

# Package ‘ChIPanalyser’

October 18, 2022

**Type** Package

**Title** ChIPanalyser: Predicting Transcription Factor Binding Sites

**Version** 1.18.0

**Date** 2021-11-18

**Author** Patrick C.N.Martin & Nicolae Radu Zabet

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

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

**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 computeGenomeWide.R parallelInternalFunctionsDev.R GenomicProfileGenericFunctions.R plotOccupancyDev.R plotOptimalHeatMapDev.R DataPreprocessingDev.R DataPreprocessingGenericFunctionsDev.R profileAccuracyEstimateDev.R

**Depends** R (>= 3.5.0), GenomicRanges, Biostrings, BSgenome, RcppRoll, parallel

**Imports** methods, IRanges,  
S4Vectors,grDevices,graphics,stats,utils,rtracklayer,ROCR,  
BiocManager,GenomeInfoDb

**Suggests** BSgenome.Dmelanogaster.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

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

**git\_branch** RELEASE\_3\_15

**git\_last\_commit** 4bc63a9

**git\_last\_commit\_date** 2022-04-26

**Date/Publication** 2022-10-18

## R topics documented:

ChIPanalyser-package . . . . .	5
averageExpPWMScore . . . . .	6
averageExpPWMScore-methods . . . . .	7
backgroundSignal . . . . .	8
backgroundSignal-methods . . . . .	9
backgroundSignal<- . . . . .	9
backgroundSignal<-methods . . . . .	10
boundMolecules . . . . .	10
boundMolecules-methods . . . . .	11
boundMolecules<- . . . . .	12
boundMolecules<-methods . . . . .	13
BPFrequency . . . . .	13
BPFrequency-methods . . . . .	14
BPFrequency<- . . . . .	14
BPFrequency<-methods . . . . .	16
ChIPanalyserData . . . . .	16
chipMean . . . . .	17
chipMean-methods . . . . .	18
chipMean<- . . . . .	18
chipMean<-methods . . . . .	19
ChIPScore-class . . . . .	19
chipSd . . . . .	21
chipSd-methods . . . . .	22
chipSd<- . . . . .	22
chipSd<-methods . . . . .	23
chipSmooth . . . . .	24
chipSmooth-methods . . . . .	25

chipSmooth<-	25
chipSmooth<-methods	26
computeChIPProfile	26
computeGenomeWideScores	28
computeOccupancy	30
computeOptimal	32
computePWMScore	34
DNASequenceLength	36
DNASequenceLength-methods	37
drop	37
drop-methods	38
genomicProfiles	39
genomicProfiles-class	40
genomicProfilesInternal-class	42
GRList-class	44
initialize-methods	44
lambdaPWM	45
lambdaPWM-methods	46
lambdaPWM<-	46
lambdaPWM<-methods	47
loci	47
loci-class	48
loci-methods	49
lociWidth	49
lociWidth-methods	50
lociWidth<-	50
lociWidth<-methods	51
maxPWMScore	51
maxPWMScore-methods	53
maxSignal	53
maxSignal-methods	54
maxSignal<-	54
maxSignal<-methods	55
minPWMScore	55
minPWMScore-methods	56
naturalLog	57
naturalLog-methods	58
naturalLog<-	58
naturalLog<-methods	59
noiseFilter	59
noiseFilter-methods	60
noiseFilter<-	60
noiseFilter<-methods	61
noOfSites	62
noOfSites-methods	63
noOfSites<-	63
noOfSites<-methods	64
nos-class	64

parameterOptions	65
parameterOptions-class	67
PFMFormat	70
PFMFormat-methods	71
PFMFormat<-	71
PFMFormat<-methods	72
ploidy	72
ploidy-methods	73
ploidy<-	73
ploidy<-methods	74
plotOccupancyProfile	75
plotOptimalHeatMaps	77
PositionFrequencyMatrix	79
PositionFrequencyMatrix-methods	80
PositionFrequencyMatrix<-	80
PositionFrequencyMatrix<-methods	81
PositionWeightMatrix	82
PositionWeightMatrix-methods	83
PositionWeightMatrix<-	83
PositionWeightMatrix<-methods	84
processingChIP	84
profileAccuracyEstimate	86
profiles-methods	87
PWMPseudocount	88
PWMPseudocount-methods	89
PWMPseudocount<-	89
PWMPseudocount<-methods	90
PWMThreshold	90
PWMThreshold-methods	91
PWMThreshold<-	92
PWMThreshold<-methods	93
removeBackground	93
removeBackground-methods	94
removeBackground<-	94
removeBackground<-methods	95
scores	95
scores-methods	96
searchSites	96
show-methods	98
stepSize	99
stepSize-methods	100
stepSize<-	100
stepSize<-methods	101
strandRule	101
strandRule-methods	102
strandRule<-	102
strandRule<-methods	103
whichstrand	104

whichstrand-methods . . . . .	105
whichstrand<- . . . . .	105
whichstrand<-methods . . . . .	106

<b>Index</b>	<b>107</b>
--------------	------------

---

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: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

## 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"),"BCDS1x.pfm")
#As an example of genome, this example will run on the Drosophila genome
```

```

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
#Building data objects
GPP <- genomicProfiles(PFM=PFM,PFMFormat="raw",BPFfrequency=DNASequenceSet)

eveLocusChip<-processingChIP(eveLocusChip,eveLocus)
# Computing Genome Wide
GenomeWide <- computeGenomeWideScores(DNASequenceSet = DNASequenceSet,
  genomicsProfiles = GPP)

#Compute PWM Scores
PWMScores <- computePWMScore(genomicsProfiles = GenomeWide,
  DNASequenceSet = DNASequenceSet,
  loci = eveLocus, chromatinState = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(genomicsProfiles = PWMScores,
  parameterOptions = OPP)

#Compute ChIP profiles
chipProfile <- computeChIPProfile(genomicProfiles = Occupancy,
  loci = eveLocus,
  parameterOptions = OPP)
#Estimating accuracy estimate
AccuracyEstimate <- profileAccuracyEstimate(genomicProfiles = chipProfile,
  ChIPScore = eveLocusChip,
  parameterOptions = OPP)

```

---

averageExpPWMScore      *Accessor for averageExpPWMScore slot in a [genomicProfiles](#) object.*

---

### Description

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

### Usage

```
averageExpPWMScore(object)
```

### Arguments

object                  object is a genomicProfiles

## Details

As a general rule, averageExpPWMScore is computed and updated internally by [computeGenomeWideScores](#). 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 [computeGenomeWideScores](#)).

## Value

Returns the averageExpPWMScore of a [genomicProfiles](#) 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 genomicProfiles object
GPP <- genomicProfiles(PFM=PFM, PFMFormat="raw")
# Extracting AllSitesAboveThreshold slot
averageExpPWMScore(GPP)

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

---

averageExpPWMScore-methods

~~ *Methods for Function* averageExpPWMScore ~~

---

## Description

~~ *Methods for function* averageExpPWMScore ~~

## Methods:

signature(object = "genomicProfilesInternal")

---

backgroundSignal      *Accessor method for the backgroundSignal slot in a parameterOptions object.*

---

### Description

Extract or access the backgroundSignal slot in a [parameterOptions](#) object.

### Usage

```
backgroundSignal(object)
```

### Arguments

object                  object is an [parameterOptions](#)

### 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 against 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 [parameterOptions](#) object.

### Value

Returns a backgroundSignal of a [parameterOptions](#) 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 <- parameterOptions()
#Viewing single value in object
backgroundSignal(OPP)
```



---

backgroundSignal-methods

~~ Methods for Function backgroundSignal ~~

---

### Description

~~ Methods for function backgroundSignal ~~

### Methods:

signature(object = "parameterOptions")

---

backgroundSignal<-     Setter method for backgroundSignal slot in a [parameterOptions](#)

---

### Description

Setter method for backgroundSignal slot in a [parameterOptions](#)

### Usage

```
backgroundSignal(object)<-value
```

### Arguments

object	object is an <a href="#">parameterOptions</a> object.
value	value is the value to be assigned to the backgroundSignal slot in <a href="#">parameterOptions</a> . backgroundSignal should be a positive value. Default value is 0.

### 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 [parameterOptions](#) object.

### Value

Returns a [parameterOptions](#) object with a new value assigned to the backgroundSignal slot.

### Author(s)

Patrick C.N. Martin <[pm16057@essex.ac.uk](mailto: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 <- parameterOptions()
# Setting new value for backgroundSignal
backgroundSignal(OPP) <- 0.2
# Viewing whole object with new updated value
OPP
#Viewing single value in object
backgroundSignal(OPP)
```

---

```
backgroundSignal<-methods
```

```
~~ Methods for Function backgroundSignal<- ~~
```

---

**Description**

```
~~ Methods for function backgroundSignal<- ~~
```

**Methods:**

```
backgroundSignal(object)<-value
```

---

```
boundMolecules      Accessor methods for boundMolecules slot in parameterOptions
                     object.
```

---

**Description**

Extract or Access boundMolecules slot in [parameterOptions](#) object.

**Usage**

```
boundMolecules(object)
```

**Arguments**

object            object is a [parameterOptions](#) 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 `parameterOptions` 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 `parameterOptions` object.

**Value**

Returns boundMolecules slot in `parameterOptions` 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 parameterOptions object
OPP <- parameterOptions()
#Checking single value by slot accessor
boundMolecules(OPP)
```

---

boundMolecules-methods

~~ *Methods for Function boundMolecules* ~~

---

**Description**

~~ Methods for function boundMolecules ~~

**Methods:**

signature(object = "parameterOptions")

---

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

---

### Description

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

### Usage

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

### Arguments

object	object is a <a href="#">parameterOptions</a> 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 [parameterOptions](#) 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 [parameterOptions](#) object.

### Value

Returns a [parameterOptions](#) object with an updated value for 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

```
# Building parameterOptions object
OPP <- parameterOptions()
# Setting new boundMolecules value in OPP
boundMolecules(OPP) <- 5000
#Checking value in whole object
OPP
#Checking single value by slot accessor
boundMolecules(OPP)
```

---

```
boundMolecules<-methods
  ~ ~ Methods for Function boundMolecules<- ~ ~
```

---

**Description**

~ ~ Methods for function boundMolecules<- ~ ~

**Methods:**

```
signature(object = "parameterOptions", value = "vector")
```

---

BPFfrequency	Accessor method for BPFfrequency slot in a <a href="#">genomicProfiles</a> object.
--------------	--

---

**Description**

Extract or Access BPFfrequency slot in a [genomicProfiles](#) object.

**Usage**

```
BPFfrequency(object)
```

**Arguments**

object            object is a [genomicProfiles](#)

**Details**

Default value is `c(0.25,0.25,0.25,0.25)` When generating a Position 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 BPFfrequency 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, BPFfrequency will compute base pair frequency from a DNA sequence. The nature of this sequence can be a `BSgenome` or a `DNASTringSet`. In order to decrease run time, it is advised to use `DNASTringSet`

**Value**

Returns BPFfrequency slot in [genomicProfiles](#) object.

**Author(s)**

Patrick C.N. Martin <[pm16057@essex.ac.uk](mailto: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 genomicProfiles object
GPP <- genomicProfiles(PFM=PFM, PFMFormat="raw")
#Extracting BPFrequency slot
BPFfrequency(GPP)
```

---

BPFfrequency-methods    *~~ Methods for Function BPFfrequency ~~*

---

## Description

*~~ Methods for function BPFfrequency ~~*

### Methods:

signature(object = "genomicProfilesInternal")

---

BPFfrequency<-                    *Setter method for BPFfrequency slot in a [genomicProfiles](#) object.*

---

## Description

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

## Usage

```
BPFfrequency(object)<-value
```

## Arguments

object	object is a <a href="#">genomicProfiles</a> 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 <a href="#">genomicProfiles</a> . A <a href="#">DNAStrngSet</a> of the organism of interest. The base pair frequency will automatically be computed and updated in <a href="#">genomicProfiles</a> (Preferred method).

**Details**

Default value is `c(0.25,0.25,0.25,0.25)` When generating a Position 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 `genomicProfiles` object. The nature of this sequence can be `aBSgenome` object or a `DNAStrngSet`. In order to decrease run time, it is advised to use `DNAStrngSet`.

**Value**

Returns a `genomicProfiles` 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"),"BCDS1x.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
# Building genomicProfiles object
GPP <- genomicProfiles(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)
```

---

```
BPFrequency<-methods    ~~ Methods for Function BPFrequency<- ~~
```

---

### Description

```
~~ Methods for function BPFrequency<- ~~
```

#### Methods:

```
signature(object = "genomicProfilesInternal", value = "DNAStringSet")
signature(object = "genomicProfilesInternal", value = "vector")
```

---

```
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 [GRanges](#) containing ChIP score 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	Accessor method for <code>chipMean</code> slot in a <code>parameterOptions</code> object.
----------	---

---

**Description**

Accessor method for `chipMean` slot in a `parameterOptions` object.

**Usage**

```
chipMean(object)
```

**Arguments**

`object`            `object` is a `parameterOptions`

**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 `parameterOptions` object.

**Value**

Returns `chipMean` slot from a `parameterOptions` 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., EisenM.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 parameterOptions object
OPP <- parameterOptions()
#Accessing chipMean slot in OPP
chipMean(OPP)
```

---

chipMean-methods      *~~ Methods for Function chipMean ~~*

---

### Description

~~ Methods for function chipMean ~~

### Methods:

chipMean(object)

---

chipMean<-      *Access methods for chipMean slot in [parameterOptions](#) object.*

---

### Description

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

### Usage

chipMean(object)<-value

### Arguments

object	object is a <a href="#">parameterOptions</a> 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 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 [parameterOptions](#) object.

