive numeric value that will be assigned to chipSd slot. Default value is 150.

Details

When computing ChIP-seq like profiles ([computeChipProfile](#), the occupancy values given by [computeOccupancy](#) are transformed into ChIP-seq like profiles. The average size of a ChIP-seq peak was described by Kaplan (Kaplan et al. , 2011). The average peak size is subject to variation. This variation is accounted for with `chipSd`. It is advised to use the standard deviation of ChIP peak width from actual ChIP-seq data. We strongly encourage to set values when building a [occupancyProfileParameters](#) object.

Value

Returns a [occupancyProfileParameters](#) object with an updated value for `chipSd`.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Kaplan T, Li X.-Y., Sabo P.J., Thomas S., Stamatoyannopoulos J.A., Biggin M.D., Eisen M.B. Quantitative models of the mechanisms that control genome-wide patterns of transcription factor binding during early *Drosophila* development, *PLoS Genet.*, 2011, vol. 7 pg. e1001290

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
# Setting new value for chipSd slot
chipSd(OPP) <- 250
```

chipSmooth *Accessor methods for chipSmooth slot in a
occupancyProfileParameters object.*

Description

Access or Extract chipSmooth slot in a [occupancyProfileParameters](#) object.

Usage

```
chipSmooth(object)
```

Arguments

object object is a [occupancyProfileParameters](#) object.

Details

When computing ChIP-seq like ([computeChipProfile](#)) profile from occupancy data (see [computeOccupancy](#)), the profiles are smoothed using a window of a given size. The default value is set at 250 base pairs. If chipSmooth is set to 0 then the profile will not be smoothed. We strongly encourage to set values when building a [occupancyProfileParameters](#) object.

Value

Returns the chipSmooth slot in an [occupancyProfileParameters](#) object.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
# Accessing chipSd slot
chipSmooth(OPP)
```

chipSmooth<- *Setter method for chipSmooth slot in [occupancyProfileParameters](#) object.*

Description

Setter method for chipSmooth slot in [occupancyProfileParameters](#) object.

Usage

```
chipSmooth(object) <- value
```

Arguments

object object is a [occupancyProfileParameters](#) object.
value value is the positive numeric value to be assigned to the chipSmooth slot in [occupancyProfileParameters](#) Default value is 250 base pairs.

Details

When computing ChIP-seq like ([computeChipProfile](#)) profile from occupancy data (see [computeOccupancy](#)), the profiles are smoothed using a window of a given size. The default value is set at 250 base pairs. If chipSmooth is set to 0 then the profile will not be smoothed. We strongly encourage to set values when building a [occupancyProfileParameters](#) object.

Value

Returns a [occupancyProfileParameters](#) object with an updated value for chipSmooth slot.

Author(s)

Patrick C.N Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
# Setting new value for chipSd slot
chipSmooth(OPP) <- 250
```

computeChipProfile *Computing ChIP-seq like profiles from Occupancy data.*

Description

computeChipProfile compute ChIP-seq like profile from occupancy data. Occupancy data is computed using [computeOccupancy](#).

Usage

```
computeChipProfile(setSequence, occupancy, occupancyProfileParameters = NULL,
  norm = TRUE, method = c("moving_kernel", "truncated_kernel", "exact"),
  peakSignificantThreshold= NULL, verbose = TRUE)
```

Arguments

- | | |
|----------------------------|---|
| setSequence | setSequence is a GRanges containing the loci of interest. It is strongly advised to name each loci / range in the GRanges . |
| occupancy | occupancy is a <code>link{genomicProfileParameters}</code> object result of the computeOccupancy function. This genomicProfileParameters object should contain a list, GRanges or a GRangesList with both PWMscore and Occupancy as meta data columns. To check if your object contains the right data, please see AllSitesAboveThreshold . |
| occupancyProfileParameters | occupancyProfileParameters is an occupancyProfileParameters object containing the desired values for each Parameter. If left as NULL, computeChipProfile will generate a new occupancyProfileParameters object using the default values (see occupancyProfileParameters). |
| norm | norm is a logical value. If TRUE, the ChIP-seq like profile will be normalised towards maximum Occupancy. If FALSE, the profile will be left as is. |
| method | method is a character string of one of the following: <code>c("moving_kernel", "truncated_kernel", "exact")</code> . If set to <code>moving_kernel</code> , the peaks will be approximated using <code>Rcpp</code> (Default). If set to <code>truncated_kernel</code> , the peaks will be approximated however this method does not require <code>Rcpp</code> . If set to <code>exact</code> , the peaks will not be approximated. |
| peakSignificantThreshold | peakSignificantThreshold is a threshold at which peaks will be selected. IMPORTANT: if you select "moving_kernel" as described in method then this threshold is a numeric value describing the peak tail height cutoff value (Default = 0.001). In the case of "truncated_kernel" and "exact", the threshold represents a distance in base pair from the peak summit at which the peak should be cut (Default = 1250). The default is set to NULL in this function. This just means that either the value is provided by user with the appropriate method. If not, the default will be selected depending on the method selected. |
| verbose | verbose is a logical value. If TRUE, progress messages will be displayed in console. If FALSE, no progress messages will be displayed in console. |

Details

computeChipProfile converts Transcription Factor occupancy to a profile resembling the one of a ChIP-seq profile. A certain set of Parameters are required in order to build ChIP profiles.

These Parameters are defined and stored in a `occupancyProfileParameters` object. These parameters are: `chipMean`, `chipSd`, `chipSmooth`, `stepSize`, `backgroundSignal`, `maxSignal` and `removeBackground`. All these Parameters have default values already stored. However, for an optimal fit, it is advised to derive these values from actual ChIP-seq data. For more information on these parameters, see `occupancyProfileParameters`. This function also requires a set of sequences in form of a `GRanges`. The sequence set are the loci of interest on which the ChIP-seq profile will be computed.

Value

Returns a list containing all ChIP-seq like profile for every combination of `ScalingFactorPWM` and `boundMolecules`. The correlation and Mean Squared Error between the predicted ChIP profile and actual ChIP-seq profile for the same loci will vary depending on the value given for `ScalingFactorPWM`

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Extracting Data
data(ChIPanalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata", package="ChIPanalyser"), "BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
# Building genomicProfileParameters object
GPP <- genomicProfileParameters(PFM=PFM, BPFrequency=DNASequenceSet)

OPP <- occupancyProfileParameters()

# Computing Genome Wide
GenomeWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP)

#Compute PWM Scores
PWMScores <- computePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GenomeWide,
  setSequence = evelocus, DNAAccessibility = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(AllSitesPWMScore = PWMScores,
  occupancyProfileParameters = OPP)
```

```
#Compute ChIP profiles
chipProfile <- computeChipProfile(setSequence = eveLocus,
  occupancy = Occupancy, occupancyProfileParameters = OPP)
chipProfile
```

```
computeGenomeWidePWMScore
```

Computing genome wide PWM scores.

Description

computeGenomeWidePWMScore scores the entire genome with a Position Weight Matrix. It is possible to restrict the scoring of the genome to only Accessible DNA.

Usage

```
computeGenomeWidePWMScore(DNASequenceSet, genomicProfileParameters,
  DNAAccessibility = NULL, verbose = TRUE)
```

Arguments

DNASequenceSet DNASequenceSet is a BSgenome or a [DNAStrngSet](#). The DNASequenceSet is the sequence of the organism of interest. For a question of ease, it is advised to use a [DNAStrngSet](#)

genomicProfileParameters

genomicProfileParameters is a [genomicProfileParameters](#) object. This object contains parameters needed to compute computeGenomeWidePWMScore. GenomicProfileParameters also contain the Position Weight Matrix. See [genomicProfileParameters](#) for more information.

DNAAccessibility

DNAAccessibility is a [GRanges](#) object containing Accessible DNA. If provided, the score will be score only on accesible DNA instead of genome wide.

verbose

verbose is a logical value. If TRUE, progress messages will be displayed in console. If FALSE, no progress message will be displayed.

Details

After creating a [genomicProfileParameters](#) object, the genome can be scored in order to extract information required for further analysis. These values will be [maxPWMScore](#), [minPWMScore](#), [averageExpPWMScore](#) and finally [DNASequenceLength](#). These values will be used in [computePWMScore](#).

Value

Returns a [genomicProfileParameters](#) object. This object will contain updated value for certain slots [maxPWMScore](#), [minPWMScore](#), [averageExpPWMScore](#) and finally [DNASequenceLength](#). It should be mentioned that certain slot will remain empty as they will be computed in other functions.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <-DNAStringSet(BSgenome.Dmelanogaster.UCSC.dm3[["chr2R"]])
names(DNASequenceSet) <- "chr2R"

# Building genomicProfileParameters object
GPP <- genomicProfileParameters(PFM=PFM, BPFrequency=DNASequenceSet)

# Computing Genome Wide
GenomeWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP)
# Computing Genome wide with DNA accessibility
GenomeWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP, DNAAccessibility = Access,
  GenomeWide = FALSE)
```

computeOccupancy

Compute Occupancy values from PWM Scores based on model.

Description

computeOccupancy will compute the Occupancy from PWM Scores. As described in detail in the vignette, ChIPAnalyser uses PWM Scores, DNA Accessibility data, the number of bound molecules and a scaling factor of Transcription Factor specificity. This function will compute occupancy using the values assigned to each variable.

Usage

```
computeOccupancy(AllSitesPWMScore, occupancyProfileParameters = NULL,
  norm = TRUE, verbose = TRUE)
```

Arguments**AllSitesPWMScore**

AllSitesPWMScore is a [genomicProfileParameters](#) object resulting from [computePWMScore](#). IT is important to use this resulting object as the occupancy will only be computed for sites above a threshold.

occupancyProfileParameters

occupancyProfileParameters is a [occupancyProfileParameters](#) object containing the adequate values assigned to each Parameter. If not Supplied (occupancyProfileParameter = NULL), a new object will be created internally using default values.

norm

norm a logical value which determines if the occupancy should be normalised or not.

verbose

verbose a logical value which determines if progress messages are printed or not.

Details

computeOccupancy will compute the Occupancy from PWM Scores. As described in detail in the vignette, ChIPanalyser uses PWM Scores, DNA Accessibility data, the number of bound molecules and a scaling factor of Transcription Factor specificity. This function will compute occupancy using the values assigned to each variable. It should also be noted that the [occupancyProfileParameters](#) object contains a set of parameters used to compute Occupancy (not only restricted to this). These parameters are often dependant on real ChIP-Seq data and will influence the goodness of fit between the predicted model and real ChIP-seq data. We strongly advise that the values assigned to each parameter should be customized in order to increase the model agreement with real world biological data.

Value

computeOccupancy will return a [genomicProfileParameters](#). The main difference will reside in the [AllSitesAboveThreshold](#) slot. This slot is generally a list or [GRangesList](#). Within these list type structures are enclosed [GRanges](#) containing the positions of site above threshold, PWM-Scores and Occupancy for each site. The series of GRanges will depend on the number of loci that are tested and the number of element in the list will depend on the various combinations of ScalingFactorPWM and boundMolecules.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Data extraction
data(ChIPanalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata", package="ChIPanalyser"), "BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome
```

```

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,BPFrequency=DNASequenceSet)
OPP <- occupancyProfileParameters()
# Computing Genome Wide
GenomeWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP)

#Compute PWM Scores
PWMScores <- computePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GenomeWide, setSequence = eveLocus,
  DNAAccessibility = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(AllSitesPWMScore = PWMScores,
  occupancyProfileParameters = OPP)
Occupancy

```

computeOptimal

compute Optimal Parameters

Description

ChIPAnalyser contains a set of functions some of which require two parameters known as [ScalingFactorPWM](#) and as [boundMolecules](#). These two parameters are not always known. `computeOptimal` will compute these values by maximising the correlation and minimising the Mean Squared Error between a predicted ChIP-seq-like profile and a real ChIP-seq profile for a given loci.

Usage

```

computeOptimal(DNASequenceSet, genomicProfileParameters, LocusProfile,
  setSequence, DNAAccessibility = NULL,
  occupancyProfileParameters = NULL, parameter = "all",
  peakMethod="moving_kernel")

```

Arguments

DNASequenceSet DNASequenceSet is a [DNAStringSet](#) or a BSgenome of the full sequence of the organism of interest.

genomicProfileParameters

genomicProfileParameters is a [genomicProfileParameters](#) object containing at least a Position Frequency Matrix or a Position Weight Matrix. It is strongly advised to customize this object to increase goodness of fit of the model when compared to real ChIP-seq data.

LocusProfile	LocusProfile is a named list containing ChIP-seq enrichments for each Loci of interest. This Profile should be normalised to a base pair level. In other words, there should be an enrichment score for each base pair of a given Locus.
setSequence	setSequence is a GRanges containing the Loci of interest.
DNAAccessibility	DNAAccessibility is a GRanges object containing Accessible DNA sites.
occupancyProfileParameters	occupancyProfileParameters is a occupancyProfileParameters object. If this object is not provided (occupancyProfileParameters = NULL), a new object will be created internally. However, it is strongly advised to tailor this object to maximise the goodness of fit of the model when compared to ChIP-seq data.
parameter	parameter is a character string which determines which parameter should be computed. There are four options c("correlation", "MSE", "theta", "all")
peakMethod	peakMethod is a character string of one of the following: c("moving_kernel", "truncated_kernel", "exact"). If set to moving_kernel, the peaks will be approximated using Rcpp (Default). If set to truncated_kernel, the peaks will be approximated however this method does not require Rcpp. If set to exact, the peaks will not be approximated.

Details

In order to backward infer the values of [ScalingFactorPWM](#) and [boundMolecules](#), it is possible to use the `computeOptimal` to find these parameters. It should be noted that this function requires a ChIP-seq data input. `LocusProfile` (ChIP-seq data) should be a named list with normalised ChIP-seq to a single base pair level. Naming should stay consistent with all other names and should represent the names of the loci of interest. The naming procedure should be similar in `setSequence`. Each range within the [GRanges](#) should be named (not to be confused with seqnames)

Value

`computeOptimal` returns a list respectively described as the optimal set of Parameters (`lambda` or [ScalingFactorPWM](#) and [boundMolecules](#)), the optimal matrix (a matrix containing accuracy estimates dependant on the parameter chosen), and finally the chosen parameter. If the parameter that was chosen was "all", then each element of this list will contain the optimal set of parameters, optimal matrices for "correlation", "Mean Squared Error" and "theta".

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata", package="ChIPAnalyser"), "BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome
```

```

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,BPFrequency=DNASequenceSet)
OPP <- occupancyProfileParameters()
#Computing Optimal set of Parameters
optimalParam <- computeOptimal(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP,
  LocusProfile = eveLocusChip,
  setSequence = eveLocus,
  DNAAccessibility = Access,
  occupancyProfileParameters = OPP,
  parameter = "all",
  peakMethod="moving_kernel")

```

computePWMScore

Compute PWM Scores of sites above threshold.

Description

computePWMScore will compute and extract all sites that exhibit a PWM Score higher than a threshold. This threshold (see [PWMThreshold](#)) will determine the percentage of total sites that should NOT be considered.

Usage

```

computePWMScore(DNASequenceSet, genomicProfileParameters,
  setSequence = NULL, DNAAccessibility = NULL, verbose = TRUE)

```

Arguments

DNASequenceSet DNASequenceSet is a [DNAStringSet](#) or a BSgenome containing the full sequence of the organism of interest.

genomicProfileParameters genomicProfileParameters is a [genomicProfileParameters](#) object resulting from the [computeGenomeWidePWMScore](#) function.

setSequence setSequence is a [GRanges](#) object containing the Loci of interest.

DNAAccessibility DNAAccessibility is a [GRanges](#) object containing Accesible DNA.

verbose verbose is a logical value indicating if progress messages should be printed or not.

Details

After determining genome wide scores, it is possible to only compute and extract high affinity sites (in the sense that they have a high PWM Score). If a `PWMThreshold` is not set by user, the default value is set at 0.7. This means that 70 % of sites will NOT be selected. Only the top 30 % will be computed and extracted. If one is interested in all PWM Scores at a genome wide scale (or accessible DNA), this is possible by setting `PWMThreshold` to zero.

Value

`computePWMScore` will return a `genomicProfileParameters` object. The `AllSitesAboveThreshold` slot will have been updated. This slot will now contain a `GRangesList` with each element being a `GRanges`. This `GRanges` will contain position of each sites (start, end and strand) and the `PWMScore` associated to that site.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDS1x.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,BPFrequency=DNASequenceSet)

# Computing Genome Wide
GenomeWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP)

#Compute PWM Scores
PWMScores <- computePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GenomeWide,
  setSequence = evelocus, DNAAccessibility = Access)
PWMScores
```

DNASequenceLength *Accessor method for DNASequenceLength slot in a [genomicProfileParameters](#)*

Description

Accessor method for DNASequenceLength slot in a [genomicProfileParameters](#)

Usage

```
DNASequenceLength(object)
```

Arguments

object object is a [genomicProfileParameters](#)

Details

The model on which is based ChIPAnalyser requires the length of the DNA sequence used to compute scores. In this circumstance, this DNA Length is the total length of the DNA of the organism of interest or the the Accessible DNA at a genome wide scale.

Value

Returns DNASequenceLength slot in a [genomicProfileParameters](#) object.

Author(s)

Patrick C. N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata", package="ChIPAnalyser"), "BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

#Building genomicProfileParameters object
GPP <- genomicProfileParameters(PFM=PFM,BPFrequency=DNASequenceSet)
```

```
# Computing Genome Wide
GenomceWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP)

DNASequenceLength(GenomceWide)
```

DNASequenceLength<- *Setter Method for* DNASequenceLength *slot in a*
[genomicProfileParameters](#) *object.*

Description

Setter Method for DNASequenceLength slot in a [genomicProfileParameters](#) object.

Usage

```
DNASequenceLength(object)<- value
```

Arguments

object	object is a genomicProfileParameters object.
value	value is a numerical value indicating the length of the DNA used and that will be assigned to DNASequenceLength slot in a genomicProfileParameters .

Details

Although it is possible to manually assign the value of the DNA Length, it is not necessary as this value is computed internally in [computeGenomeWidePWMScore](#).

Value

Returns a [genomicProfileParameters](#) object with an updated value for the DNASequenceLength slot.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

# Building genomicProfileParameters object
GPP <- genomicProfileParameters(PFM=PFM,BPFrequency=DNASequenceSet)
# Setting new value for slot
DNASequenceLength(GPP)<- 1500000
# !! This value is computed internally !!
```

genomicProfileParameters

Genomic Profile Parameter Object

Description

genomicProfileParameter is a data/parameter storing object. The main purpose of this object is to store parameters necessary for further computation but also store results. This makes parsing data and parameters easier between the different of functions available in ChIPAnalyser.

Usage

```
genomicProfileParameters(PWM = NULL, PFM = NULL,PFMFormat="raw",
  ScalingFactorPWM = 1,
  PWMpseudocount = 1, noOfSites = 0, BPFrequency = rep(0.25, 4),
  naturalLog = FALSE, PWMThreshold = 0.7, strandRule = "max",
  whichstrand = "+-")
```

Arguments

PFM PFM is a position Frequency matrix. This matrix can either be supplied directly as a matrix (in the following form: 1 row for each base pair - ACTG in order - and a number of rows equal to the number of base pairs in the binding site) or can be supplied as path to a file containing the Position Frequency Matrix. At least ONE of either the PFM or PWM must be supplied. This is the minimum to build a genomicProfileParameters object. NOTE: Although the object will build, NO computation can be carried out without at least a PFM or a PWM.

PWM	PWM is a position weight matrix. This can be provided in matrix form (1 row for each base pair - ACTG in order - and a number of rows equal to the number of base pairs in the binding site). If a PFM is provided the PWM will be computed internally. NOTE: Although the object will build, NO computation can be carried out without at least a PFM or a PWM.
PFMFormat	PFMFormat is a character string describing the format of the file used (if any) in PositionFrequencyMatrix. ChIPanalyser handles most PFM formats thus this argument should take one of the following: raw, transfac, JASPAR or sequences.
ScalingFactorPWM	ScalingFactorPWM: A vector (or single value) containing values for the ScalingFactorPWM (Also known as lambda).Default:1
PWMPseudocount	PWMPseudocount: A numeric value describing a PWMPseudocount for PWM computation. Default:1
BPFrequency	BPFrequency: Base Pair Frequency in the genome (if a DNA sequence is provided (as a DNASTringSet or BSgenome) , will be automatically computed internally). Default:c(0.25,0.25,0.25,0.25)
naturalLog	naturalLog: A logical value describing if natural Log will be used to compute the PWM (if FALSE then log2 will be used). Default: TRUE
noOfSites	noOfSites: A Positive integer describing number of sites (in base pair) should be used from the PFM to compute PWM. Default =0 (Full width of binding site will be used when set to 0)
PWMThreshold	PWMThreshold: Threshold at which PWM Score should be selected (only sites above threshold will be selected - between 0 and 1)
strandRule	strandRule: 'mean', 'max' or 'sum' will determine how strand should be handle for computing PWM Scores. Default : 'max'
whichstrand	whichstrand: '+', '-' or '+-' on which strand should PWM Score be computed. Default: '+-'