### Value

Returns a [parameterOptions](#) 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 parameterOptions object
OPP <- parameterOptions()
# Setting new value for slot
chipMean(OPP) <- 250
```

---

chipMean<-methods      *~~ Methods for Function chipMean<- ~~*

---

**Description**

*~~ Methods for function chipMean<- ~~*

**Methods:**

chipMean(object)<-value

---

ChIPScore-class      *Class "ChIPScore"*

---

**Description**

ChIPScore is the result of the [processingChIP](#) function. This object contains the extracted ChIP Score from ChIP data, the loci of interest and optional parameters associated to ChIPanalyzer. The loci of interest will either be user provided or the top n regions as defined by the reduce argument in processingChIP. This object has the sole purpose of aiding the storage and parsing of data and parameters.

**Objects from the Class**

Object of this class are created internally and will be parsed to other objects as is.

**Slots**

**scores:** Object of class "list" List of extracted ChIP scores  
**loci:** Object of class "loci" GRanges containing loci of interest  
**ploidy:** Object of class "numeric" Ploidy level of the organism  
**boundMolecules:** Object of class "vector" Number of Bound molecules to the DNA  
**backgroundSignal:** Object of class "numeric" ChIP background signal (average ChIP score)  
**maxSignal:** Object of class "numeric" max ChIP signal  
**lociWidth:** Object of class "numeric" Width of loci if reduce is used and no loci are provided  
**chipMean:** Object of class "numeric" Average ChIP peak width  
**chipSd:** Object of class "numeric" Standard Deviation of ChIP peak width  
**chipSmooth:** Object of class "vector" Smoothing window width for ChIP score  
**stepSize:** Object of class "numeric" Defining resolution size of ChIP like profiles (10bp = signal will be only considered every 10bp)  
**removeBackground:** Object of class "numeric" Signal Threshold to be removed. Default removes all negative scores  
**noiseFilter:** Object of class "character" Type of noise filter to be used on ChIP data.  
**PWMThreshold:** Object of class "numeric" Threshold of PWM scores that will be selected  
**strandRule:** Object of class "character" Rule to compute strand score (max, mean or sum)  
**whichstrand:** Object of class "character" Which strand should be used to compute PWM scores.  
**lambdaPWM:** Object of class "vector" Lambda value - Scaling factor to the PWM  
**naturalLog:** Object of class "logical" PFM to PWM conversion log transform ( natural log or log2)  
**noOfSites:** Object of class "nos" Number of Sites in the PWM that should be used to compute PWM scores.  
**PWMpseudocount:** Object of class "numeric" PWM pseudocount value for PFM to PWM conversion.  
**paramTag:** Object of class "character" Internal Tag - Code progression

**Extends**

Class "[parameterOptions](#)", directly.

**Methods**

**.loci<-** signature(object = "ChIPScore", value = "loci"): ...  
**.scores<-** signature(object = "ChIPScore", value = "list"): ...  
**initialize** signature(.Object = "ChIPScore"): ...  
**loci** signature(object = "ChIPScore"): ...  
**scores** signature(object = "ChIPScore"): ...  
**show** signature(object = "ChIPScore"): ...

**Author(s)**

Patrick C.N. Martin

**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**

[processingChIP](#)

**Examples**

```
showClass("ChIPScore")
```

---

chipSd

*Accessor method for chipSd slot in a [parameterOptions](#) object.*

---

**Description**

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

**Usage**

```
chipSd(object)
```

**Arguments**

object            object is a [parameterOptions](#)

**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 [parameterOptions](#) object.

**Value**

Returns a [parameterOptions](#) 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 parameterOptions object
OPP <- parameterOptions()
# Accessing chipSd slot
chipSd(OPP)
```

---

chipSd-methods            *~~ Methods for Function chipSd ~~*

---

## Description

*~~ Methods for function chipSd ~~*

### Methods:

chipSd(object)

---

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

---

## Description

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

## Usage

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

## Arguments

object	object is <a href="#">parameterOptions</a> 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 [parameterOptions](#) object.

## Value

Returns a [parameterOptions](#) 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 parameterOptions object
OPP <- parameterOptions()
# Setting new value for chipSd slot
chipSd(OPP) <- 250
```

---

chipSd<-methods      *~~ Methods for Function chipSd<- ~~*

---

## Description

*~~ Methods for function chipSd<- ~~*

## Methods:

`chipSd(object)<-value`

---

chipSmooth	<i>Accessor methods for chipSmooth slot in a <a href="#">parameterOptions</a> object.</i>
------------	---

---

### Description

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

### Usage

```
chipSmooth(object)
```

### Arguments

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

### Value

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

### Author(s)

Patrick C.N. Martin <[pm16057@essex.ac.uk](mailto: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 parameterOptions object
OPP <- parameterOptions()
# Accessing chipSd slot
chipSmooth(OPP)
```



---

chipSmooth-methods    *~~ Methods for Function chipSmooth ~~*

---

**Description**

~~ Methods for function chipSmooth ~~

**Methods:**

signature(object = "parameterOptions")

---

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

---

**Description**

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

**Usage**

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

**Arguments**

object	object is a <a href="#">parameterOptions</a> object.
value	value is the positive numeric value to be assigned to the chipSmooth slot in <a href="#">parameterOptions</a> 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 [parameterOptions](#) object.

**Value**

Returns a [parameterOptions](#) 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 parameterOptions object
OPP <- parameterOptions()
# Setting new value for chipSd slot
chipSmooth(OPP) <- 250
```

---

chipSmooth<-methods    *~~ Methods for Function chipSmooth<- ~~*

---

**Description**

~~ Methods for function chipSmooth<- ~~

**Methods:**

signature(object = "parameterOptions", value = "vector")

---

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(genomicProfiles, loci, parameterOptions = NULL,
  norm = TRUE, method = c("moving_kernel", "truncated_kernel", "exact"),
  peakSignificantThreshold= NULL, cores=1, verbose = TRUE)
```

**Arguments**

**genomicProfiles**  
genomicProfiles is the result of [computeOccupancy](#). This object should be a [genomicProfiles](#) object.

**loci**  
loci is either a [GRanges](#) or [ChIPScore](#) object. ChIPScore-class will be the result of [processingChIP](#). This object represents the set of Loci you are interested in analysing. If you have followed the full ChIPanalyser pipe line, you would have used the processingChIP function that would return a ChIPScore-class object containing your loci of interest. GRanges are also supported if you are only using part of the pipeline.

parameterOptions	parameterOptions is a <a href="#">parameterOptions</a> object. This object is used to store the numerous parameters offered by ChIPAnalyser. This argument is optional as all arguments are also parse in both ChIPScore-class and genomicProfiles objects. If you wanted to make some last minute changes, <a href="#">parameterOptions</a> is the way to go. We recommend that you set your desired options before hand.
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: 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.
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.
cores	cores is the number of cores that will be used to compute ChIP profiles.
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. Internally a few parameters are required to build a ChIP like profile. These parameters are either defined and stored in a [ChIPScore](#) object (Parameters are updated based on your ChIP data), a [genomicProfiles](#) (user defined at the start of the analysis) or a [parameterOptions](#) (if you want to update values as you go along)

## Value

Returns a [genomicProfiles](#) object containing all ChIP-seq like profile for every combination of [lambdaPWM](#) and [boundMolecules](#) provided by the user.

## 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)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
# Building genomicProfiles object
GPP <- genomicProfiles(PFM=PFM, PFMFormat="raw",BPFrequency=DNASequenceSet)

# Computing Genome Wide
GenomeWide <- computeGenomeWideScores(genomicProfiles = GPP,
  DNASequenceSet = DNASequenceSet)

#Compute PWM Scores
PWMScores <- computePWMScore(genomicProfiles = GenomeWide,
  DNASequenceSet = DNASequenceSet, loci = eveLocus, chromatinState = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(genomicProfiles = PWMScores)

#Compute ChIP profiles
chipProfile <- computeChIPProfile(genomicProfiles=Occupancy,loci=eveLocus)
chipProfile

```

---

```
computeGenomeWideScores
```

*Computing Genome Wide scores*

---

**Description**

computeGenomeWideScores compute the max and min PWM score over the entire genome.

**Usage**

```
computeGenomeWideScores(genomicProfiles, DNASequenceSet, chromatinState = NULL, parameterOptions = NULL)
```

**Arguments**

genomicProfiles

genomicProfiles is a genomicProfiles object containing the PFM, PWM of interest.

DNASequenceSet	DNASequenceSet is a BSgenome or DNASTringSet containing the sequence of the organism of interest.
chromatinState	chromatinState is a GRanges object containing the chromatin States. This can either represent regions of accessible DNA or Chromatin state affinities.
parameterOptions	parameterOptions is a parameterOptions object containing parameters that you wish to change. The genomicProfiles object will be updated using the values assigned to parameterOptions
cores	cores is the number of cores that will be used (Numeric value - Default = 1 )
verbose	verbose is a logical value that will determine if internal progress message will be printed.

### Details

computeGenomeWideScores function computes PWM scores over the entire genome (or accessible Genome if chromatin State are provided ). Genome wide scores are used to determine the maximum and minimum PWM score as well as the average exponential score. These scores will in turn be used to determine which score are above the PWM threshold. The average exponential score is an integrable part of the equation used to compute Occupancy. Using default settings, ChIPanalyser will only compute occupancy on the top 70% of PWM scores. This threshold can be changed. See [PWMThreshold](#)

### Value

Returns a genomicProfiles object with updated values for max score, min score and average-ExpPWMScore.

### 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

```
if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
# Building genomicProfiles object
GPP <- genomicProfiles(PFM=PFM, BPFfrequency=DNASequenceSet)
```

```
# Computing Genome Wide
GenomeWide <- computeGenomeWideScores(genomicProfiles = GPP,
  DNASequenceSet = DNASequenceSet)
```

---

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(genomicProfiles,parameterOptions = NULL,
  norm = TRUE, verbose = TRUE)
```

### Arguments

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

**parameterOptions**  
parameterOptions is a [parameterOptions](#) object containing the adequate values assigned to each Parameter. If not Supplied (parameterOptions = 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 [parameterOptions](#) 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 an real ChIP-seq data. We strongly advise that the values assigned to each parameter should be customized in order to increase the model ageement with real world biological data.

**Value**

computeOccupancy will return a [genomicProfiles](#). The main difference will reside in the [profiles](#) slot. This slot is generally a list or [GRangesList](#). Within these list type structures are enclosed [GRanges](#) containing the positions of site above threshold, PWMScores 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 lambdaPWM 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)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

#Building data objects
GPP <- genomicProfiles(PFM=PFM,BPFrequency=DNASequenceSet)
OPP <- parameterOptions()
# Computing Genome Wide
GenomeWide <- computeGenomeWideScores(genomicProfiles = GPP,
  DNASequenceSet = DNASequenceSet)

#Compute PWM Scores
PWMScores <- computePWMScore(genomicProfiles = GenomeWide,
  DNASequenceSet = DNASequenceSet,
  loci = eveLocus,
  chromatinState = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(genomicProfiles = PWMScores,
  parameterOptions = OPP)
Occupancy
```

---

computeOptimal	<i>compute Optimal Parameters</i>
----------------	-----------------------------------

---

## Description

ChIPAnalyser contains a set of functions some of which require two parameters known as [lambdaPWM](#) 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(genomicProfiles, DNASequenceSet, ChIPScore, chromatinState = NULL,
  parameterOptions = NULL, optimalMethod = "all", rank=FALSE, returnAll=TRUE,
  peakMethod="moving_kernel", cores=1)
```

## Arguments

genomicProfiles	genomicProfiles is a <a href="#">genomicProfiles</a> 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.
DNASequenceSet	DNASequenceSet is a <a href="#">DNAStrngSet</a> or a BSgenome of the full sequence of the organism of interest.
ChIPScore	ChIPScore 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.
chromatinState	chromatinState is a <a href="#">GRanges</a> object containing either accesible sites or DNA affinity scores.
parameterOptions	<a href="#">parameterOptions</a> is a <a href="#">parameterOptions</a> object. If this object is not provided (parameterOptions = 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.
optimalMethod	optimalMethod is a character string which determines which method for optimal parameter selection should be selected. optimalMethod can be one of the following: pearson, spearman, kendall, ks, fscore, geometric, MSE, or all. Default is set to all.
rank	rank is a logical value indicating if optimal parameters should be based on rank (parameter combination occurring the most over all regions) or average score (best performing combination of parameters on average over all regions selected). DEFAULT = FALSE



returnAll	returnAll is a logical value indicating if all internal objects should be returned. DEFAULT = TRUE. Internal objects are the following: Occupancy Scores, ChIP like profiles, goodness of fit metrics and optimal parameters. If set to FALSE, computeOptimal will only return the optimal parameters.
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.
cores	cores is the number cores that will be used to compute optimal set of parameters.

### Details

In order to backward infer the values of `lambdaPWM` and `boundMolecules`, it is possible to use the `computeOptimal` to find these parameters. It should be noted that this functions requires a ChIP-seq data input. `ChIPScore` (ChIP-seq data). This should be the output of the `processingChIP` function.

### Value

`computeOptimal` returns a list respectively described as the optimal set of Parameters (`lambda - lambdaPWM` 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 all of the aforementioned parameters (see `optimalMethod`).

### 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)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
eveLocusChip<-processingChIP(eveLocusChip,eveLocus)
```

```

#Building data objects
GPP <- genomicProfiles(PFM=PFM,BPFrequency=DNASequenceSet)
OPP <- parameterOptions()
#Computing Optimal set of Parameters
optimalParam <- computeOptimal(genomicProfiles = GPP,
  DNASequenceSet = DNASequenceSet,
  ChIPScore = eveLocusChip,
  chromatinState = Access,
  parameterOptions = OPP,
  parameter = "all",
  peakMethod="moving_kernel")

```

---

computePWMScore	<i>Compute PWM Scores of sites above threshold.</i>
-----------------	---

---

## 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(genomicProfiles, DNASequenceSet,
  loci = NULL, chromatinState = NULL, parameterOptions=NULL, cores=1, verbose = TRUE)

```

## Arguments

DNASequenceSet	DNASequenceSet is a <a href="#">DNAStringSet</a> or a BSgenome containing the full sequence of the organism of interest.
genomicProfiles	genomicProfiles is a <a href="#">genomicProfiles</a> object resulting from the <a href="#">computeGenomeWideScores</a> function.
loci	loci is a <a href="#">GRanges</a> object containing the Loci of interest or a <a href="#">ChIPScore</a> object result of <a href="#">processingChIP</a> function.
parameterOptions	parameterOptions is a <a href="#">parameterOptions</a> object containing parameters that you wish to parse/change when computing PWM Scores.
chromatinState	chromatinState is a <a href="#">GRanges</a> object sites of accessible DNA or DNA affinity scores.
cores	cores is the number of cores used to compute PWM Scores.
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 `genomicProfiles` object. The `profiles` 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 PWM Score 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"),"BCDSLx.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
eveLocusChip<-processingChIP(eveLocusChip,eveLocus)
#Building data objects
GPP <- genomicProfiles(PFM=PFM,BPFrequency=DNASequenceSet)

# Computing Genome Wide
GenomeWide <- computeGenomeWideScores(DNASequenceSet = DNASequenceSet,
  genomicProfiles = GPP)

#Compute PWM Scores
PWMScores <- computePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfiles = GenomeWide,
  loci = eveLocusChip, chromatinState = Access)
PWMScores
```

---

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

---

### Description

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

### Usage

```
DNASequenceLength(object)
```

### Arguments

object                  object is a [genomicProfiles](#)

### 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 [genomicProfiles](#) 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"),"BCDS1x.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
```

```

library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)

#Building genomicProfiles object
GPP <- genomicProfiles(PFM=PFM,BPFrequency=DNASequenceSet)
# Computing Genome Wide
GenomceWide <- computeGenomeWideScores(DNASequenceSet = DNASequenceSet,
    genomicProfiles = GPP)

DNASequenceLength(GenomceWide)

```

---

DNASequenceLength-methods

~~ Methods for Function DNASequenceLength ~~

---

### Description

~~ Methods for function DNASequenceLength ~~

### Methods:

signature(object = "genomicProfilesInternal")

---

drop

Accessor Method for the drop slot in a [genomicProfiles](#) object.

---

### Description

Accessor Method for the drop slot in a [genomicProfiles](#) object.

### Usage

drop(object)

### Arguments

object            object is a [genomicProfiles](#) 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 accesible 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 <- genomicProfiles(PFM=PFM,PFMFormat="raw")

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

drop(GPP)
```

---

drop-methods

*~~ Methods for Function drop ~~*

---

**Description**

*~~ Methods for function drop ~~*

**Methods:**

signature(object = "genomicProfilesInternal")

---

genomicProfiles	<i>Genomic Profile object</i>
-----------------	-------------------------------

---

### Description

genomicProfiles is an S4 object serving two purposes: (i) storing internal computed data and (ii) storing parameter options. This object is parsed through the different steps of the pipeline to facilitate that parsing and changing of parameters.

### Usage

```
genomicProfiles(..., parameterOptions = NULL, genomicProfiles = NULL, ChIPScore = NULL)
```

### Arguments

...	Any of the user available slots in genomicProfiles.
parameterOptions	If some parameters were already previously computed or stored in a parameterOptions, parsing this object will use those values instead of the default ones.
genomicProfiles	If some parameters were already previously computed or stored in a genomicProfiles, parsing this object will use those values instead of the default ones.
ChIPScore	If some parameters were already previously computed or stored in a ChIPScore, parsing this object will use those values instead of the default ones.

### Details

The genomicProfiles object serves the purpose of storing, and parsing parameters and computed data between the different steps of the pipeline. When creating a genomicProfiles object it is possible to use previously computed values by simply parsing the object to the constructor function.

### Value

Returns a genomicProfiles object with updated slots for all parameters parsed.

### 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

[genomicProfiles](#)  
[parameterOptions](#)

**Examples**

```
PFM <- file.path(system.file("extdata",package="ChIPAnalyser"),"BCDS1x.pfm")
genomicProfiles()
genomicProfiles(PFM=PFM,PFMFormat="raw")
```

---

```
genomicProfiles-class Class "genomicProfiles"
```

---

**Description**

genomicProfiles is an S4 object serving two purposes: (i) storing internal computed data and (ii) storing parameter options. This object is parsed through the different steps of the pipeline to facilitate that parsing and changing of parameters.

**Objects from the Class**

Objects can be created by calls of the form `genomicProfiles(ploidy, boundMolecules, backgroundSignal, maxSignal, lociWidth, chipMean, chipSd, chipSmooth, stepSize, noiseFilter, removeBackground, lambdaPWM, PWMpseudocount, naturalLog, noOfSites, PWMthreshold, strandRule, whichstrand, PFM, PWM, PFMFormat, BPFrequency, minPWMScore, maxPWMScore, profiles, DNASequenceLength, averageExpPWMScore)`.

**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

**BPFrequency:** Object of class "vector": Base Pair Frequency in the genome (if a DNA sequence is provided (as a [DNAStrngSet](#) or BSgenome), will be automatically computed internally). Default:c(0.25,0.25,0.25,0.25)

**minPWMScore:** Object of class "vector": Lowest PWM score accross the genome (computed and updated internally)

**maxPWMScore:** Object of class "vector": Highest PWM score across the genome (computed and updated internally)

**profiles:** Object of class "GRList": Contains GRanges with sites above threshold and associated metrics (PWMScore and Occupancy) - Computed Internally

**DNASequenceLength:** Object of class "vector": Length of the Genome (or accesible genome) - computed internally

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

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



**drop:** Object of class "vector": Stores Loci that do contain accesible DNA if it were to be the case (computed and updated internally)  
**tags:** Object of class "character" ~Internal Tags~  
**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  
**lociWidth:** Object of class "numeric" ~~  
**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)  
**noiseFilter:** Object of class "character" ~Describes the noiseFilter method that will be applied to ChIP data (Zero, mean, median, sigmoid)~  
**PWMThreshold:** Object of class "numeric": Threshold at which PWM Score should be selected (only sites above threshold will be selected - between 0 and 1)  
**strandRule:** Object of class "character": "mean", "max" or "sum" will dertermine 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: "+-"  
**lambdaPWM:** Object of class "vector" A vector (or single value) contaning values for lambdaPWM Default:1  
**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 "nos" 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)  
**PWMPseudocount:** Object of class "numeric": A numeric value describing a PWMPseudocount for PWM computation. Default:1  
**paramTag:** Object of class "character" ~Internal~

## Extends

Class "[genomicProfilesInternal](#)", directly. Class "[parameterOptions](#)", directly.

**Methods**

**initialize** signature(.Object = "genomicProfiles"): ...  
**show** signature(object = "genomicProfiles"): ...

**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**

[genomicProfiles parameterOptions](#)

**Examples**

```
showClass("genomicProfiles")
```

---

```
genomicProfilesInternal-class
      Class "genomicProfilesInternal"
```

---

**Description**

Non exported class. Represents the stripped down version of genomicProfiles.

**Objects from the Class**

Created Internally.

**Slots**

PWM: Object of class "matrix" ~~  
 PFM: Object of class "matrix" ~~  
 PFMFormat: Object of class "character" ~~  
 BPFrequency: Object of class "vector" ~~  
 minPWMScore: Object of class "vector" ~~  
 maxPWMScore: Object of class "vector" ~~  
 profiles: Object of class "GRLList" ~~  
 DNASequenceLength: Object of class "vector" ~~  
 averageExpPWMScore: Object of class "vector" ~~  
 ZeroBackground: Object of class "vector" ~~  
 drop: Object of class "vector" ~~  
 tags: Object of class "character" ~~

**Methods**

**.averageExpPWMScore**<- signature(object = "genomicProfilesInternal", value = "numeric"): ...

**.DNASequenceLength**<- signature(object = "genomicProfilesInternal", value = "vector"): ...

**.drop**<- signature(object = "genomicProfilesInternal", value = "vector"): ...

**.generatePWM** signature(object = "genomicProfilesInternal"): ...

**.maxPWMScore**<- signature(object = "genomicProfilesInternal", value = "vector"): ...

**.minPWMScore**<- signature(object = "genomicProfilesInternal", value = "vector"): ...

**.profiles**<- signature(object = "genomicProfilesInternal", value = "GRLList"): ...

**.tags** signature(object = "genomicProfilesInternal"): ...

**.tags**<- signature(object = "genomicProfilesInternal", value = "character"): ...

**averageExpPWMScore** signature(object = "genomicProfilesInternal"): ...

**BPFrequency** signature(object = "genomicProfilesInternal"): ...

**BPFrequency**<- signature(object = "genomicProfilesInternal", value = "DNAStrngSet"): ...

**BPFrequency**<- signature(object = "genomicProfilesInternal", value = "vector"): ...

**DNASequenceLength** signature(object = "genomicProfilesInternal"): ...

**drop** signature(object = "genomicProfilesInternal"): ...

**maxPWMScore** signature(object = "genomicProfilesInternal"): ...

**minPWMScore** signature(object = "genomicProfilesInternal"): ...

**PFMFormat** signature(object = "genomicProfilesInternal"): ...

**PFMFormat**<- signature(object = "genomicProfilesInternal", value = "character"): ...

**PositionFrequencyMatrix** signature(object = "genomicProfilesInternal"): ...

**PositionFrequencyMatrix**<- signature(object = "genomicProfilesInternal", value = "character"): ...

**PositionFrequencyMatrix**<- signature(object = "genomicProfilesInternal", value = "matrix"): ...

**PositionWeightMatrix** signature(object = "genomicProfilesInternal"): ...

**PositionWeightMatrix**<- signature(object = "genomicProfilesInternal", value = "matrix"): ...

**profiles** signature(object = "genomicProfilesInternal"): ...

**Author(s)**

Patrick C. N. Martin <pml6057@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**

[genomicProfiles](#)  
[parameterOptions](#)

**Examples**

```
showClass("genomicProfilesInternal")
```

---

GRList-class	Class "GRList"
--------------	----------------

---

**Description**

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

**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")
```

---

initialize-methods	~~ <i>Methods for Function initialize</i> ~~
--------------------	--

---

**Description**

~~ *Methods for function initialize* ~~

**Methods:**

signature(.Object = "ChIPScore") Initialize ChIPScore  
signature(.Object = "genomicProfiles") Initialize genomicProfiles  
signature(.Object = "parameterOptions") Initialize parameterOptions

---

lambdaPWM	<i>Accessor Method for the lambdaPWM slot in a <a href="#">parameterOptions</a> object</i>
-----------	--

---

### Description

Accessor Method for the lambdaPWM slot in a [parameterOptions](#) object

### Usage

```
lambdaPWM(object)
```

### Arguments

object            object is [parameterOptions](#) object

### Details

The model underlying ChIPanalyser internally infers two parameters: number of bound molecules and lambda. Lambda represents a scaling factor for the Position weight matrix (PWM). This can be described as how well does a TF discriminate between high affinity and very high affinity sites.