Details

Most functions in ChIPanalyser require some form of a genomicProfileParameters object. It is required to build a genomicProfileParameters object before starting any computation. Most functions will return a genomicProfileParameters and thus this object should be parsed to the next function and so on. Most parameters are customisable and we strongly advise to do so. This can either be done directly (advised method) when building the genomicProfileParameters or by using setter methods. The former is advised as certain value are computed internally and require that parameters are set before hand (e.g BPFrequency). It is important to note that not all slots/parameters can be set by user. The value assigned to these slots are computed internally. For more information on this slots, see [genomicProfileParameters-class](#). The following describes the default values and which slot are computed internally:

```

PWM = To be supplied by user
PFM = To be supplied by user
PFMFormat = "raw"
ScalingFactorPWM = 1
PWMPseudocount = 1
BPFrequency = c(0.25,0.25,0.25,0.25)
naturalLog = TRUE
noOfSites = 0

```

minPWMScore = Internally Computed
 maxPWMScore = Internally Computed
 PWMThreshold = 0.7
 AllSitesAboveThreshold = Internally Computed
 DNASequenceLength = Internally Computed (may still be provided by user see [DNASequenceLength<-](#))
 averageExpPWMScore = Internally Computed
 strandRule = 'max'
 whichstrand = '+-'
 NoAccess = Internally Computed
 ZeroBackground = Internally Computed

Value

Returns a genomicProfileParameters object

Author(s)

Patrick C. N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

See Also

[genomicProfileParameters-class](#) [occupancyProfileParameters](#) [occupancyProfileParameters-class](#)

Examples

```

#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome
if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

# Building genomicProfileParameters object
GPP <- genomicProfileParameters(PFM=PFM, ScalingFactorPWM=c(1,1.5,2),
  BPFrequency=DNASequenceSet)

```

```
genomicProfileParameters-class
      Class "genomicProfileParameters"
```

Description

`genomicProfileParameters` is a data/parameter storing object. The main purpose of this object is to store parameters necessary for further computation but also store results. This makes parsing data and parameters easier between the different of functions available in `ChIPAnalyser`.

Objects from the Class

Objects can be created by calls of the form `genomicProfileParameters(PWM, PFM, PFMFormat, ScalingFactorPWM, PWM` and `PFM` slots are the only slots that are absolutely required (At least one of the two). All other slots have a provided default value. We strongly advise you to tailor these parameters to increase goodness of fit between the predicted model and real ChIP-seq data.

Slots

- PWM:** Object of class "matrix": A Position Weight Matrix (either supplied or internally computed if PFM is provided)
- PFM:** Object of class "matrix": A Position Frequency Matrix (may also be a path to file containing PFM)
- PFMFormat:** Object of class "character": A character string of one of the following: raw, trans-fac, JASPAR or sequences
- ScalingFactorPWM:** Object of class "vector": A vector (or single value) containing values for the ScalingFactorPWM (Also known as lambda). Default: 1
- PWMpseudocount:** Object of class "numeric": A numeric value describing a PWMpseudocount for PWM computation. Default: 1
- BPFrequency:** Object of class "vector": Base Pair Frequency in the genome (if a DNA sequence is provided (as a [DNAStringSet](#) or `BSgenome`), will be automatically computed internally). Default: `c(0.25,0.25,0.25,0.25)`
- naturalLog:** Object of class "logical": A logical value describing if natural Log will be used to compute the PWM (if FALSE then log2 will be used). Default: TRUE
- noOfSites:** Object of class "vector": A Positive integer describing number of sites (in base pair) should be used from the PFM to compute PWM. Default = 0 (Full width of binding site will be used when set to 0)
- minPWMScore:** Object of class "vector": Lowest PWM score across the genome (computed and updated internally)
- maxPWMScore:** Object of class "vector": Highest PWM score across the genome (computed and updated internally)
- PWMThreshold:** Object of class "numeric": Threshold at which PWM Score should be selected (only sites above threshold will be selected - between 0 and 1)
- AllSitesAboveThreshold:** Object of class "GRList": Contains GRanges with sites above threshold and associated metrics (PWMScore and Occupancy) - Computed Internally
- DNASequencesLength:** Object of class "vector": Length of the Genome (or accessible genome) - computed internally

averageExpPWMScore: Object of class "vector": Average exponential PWM score across the genome (or accesible genome) - computed internally

strandRule: Object of class "character": "mean", "max" or "sum" will determine how strand should be handle for computing PWM Scores. Default : "max"

whichstrand: Object of class "character": "+", "-" or "+-" on which strand should PWM Score be computed. Default: "+-"

NoAccess: Object of class "vector": Stores Loci that do contain accesible DNA if it were to be the case (computed and updated internally)

ZeroBackground: Object of class "vector": Internal background value (computed internally)

Extends

Class [occupancyProfileParameters](#), directly.

Methods

AllSitesAboveThreshold signature(object = "genomicProfileParameters"): Slot Accessor Method

averageExpPWMScore signature(object = "genomicProfileParameters"): Slot Accessor Method

BPFrequency<- signature(object = "genomicProfileParameters", value = "vector"):Slot Setter Method

BPFrequency<- signature(object = "genomicProfileParameters", value = "DNAStrngSet"):Slot Setter Method

BPFrequency signature(object = "genomicProfileParameters"): Slot Accessor Method

DNASequenceLength<- signature(object = "genomicProfileParameters", value = "vector"): Slot Setter Method

DNASequenceLength signature(object = "genomicProfileParameters"): Slot Accessor Method

maxPWMScore signature(object = "genomicProfileParameters"): Slot Accessor Method

minPWMScore signature(object = "genomicProfileParameters"): Slot Accessor Method

naturalLog<- signature(object = "genomicProfileParameters", value = "logical"): Slot Setter Method

naturalLog signature(object = "genomicProfileParameters"): Slot Accessor Method

NoAccess signature(object = "genomicProfileParameters"): Slot Accessor Method

noOfSites<- signature(object = "genomicProfileParameters", value = "vector"): Slot Setter Method

noOfSites signature(object = "genomicProfileParameters"): Slot Accessor Method

PositionFrequencyMatrix<- signature(object = "genomicProfileParameters", value = "matrix"): Slot Setter Method

PositionFrequencyMatrix<- signature(object = "genomicProfileParameters", value = "character"): Slot Setter Method

PositionFrequencyMatrix signature(object = "genomicProfileParam"): Slot Accessor Method

```

PFMFormat<- signature(object = "genomicProfileParameters", value = "character"):
  Slot Setter Method
PFMFormat signature(object = "genomicProfileParam": Slot Accessor Method
PositionWeightMatrix<- signature(object = "genomicProfileParameters", value = "matrix"):Slot
  Setter Method
PositionWeightMatrix signature(object = "genomicProfileParameters"): Slot Ac-
  cessor Method
PWMpseudocount<- signature(object = "genomicProfileParameters", value = "numeric"):
  Slot Setter Method
PWMpseudocount signature(object = "genomicProfileParameters"):Slot Accessor
  Method
PWMThreshold<- signature(object = "genomicProfileParameters", value = "numeric"):
  Slot Setter Method
PWMThreshold signature(object = "genomicProfileParameters"): Slot Accessor
  Method
ScalingFactorPWM<- signature(object = "genomicProfileParameters", value = "vector"):
  Slot Setter Method
ScalingFactorPWM signature(object = "genomicProfileParameters"): Slot Accessor
  Method
strandRule<- signature(object = "genomicProfileParameters", value = "character"):
  Slot Setter Method
strandRule signature(object = "genomicProfileParameters"): Slot Accessor Method
whichstrand<- signature(object = "genomicProfileParameters", value = "character"):
  Slot Setter Method
whichstrand signature(object = "genomicProfileParameters"): Slot Accessor
  Method

```

Author(s)

Partick C. N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

See Also

[genomicProfileParameters](#) [occupancyProfileParameters-class](#) [occupancyProfileParameters](#)

Examples

```

#Data extraction
data(ChIPanalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){

```

```

source("https://bioconductor.org/biocLite.R")
biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

# Building genomicProfileParameters object
GPP <- genomicProfileParameters(PFM=PFM, ScalingFactorPWM=c(1,1.5,2),
  BPFrequency=DNASequenceSet)

```

GRList-class

Class "GRList"

Description

Virtual Class to handle multiple data types for one slot ([AllSitesAboveThreshold](#))

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

[GRList-class](#) The purpose of this virtual class is to store data of two different formats in one slot: GRangesList and Lists

Author(s)

Patrick C. N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
showClass("GRList")
```

maxPWMScore

Accessor function for maxPWMScore slot in a [genomicProfileParameters](#) object.

Description

Accessor function for maxPWMScore slot in a [genomicProfileParameters](#) object.

Usage

```
maxPWMScore(object)
```

Arguments

object object is a [genomicProfileParameters](#) object.

Details

maxPWMScore is a numerical value that can be described as the highest PWM score computed at a genome wide scale. This value is computed and updated in the [genomicProfileParameters](#) object after using the [computeGenomeWidePWMScore](#).

Value

Returns the value of assigned to the maxPWMScore slot in a [genomicProfileParameters](#) object.

Author(s)

Patrick C. N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata", package="ChIPAnalyser"), "BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM)

# Computing Genome Wide
GenomeWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP)
maxPWMScore(GenomeWide)
## If used before computeGenomeWidePWMScore, will return NULL
```

maxSignal	<i>Accessor method for the maxSignal slot in a occupancyProfileParameters object.</i>
-----------	---

Description

Accessor method for the maxSignal slot in a [occupancyProfileParameters](#) object.

Usage

```
maxSignal(object)
```

Arguments

object object is a [occupancyProfileParameters](#) object.

Details

In the context of ChIPanalyser, maxSignal represents the maximum normalised ChIP-Seq signal of a given Transcription factor (or DNA binding protein). Although, A default value of 1 has been assigned to this slot, we strongly recommend to tailor this value accordingly. We strongly encourage to set values when building a [occupancyProfileParameters](#) object.

Value

Returns the value assigned to the maxSignal slot in a [occupancyProfileParameters](#) object.