### Value

Returns the value assigned to the lambdaPWM slot in a [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(lambdaPWM=1)
#Setting new Value for lambdaPWM
lambdaPWM(GPP)
```

---

lambdaPWM-methods      *~~ Methods for Function lambdaPWM ~~*

---

### Description

~~ Methods for function lambdaPWM ~~

#### Methods:

lambdaPWM(object)

---

lambdaPWM<-      *Setter Method for the lambdaPWM slot in a [parameterOptions](#) object*

---

### Description

Setter Method for the lambdaPWM slot in a [parameterOptions](#) object

### Usage

```
lambdaPWM(object)<-value
```

### Arguments

object	object is <a href="#">parameterOptions</a> object
value	value is the numeric value to be assigned to the lambdaPWM slot. Default set at 1.

### Details

The model underlying ChIPanalyser internally infers two parameters: number of bound molecules and lambda. Lambda represents a scaling factor for the Position weight matrix (PWM). This can be described as how well does a TF discriminate between high affinity and very high affinity sites.

### Value

Returns the value assigned to the lambdaPWM slot in a [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(lambdaPWM=1)
#Setting new Value for lambdaPWM
lambdaPWM(GPP) <- 2
```

---

lambdaPWM<-methods      *~~ Methods for Function lambdaPWM<- ~~*

---

**Description**

Setter method for the lambdaPWM slot in the parameterOptions

**Methods:**

lambdaPWM(object)<-value

---

loci                      *Accessor Method for the loci slot in a [ChIPScore](#) object*

---

**Description**

Setter Method for the loci slot in a [ChIPScore](#) object

**Usage**

```
loci(object)
```

**Arguments**

object                  object is [ChIPScore](#) object

**Details**

When using the [processingChIP](#), this functions will return a name GRanges with the loci of interest. These loci will either result from user input or extracted from the CHIP profiles (see [processingChIP](#) and [lociWidth](#)). This functions enables you to extract those loci from the [ChIPScore](#) object.

**Value**

Returns the value assigned to the loci slot in a [ChIPScore](#) 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)

eveLocusChip<-processingChIP(eveLocusChip,eveLocus)
loci(eveLocusChip)
```

---

loci-class

*Class "loci"*

---

**Description**

Setter for Loci of interest parsed to or extracted from the ChIPScore object

**Objects from the Class**

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

**Methods**

**.loci<-** signature(object = "ChIPScore", value = "loci"): ...

**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**

[ChIPScore](#)

**Examples**

```
showClass("loci")
```



---

loci-methods                      *~~ Methods for Function loci ~~*

---

**Description**

Accessor method for the loci slot in [ChIPScore](#)

**Methods:**

loci{Object} Loci of interest parsed to or extracted from the ChIPScore object

---

lociWidth                      *Accessor Method for the lociWidth slot in a [parameterOptions](#) object*

---

**Description**

Setter Method for the lociWidth slot in a [parameterOptions](#) object

**Usage**

lociWidth(object)

**Arguments**

object                      object is [parameterOptions](#) object

**Details**

When using the [processingChIP](#) function, the provided ChIP scores will be split into bins of a given size. lociWidth determines the Size of that bin. Default is set at 20 000 bp. This means that the ChIP profiles provided will be split into bins of 20 000 bp over the entire profile provided if no loci of interest is provided.

**Value**

Returns the value assigned to the lociWidth slot in a [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(lociWidth=20000)
#Accessing new Value for lociWidth
lociWidth(GPP)
```

---

lociWidth-methods      *~~ Methods for Function lociWidth ~~*

---

**Description**

Accessor method for the loci slot in [ChIPScore](#)

**Methods:**

lociWidth(object) Setting width of regions when using the reduce argument and NOT providing your own loci when using the processingChIP function.

---

lociWidth<-              *Setter Method for the lociWidth slot in a [parameterOptions](#) object*

---

**Description**

Setter Method for the lociWidth slot in a [parameterOptions](#) object

**Usage**

```
lociWidth(object)<-value
```

**Arguments**

object                  object is [parameterOptions](#) object  
value                    value is the numeric value to be assigned to the lociWidth slot. Default set at 1.

**Details**

When using the [processingChIP](#) function, the provided ChIP scores will be split into bins of a given size. lociWidth determines the Size of that bin. Default is set at 20 000 bp. This mean that the ChIP profiles provided will be split into bins of 20 000 bp over the entire profile provided if no loci of interest is provided.

**Value**

Returns the value assigned to the lociWidth slot in a [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(lociWidth=20000)
#Setting new Value for lociWidth
lociWidth(GPP) <- 30000
```

---

lociWidth<-methods      *~~ Methods for Function lociWidth<- ~~*

---

**Description**

Setter method for the loci slot in [ChIPScore](#)

**Methods:**

lociWidth(Object)<-value

---

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

---

**Description**

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

**Usage**

maxPWMScore(object)

## Arguments

object            object is a [genomicProfiles](#) 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 [genomicProfiles](#) object after using the [computeGenomeWideScores](#).

## Value

Returns the value of assigned to the maxPWMScore slot in a [genomicProfiles](#) 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"),"BCDS1x.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
#Building data objects
GPP <- genomicProfiles(PFM=PFM,PFMFormat="raw")

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

---

maxPWMScore-methods    *~~ Methods for Function maxPWMScore ~~*

---

**Description**

Accessor method for maxPWMScore

**Methods:**

maxPWMScore(object)

---

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

---

**Description**

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

**Usage**

maxSignal(object)

**Arguments**

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

**Value**

Returns the value assigned to the maxSignal slot in a [parameterOptions](#) 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 parameterOptions object
OPP <- parameterOptions()
#Setting new Value for maxSignal
maxSignal(OPP)
```

---

`maxSignal-methods`      *~~ Methods for Function maxSignal ~~*

---

**Description**

Accessor method for `maxSignal`

**Methods:**

`maxSignal(object)` Maximum ChIP signal extracted from ChIP data (see `processingChIP`)

---

`maxSignal<-`                      *Setter method for maxSignal slot in a [parameterOptions](#) object.*

---

**Description**

Setter method for `maxSignal` slot in a [parameterOptions](#) object.

**Usage**

```
maxSignal(object) <- value
```

**Arguments**

<code>object</code>	<code>object</code> is a <a href="#">parameterOptions</a> object.
<code>value</code>	<code>value</code> is a numerical value to be assigned to the <code>maxSignal</code> slot.

**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 [parameterOptions](#) object.

**Value**

Returns a [parameterOptions](#) with an updated value for `maxSignal`.

**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 parameterOptions object
OPP <- parameterOptions()
#Setting new Value for maxSignal
maxSignal(OPP) <- 1.8
```

---

maxSignal<-methods      *~~ Methods for Function maxSignal<- ~~*

---

**Description**

Setter method for maxSignal

**Methods:**

maxSignal(Object)<-value Maximum ChIP signal extracted from ChIP data (see processingChIP)

---

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

---

**Description**

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

**Usage**

```
minPWMScore(object)
```

**Arguments**

object      object is a [genomicProfiles](#) 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 [computeGenomeWideScores](#) function.

**Value**

Returns the value assigned to the minPWMScore slot in a [genomicProfiles](#) 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"),"BCDS1x.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
#Building data objects
GPP <- genomicProfiles(PFM=PFM,PFMFormat="raw")

# Computing Genome Wide
GenomceWide <- computeGenomeWideScores(DNASequenceSet = DNASequenceSet,
  genomicProfiles = GPP)
minPWMScore(GenomceWide)

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

---

minPWMScore-methods    *~~ Methods for Function minPWMScore ~~*

---

**Description**

Accessor for minPWMScore

**Methods:**

minPWMScore(object) Minimum PWM score computed during the computeGenomeWideScores step.



---

naturalLog	Accessor method the naturalLog slot in a <a href="#">parameterOptions</a> object.
------------	---

---

**Description**

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

**Usage**

```
naturalLog(object)
```

**Arguments**

object            object is [parameterOptions](#) 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 [parameterOptions](#) object.

**Value**

Returns the value assigned to the naturalLog slot in a [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(naturalLog=TRUE)
#Setting new Value for naturalLog
naturalLog(GPP)
```

---

naturalLog-methods     *~~ Methods for Function naturalLog ~~*

---

### Description

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

### Methods:

naturalLog(object)

---

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

---

### Description

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

### Usage

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

### Arguments

object	object is a <a href="#">parameterOptions</a> 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 [parameterOptions](#) object.

### Value

Returns [parameterOptions](#) 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)

#Building data objects
OPP <- parameterOptions(naturalLog=TRUE)
#Setting new Value for naturalLog
naturalLog(OPP) <- FALSE
```

---

naturalLog<-methods     *~~ Methods for Function naturalLog<- ~~*

---

**Description**

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

**Methods:**

naturalLog(object)<-value

---

noiseFilter                    *Accessor Method for the noiseFilter slot in a [parameterOptions](#) object*

---

**Description**

Accessor Method for the noiseFilter slot in a [parameterOptions](#) object

**Usage**

```
noiseFilter(object)
```

**Arguments**

object                    object is [parameterOptions](#) object

**Details**

Noise filtering method that should be used on ChIP-seq data. Four methods are available: Zero, Mean, Median and Sigmoid. Zero removes all ChIP-seq scores bellow zero, mean under the mean score, median under median score and sigmoid assigns a weight to each score based on a logistic regression curve. Mid point is set at 95 95 quantile of ChIP-seq scores. Below midpoint will receive a score between 0 and 1 , everything above will receive a score between 1 and 2

**Value**

Returns the value assigned to the noiseFilter slot in a [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(noiseFilter="sigmoid")
#Setting new Value for noiseFilter
noiseFilter(GPP)
```

---

noiseFilter-methods    *~~ Methods for Function noiseFilter ~~*

---

**Description**

Accessor method for noiseFilter

**Methods:**

noiseFilter(object) Noise Filter that will be applied to ChIP scores

---

noiseFilter<-                    *Setter Method for the noiseFilter slot in a [parameterOptions](#) object*

---

**Description**

Setter Method for the noiseFilter slot in a [parameterOptions](#) object

**Usage**

```
noiseFilter(object) <- value
```

**Arguments**

object            object is [parameterOptions](#) object  
value            value is the value to be assigned to the noiseFilter slot (zero - mean - median -sigmoid)

**Details**

Noise filtering method that should be used on ChIP-seq data. Four methods are available: Zero, Mean, Median and Sigmoid. Zero removes all ChIP-seq scores bellow zero, mean under the mean score, median under median score and sigmoid assigns a weight to each score based on a logistic regression curve. Mid point is set at 95 95 quantile of ChIP-seq scores. Below midpoint will receive a score between 0 and 1 , everything above will receive a score between 1 and 2

**Value**

Returns the value assigned to the noiseFilter slot in a [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(noiseFilter="sigmoid")
#Setting new Value for noiseFilter
noiseFilter(GPP) <-"zero"
```

---

noiseFilter<-methods    *~~ Methods for Function noiseFilter<- ~~*

---

**Description**

Setter method for noiseFilter

**Methods:**

noiseFilter(object)<-value Noise Filter that will be applied to ChIP scores

---

noOfSites	<i>Accessor Method for the noOfSites slot in a <a href="#">parameterOptions</a> object</i>
-----------	--

---

### Description

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

### Usage

```
noOfSites(object)
```

### Arguments

object            object is [parameterOptions](#) object

### 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 "all". In this case, all sites will be used to compute the PWM.

### Value

Returns the value assigned to the noOfSites slot in a [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(noOfSites="all")
#Setting new Value for naturalLog
noOfSites(GPP)
```

---

noOfSites-methods      *~~ Methods for Function noOfSites ~~*

---

**Description**

~~ Methods for function noOfSites ~~

**Methods:**

signature(object = "parameterOptions")

---

noOfSites<-                      *Setter Method for the noOfSites slot in a [parameterOptions](#) object.*

---

**Description**

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

**Usage**

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

**Arguments**

object                      object is a [parameterOptions](#) 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 "all". In this case, all sites will be used to compute the PWM.

**Value**

Returns a [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(noOfSites=0)
#Setting new Value for naturalLog
noOfSites(GPP) <- 8
```

---

```
noOfSites<-methods      ~~ Methods for Function noOfSites<- ~~
```

---

**Description**

Setter method for noOfSites

**Methods:**

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

---

```
nos-class              Class "nos"
```

---

**Description**

Virtual class to handle Number of Sites

**Objects from the Class**

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

**Methods**

No methods defined with class "nos" in the signature.

**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**

```
showClass("nos")
```



---

parameterOptions	<i>parameter Options object</i>
------------------	---------------------------------

---

## Description

parameterOptions is an object used to store and parse the various parameters needed throughout this analysis pipeline.

## Usage

```
parameterOptions(ploidy = 2, boundMolecules = 1000, backgroundSignal = 0, maxSignal = 1, lociWidth = 2000)
```

## Arguments

ploidy	<b>ploidy</b> : A numeric Value describing the ploidy of the organism. Default: 2
boundMolecules	<b>boundMolecules</b> : A vector (or single value) containing the number of bound Molecules (bound Transcription Factors): Default: 1000
backgroundSignal	<b>backgroundSignal</b> : A numeric value describing the ChIP-seq background Signal (average signal from real ChIP seq data). Default: 0
maxSignal	<b>maxSignal</b> : A numeric value describing the highest ChIP-seq signal (from real ChIP-seq data). Default: 1
lociWidth	<b>lociWidth</b> : A numeric value describing the width of the bins used to split ChIP profiles parsed to <a href="#">processingChIP</a> . Default = 20000
chipMean	<b>chipMean</b> : A numeric value describing the mean width of a ChIP- seq peak: Default:200
chipSd	<b>chipSd</b> : A numeric value describing the standard deviation of ChIP-seq peaks. Default: 200
chipSmooth	<b>chipSmooth</b> : A numeric value describing the width of the window used to smooth Occupancy profiles into ChIP profiles. Default:250
stepSize	<b>stepSize</b> : A numeric value describing the step Size (in base pairs) between each ChIP-seq score. Default:10 (Scored every 10 base pairs)
removeBackground	<b>removeBackground</b> : A numeric value describing the value at which score should be removed. Default:0 (If negative scores then remove)
noiseFilter	<b>noiseFilter</b> : A character string of one of the following: Zero, Mean, Median, or Sigmoid. Noise filter that will be applied to the ChIP Score during the processingChIP step.
naturalLog	<b>naturalLog</b> : A logical value describing if natural Log will be used to compute the PWM (if FALSE then log2 will be used). Default: TRUE
noOfSites	<b>noOfSites</b> : 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	<b>PWMThreshold:</b> Threshold at which PWM Score should be selected (only sites above threshold will be selected - between 0 and 1)
strandRule	<b>strandRule:</b> 'mean', 'max' or 'sum' will determine how strand should be handle for computing PWM Scores. Default : 'max'
whichstrand	<b>whichstrand:</b> '+', '-' or '+-' on which strand should PWM Score be computed. Default: '+-'
PWmpseudocount	<b>PWmpseudocount:</b> A numeric value describing a PWmpseudocount for PWM computation. Default:1
lambdaPWM	A vector (or single value) containing values for the ScalingFactorPWM (Also known as lambda).Default:1

### Details

ChIPAnalyser requires a lot of parameters. parameterOptions was created with the intent of storing and parsing these numerous arguments to the different functions. All parameters in this object are optional although strongly recommend. Some parameters are extracted and updated from function along the pipeline e.g. maxSignal and backgroundSignal are extracted during the processingChIP step. These parameters will be automatically parsed. If you do not wish to use them ( or any other parameter) simply parse a new parameterOptions object with your desired parameters.

### Value

Returns a parameterOptions with updated values.

### 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

[genomicProfiles](#)

### Examples

```
#
parameterOptions(ploidy = 2, boundMolecules = 1000, backgroundSignal = 0,
  maxSignal = 1, lociWidth = 20000, chipMean = 200, chipSd = 200,
  chipSmooth = 250, stepSize = 10, removeBackground = 0, noiseFilter = "zero",
  naturalLog = TRUE, noOfSites = "all", PWMThreshold = 0.7,
  strandRule = "max", whichstrand = "+-", PWmpseudocount = 1,
  lambdaPWM = 1)
```

---

 parameterOptions-class

 Class "parameterOptions"
 

---

### Description

parameterOptions is an object used to store and parse the various parameters needed throughout this analysis pipeline.

### Objects from the Class

Objects can be created by calls of the form parameterOptions(ploidy, boundMolecules, backgroundSignal, maxSignal, lociWidth, chipMean, chipSd, chipSmooth, stepSize, noiseFilter, removeBackground, lambdaPWM, PWMpseudocount, naturalLog, noOfSites, PWMThreshold, strandRule, whichstrand).

### 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

**lociWidth:** Object of class "numeric": A numeric value describing bin size when splitting ChIP seq scores). Default: 20 000

**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)

**noiseFilter:** Object of class "character" Describes noiseFilter method applied to ChIP scores

**PWMThreshold:** Object of class "numeric": Threshold at which PWM Score should be selected (only sites above threshold will be selected - between 0 and 1)

**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: "+-"

**lambdaPWM:** Object of class "vector" A vector (or single value) containing values for lambdaPWM Default:1

**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 "nos" 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)

**PWMpseudocount:** Object of class "numeric": A numeric value describing a PWMpseudocount for PWM computation. Default:1

**paramTag:** Object of class "character" ~Internal~

## Methods

**.paramTag** signature(object = "parameterOptions"): ...

**.paramTag<-** signature(object = "parameterOptions", value = "character"): ...

**.ZeroBackground** signature(object = "parameterOptions"): ...

**.ZeroBackground<-** signature(object = "parameterOptions", value = "vector"): ...

**backgroundSignal** signature(object = "parameterOptions"): ...

**backgroundSignal<-** signature(object = "parameterOptions", value = "numeric"): ...

**boundMolecules** signature(object = "parameterOptions"): ...

**boundMolecules<-** signature(object = "parameterOptions", value = "vector"): ...

**chipMean** signature(object = "parameterOptions"): ...

**chipMean<-** signature(object = "parameterOptions", value = "numeric"): ...

**chipSd** signature(object = "parameterOptions"): ...

**chipSd<-** signature(object = "parameterOptions", value = "numeric"): ...

**chipSmooth** signature(object = "parameterOptions"): ...

**chipSmooth<-** signature(object = "parameterOptions", value = "vector"): ...

**initialize** signature(.Object = "parameterOptions"): ...

**lambdaPWM** signature(object = "parameterOptions"): ...

**lambdaPWM<-** signature(object = "parameterOptions", value = "vector"): ...

**lociWidth** signature(object = "parameterOptions"): ...

**lociWidth<-** signature(object = "parameterOptions", value = "numeric"): ...

**maxSignal** signature(object = "parameterOptions"): ...

**maxSignal<-** signature(object = "parameterOptions", value = "numeric"): ...

**naturalLog** signature(object = "parameterOptions"): ...

**naturalLog<-** signature(object = "parameterOptions", value = "logical"): ...

**noiseFilter** signature(object = "parameterOptions"): ...

**noiseFilter<-** signature(object = "parameterOptions", value = "character"): ...

**noOfSites** signature(object = "parameterOptions"): ...  
**noOfSites<-** signature(object = "parameterOptions", value = "character"): ...  
**noOfSites<-** signature(object = "parameterOptions", value = "numeric"): ...  
**ploidy** signature(object = "parameterOptions"): ...  
**ploidy<-** signature(object = "parameterOptions", value = "numeric"): ...  
**PWMpseudocount** signature(object = "parameterOptions"): ...  
**PWMpseudocount<-** signature(object = "parameterOptions", value = "numeric"): ...  
**PWMThreshold** signature(object = "parameterOptions"): ...  
**PWMThreshold<-** signature(object = "parameterOptions", value = "numeric"): ...  
**removeBackground** signature(object = "parameterOptions"): ...  
**removeBackground<-** signature(object = "parameterOptions", value = "vector"): ...  
**show** signature(object = "parameterOptions"): ...  
**stepSize** signature(object = "parameterOptions"): ...  
**stepSize<-** signature(object = "parameterOptions", value = "numeric"): ...  
**strandRule** signature(object = "parameterOptions"): ...  
**strandRule<-** signature(object = "parameterOptions", value = "character"): ...  
**whichstrand** signature(object = "parameterOptions"): ...  
**whichstrand<-** signature(object = "parameterOptions", value = "character"): ...

### 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

[genomicProfiles](#)

### Examples

```
showClass("parameterOptions")
```

---

PFMFormat                      *Accessor method for the PFMFormat slot in a [genomicProfiles](#) object*

---

### Description

Accessor method for the PFMFormat slot in a [genomicProfiles](#) object

### Usage

```
PFMFormat(object)
```

### Arguments

object                      object is a [genomicProfiles](#) 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 [genomicProfiles](#) with a [PositionFrequencyMatrix](#) 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 [genomicProfiles](#)

### 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 <- genomicProfiles(PFM=PFM, PFMFormat="raw")
#Setting New value for PFMFormat
PFMFormat(GPP)
```

---

PFMFormat-methods      *~~ Methods for Function PFMFormat ~~*

---

**Description**

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

**Methods:**

PFMFormat(object)

---

PFMFormat<-                      *Setter method for the PFMFormat slot in a [genomicProfiles](#) object*

---

**Description**

Setter method for the PFMFormat slot in a [genomicProfiles](#) object

**Usage**

PFMFormat(object) <- value

**Arguments**

object	object is a <a href="#">genomicProfiles</a> object
value	value is character string of one of the following: "raw", "transfac", "JASPAR" or "sequences". If loading a <a href="#">PositionFrequencyMatrix</a> from a file, the format of the file should specified. Default is raw.

**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 [genomicProfiles](#) with a PositionFrequency-Matrix file, this will not influence the parsing of the file.

**Value**

Returns a [genomicProfiles](#) object with an updated value for the PFMFormat 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
#### THIS IS THE PREFERRED METHOD FOR SETTING PFMFormat
GPP <- genomicProfiles(PFM=PFM,PFMFormat="raw")
#Setting New value for PFMFormat
PFMFormat(GPP) <- "raw"
```

---

PFMFormat<-methods      *~~ Methods for Function PFMFormat<- ~~*

---

## Description

Setter method for the PFMFormat slot in a [genomicProfiles](#) object

### Methods:

PFMFormat(object)<-value

---

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

---

## Description

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

## Usage

```
ploidy(object)
```

## Arguments

object                      object is a [parameterOptions](#) 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 [parameterOptions](#) 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 parameterOptions object
OPP <- parameterOptions()
#Setting new Value for maxSignal
ploidy(OPP)
```

---

ploidy-methods            *~~ Methods for Function ploidy ~~*

---

**Description**

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

**Methods:**

ploidy(object)

---

ploidy<-                    *Setter Method for the ploidy slot in an parameterOptions object*

---

**Description**

Setter Method for the ploidy slot in an [parameterOptions](#) object

**Usage**

```
ploidy(object)<- value
```

**Arguments**

object                    object is a [parameterOptions](#) object  
value                      value is a positive integer that describes the ploidy of the organism of interest.

**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 however this only considers simple polyploidy (or haploidy). The model does not consider hybrids such as wheat.