Author(s)

Patrick C.N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
#Setting new Value for maxSignal
maxSignal(OPP)
```

minPWMScore	<i>Accessor method the minPWMScore slot in a genomicProfileParameters object</i>
-------------	--

Description

Accessor method the minPWMScore slot in a [genomicProfileParameters](#) object

Usage

```
minPWMScore(object)
```

Arguments

object object is a [genomicProfileParameters](#) object.

Details

minPWMScore can be described as the lowest PWM score computed at a genome wide scale. Although it is possible to assign a value to minPWMScore, we strongly advise to use the value computed and assigned internally. This value is computed in the [computeGenomeWidePWMScore](#) function.

Value

Returns the value assigned to the minPWMScore slot in a [genomicProfileParameters](#) object.

Author(s)

Patrick C. N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM)
```

```
# Computing Genome Wide
GenomceWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP)
minPWMScore(GenomceWide)

## If used before computeGenomeWidePWMScore, will return NULL
```

naturalLog *Accessor method the naturalLog slot in a genomicProfileParameters object.*

Description

Accessor method the naturalLog slot in a [genomicProfileParameters](#) object.

Usage

```
naturalLog(object)
```

Arguments

object object is [genomicProfileParameters](#) object.

Details

During the computation of a Position Weight Matrix, the Position Probability Matrix (derived from a Position Frequency Matrix) is log transformed. This parameter provides which "log transform" will be used. If TRUE, the Natural Log will be used (ln). If FALSE, log2 will be used. We strongly encourage to set values when building a [genomicProfileParameters](#) object.

Value

Returns the value assigned to the naturalLog slot in a [genomicProfileParameters](#) object.

Author(s)

Patrick C.N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,naturalLog=TRUE)
#Setting new Value for naturalLog
naturalLog(GPP)
```

naturalLog<- *Setter method for the naturalLog slot in a genomicProfileParameters object.*

Description

Setter method for the naturalLog slot in a [genomicProfileParameters](#) object.

Usage

```
naturalLog(object)<- value
```

Arguments

object	object is a genomicProfileParameters object.
value	value is a logical value that will determine if the natural log or log2 should be used for the computation of the Position Weight Matrix.

Details

During the computation of a Position Weight Matrix, the Position Probability Matrix (derived from a Position Frequency Matrix) is log transformed. This parameter provides which "log transform" will be used. If TRUE, the Natural Log will be used (ln). If FALSE, log2 will be used. We strongly encourage to set values when building a [genomicProfileParameters](#) object.

Value

Returns [genomicProfileParameters](#) object with an updated value for the naturalLog slot.

Author(s)

Patrick C.N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,naturalLog=TRUE)
#Setting new Value for naturalLog
naturalLog(GPP) <- FALSE
```

NoAccess *Accessor Method for the NoAccess slot in a*
genomicProfileParameters object.

Description

Accessor Method for the NoAccess slot in a [genomicProfileParameters](#) object.

Usage

```
NoAccess(object)
```

Arguments

object object is a [genomicProfileParameters](#) object.

Details

During certain computations, it is possible that the Loci of interest do not show any overlap with accessible DNA. If this were to be the case, a warning message will appear in the console but these inaccessible Loci will be stored in this slot. It is also for these reasons that it is imperative for Loci of interest to be named (in this case, a named [GRanges](#)).

Value

Returns a character string with loci containing no accessible DNA.

Author(s)

Patrick C.N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM)

# Loci with no acces - a warning message will be issued
#if loci do no contain accesible DNA
# Otherwise this slot will remain empty

NoAccess(GPP)
```

noOfSites	<i>Accessor Method for the noOfSites slot in a genomicProfileParameters object</i>
-----------	--

Description

Accessor Method for the noOfSites slot in a [genomicProfileParameters](#) object

Usage

```
noOfSites(object)
```

Arguments

object object is [genomicProfileParameters](#) object

Details

While computing Position Weight Matricies (PWM) from Position Frequency Matricies (PFM), it is possible to restrict the number of sites that will be used to compute the PWM. The default is set at 0. In this case, all sites will be used to compute the PWM.

Value

Returns the value assigned to the noOfSites slot in a [genomicProfileParameters](#) object.

Author(s)

Patrick C. N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"), "BCDSLx.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,noOfSites=0)
#Setting new Value for naturalLog
noOfSites(GPP)
```

noOfSites<- *Setter Method for the noOfSites slot in a genomicProfileParameters object.*

Description

Setter Method for the noOfSites slot in a [genomicProfileParameters](#) object.

Usage

```
noOfSites(object) <- value
```

Arguments

object object is a [genomicProfileParameters](#) object.
value value is a positive integer that will be assigned to the noOfSites slot.

Details

While computing Position Weight Matrices (PWM) from Position Frequency Matrices (PFM), it is possible to restrict the number of sites that will be used to compute the PWM. The default is set at 0. In this case, all sites will be used to compute the PWM.

Value

Returns a [genomicProfileParameters](#) object with an updated value for the noOfSites slot.

Author(s)

Patrick C.N. Martin <p.martin@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,noOfSites=0)
#Setting new Value for naturalLog
noOfSites(GPP) <- 8
```

occupancyProfileParameters

Occupancy Profile Parameters

Description

occupancyProfileParameters is a parameter storage object. Similarly to [genomicProfileParameters](#), this object serves the purpose of handling the set of parameters needed when using certain functions.

Usage

```
occupancyProfileParameters(ploidy = 2 , boundMolecules = 1000 ,
  backgroundSignal = 0 , maxSignal = 1 , chipMean = 150 ,
  chipSd = 150 , chipSmooth = 250 , stepSize = 10 ,
  removeBackground = 0 , thetaThreshold = 0.1 )
```

Arguments

ploidy	ploidy : A numeric Value describing the ploidy of the organism. Default: 2
boundMolecules	boundMolecules : A vector (or single value) containing the number of bound Molecules (bound Transcription Factors): Default: 1000
backgroundSignal	backgroundSignal : A numeric value describing the ChIP-seq background Signal (average signal from real ChIP seq data). Default: 0
maxSignal	maxSignal : A numeric value describing the highest ChIP-seq signal (from real ChIP-seq data). Default: 1
chipMean	chipMean : A numeric value describing the mean width of a ChIP- seq peak: Default:150
chipSd	chipSd : A numeric value describing the standard deviation of ChIP-seq peaks. Default: 150
chipSmooth	chipSmooth : A numeric value describing the width of the window used to smooth Occupancy profiles into ChIP profiles. Default:250
stepSize	stepSize : A numeric value describing the step Size (in base pairs) between each ChIP-seq score. Default:10 (Scored every 10 base pairs)
removeBackground	removeBackground : A numeric value describing the value at which score should be removed. Default:0 (If negative scores then remove)

thetaThreshold [thetaThreshold](#): A numeric value describing a threshold at which low ratios of correlation over Mean Squared Error (for accuracy estimation) should be removed. Default: 0.1

Details

The purpose of this occupancyProfileParameters object is to store parameters used during certain of the functions available in ChIPAnalyser. As opposed to genomicProfileParameters, this object is not required for a function to run as all parameters have default values (the object will be created internally as default values will be used - This also means that no argument is required for this object to be built). However, it is strongly advised to set these parameters to increase the goodness of fit between the predicted profiles and the real ChIP-seq profiles. As an example, maxSignal is the maximum signal of the real ChIP-seq data, this signal will depend on the data being used thus should be tuned to optimise the model.

Default values are set at:

```
ploidy = 2
boundMolecules = 1000
backgroundSignal = 0
maxSignal = 1
chipMean = 150
chipSd = 150
chipSmooth = 250
stepSize = 10
removeBackground = 0
thetaThreshold = 0.1
```

Value

Returns an occupancyProfileParameters object

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. Nucleic Acids Res., 43, 84–94.

See Also

[occupancyProfileParameters-class](#) [genomicProfileParameters](#) [genomicProfileParameters-class](#)

Examples

```
OPP <- occupancyProfileParameters()
# OR
OPP <- occupancyProfileParameters(ploidy=2,
  boundMolecules=5000, backgroundSignal=0)
```

```
occupancyProfileParameters-class
      Class "occupancyProfileParameters"
```

Description

occupancyProfileParameters is a parameter storage object. Similarly to [genomicProfileParameters](#), this object serves the purpose of handling the set of parameters needed when using certain functions.

Objects from the Class

Objects can be created by calls of the form `occupancyProfileParameters(ploidy = 2 , boundMolecules = 1000 ,`

Slots

ploidy: Object of class "numeric": A numeric Value describing the ploidy of the organism. Default: 2

boundMolecules: Object of class "vector": A vector (or single value) containing the number of bound Molecules (bound Transcription Factors): Default: 1000

backgroundSignal: Object of class "numeric": A numeric value describing the ChIP-seq background Signal (average signal from real ChIP seq data). Default: 0

maxSignal: Object of class "numeric": A numeric value describing the highest ChIP-seq signal (from real ChIP-seq data). Default: 1

chipMean: Object of class "numeric": A numeric value describing the mean width of a ChIP- seq peak. Default:150

chipSd: Object of class "numeric": A numeric value describing the standard deviation of ChIP-seq peaks. Default: 150

chipSmooth: Object of class "vector": A numeric value describing the width of the window used to smooth Occupancy profiles into ChIP profiles. Default:250

stepSize: Object of class "numeric": A numeric value describing the step Size (in base pairs) between each ChIP-seq score. Default:10 (Scored every 10 base pairs)

removeBackground: Object of class "numeric": A numeric value describing the value at which score should be removed. Default:0 (If negative scores then remove)

thetaThreshold: Object of class "numeric": A numeric value describing a threshold at which low ratios of correlation over Mean Squared Error (for accuracy estimation) should be removed. Default: 0.1

Methods

backgroundSignal<- signature(object = "occupancyProfileParameters", value = "numeric"): Slot Setter Method

backgroundSignal signature(object = "occupancyProfileParameters"): Slot Accessor Method

boundMolecules<- signature(object = "occupancyProfileParameters", value = "vector"): Slot Setter Method

boundMolecules signature(object = "occupancyProfileParameters"): Slot Accessor Method

```

chipMean<- signature(object = "occupancyProfileParameters", value = "numeric"):
  Slot Setter Method
chipMean signature(object = "occupancyProfileParameters"): Slot Accessor Method
chipSd<- signature(object = "occupancyProfileParameters", value = "numeric"):
  Slot Setter Method
chipSd signature(object = "occupancyProfileParameters"): Slot Accessor Method
chipSmooth<- signature(object = "occupancyProfileParameters", value = "vector"):
  Slot Setter Method
chipSmooth signature(object = "occupancyProfileParameters"): Slot Accessor
  Method
maxSignal<- signature(object = "occupancyProfileParameters", value = "numeric"):
  Slot Setter Method.
maxSignal signature(object = "occupancyProfileParameters"): Slot Accessor
  Method
ploidy<- signature(object = "occupancyProfileParameters", value = "numeric"):
  Slot Setter Method
ploidy signature(object = "occupancyProfileParameters"): Slot Accessor Method
removeBackground<- signature(object = "occupancyProfileParameters", value = "vector"): Slot
  Setter Method
removeBackground signature(object = "occupancyProfileParameters"): Slot Setter
  Method
show signature(object = "occupancyProfileParameters"): Slot Accessor Method
stepSize<- signature(object = "occupancyProfileParameters", value = "numeric"):
  Slot Setter Method
stepSize signature(object = "occupancyProfileParameters"): Slot Accessor Method
thetaThreshold<- signature(object = "occupancyProfileParameters", value = "numeric"):
  Slot Setter Method
thetaThreshold signature(object = "occupancyProfileParameters"): Slot Accessor
  Method

```

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

See Also

[occupancyProfileParameters](#) [occupancyProfileParameters-class](#) [genomicProfileParameters](#)

Examples