**Value**

Returns a [parameterOptions](#) object with an updated value for the ploidy 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 parameterOptions object
OPP <- parameterOptions()
#Setting new Value for maxSignal
ploidy(OPP) <- 2
```

---

ploidy<-methods      *~~ Methods for Function ploidy<- ~~*

---

**Description**

Setter Method for the ploidy slot in an [parameterOptions](#) object

**Methods:**

ploidy(object)<-value

---

plotOccupancyProfile *Plot Occupancy Profiles*

---

### Description

plotOccupancyProfile plots the predicted profiles. If provided, this functions will also plot ChIP-seq profiles, PWMScores (or Occupancy), chromatin States, Goodness of Fit estimates and gene information.

### Usage

```
plotOccupancyProfile(predictedProfile, ChIPScore = NULL, chromatinState = NULL
, occupancy = NULL, goodnessOfFit = NULL, PWM=FALSE,
geneRef = NULL, axis=TRUE, ...)
```

### Arguments

predictedProfile	predictedProfile is a either <a href="#">GRanges</a> containing the predicted profiles for one loci, all loci selected for one paramter, or all loci selected for all parameter combinations selected. (see <a href="#">searchSites</a> )
ChIPScore	ChIPScore is a ChIPscore object containing ChIPscore (or a list of numeric values representing ChIP scores (Experimental ChIP))
chromatinState	chromatinState is a <a href="#">GRanges</a> containing accesible DNA sites or chromatin States.
occupancy	occupancy is a <a href="#">GRanges</a> or a genomicProfiles object contaning PWM scores and Occupancy ( see computeOccupancy)
goodnessOfFit	goodnessOfFit results of the <a href="#">profileAccuracyEstimate</a> function.
PWM	PWM is a logical value that in the case occupancy is provided which of occupancy scores of PWM scores hould be plotted. Default set at FALSE
geneRef	geneRef is a <a href="#">GRanges</a> containing gene information on exons,introns, UTR's, enhancers or any other genetic element to be plotted.
axis	axis is a logical value indicating if axes should be plotted or not.
...	Any other graphical Parameter of the following : col, density, border, lty, lwd, cex, cex.axis, xlab, ylab, xlim, ylim, las and axislables. Each Parameter will be parsed in the same order as the arguments to this function. If the name of the argument is specified, the argument value will be parsed to the correct internal plotting. See examples below

### Details

Once the predicted ChIP-seq like profiles have been computed, it is possible to plot these profiles.

**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. **IMPORTANT:** the colours described above represent the default colour values. If colours are provided by the user, they will be updated in plotting order or if the argument has been speicified assigned to said argument.

**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)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
#Building data objects
GPP <- genomicProfiles(PFM=PFM,BPFrequency=DNASequenceSet)

# Computing Genome Wide
GenomeWide <- computeGenomeWideScores(DNASequenceSet = DNASequenceSet,
  genomicProfiles = GPP)

#Compute PWM Scores
PWMScores <- computePWMScore(DNASequenceSet = DNASequenceSet,
  genomicProfiles = GenomeWide,
  loci = eveLocus, chromatinState = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(genomicProfiles = PWMScores)

#Compute ChIP profiles
```

```

chipProfile <- computeChipProfile(loci = eveLocus,
  genomicProfiles = Occupancy)

#Plotting Profile
plotOccupancyProfile(predictedProfile=chipProfile,
  ChIPScore = eveLocusChip,
  chromatinState = Access,
  occupancy = Occupancy,
  geneRef =geneRef)

## Changing graphical Parameters

#### In this examples ####
### predictedProfile will be red
### chipProfile will be blue
### and DNAAccessibility will be green
plotOccupancyProfile(predictedProfile=chipProfile,
  ChIPScore = eveLocusChip,
  chromatinState = Access,
  occupancy = Occupancy,
  geneRef =geneRef,col=c("red", "blue", "green"))

### If name is specified
plotOccupancyProfile(predictedProfile=chipProfile,
  ChIPScore = eveLocusChip,
  chromatinState = Access,
  occupancy = Occupancy,
  geneRef =geneRef,col=c("DNAAccessibility"="red", "blue", "green"))

```

---

plotOptimalHeatMaps    *Heat Map of optimal Parameters*

---

## Description

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

## Usage

```
plotOptimalHeatMaps(optimalParam, contour=TRUE, col=NULL, main=NULL, layout=TRUE, overlay=FALSE)
```

## Arguments

**optimalParam**    optimalParam is a list containing containing optimal matrices (or only one if only one paramter was selected). These matrices are the result of the computeOptimal function

contour	parameter is logical. Should contour lines be plotted?
col	col vector of colours to be used for each heat map. If none are specified, rainbow colours will be used. NOTE: colour vector will be recycled if not enough colours are provided.
main	main title.
layout	layout is either TRUE or FALSE specifying if standard layout should be used or not. If TRUE, each heat map will be plotted on an individual page with a heat map scale of the right side.
overlay	overlay is either TRUE or FALSE specifying if an overlay plot should be produced. The overlay plot takes the top 10 percent of best performing parameters per scoring metric and overlays them in a single plot. The resulting plots shows the optimal set of parameters for all metrics combined.

### Details

Once the optimal set of Parameters ( [lambdaPWM](#) and [boundMolecules](#) ), it is possible to plot the results in the form of a heat map. Each heat map will be plotted in a separate page if layout = TRUE, If layout= FALSE, it is up to the user to define how they wish to layout their heat maps.

### Value

Returns a heat map of optimal combinations of [lambdaPWM](#) and [boundMolecules](#). The x axis represents the different value assigned to lambda ( [lambdaPWM](#) ) and the y axis represents the different values to boundMolecules ( [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"),"BCDS1x.pfm")
#As an example of genome, this example will run on the Drosophila genome

if(!require("BSgenome.Dmelanogaster.UCSC.dm3", character.only = TRUE)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
```

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

#Computing Optimal set of Parameters
optimalParam <- computeOptimal(genomicProfiles = GPP,
  DNASquenceSet = DNASquenceSet,
  ChIPScore = eveLocusChip,
  chromatinState = Access,
  parameterOptions = OPP,
  parameter = "all",
  peakMethod="moving_kernel")
plotOptimalHeatMaps(optimalParam)
```

---

PositionFrequencyMatrix

*Accessor method for the PFM slot in a genomicProfiles object*

---

### Description

Accessor method for the PFM slot in a genomicProfiles object

### Usage

```
PositionFrequencyMatrix(object)
```

### Arguments

object            object is a [genomicProfiles](#) object

### Details

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

### Value

Returns the Position Frequency Matrix (PFM slot) used to compute the [PositionWeightMatrix](#) in a [genomicProfiles](#) 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 genomicProfiles object
GPP<-genomicProfiles(PFM=PFM,PFMFormat="raw")
# Accessing Slot
PositionFrequencyMatrix(GPP)
```

---

PositionFrequencyMatrix-methods

*~~ Methods for Function PositionFrequencyMatrix ~~*

---

## Description

Accessor method for the PFM slot in a `genomicProfiles` object

### Methods:

`PositionFrequencyMatrix(object)`

---

PositionFrequencyMatrix<-

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

---

## Description

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

## Usage

`PositionFrequencyMatrix(object)<- value`

## Arguments

<code>object</code>	object is a <a href="#">genomicProfiles</a> object
<code>value</code>	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 [genomicProfiles](#). If after building a [genomicProfiles](#), 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 [genomicProfiles](#) object has already been created and only the PFM is supplied/updated , then the Positon Weight Matrix will automatically updated as well.

**Value**

Returns a [genomicProfiles](#) 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 genomicProfiles object
GPP<-genomicProfiles()
#Setting PFM
PositionFrequencyMatrix(GPP) <- PFM
```

---

PositionFrequencyMatrix<-methods

~~ *Methods for Function* PositionFrequencyMatrix<- ~~

---

**Description**

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

**Methods:**

PositionFrequencyMatrix(object)<-"path/to/file/"

PositionFrequencyMatrix(object)<-value

---

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

---

### Description

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

### Usage

```
PositionWeightMatrix(object)
```

### Arguments

object            object is a [genomicProfiles](#)

### Details

After creating a [genomicProfiles](#) 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"), "BCDSLx.pfm")
#Building genomicProfiles object
GPP<-genomicProfiles(PFM=PFM,PFMFormat="raw")
# Accessing Slot
PositionWeightMatrix(GPP)
```

---

PositionWeightMatrix-methods

*~~ Methods for Function PositionWeightMatrix ~~*

---

### Description

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

### Methods:

PositionWeightMatrix(object)

---

PositionWeightMatrix<-

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

---

### Description

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

### Usage

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

### Arguments

object	object is a <a href="#">genomicProfiles</a> 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 [genomicProfiles](#) object has already been created. In that case, we advise to first create a [genomicProfiles](#) 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 [genomicProfiles](#) 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 genomicProfiles object
GPP <- genomicProfiles()
#Setting PWM to PositionWeightMatrix slot
PWM <- matrix(runif(32,-10,20), ncol=8)
rownames(PWM) <- c("A","C","T","G")
PositionWeightMatrix(GPP) <- PWM
```

---

PositionWeightMatrix<-methods

~~ *Methods for Function* PositionWeightMatrix<- ~~

---

**Description**

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

**Methods:**

PositionWeightMatrix(object)<-value

---

processingChIP

*Pre-processing ChIP-seq data*

---

**Description**

processingChIP will process and extract ChIP scores at a set of loci of interest.

**Usage**

```
processingChIP(profile, loci=NULL, reduce=NULL,
               peaks=NULL, chromatinState=NULL, parameterOptions=NULL,
               cores=1)
```

**Arguments**

profile	profile is a path to a UCSC format file, a GRanges or data frame. The input data should contain 4 columns:chromosome, start , end and score.
loci	loci is <a href="#">GRanges</a> describing the loci at which ChIP scores should be extracted. If NULL, a set of Loci will extracted from profile. The data provided will then be split into bins of width equal to lociWidth (Default 20kbp) Default=NULL
reduce	reduce is a the top regions to select based on the mean ChIP score. If peaks are provided, regions overlapplying with known peaks will be selected based on highest ChIP score. If NULL, all regions will be considered. Default=NULL
parameterOptions	parameterOptions is an <a href="#">parameterOptions</a> object containing chip Parameters to be parsed for ChIP score extraction. If NULL, <a href="#">parameterOptions</a> will be built internally with default ChIP extraction parameters (see <a href="#">chipSmooth</a> , <a href="#">chipSd</a> and <a href="#">chipMean</a> ) Default=NULL
peaks	peaks is a path to UCSC format file or a GRanges object containing location of ChIP peaks. Default=NULL
chromatinState	chromatinState is a <a href="#">GRanges</a> containing Accessible DNA or chromatin States If provided, regions will be selected only if they contain accesible DNA. Default=NULL
cores	cores is the number of cores used to extract ChIP scores. Default = 1

**Details**

When using [computeOptimal](#), it is required to supply real ChIP data in order to have a point of comparison. The corralation and MSE Scores are computed based of how well the model fits biological data. processingChIP will extract this data from ChIP data at loci of interest. When using the reduce option, this function will only select the top regions based on peak height or mean ChIP score. processingChIP will also extract maxSignal and backgroundSignal from ChIP data and parse it to an [parameterOptions](#) object.

**Value**

Returns a ChIPScore object containing extracted (and normalised) ChIP scores, the loci of interest and newly extracted Parameters(e.g. maxSignal)

**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)

## Extracting ChIP scores at loci of interest

ChIP<-processingChIP(profile=eveLocusChip, loci=eveLocus)
```

---

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(genomicProfiles,ChIPScore,
  parameterOptions=NULL,method="all",cores=1)
```

**Arguments**

genomicProfiles	genomicProfiles is the result of <a href="#">computeChIPProfile</a>
ChIPScore	ChIPScore is the result of processingChIP. Extracted/Normalised experimental ChIP scores.
parameterOptions	parameterOptions is a <a href="#">parameterOptions</a> object for parameter specification.
method	method is the method that will be used to assess model quality against ChIP-seq data. Method can be one of the following: pearson, spearman, kendall, ks, geometric,fscore, MSE,or all.Fscore contains f-score, precision,recall, MCC, Accuracy and AUC ROC.
cores	cores is the number of cores used to extract ChIP scores. Default = 1

**Details**

In order to assess the quality of the model against experimental ChIP-seq data, ChIPAnalyser offers a wide range of method to choose from. These methods are also used when computing optimal parameters.

**Value**

Returns list of goodness of fit metrics for each loci and each parameter selected.