```

OPP <- occupancyProfileParameters()
# OR
OPP <- occupancyProfileParameters(ploidy=2,
  boundMolecules=5000, backgroundSignal=0)

```

PFMFormat	<i>Accesor method for the</i>	PFMFormat	<i>slot in a</i>
	genomicProfileParameters	<i>object</i>	

Description

Accesor method for the PFMFormat slot in a [genomicProfileParameters](#) object

Usage

```
PFMFormat(object)
```

Arguments

object object is a [genomicProfileParameters](#) object

Details

If loading a [PositionFrequencyMatrix](#) from a file, the format of the file should be specified. Default is raw. Please keep in mind that this argument is used when parsing the [PositionFrequencyMatrix](#) file. IF this argument is changed after building the [genomicProfileParameters](#) with a Position-FrequencyMatrix file, this will not influence the parsing of the file. PFMFormat can be one of the following: "raw", "transfac", "JASPAR" or "sequences"

Value

Returns the value assigned to the PFMFormat slot a [genomicProfileParameters](#)

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDS1x.pfm")
#Building data objects
#### THIS IS THE PREFERRED METHOD FOR SETTING PFMFormat
GPP <- genomicProfileParameters(PFM=PFM,PFMFormat="raw")
#Setting New value for PFMFormat
PFMFormat(GPP)
```

ploidy	<i>Accessor method for the ploidy slot in a occupancyProfileParameters object</i>
--------	---

Description

Accessor method for the ploidy slot in a [occupancyProfileParameters](#) object

Usage

```
ploidy(object)
```

Arguments

object object is a [occupancyProfileParameters](#) object

Details

Default value for ploidy is set a 2. It should be mentioned that ChIPanalyser is based on a model that also considers the ploidy of the organism of interest however this only considers simple polyploidy (or haploidy). The model does not consider hybrids such as wheat.

Value

Returns the value assigned to the ploidy slot in a [occupancyProfileParameters](#) object

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
#Setting new Value for maxSignal
ploidy(OPP)
```

plotOccupancyProfile *Plot Occupancy Profiles*

Description

plotOccupancyProfile plots the predicted profiles. If provided, this functions will also plot ChIP-seq profiles, PWM Scores, DNA Accessibility, accuracy estimates and gene information.

Usage

```
plotOccupancyProfile(predictedProfile, setSequence,
  profileAccuracy = NULL, chipProfile = NULL, occupancy = NULL, PWM=FALSE,
  DNAAccessibility = NULL, occupancyProfileParameters = NULL, geneRef = NULL)
```

Arguments

predictedProfile
predictedProfile is a [GRanges](#) containing predicted ChIP-seq scores for ONE loci and ONE combination of lambda (See [ScalingFactorPWM](#)) and [boundMolecules](#).

setSequence
setSequence is a [GRanges](#) containing the locus of interest. This locus MUST be named.

profileAccuracy
profileAccuracy is the accuracy estimate resulting from [profileAccuracyEstimate](#) for only one Locus and combination of lambda and boundMolecules

chipProfile
chipProfile is a numeric vector containing real ChIP-seq scores.

occupancy
occupancy is a [GRanges](#) containing PWM Score and predicted Occupancies.

DNAAccessibility
DNAAccessibility is a [GRanges](#) containing accesible DNA sites

PWM
PWM is a logical value that in the case occupancy is provided which of occupancy scores of PWM scores hshould be plotted. Default set at FALSE

occupancyProfileParameters
[occupancyProfileParameters](#) is [occupancyProfileParameters](#) object containing parameters used for previous computations (see vignette for pipeline/function order)

geneRef
geneRef is a list of [GRanges](#) containing gene information on exons,introns, UTR's and enhancers

Details

Once the predicted ChIP-seq like profiles have been computed, it is possible to plot these profiles. The important aspect to keep in mind with plotOccupancyProfile is that in only computes one profile at a time. This means that the output of previous functions should be subsetted carefully. This leaves to the user more flexibility in what and how profiles should be plotted. Keep in mind that the output of different function are more or less nested.

Value

Returns a profile plot with "Occupancy" on the y axis and DNA position on the the X- axis. If the orange line is the predicted profile, the grey shaded area represents real ChIP-seq data. The yellow boxes represent regions on NON-accessible DNA. On the lower part of the plot, gene information is plotted with respect to the strand they are localised on. Finally, the blue vertical lines represent sites of either high Occupancy or high PWM score depending on which had been selected. The minimal plot will only contain the predicted profile. The more data is provided the more will be plotted.

Author(s)

Patrick C.N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,BPFrequency=DNASequenceSet)
OPP <- occupancyProfileParameters()

# Computing Genome Wide
GenomeWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP)

#Compute PWM Scores
PWMScores <- computePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GenomeWide,
  setSequence = eveLocus, DNAAccessibility = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(AllSitesPWMScore = PWMScores,
  occupancyProfileParameters = OPP)

#Compute ChIP profiles
chipProfile <- computeChipProfile(setSequence = eveLocus,
  occupancy = Occupancy, occupancyProfileParameters = OPP)
#Estimating accuracy estimate
AccuracyEstimate <- profileAccuracyEstimate(LocusProfile = eveLocusChip,
  predictedProfile = chipProfile, occupancyProfileParameters = OPP)
#Plotting Profile
plotOccupancyProfile(predictedProfile=chipProfile[[1]][[1]],
```

```

setSequence=eveLocus,
profileAccuracy = AccuracyEstimate[[1]][[1]],
chipProfile = eveLocusChip[[1]],
occupancy = AllSitesAboveThreshold(Occupancy)[[1]][[1]],
DNAAccessibility = Access,
occupancyProfileParameters = OPP,
geneRef =geneRef)

```

plotOptimalHeatMaps *Heat Map of optimal Parameters*

Description

plotOptimalHeatMaps will plot heat maps of optimal Parameters and highlight the optimal combination of [ScalingFactorPWM](#) and [boundMolecules](#)

Usage

```
plotOptimalHeatMaps(optimalParam, parameter = "all", Contour = TRUE)
```

Arguments

optimalParam	optimalParam is a list resulting of computeOptimal . This list should contain three elements: optimal Parameters, optimal Matrix and parameter.
parameter	parameter is a character string representing which parameter should be plotted. There are four possibilities: "correlation", "MSE", "theta" and "all"
Contour	Contour is a logical value indicating if contour lines should be plotted.

Details

Once the optimal set of Parameters ([ScalingFactorPWM](#) and [boundMolecules](#)), it is possible to plot the results in the form of a heat map. There are four possible heat maps: a "correlation" heat map that will only show the correlation between a predicted Profile and true ChIP-seq profiles for each combination of Parameters ; "MSE" will show a similar heat map only with the Mean Squared Error associated to each predicted profile, true profile and parameter combination; "theta" is a in house metric describing the best fit between high correlation and low MSE (see [thetaThreshold](#)); finally "all" will plot all of the above.

Value

Returns a heat map of optimal combinations of [ScalingFactorPWM](#) and [boundMolecules](#). The x axis represents the different value assigned to lambda ([ScalingFactorPWM](#)) and the y axis represents the different values to boundMolecules ([boundMolecules](#)). The hilighted box describes the optimal combination of parameters to either maximise "correlation" and "theta" or to minimise "MSE".

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,BPFrequency=DNASequenceSet)
OPP <- occupancyProfileParameters()
#Computing Optimal set of Parameters
optimalParam <- computeOptimal(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP,
  LocusProfile = eveLocusChip,
  setSequence = eveLocus,
  DNAAccessibility = Access,
  occupancyProfileParameters = OPP,
  parameter = "all")
plotOptimalHeatMaps(optimalParam, parameter="all")
```

PositionFrequencyMatrix

Accessor method for the PFM slot in a genomicProfileParameters object

Description

Accessor method for the PFM slot in a genomicProfileParameters object

Usage

```
PositionFrequencyMatrix(object)
```

Arguments

object object is a [genomicProfileParameters](#) object

Details

After creating a [genomicProfileParameters](#) object, it is possible to access the Position Frequency Matrix slot. However this slot will be empty if the [genomicProfileParameters](#) object was built using directly a Position Weight Matrix. See [genomicProfileParameters](#)

Value

Returns the Position Frequency Matrix (PFM slot) used to compute the [PositionWeightMatrix](#) in a [genomicProfileParameters](#) object

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"), "BCDSLx.pfm")
#Building genomicProfileParameters object
GPP<-genomicProfileParameters(PFM=PFM)
# Accessing Slot
PositionFrequencyMatrix(GPP)
```

PositionFrequencyMatrix<-

Setter method for the PFM slot in a [genomicProfileParameters](#) object

Description

Setter method for the PFM slot in a [genomicProfileParameters](#) object

Usage

```
PositionFrequencyMatrix(object)<- value
```

Arguments

object	object is a genomicProfileParameters object
value	value can be of two forms. Either a matrix in the form of a Position Frequency Matrix or a path/to/file character string.

Details

The Position Frequency Matrix is one of the fundamental object that needs to be supplied to a [genomicProfileParameters](#). If after building a [genomicProfileParameters](#), only the Position Frequency Matrix needs to be modified then it is possible to manually update the value of this matrix using the function above. There are two options for the type of data that may be supplied to the PFM slot: a matrix in the form of a Position Frequency Matrix (matrix with four rows - one for each base pair (ACTG) and a number of columns equal to the number of sites in the binding site), or it is possible (also recommended) to provide a path to the file containing the Position Frequency Matrix. This Position Frequency Matrix file may come in multiple form such as RAW, Transfac or JASPAR. WARNING: if a [genomicProfileParameters](#) object has already been created and only the PFM is supplied/updated, then the Position Weight Matrix will automatically updated as well.

Value

Returns a [genomicProfileParameters](#) with an updated PFM slot (as described above this will lead to an updated PositionWeightMatrix).

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#Building genomicProfileParameters object
GPP<-genomicProfileParameters()
#Setting PFM
PositionFrequencyMatrix(GPP) <- PFM
```

PositionWeightMatrix *Accessor Method for the PWM slot in a [genomicProfileParameters](#) object*

Description

Accessor Method for the PWM slot in a [genomicProfileParameters](#) object

Usage

```
PositionWeightMatrix(object)
```

Arguments

object object is a [genomicProfileParameters](#)

Details

After creating a [genomicProfileParameters](#) object, it is possible to access the Position Weight Matrix stored in this slot. This slot should always contain something. This slot is either supplied by user or directly computed from a Position Frequency Matrix when supplied.

Value

Returns a matrix in the form of a Position Weight Matrix

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata", package="ChIPAnalyser"), "BCDS1x.pfm")
#Building genomicProfileParameters object
GPP<-genomicProfileParameters(PFM=PFM)
# Accessing Slot
PositionWeightMatrix(GPP)
```

PositionWeightMatrix<-

Setter Method for the PositionWeightMatrix slot in a [genomicProfileParameters](#) object

Description

Setter Method for the PositionWeightMatrix slot in a [genomicProfileParameters](#) object

Usage

```
PositionWeightMatrix(object) <- value
```

Arguments

object	object is a genomicProfileParameters object
value	value is a matrix in the form of a Position Weight Matrix.

Details

If a Position Weight Matrix is readily available, it is possible to directly assign this matrix to the PWM slot. However, this is only possible if a `genomicProfileParameters` object has already been created. In that case, we advise to first create a `genomicProfileParameters` object. It should be noted that this Position Weight Matrix will be automatically computed from a Position Frequency Matrix. If no Position Frequency Matrix are available, then a Position Weight Matrix can be directly assigned to this slot.

Value

Returns a `genomicProfileParameters` object with an updated value for the PWM slot

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Building genomicProfileParameters object
GPP <- genomicProfileParameters()
#Setting PWM to PositionWeightMatrix slot
PWM <- matrix(runif(32,-10,20), ncol=8)
rownames(PWM) <- c("A","C","T","G")
PositionWeightMatrix(GPP) <- PWM
```

profileAccuracyEstimate

Estimating Accuracy of predicted Profiles

Description

`profileAccuracyEstimate` will compare the predicted ChIP-seq-like profile to real ChIP-seq data and return a set of metrics describing how accurate the predicted model is compared to real data.

Usage

```
profileAccuracyEstimate(LocusProfile, predictedProfile,
  occupancyProfileParameters = NULL)
```

Arguments

LocusProfile LocusProfile is a list containing normalised ChIP-seq profiles of the Loci of interest (real data). Each profile is a numeric vector of length equals to the length of the locus in base pair.

predictedProfile

predictedProfile is the result of the `computeChipProfile` function. Generally, the output of this function comes in the form of a list of `GRangesList`. Each `GRangesList` contains a `GRanges` with the predicted ChIP-seq-like profiles for each Loci of interest.

occupancyProfileParameters

occupancyProfileParameters is a `occupancyProfileParameters` object

Details

The accuracy of the predicted profile may be estimated by measuring correlation, Mean Squared Error and theta (in house metric based on a modified ratio of correlation over MSE) between predicted Profiles and real ChIP-seq data. Actual ChIP-seq profiles should be normalised to a base pair level (Enrichment divided by the width of the range for that given score - the end result is a numeric vector of length equals to the length of the locus in base pairs). It should be noted that if an `occupancyProfileParameters` object is not supplied, then one will be created internally. However, we strongly advise to use the same `occupancyProfileParameters` object used previously.

Value

Returns a list of lists. Each element in the list represents a combination of lambda (see `ScalingFactorPWM`) and bound molecules (see `boundMolecules`) and the list within each element is the list of Loci of interest. Finally, at the core of these lists is a named vector containing correlation and MSE for the given Loci but also meanCorr, meanMSE and meanTheta for all loci for a given combination of Lambda and bound molecules.

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Data extraction
data(ChIPAnalyserData)
# path to Position Frequency Matrix
PFM <- file.path(system.file("extdata", package="ChIPAnalyser"), "BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

#Building data objects
GPP <- genomicProfileParameters(PFM=PFM,BPFrequency=DNASequenceSet)
OPP <- occupancyProfileParameters()
```

```

# Computing Genome Wide
GenomeWide <- computeGenomeWidePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GPP)

#Compute PWM Scores
PWMScores <- computePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfileParameters = GenomeWide,
  setSequence = eveLocus, DNAAccessibility = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(AllSitesPWMScore = PWMScores,
  occupancyProfileParameters = OPP)

#Compute ChIP profiles
chipProfile <- computeChipProfile(setSequence = eveLocus,
  occupancy = Occupancy,
  occupancyProfileParameters = OPP)
#Estimating accuracy estimate
AccuracyEstimate <- profileAccuracyEstimate(LocusProfile = eveLocusChip,
  predictedProfile = chipProfile,
  occupancyProfileParameters = OPP)

```

PWMpseudocount *Accessor Method for a PWMpseudocount slot in a genomicProfileParameters*

Description

Accessor Method for a PWMpseudocount slot in a [genomicProfileParameters](#)

Usage

```
PWMpseudocount(object)
```

Arguments

object object is a [genomicProfileParameters](#) object.

Details

In the context of Position Weight Matrices, the pseudocount is used to avoid 0 probabilities during the transformation of Position Frequency Matrix to a Position Probability Matrix and finally to a Position Weight Matrix. It is essentially a sample correction that is added in the case of small sample size. The effect of the base pair to which a pseudocount was assigned will not influence the model nor will create mathematical issues such as infinities or zero division. Default is set at 1.

Value

Returns the value assigned to a PWMpseudocount slot in a [genomicProfileParameters](#) object

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPanalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPanalyser"),"BCDSLx.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM, PWMpseudocount=0)
#Accessing slot value
PWMpseudocount(GPP)
```

PWMpseudocount<- *Setter Method for the pseudocount slot in a genomicProfileParameters object*

Description

Setter Method for the pseudocount slot in a [genomicProfileParameters](#) object

Usage

```
PWMpseudocount(object) <- value
```

Arguments

object	object is a genomicProfileParameters object
value	value is a numeric value that will be assigned to the pseudocount slot. Default is set at 1

Details

In the context of Position Weight Matrices, the pseudocount is used to avoid 0 probabilities during the transformation of Position Frequency Matrix to a Position Probability Matrix and finally to a Position Weight Matrix. It is essentially a sample correction that is added in the case of small sample size. The effect of the base pair to which a pseudocount was assigned will not influence the model nor will create mathematical issues such as infinities or zero division.

Value

Returns a [genomicProfileParameters](#) object with an updated value for the pseudocount slot.

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDS1x.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM, PWMpseudocount=0)
#Setting Value for new PWMpseudocount
PWMpseudocount(GPP) <- 1
```

PWMThreshold	<i>Accessor method for the PWMThreshold slot in a genomicProfileParameters object</i>
--------------	---

Description

Accessor method for the PWMThreshold slot in a [genomicProfileParameters](#) object

Usage

```
PWMThreshold(object)
```

Arguments

object object is a [genomicProfileParameters](#) object

Details

The computePWMScore function requires a so-called PWM Threshold. This threshold represents the Threshold at which PWM Score should be selected. The PWMThreshold is a positive numeric value (between 0 and 1. If set at 0, all sites will be selected. If set at 0.7 (Default value), then 70 % of PWM Score (and by extension binding sites) will be IGNORED. The top 30 % will be selected.

Value

Returns the value assigned to the PWMThreshold slot in a [genomicProfileParameters](#) object

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM, PWMThreshold=0.7)
#Accessing Value for PWMThreshold
PWMThreshold(GPP)
```

PWMThreshold<- *Setter Method for the PWMThreshold slot in a genomicProfileParameters object*

Description

Setter Method for the PWMThreshold slot in a [genomicProfileParameters](#) object

Usage

```
PWMThreshold(object) <- value
```

Arguments

object	object is a genomicProfileParameters object
value	value is a numeric value (between 0 and 1) to be assigned to the PWMThreshold slot in genomicProfileParameters object. Default is set at 0.7

Details

The computePWMScore function requires a so-called PWM Threshold. This threshold represents the Threshold at which PWM Score should be selected. The PWMThreshold is a positive numeric value (between 0 and 1. If set at 0, all sites will be selected. If set at 0.7 (Default value), then 70 % of PWM Score (and by extension binding sites) will be IGNORED. The top 30 % will be selected.

Value

Returns [genomicProfileParameters](#) object with an updated value for the PWMThreshold slot

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM, PWMThreshold=0.7)
#Setting Value for new PWMThreshold
PWMThreshold(GPP) <- 0.8
```

removeBackground	<i>Accessor Method for the removeBackground slot in a occupancyProfileParameters object</i>
------------------	---

Description

Accessor Method for the removeBackground slot in a [occupancyProfileParameters](#) object

Usage

```
removeBackground(object)
```

Arguments

object object is a [occupancyProfileParameters](#) object

Details

A numeric value describing a threshold at which Occupancy signals must be removed (Default is set at 0). The removal of Occupancy signals will occur when computing [computeOccupancy](#) (see [computeOccupancy](#) function)

Value

Returns the value assigned to the removeBackground slot in a [occupancyProfileParameters](#) object

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
#Accessing Value for removeBackground
removeBackground(OPP)
```

removeBackground<- *Setter Method for the removeBackground slot in a
occupancyProfileParameters object*

Description

Setter Method for the removeBackground slot in a occupancyProfileParameters object

Usage

```
removeBackground(object) <-value
```

Arguments

object	object is an occupancyProfileParameters object
value	value is positive numerical value to be assigned to the removeBackground slot in a occupancyProfileParameters object. Default is set a 0.

Details

A numeric value describing a threshold at which Occupancy signals must be removed (Default is set at 0). The removal of Occupancy signals will occur when computing [computeOccupancy](#) (see [computeOccupancy](#) function)

Value

Returns an [occupancyProfileParameters](#) object with an updated value for the removeBackground slot

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
#Building occupancyProfileParameters object  
OPP <- occupancyProfileParameters()  
#Setting new Value for removeBackground  
removeBackground(OPP) <- 0.1
```

ScalingFactorPWM *Accessor method for the lambda slot in a genomicProfileParameters object*

Description

Accessor method for the lambda slot in a [genomicProfileParameters](#) object

Usage

```
ScalingFactorPWM(object)
```

Arguments

object object is a [genomicProfileParameters](#) object

Details

The model used in ChIPanalyser relies on a scaling factor (scaling Transcription Factor binding affinity) for the PWM Score (or binding energy). It is also referred to as lambda. In the case a vector of values is provided, each value will be either computed individually (as it will be the case in [computeGenomeWidePWMScore](#)) or in combination with [boundMolecules](#) (as it will be the case in [computeOccupancy](#)).