**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)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
# Building genomicProfiles object
GPP <- genomicProfiles(PFM=PFM, BPFrequency=DNASequenceSet)

# Computing Genome Wide
GenomeWide <- computeGenomeWideScore(genomicProfiles = GPP,
  DNASequenceSet = DNASequenceSet)

#Compute PWM Scores
PWMScores <- computePWMScore(genomicProfiles = GenomeWide,
  DNASequenceSet = DNASequenceSet, loci = eveLocus, chromatinState = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(genomicProfiles = PWMScores)

#Compute ChIP profiles
chipProfile <- computeChIPProfile(genomicProfiles=Occupancy,loci=eveLocus)
#Estimating accuracy estimate
AccuracyEstimate <- profileAccuracyEstimate(genomicProfiles = chipProfile,
  ChIPScore = eveLocusChip,
  occupancyProfileParameters = OPP)
```

**Description**

Accessor method for profiles in a [genomicProfiles](#) object

**Methods:**

profiles(object) Computed PWM scores, Occupancy or ChIP-seq like profiles for loci of interest and parameter combination of interest.

---

PWMpseudocount	<i>Accessor Method for a PWMpseudocount slot in a <a href="#">parameterOptions</a></i>
----------------	--

---

**Description**

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

**Usage**

PWMpseudocount(object)

**Arguments**

object            object is a [parameterOptions](#) 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 [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(PWMpseudocount=0)
#Accessing slot value
PWMpseudocount(GPP)
```

---

PWMpseudocount-methods

~~ *Methods for Function PWMpseudocount* ~~

---

**Description**

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

**Methods:**

PWMpseudocount(object)

---

PWMpseudocount<-      *Setter Method for the pseudocount slot in a [parameterOptions](#) object*

---

**Description**

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

**Usage**

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

**Arguments**

object	object is a <a href="#">parameterOptions</a> 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 parameterOptions 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)

#Building data objects
GPP <- parameterOptions( PWMpseudocount=0)
#Setting Value for new PWMpseudocount
PWMpseudocount(GPP) <- 1
```

---

PWMpseudocount<-methods

*~~ Methods for Function PWMpseudocount<- ~~*

---

**Description**

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

**Methods:**

PWMpseudocount(object)<-value

---

PWMThreshold

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

---

**Description**

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

**Usage**

PWMThreshold(object)

**Arguments**

object                    object is a [parameterOptions](#) 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 [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(PWMThreshold=0.7)
#Accessing Value for PWMThreshold
PWMThreshold(GPP)
```

---

PWMThreshold-methods    *~~ Methods for Function PWMThreshold ~~*

---

**Description**

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

**Methods:**

PWMThreshold(object)

---

PWMThreshold<-            *Setter Method for the PWMThreshold slot in a [parameterOptions](#) object*

---

## Description

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

## Usage

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

## Arguments

object	object is a <a href="#">parameterOptions</a> object
value	value is a numeric value (between 0 and 1) to be assigned to the PWMThreshold slot in <a href="#">parameterOptions</a> 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 [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(PWMThreshold=0.7)
#Setting Value for new PWMThreshold
PWMThreshold(GPP) <- 0.8
```

---

PWMThreshold<-methods    *~~ Methods for Function PWMThreshold<- ~~*

---

**Description**

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

**Methods:**

PWMThreshold(object)<-value

---

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

---

**Description**

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

**Usage**

removeBackground(object)

**Arguments**

object                object is a [parameterOptions](#) 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 [parameterOptions](#) 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 parameterOptions object
OPP <- parameterOptions()
#Accessing Value for removeBackground
removeBackground(OPP)
```

---

removeBackground-methods

~~ *Methods for Function removeBackground* ~~

---

**Description**

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

**Methods:**

removeBackground(object)

---

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

---

**Description**

Setter Method for the removeBackground slot in a parameterOptions object

**Usage**

removeBackground(object) <-value

**Arguments**

object	object is an <a href="#">parameterOptions</a> object
value	value is positive numerical value to be assigned to the removeBackground slot in a <a href="#">parameterOptions</a> 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 [parameterOptions](#) 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 parameterOptions object
OPP <- parameterOptions()
#Setting new Value for removeBackground
removeBackground(OPP) <- 0.1
```

---

```
removeBackground<-methods
```

```
~~ Methods for Function removeBackground<- ~~
```

---

**Description**

Setter Method for the removeBackground slot in a parameterOptions object

**Methods:**

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

---

```
scores
```

```
Accessor Method for the scores slot in a ChIPScore object
```

---

**Description**

Setter Method for the scores slot in a [ChIPScore](#) object

**Usage**

```
scores(object)
```

**Arguments**

object            object is [ChIPScore](#) object

**Details**

When using the [processingChIP](#), this functions will return a name list of normalised ChIP scores at loci of interest. This functions enables you to extract those scores from the [ChIPScore](#) object.

**Value**

Returns the value assigned to the scores slot in a [ChIPScore](#) 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)

eveLocusChip<-processingChIP(eveLocusChip,eveLocus)
str(scores(eveLocusChip))
```

---

scores-methods            *~~ Methods for Function scores ~~*

---

**Description**

Accessor method for scores slot in a [ChIPScore](#) object.

**Methods:**

scores(object) Extracted and normalised ChIP scores at loci of interest.

---

searchSites            *Searching function for Sites above threshold and predicted ChIP-seq Profiles*

---

**Description**

searchSites is function enabling quick extraction and search for parameter combinations and/or loci in any [genomicProfiles](#) object from computeOccupancy onwards.

**Usage**

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



**Arguments**

Sites	Sites is either a <a href="#">genomicProfiles</a> or the result of <code>computeOptimal</code>
lambdaPWM	lambdaPWM is a numeric vector describing the <code>ScalingFactors</code> 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**

When testing numerous combinations of lambdaPWM and boundMolecules on top of many loci, it can become challenging to navigate the large data output 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. If a value for lambdaPWM is user provided then only this lambdaPWM will be selected (all boundMolecules and loci will also be selected). searchSites also works on the result of `computeOptimal`.

**Value**

Returns object of same time as parsed to this function with only the parameters and/or loci selected.

**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)){
  if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
  BiocManager::install("BSgenome.Dmelanogaster.UCSC.dm3")
}
library(BSgenome.Dmelanogaster.UCSC.dm3)
DNASequenceSet <- getSeq(BSgenome.Dmelanogaster.UCSC.dm3)
# Building genomicProfiles object
GPP <- genomicProfiles(PFM=PFM, BPFrequency=DNASequenceSet)
```

```

# Computing Genome Wide
GenomeWide <- computeGenomeWideScore(genomicProfiles = GPP,
  DNASequenceset = DNASequenceset)

#Compute PWM Scores
PWMScores <- computePWMScore(genomicProfiles = GenomeWide,
  DNASequenceset = DNASequenceset, loci = eveLocus, chromatinState = Access)
#Compute Occupancy
Occupancy <- computeOccupancy(genomicProfiles = PWMScores)
searchSites(Occupancy,ScalingFactor=c(1,4), BoundMolecules = c(1,100),
  Locus="eve")

#Compute ChIP profiles
chipProfile <- computeChIPProfile(genomicProfiles=Occupancy,loci=eveLocus)
searchSites(chipProfile,ScalingFactor=c(1,4), BoundMolecules = c(1,100),
  Locus="eve")

optimalParam <- computeOptimal(genomicProfiles = GPP,
  DNASequenceset = DNASequenceset,
  ChIPScore = eveLocusChip,
  chromatinState = Access,
  parameterOptions = OPP,
  parameter = "all",
  peakMethod="moving_kernel")

searchSites(optimalParam,ScalingFactor=c(1,4), BoundMolecules = c(1,100),
  Locus="eve")

```

---

show-methods

~~ *Methods for Function show* ~~

---

## Description

Show methods for various objects

## Methods:

signature(object = "ChIPScore")

signature(object = "genomicProfiles")

signature(object = "parameterOptions")

---

stepSize	<i>Accessor method of the stepSize slot in <a href="#">parameterOptions</a> object</i>
----------	--

---

### Description

Accessor method of the stepSize slot in [parameterOptions](#) object

### Usage

```
stepSize(object)
```

### Arguments

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

### 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 the value assigned to the stepSize slot in a [parameterOptions](#)

### 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 parameterOptions object
OPP <- parameterOptions()
#Setting new Value for maxSignal
stepSize(OPP)
```

---

```
stepSize-methods      ~~ Methods for Function stepSize ~~
```

---

**Description**

Accessor method of the stepSize slot in [parameterOptions](#) object

**Methods:**

```
stepSize(object)
```

---

```
stepSize<-          Setter Method for the stepSize slot in a parameterOptions
```

---

**Description**

Setter Method for the stepSize slot in a [parameterOptions](#)

**Usage**

```
stepSize(object) <- value
```

**Arguments**

`object` object is a [parameterOptions](#) object

`value` value is a positive numeric value that will be assigned to the stepSize slot in a [parameterOptions](#) object. Default is set at 10 base pairs.

**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 [parameterOptions](#) 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 parameterOptions object
OPP <- parameterOptions()
#Setting new Value for maxSignal
stepSize(OPP) <- 20
```

---

```
stepSize<-methods      ~~ Methods for Function stepSize<- ~~
```

---

**Description**

Setter Method for the stepSize slot in a [parameterOptions](#)

**Methods:**

```
stepSize(object)<-value
```

---

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

---

**Description**

Accessor Method for the strandRule slot in a [parameterOptions](#) object

**Usage**

```
strandRule(object)
```

**Arguments**

object                object is a [parameterOptions](#) 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 [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions( strandRule="max")
#Accesssing Value for strandRule
strandRule(GPP)
```

---

strandRule-methods     *~~ Methods for Function strandRule ~~*

---

**Description**

Accessor Method for the strandRule slot in a [parameterOptions](#) object

**Methods:**

strandRule(object)

---

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

---

**Description**

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

**Usage**

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

**Arguments**

object            object is a [parameterOptions](#) 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 [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions(strandRule="max")
#Setting New Value for strandRule
strandRule(GPP) <- "mean"
```

---

strandRule<-methods    *~~ Methods for Function strandRule<- ~~*

---

**Description**

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

**Methods:**

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

---

whichstrand	<i>Accessor method for the whichstrand slot in a <a href="#">parameterOptions</a> object</i>
-------------	--

---

### Description

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

### Usage

```
whichstrand(object)
```

### Arguments

object            object is a [parameterOptions](#) 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 [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions( whichstrand="+-")
#Setting New Value for whichstrand
whichstrand(GPP)
```



---

whichstrand-methods    *~~ Methods for Function whichstrand ~~*

---

**Description**

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

**Methods:**

whichstrand(object)

---

whichstrand<-            *Setter method for the whichstrand slot in a [parameterOptions](#) object*

---

**Description**

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

**Usage**

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

**Arguments**

object	object is a <a href="#">parameterOptions</a> 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 [parameterOptions](#) 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)