Value

Returns the value or vector assigned to the lambda slot in a [genomicProfileParameters](#) object

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPanalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata", package="ChIPanalyser"), "BCDSLx.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM, ScalingFactorPWM=c(0.5,1,1.5))
#Accessing value for ScalingFactorPWM
ScalingFactorPWM(GPP)
```

ScalingFactorPWM<- *Setter method for the lambda slot in a [genomicProfileParameters](#) object*

Description

Setter method for the lambda slot in a [genomicProfileParameters](#) object

Usage

```
ScalingFactorPWM(object) <- value
```

Arguments

object	object is a genomicProfileParameters object
value	value is a positive numeric value or a vector of positive numeric values that will be assigned to the lambda slot in a genomicProfileParameters object. Default is set at 1

Details

The model used in ChIPanalyser relies on a scaling factor (scaling Transcription Factor binding affinity) for the PWM Score (or binding energy). It is also referred to as lambda. In the case a vector of values is provided, each value will be either computed individually (as it will be the case in [computeGenomeWidePWMScore](#)) or in combination with [boundMolecules](#) (as it will be the case in [computeOccupancy](#)).

Value

Returns a [genomicProfileParameters](#) object with an updated value for the lambda slot.

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPanalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPanalyser"),"BCDS1x.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM, ScalingFactorPWM=c(0.5,1,1.5))
#Setting New value for ScalingFactorPWM
ScalingFactorPWM(GPP) <- 2
```

```
#OR
ScalingFactorPWM(GPP) <- c(2,3,4)
```

searchSites	<i>Searching function for Sites above threshold and predicted ChIP-seq Profiles</i>
-------------	---

Description

searchSites is function enabling quick extraction and search for the [genomicProfileParameters](#)'s [AllSitesAboveThreshold](#) slot or a list resulting from [computeChipProfile](#).

Usage

```
searchSites(Sites,ScalingFactor="all",BoundMolecules="all", Locus="all")
```

Arguments

Sites	Sites is either a genomicProfileParameters object or a list (result of computeChipProfile)
ScalingFactor	ScalingFactor is a numeric vector describing the ScalingFactors that should be searched within Sites.
BoundMolecules	BoundMolecules is a numeric vector describing the BoundMolecules that should be searched within Sites.
Locus	Locus is a character vector describing the Loci that should be searched within Sites.

Details

After PWM Scores and/or occupancy have been computed (see [computePWMScore](#) and [computeOccupancy](#), it is possible to extract and search the [AllSitesAboveThreshold](#) slot. As this slot may be fairly large it can become difficult to navigate. searchSites will make searching in this slot a lot easier. If all arguments are left at their default value of "all", then all Parameters will be searched thus returning the full list of Sites above threshold. searchSites also works on the result of [computeChipProfile](#).

Value

Returns a list or GRangesList.

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Details

It possible to restrict the size of the ChIP-seq-like profile produced by `computeChipProfile`. Instead of returning ChIP-seq like score for each base pair, it is possible to skip base pairs and only return the predicted enrichment score for every "n" base pair (n is the value assigned to `stepSize`). This will reduce the size of the output data (unless step size is very large, this will not affect the accuracy of the model). Default is set at 10 base pairs.

Value

Returns a `occupancyProfileParameters` object with an updated value for the `stepSize` slot.

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
#Setting new Value for maxSignal
stepSize(OPP) <- 20
```

strandRule	<i>Accessor Method for the strandRule slot in a genomicProfileParameters object</i>
------------	---

Description

Accessor Method for the `strandRule` slot in a `genomicProfileParameters` object

Usage

```
strandRule(object)
```

Arguments

`object` `object` is a `genomicProfileParameters` object

Details

When computing the PWM Scores and if `whichstrand` is set to "+-", `strandRule` will determine how to handle both strands (one of three options : "mean", "max", "sum"). If set to "mean", the average PWM Score of both strand will be computed. If set to "max", the highest PWM score between each strand will be selected and finally "sum" will sum both score together. Default set at "max"

Value

Returns the value assigned to strandRule slot (one of three options : "mean", "max", "sum") in a [genomicProfileParameters](#) object

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDSLx.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM, strandRule="max")
#Accesssing Value for strandRule
strandRule(GPP)
```

strandRule<- *Setter method for the strandRule slot in a [genomicProfileParameters](#) object.*

Description

Setter method for the strandRule slot in a [genomicProfileParameters](#) object.

Usage

```
strandRule(object) <- value
```

Arguments

object	object is a genomicProfileParameters object
value	value is a character string and can be one of the following 'mean', 'max', 'sum'. This will only apply if whichstrand is '+-'. Default set at 'max'

Details

When computing the PWM Scores and if [whichstrand](#) is set to '+-', strandRule will determine how to handle both strands (one of three options : 'mean', 'max', 'sum'). If set to 'mean', the average PWM Score of both strand will be computed. If set to 'max', the highest PWM score between each strand will be selected and finally 'sum' will sum both score together. Default set at 'max'

Value

Returns a [genomicProfileParameters](#) object with an updated value for the strandRule slot

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPanalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata", package="ChIPanalyser"), "BCDS1x.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM, strandRule="max")
#Setting New Value for strandRule
strandRule(GPP) <- "mean"
```

thetaThreshold	<i>Accessor method for the thetaThreshold slot in a occupancyProfileParameters object</i>
----------------	---

Description

Accessor method for the thetaThreshold slot in a [occupancyProfileParameters](#) object

Usage

```
thetaThreshold(object)
```

Arguments

object object is a [occupancyProfileParameters](#) object

Details

In order to measure the accuracy of predicted models against real ChIP-seq data, ChIPanalyser uses three "measures": correlation, Mean Squared Error and Theta. Theta is an in house metric that can be described as a modified ratio of correlation over Mean Squared Error. The higher the value of theta, the better the fit becomes. The theta threshold serves the purpose of avoiding extreme values and thus clarifying optimal parameter heat maps (see [plotOptimalHeatMaps](#)). Default is set at 0.1

Value

Returns the value assigned to the thetaThreshold slot in a [occupancyProfileParameters](#)

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
#Setting new Value for thetaThreshold
thetaThreshold(OPP)
```

`thetaThreshold<-` *Setter method the thetaThreshold slot in a [occupancyProfileParameters](#) object*

Description

Setter method the `thetaThreshold` slot in a [occupancyProfileParameters](#) object

Usage

```
thetaThreshold(object) <- value
```

Arguments

<code>object</code>	object is a occupancyProfileParameters
<code>value</code>	value is a positive numeric value to be assigned to the <code>thetaThreshold</code> slot. Default is set at 0.1

Details

In order to measure the accuracy of predicted models against real ChIP-seq data, ChIPanalyzer uses three "measures": correlation, Mean Squared Error and Theta. Theta is an in house metric that can be described as a modified ratio of correlation over Mean Squared Error. The higher the value of theta, the better the fit becomes. The theta threshold serves the purpose of avoiding extreme values and thus clarifying optimal parameter heat maps (see [plotOptimalHeatMaps](#)). Default is set at 0.1

Value

Returns an [occupancyProfileParameters](#) object with an updated value for `thetaThreshold`

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Building occupancyProfileParameters object
OPP <- occupancyProfileParameters()
#Setting new Value for thetaThreshold
thetaThreshold(OPP) <- 0.1
```

whichstrand	<i>Accessor method for the whichstrand slot in a genomicProfileParameters object</i>
-------------	--

Description

Accessor method for the whichstrand slot in a [genomicProfileParameters](#) object

Usage

```
whichstrand(object)
```

Arguments

object object is a [genomicProfileParameters](#) object

Details

PWM Score may be computed on either the positive strand ("+"), the negative strand ("-") or on both strands ("+-").

Value

Returns on which strand PWM Scores should be computed (whichstrand in a [genomicProfileParameters](#) object)

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata", package="ChIPAnalyser"), "BCDS1x.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM, whichstrand="+-")
#Setting New Value for whichstrand
whichstrand(GPP)
```

whichstrand<- *Setter method for the whichstrand slot in a genomicProfileParameters object*

Description

Setter method for the whichstrand slot in a [genomicProfileParameters](#) object

Usage

```
whichstrand(object) <- value
```

Arguments

object	object is a genomicProfileParameters object
value	value is a character string specifying which strand should be used to compute PWM Scores. The three available options are the following: "+", "-" or "+-". Default is "+-"

Details

PWM Score may be computed on either the positive strand ("+"), the negative strand ("-") or on both strands ("+-").

Value

Returns a [genomicProfileParameters](#) object with an updated value for the whichstrand slot

Author(s)

Patrick C. N. Martin <pm16057@essex.ac.uk>

References

Zabet NR, Adryan B (2015) Estimating binding properties of transcription factors from genome-wide binding profiles. *Nucleic Acids Res.*, 43, 84–94.

Examples

```
# Loading data
data(ChIPAnalyserData)
#Loading PFM files
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDS1x.pfm")
#Building data objects
GPP <- genomicProfileParameters(PFM=PFM, whichstrand="+-")
#Setting New Value for whichstrand
whichstrand(GPP) <- "+"
```

Index

*Topic **classes**

genomicProfileParameters-class, [36](#)
 GRList-class, [39](#)
 occupancyProfileParameters-class,
[51](#)

*Topic **package**

ChIPAnalyser-package, [3](#)
 .ZeroBackground,genomicProfileParameters-method
 (genomicProfileParameters-class),
[36](#)
 .ZeroBackground<- ,genomicProfileParameters,vector-method
 (genomicProfileParameters-class),
[36](#)

Access (ChIPAnalyserData), [15](#)
 AllSitesAboveThreshold, [7](#), [22](#), [26](#), [30](#), [39](#),
[74](#)

AllSitesAboveThreshold,genomicProfileParameters-method
 (genomicProfileParameters-class),
[36](#)

AllSitesAboveThreshold-methods
 (AllSitesAboveThreshold), [7](#)

averageExpPWMScore, [8](#), [24](#)
 averageExpPWMScore,genomicProfileParameters-method
 (genomicProfileParameters-class),
[36](#)

averageExpPWMScore-methods
 (averageExpPWMScore), [8](#)

backgroundSignal, [9](#), [23](#), [49](#)

backgroundSignal,occupancyProfileParameters-method
 (occupancyProfileParameters-class),
[51](#)

backgroundSignal-methods
 (backgroundSignal), [9](#)

backgroundSignal<- , [10](#)

backgroundSignal<- ,occupancyProfileParameters,numeric-method
 (occupancyProfileParameters-class),
[51](#)

backgroundSignal<--methods
 (backgroundSignal<-), [10](#)

boundMolecules, [8](#), [11](#), [23](#), [27](#), [28](#), [49](#), [57](#), [59](#),
[65](#), [72](#), [73](#)

boundMolecules,occupancyProfileParameters-method
 (occupancyProfileParameters-class),
[51](#)

boundMolecules-methods
 (boundMolecules), [11](#)

boundMolecules<- , [12](#)

boundMolecules<- ,occupancyProfileParameters,vector-method
 (occupancyProfileParameters-class),
[51](#)

boundMolecules<--methods
 (boundMolecules<-), [12](#)