#Building data objects
GPP <- parameterOptions( whichstrand="+-")
#Setting New Value for whichstrand
whichstrand(GPP) <- "+"
```

---

whichstrand<-methods    *~~ Methods for Function whichstrand<- ~~*

---

**Description**

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

**Methods:**

whichstrand(object)<-value

# Index

## \* classes

- ChIPScore-class, 19
- genomicProfiles-class, 40
- genomicProfilesInternal-class, 42
- GRList-class, 44
- loci-class, 48
- nos-class, 64
- parameterOptions-class, 67

## \* methods

- averageExpPWMScore-methods, 7
- backgroundSignal-methods, 9
- backgroundSignal<-methods, 10
- boundMolecules-methods, 11
- boundMolecules<-methods, 13
- BPFrequency-methods, 14
- BPFrequency<-methods, 16
- chipMean-methods, 18
- chipMean<-methods, 19
- chipSd-methods, 22
- chipSd<-methods, 23
- chipSmooth-methods, 25
- chipSmooth<-methods, 26
- DNASequenceLength-methods, 37
- drop-methods, 38
- initialize-methods, 44
- lambdaPWM-methods, 46
- lambdaPWM<-methods, 47
- loci-methods, 49
- lociWidth-methods, 50
- lociWidth<-methods, 51
- maxPWMScore-methods, 53
- maxSignal-methods, 54
- maxSignal<-methods, 55
- minPWMScore-methods, 56
- naturalLog-methods, 58
- naturalLog<-methods, 59
- noiseFilter-methods, 60
- noiseFilter<-methods, 61
- noOfSites-methods, 63

- noOfSites<-methods, 64
- PFMFormat-methods, 71
- PFMFormat<-methods, 72
- ploidy-methods, 73
- ploidy<-methods, 74
- PositionFrequencyMatrix-methods, 80
- PositionFrequencyMatrix<-methods, 81
- PositionWeightMatrix-methods, 83
- PositionWeightMatrix<-methods, 84
- profiles-methods, 87
- PWMPseudocount-methods, 89
- PWMPseudocount<-methods, 90
- PWMThreshold-methods, 91
- PWMThreshold<-methods, 93
- removeBackground-methods, 94
- removeBackground<-methods, 95
- scores-methods, 96
- show-methods, 98
- stepSize-methods, 100
- stepSize<-methods, 101
- strandRule-methods, 102
- strandRule<-methods, 103
- whichstrand-methods, 105
- whichstrand<-methods, 106

## \* package

- ChIPAnalyser-package, 5
- .DNASequenceLength<- , genomicProfilesInternal , vector-method (genomicProfilesInternal-class), 42
- .ZeroBackground , parameterOptions-method (parameterOptions-class), 67
- .ZeroBackground<- , parameterOptions , vector-method (parameterOptions-class), 67
- .averageExpPWMScore<- , genomicProfilesInternal , numeric-method (genomicProfilesInternal-class), 42
- .drop<- , genomicProfilesInternal , vector-method

- (genomicProfilesInternal-class), 42
- .generatePWM, genomicProfilesInternal-method (genomicProfilesInternal-class), 42
- .maxPWMScore<-, genomicProfilesInternal, vector-method (genomicProfilesInternal-class), 42
- .minPWMScore<-, genomicProfilesInternal, vector-method (genomicProfilesInternal-class), 42
- .paramTag, parameterOptions-method (parameterOptions-class), 67
- .paramTag<-, parameterOptions, character-method (parameterOptions-class), 67
- .profiles<-, genomicProfilesInternal, GRList-method (genomicProfilesInternal-class), 42
- .tags, genomicProfilesInternal-method (genomicProfilesInternal-class), 42
- .tags<-, genomicProfilesInternal, character-method (genomicProfilesInternal-class), 42
- Access (ChIPAnalyserData), 16
- averageExpPWMScore, 6
- averageExpPWMScore, genomicProfilesInternal-method (genomicProfilesInternal-class), 42
- averageExpPWMScore-methods, 7
- backgroundSignal, 8, 65
- backgroundSignal, parameterOptions-method (parameterOptions-class), 67
- backgroundSignal-methods, 9
- backgroundSignal<-, 9
- backgroundSignal<-methods, 10
- backgroundSignal<-, parameterOptions, numeric-method (parameterOptions-class), 67
- boundMolecules, 10, 27, 32, 33, 65, 77, 78
- boundMolecules, parameterOptions-method (parameterOptions-class), 67
- boundMolecules-methods, 11
- boundMolecules<-, 12
- boundMolecules<-methods, 13
- boundMolecules<-, parameterOptions, vector-method (parameterOptions-class), 67
- BPFrequency, 13
- BPFrequency, genomicProfilesInternal-method (genomicProfilesInternal-class), 42
- BPFrequency-methods, 14
- BPFrequency<-, 14
- BPFrequency<-methods, 16
- BPFrequency<-, genomicProfilesInternal, DNAStrngSet-method (genomicProfilesInternal-class), 42
- BPFrequency<-, genomicProfilesInternal, vector-method (genomicProfilesInternal-class), 42
- ChIPAnalyser (ChIPAnalyser-package), 5
- ChIPAnalyser-package, 5
- ChIPAnalyserData, 16
- chipMean, 17, 17, 65, 85
- chipMean, parameterOptions-method (parameterOptions-class), 67
- chipMean-methods, 18
- chipMean<-, 18
- chipMean<-methods, 19
- chipMean<-, parameterOptions, numeric-method (parameterOptions-class), 67
- ChIPScore, 26, 27, 34, 47–51, 95, 96
- ChIPScore-class, 19
- chipSd, 21, 65, 85
- chipSd, parameterOptions-method (parameterOptions-class), 67
- chipSd-methods, 22
- chipSd<-, 22
- chipSd<-methods, 23
- chipSd<-, parameterOptions, numeric-method (parameterOptions-class), 67
- chipSmooth, 24, 65, 85
- chipSmooth, parameterOptions-method (parameterOptions-class), 67
- chipSmooth-methods, 25
- chipSmooth<-, 25
- chipSmooth<-methods, 26
- chipSmooth<-, parameterOptions, vector-method (parameterOptions-class), 67
- computeChIPProfile, 8, 9, 17, 18, 21, 23–25, 26, 86, 99, 100
- computeGenomeWideScores, 7, 28, 34, 52, 55
- computeOccupancy, 8, 9, 11, 12, 24–26, 30, 93, 94
- computeOptimal, 11, 12, 32, 85
- computePWMScore, 30, 34, 35

- DNASequencLength, [7](#), [36](#)  
 DNASequencLength, genomicProfilesInternal-method  
     (genomicProfilesInternal-class),  
     [42](#)  
 DNASequencLength-methods, [37](#)  
 DNAStrngSet, [13–15](#), [32](#), [34](#), [40](#)  
 drop, [37](#)  
 drop, genomicProfilesInternal-method  
     (genomicProfilesInternal-class),  
     [42](#)  
 drop-methods, [38](#)  
  
 eveLocus (ChIPAnalyserData), [16](#)  
 eveLocusChip (ChIPAnalyserData), [16](#)  
  
 geneRef (ChIPAnalyserData), [16](#)  
 genomicProfiles, [6](#), [7](#), [13–15](#), [26](#), [27](#), [30–32](#),  
     [34–37](#), [39](#), [39](#), [42](#), [44](#), [51](#), [52](#), [55](#), [56](#),  
     [66](#), [69–72](#), [79–84](#), [88](#), [96](#), [97](#)  
 genomicProfiles-class, [40](#)  
 genomicProfilesInternal, [41](#)  
 genomicProfilesInternal-class, [42](#)  
 GRanges, [16](#), [26](#), [31](#), [32](#), [34](#), [35](#), [37](#), [75](#), [85](#)  
 GRangesList, [31](#), [35](#)  
 GRList-class, [44](#), [44](#)  
  
 initialize, ChIPScore-method  
     (initialize-methods), [44](#)  
 initialize, genomicProfiles-method  
     (initialize-methods), [44](#)  
 initialize, parameterOptions-method  
     (initialize-methods), [44](#)  
 initialize-methods, [44](#)  
  
 lambdaPWM, [27](#), [32](#), [33](#), [45](#), [77](#), [78](#)  
 lambdaPWM, parameterOptions-method  
     (parameterOptions-class), [67](#)  
 lambdaPWM-methods, [46](#)  
 lambdaPWM<-, [46](#)  
 lambdaPWM<-methods, [47](#)  
 lambdaPWM<-, parameterOptions, vector-method  
     (parameterOptions-class), [67](#)  
 loci, [47](#)  
 loci, ChIPScore-method  
     (ChIPScore-class), [19](#)  
 loci-class, [48](#)  
 loci-methods, [49](#)  
 lociWidth, [49](#), [65](#)  
 lociWidth, parameterOptions-method  
     (parameterOptions-class), [67](#)  
  
 lociWidth-methods, [50](#)  
 lociWidth<-, [50](#)  
 lociWidth<-methods, [51](#)  
 lociWidth<-, parameterOptions, numeric-method  
     (parameterOptions-class), [67](#)  
  
 maxPWMScore, [51](#)  
 maxPWMScore, genomicProfilesInternal-method  
     (genomicProfilesInternal-class),  
     [42](#)  
 maxPWMScore-methods, [53](#)  
 maxSignal, [8](#), [9](#), [53](#), [65](#)  
 maxSignal, parameterOptions-method  
     (parameterOptions-class), [67](#)  
 maxSignal-methods, [54](#)  
 maxSignal<-, [54](#)  
 maxSignal<-methods, [55](#)  
 maxSignal<-, parameterOptions, numeric-method  
     (parameterOptions-class), [67](#)  
 minPWMScore, [55](#)  
 minPWMScore, genomicProfilesInternal-method  
     (genomicProfilesInternal-class),  
     [42](#)  
 minPWMScore-methods, [56](#)  
  
 naturalLog, [57](#), [65](#)  
 naturalLog, parameterOptions-method  
     (parameterOptions-class), [67](#)  
 naturalLog-methods, [58](#)  
 naturalLog<-, [58](#)  
 naturalLog<-methods, [59](#)  
 naturalLog<-, parameterOptions, logical-method  
     (parameterOptions-class), [67](#)  
 noiseFilter, [59](#), [65](#)  
 noiseFilter, parameterOptions-method  
     (parameterOptions-class), [67](#)  
 noiseFilter-methods, [60](#)  
 noiseFilter<-, [60](#)  
 noiseFilter<-methods, [61](#)  
 noiseFilter<-, parameterOptions, character-method  
     (parameterOptions-class), [67](#)  
 noOfSites, [62](#), [65](#)  
 noOfSites, parameterOptions-method  
     (parameterOptions-class), [67](#)  
 noOfSites-methods, [63](#)  
 noOfSites<-, [63](#)  
 noOfSites<-methods, [64](#)  
 noOfSites<-, parameterOptions, character-method  
     (parameterOptions-class), [67](#)

- noOfSites<- ,parameterOptions,numeric-method  
(parameterOptions-class), 67
- nos-class, 64
- parameterOptions, 8–12, 17, 18, 20–25, 27,  
30, 32, 34, 39, 41, 42, 44–46, 49–51,  
53, 54, 57–63, 65, 72–74, 85, 86,  
88–94, 99–106
- parameterOptions-class, 67
- PFMFormat, 70
- PFMFormat,genomicProfilesInternal-method  
(genomicProfilesInternal-class),  
42
- PFMFormat-methods, 71
- PFMFormat<- , 71
- PFMFormat<-methods, 72
- PFMFormat<- ,genomicProfilesInternal,character-method  
(genomicProfilesInternal-class),  
42
- ploidy, 65, 72
- ploidy,parameterOptions-method  
(parameterOptions-class), 67
- ploidy-methods, 73
- ploidy<- , 73
- ploidy<-methods, 74
- ploidy<- ,parameterOptions,numeric-method  
(parameterOptions-class), 67
- plotOccupancyProfile, 8, 9, 75
- plotOptimalHeatMaps, 77
- PositionFrequencyMatrix, 70, 71, 79
- PositionFrequencyMatrix,genomicProfilesInternal-method  
(genomicProfilesInternal-class),  
42
- PositionFrequencyMatrix-methods, 80
- PositionFrequencyMatrix<- , 80
- PositionFrequencyMatrix<-methods, 81
- PositionFrequencyMatrix<- ,genomicProfilesInternal,character-method  
(genomicProfilesInternal-class),  
42
- PositionFrequencyMatrix<- ,genomicProfilesInternal,numeric-method  
(genomicProfilesInternal-class),  
42
- PositionWeightMatrix, 79, 82
- PositionWeightMatrix,genomicProfilesInternal-method  
(genomicProfilesInternal-class),  
42
- PositionWeightMatrix-methods, 83
- PositionWeightMatrix<- , 83
- PositionWeightMatrix<-methods, 84
- PositionWeightMatrix<- ,genomicProfilesInternal,matrix-method  
(genomicProfilesInternal-class),  
42
- processingChIP, 19, 21, 26, 33, 34, 47, 49,  
50, 65, 84, 95
- profileAccuracyEstimate, 8, 9, 75, 86
- profiles, 31, 35, 44
- profiles (profiles-methods), 87
- profiles,genomicProfilesInternal-method  
(genomicProfilesInternal-class),  
42
- profiles-methods, 87
- PWMpseudocount, 66, 88
- PWMpseudocount,parameterOptions-method  
(parameterOptions-class), 67
- PWMpseudocount-methods, 89
- PWMpseudocount<- , 89
- PWMpseudocount<-methods, 90
- PWMpseudocount<- ,parameterOptions,numeric-method  
(parameterOptions-class), 67
- PWMThreshold, 29, 34, 35, 66, 90
- PWMThreshold,parameterOptions-method  
(parameterOptions-class), 67
- PWMThreshold-methods, 91
- PWMThreshold<- , 92
- PWMThreshold<-methods, 93
- PWMThreshold<- ,parameterOptions,numeric-method  
(parameterOptions-class), 67
- removeBackground, 65, 93
- removeBackground,parameterOptions-method  
(parameterOptions-class), 67
- removeBackground-methods, 94
- removeBackground<- , 94
- removeBackground<-methods, 95
- removeBackground<- ,parameterOptions,vector-method  
(parameterOptions-class), 67
- scores, 95
- scores,ChIPScore-method  
(ChIPScore-class), 19
- scores-methods, 96
- searchSites, 75, 96
- showChIPScore-method (show-methods), 98
- show,genomicProfiles-method  
(show-methods), 98
- show,parameterOptions-method  
(show-methods), 98
- show-methods, 98

stepSize, [65](#), [99](#)  
stepSize,parameterOptions-method  
    (parameterOptions-class), [67](#)  
stepSize-methods, [100](#)  
stepSize<-, [100](#)  
stepSize<-methods, [101](#)  
stepSize<-,parameterOptions,numeric-method  
    (parameterOptions-class), [67](#)  
strandRule, [66](#), [101](#)  
strandRule,parameterOptions-method  
    (parameterOptions-class), [67](#)  
strandRule-methods, [102](#)  
strandRule<-, [102](#)  
strandRule<-methods, [103](#)  
strandRule<-,parameterOptions,character-method  
    (parameterOptions-class), [67](#)

whichstrand, [66](#), [101](#), [103](#), [104](#)  
whichstrand,parameterOptions-method  
    (parameterOptions-class), [67](#)  
whichstrand-methods, [105](#)  
whichstrand<-, [105](#)  
whichstrand<-methods, [106](#)  
whichstrand<-,parameterOptions,character-method  
    (parameterOptions-class), [67](#)