BPFrequency, [13](#), [34](#)

BPFrequency,genomicProfileParameters-method
 (genomicProfileParameters-class),
[36](#)

BPFrequency-methods (BPFrequency), [13](#)

BPFrequency<- , [14](#)

BPFrequency<- ,genomicProfileParameters,DNAStringSet-method
 (genomicProfileParameters-class),
[36](#)

BPFrequency<- ,genomicProfileParameters,vector-method
 (genomicProfileParameters-class),
[36](#)

BPFrequency<--methods (BPFrequency<-),
[14](#)

ChIPAnalyser (ChIPAnalyser-package), [3](#)

ChIPAnalyser-package, [3](#)

ChIPAnalyserData, [15](#)

chipMean, [16](#), [16](#), [23](#), [49](#)

chipMean,occupancyProfileParameters-method
 (occupancyProfileParameters-class),
[51](#)

chipMean-methods (chipMean), [16](#)

chipMean<- , [17](#)

chipMean<- ,occupancyProfileParameters,numeric-method
 (occupancyProfileParameters-class),
[51](#)

chipMean<--methods (chipMean<-), [17](#)

chipSd, [18](#), [23](#), [49](#)

chipSd,occupancyProfileParameters-method
 (occupancyProfileParameters-class),
[51](#)

chipSd-methods (chipSd), [18](#)

- chipSd<-, 19
- chipSd<-, occupancyProfileParameters, numeric-method (occupancyProfileParameters-class), 51
- chipSd<--methods (chipSd<-), 19
- chipSmooth, 20, 23, 49
- chipSmooth, occupancyProfileParameters-method (occupancyProfileParameters-class), 51
- chipSmooth-methods (chipSmooth), 20
- chipSmooth<-, 21
- chipSmooth<-, occupancyProfileParameters, vector-method (occupancyProfileParameters-class), 51
- chipSmooth<--methods (chipSmooth<-), 21
- computeChipProfile, 7, 10, 11, 16–21, 22, 65, 74, 76, 77
- computeGenomeWidePWMScore, 9, 24, 29, 32, 40, 43, 72, 73
- computeOccupancy, 7, 8, 10–12, 20–22, 25, 70–74
- computeOptimal, 12, 27, 59
- computePWMScore, 7, 24, 26, 29, 30, 74
- DNASequenceLength, 9, 24, 31
- DNASequenceLength, genomicProfileParameters-method (genomicProfileParameters-class), 36
- DNASequenceLength-methods (DNASequenceLength), 31
- DNASequenceLength<-, 32
- DNASequenceLength<-, genomicProfileParameters, vector-method (genomicProfileParameters-class), 36
- DNASequenceLength<--methods (DNASequenceLength<-), 32
- DNAStrngSet, 13, 14, 24, 27, 29, 34, 36
- eveLocus (ChIPAnalyserData), 15
- eveLocusChip (ChIPAnalyserData), 15
- geneRef (ChIPAnalyserData), 15
- genomicProfileParameters, 7–9, 13–15, 22, 24, 26, 27, 29–32, 33, 38–40, 43–54, 60–64, 66–69, 72–74, 77, 78, 81, 82
- genomicProfileParameters-class, 36
- GRanges, 7, 8, 16, 22–24, 26, 28–30, 46, 57, 65
- GRangesList, 7, 8, 26, 30, 65
- GRList-class, 39, 39
- maxPWMScore, 24, 39
- maxPWMScore, genomicProfileParameters-method (genomicProfileParameters-class), 36
- maxPWMScore-methods (maxPWMScore), 39
- maxSignal, 10, 11, 23, 41, 49
- maxSignal, occupancyProfileParameters-method (occupancyProfileParameters-class), 51
- maxSignal-methods (maxSignal), 41
- maxSignal<-, 42
- maxSignal<-, occupancyProfileParameters, numeric-method (occupancyProfileParameters-class), 51
- maxSignal<--methods (maxSignal<-), 42
- minPWMScore, 24, 43
- minPWMScore, genomicProfileParameters-method (genomicProfileParameters-class), 36
- minPWMScore-methods (minPWMScore), 43
- names, 8
- naturalLog, 34, 44
- naturalLog, genomicProfileParameters-method (genomicProfileParameters-class), 36
- naturalLog-methods (naturalLog), 44
- naturalLog<-, 45
- naturalLog<-, genomicProfileParameters, logical-method (genomicProfileParameters-class), 36
- naturalLog<--methods (naturalLog<-), 45
- NoAccess, 46
- NoAccess, genomicProfileParameters-method (genomicProfileParameters-class), 36
- NoAccess-methods (NoAccess), 46
- noOfSites, 34, 47
- noOfSites, genomicProfileParameters-method (genomicProfileParameters-class), 36
- noOfSites-methods (noOfSites), 47
- noOfSites<-, 48
- noOfSites<-, genomicProfileParameters, vector-method (genomicProfileParameters-class), 36
- noOfSites<--methods (noOfSites<-), 48
- occupancyProfileParameters, 9–13, 16–23, 26, 28, 35, 37, 38, 41, 42, 49, 52, 55–57, 65, 70, 71, 75, 76, 79, 80
- occupancyProfileParameters-class, 51
- PFMFormat, 53
- PFMFormat, genomicProfileParameters-method (genomicProfileParameters-class), 36

- PFMFormat-methods (PFMFormat), 53
- PFMFormat<- , 54
- PFMFormat<- , genomicProfileParameters, character-method
(genomicProfileParameters-class),
36
- PFMFormat<--methods (PFMFormat<-), 54
- ploidy, 49, 55
- ploidy, occupancyProfileParameters-method
(occupancyProfileParameters-class),
51
- ploidy-methods (ploidy), 55
- ploidy<- , 56
- ploidy<- , occupancyProfileParameters, numeric-method
(occupancyProfileParameters-class),
51
- ploidy<--methods (ploidy<-), 56
- plotOccupancyProfile, 10, 11, 57
- plotOptimalHeatMaps, 59, 79, 80
- PositionFrequencyMatrix, 53, 54, 60
- PositionFrequencyMatrix, genomicProfileParameters-method
(genomicProfileParameters-class),
36
- PositionFrequencyMatrix-methods
(PositionFrequencyMatrix), 60
- PositionFrequencyMatrix<- , 61
- PositionFrequencyMatrix<- , genomicProfileParameters, character-method
(genomicProfileParameters-class),
36
- PositionFrequencyMatrix<- , genomicProfileParameters, matrix-method
(genomicProfileParameters-class),
36
- PositionFrequencyMatrix<--methods
(PositionFrequencyMatrix<-), 61
- PositionWeightMatrix, 61, 62
- PositionWeightMatrix, genomicProfileParameters-method
(genomicProfileParameters-class),
36
- PositionWeightMatrix-methods
(PositionWeightMatrix), 62
- PositionWeightMatrix<- , 63
- PositionWeightMatrix<- , genomicProfileParameters, matrix-method
(genomicProfileParameters-class),
36
- PositionWeightMatrix<--methods
(PositionWeightMatrix<-), 63
- profileAccuracyEstimate, 10, 11, 57, 64
- PWMPseudocount, 34, 66
- PWMPseudocount, genomicProfileParameters-method
(genomicProfileParameters-class),
36
- PWMPseudocount-methods
(PWMPseudocount), 66
- PWMPseudocount<- , 67
- PWMPseudocount<- , genomicProfileParameters, numeric-method
(genomicProfileParameters-class),
36
- PWMPseudocount<--methods
(PWMPseudocount<-), 67
- PWMThreshold, 7, 8, 29, 30, 34, 68
- PWMThreshold, genomicProfileParameters-method
(genomicProfileParameters-class),
36
- PWMThreshold-methods (PWMThreshold), 68
- PWMThreshold<- , 69
- PWMThreshold<- , genomicProfileParameters, numeric-method
(genomicProfileParameters-class),
36
- PWMThreshold<--methods
(PWMThreshold<-), 69
- removeBackground, 23, 49, 70
- removeBackground, occupancyProfileParameters-method
(occupancyProfileParameters-class),
51
- removeBackground-methods
(removeBackground), 70
- removeBackground<- , 71
- removeBackground<- , occupancyProfileParameters, vector-method
(occupancyProfileParameters-class),
51
- removeBackground<--methods
(removeBackground<-), 71
- ScalingFactorPWM, 8, 23, 27, 28, 34, 57, 59,
65, 72
- ScalingFactorPWM, genomicProfileParameters-method
(genomicProfileParameters-class),
36
- ScalingFactorPWM-methods
(ScalingFactorPWM), 72
- ScalingFactorPWM<- , 73
- ScalingFactorPWM<- , genomicProfileParameters, vector-method
(genomicProfileParameters-class),
36
- ScalingFactorPWM<--methods
(ScalingFactorPWM<-), 73
- searchSites, 74
- show, occupancyProfileParameters-method
(occupancyProfileParameters-class),
51
- stepSize, 23, 49, 75
- stepSize, occupancyProfileParameters-method
(occupancyProfileParameters-class),
51
- stepSize-methods (stepSize), 75

stepSize<-, 76
stepSize<-, occupancyProfileParameters, numeric-method
 (occupancyProfileParameters-class),
 51
stepSize<--methods (stepSize<-), 76
strandRule, 34, 77
strandRule, genomicProfileParameters-method
 (genomicProfileParameters-class),
 36
strandRule-methods (strandRule), 77
strandRule<-, 78
strandRule<-, genomicProfileParameters, character-method
 (genomicProfileParameters-class),
 36
strandRule<--methods (strandRule<-), 78

thetaThreshold, 50, 59, 79
thetaThreshold, occupancyProfileParameters-method
 (occupancyProfileParameters-class),
 51
thetaThreshold-methods
 (thetaThreshold), 79
thetaThreshold<-, 80
thetaThreshold<-, occupancyProfileParameters, numeric-method
 (occupancyProfileParameters-class),
 51
thetaThreshold<--methods
 (thetaThreshold<-), 80

whichstrand, 34, 77, 78, 81
whichstrand, genomicProfileParameters-method
 (genomicProfileParameters-class),
 36
whichstrand-methods (whichstrand), 81
whichstrand<-, 82
whichstrand<-, genomicProfileParameters, character-method
 (genomicProfileParameters-class),
 36
whichstrand<--methods (whichstrand<-),
 82