

# ChIPseqR

October 25, 2011

---

BindScore-class      *Class "BindScore"*

---

## Description

This class provides the infrastructure to store results of ChIP-seq analysis.

## Usage

```
## S4 method for signature 'BindScore'
binding(x)
## S4 method for signature 'BindScore'
chrLength(x, subset)
## S4 method for signature 'BindScore'
cutoff(x, type=c("score", "pvalue"))
## S4 replacement method for signature 'BindScore'
cutoff(x, type=c("score", "pvalue")) <- value
## S4 method for signature 'BindScore'
head(x, n=6, by=c("score", "position"), ...)
## S4 method for signature 'BindScore'
lapply(X, FUN, ...)
## S4 method for signature 'BindScore'
length(x)
## S4 replacement method for signature 'BindScore'
length(x) <- value
## S4 method for signature 'BindScore'
max(x, ..., na.rm=TRUE)
## S4 method for signature 'BindScore'
min(x, ..., na.rm=TRUE)
## S4 method for signature 'BindScore'
names(x)
## S4 replacement method for signature 'BindScore,ANY'
names(x) <- value
## S4 method for signature 'BindScore'
nullDist(x)
## S4 replacement method for signature 'BindScore'
nullDist(x) <- value
## S4 method for signature 'BindScore'
```

```

peaks(x, ...)
## S4 method for signature 'BindScore'
range(x, ..., na.rm=TRUE)
## S4 method for signature 'BindScore'
score(x)
## S4 method for signature 'BindScore'
support(x)
## S4 method for signature 'BindScore'
tail(x, n=6, by=c("score", "position"), ...)
BindScore(call, score=list(), pvalue=list(), peaks=list(), cutoff=c(-Inf, 1), nu

```

### Arguments

x	Object of class BindScore.
X	Object of class BindScore.
subset	Index vector indicating a subset of x. If subset is missing everything is selected.
type	A string indicating which type of cut-off should be returned or changed. Either "score" or "pvalue"
n	Number of entries to show.
by	A string indicating whether the output should be sorted by score or by position in the genome.
na.rm	Logical indication whether NAs should be ignored.
FUN	Function to apply to results for each chromosome.
value	Replacement value.
call	Function call used to generate the values of the other slots.
score	List of binding site scores. One component per chromosome.
pvalue	List of binding site p-values. One component per chromosome.
peaks	List of significant peaks in binding site score. One component per chromosome.
cutoff	Numeric vector of length two indicating the significance cut-off in terms of score and p-value.
nullDist	Parameters of the null distribution.
names	Character vector providing names for chromosomes.
start	Integer indicating position of first binding site score.
compress	Logical indicating whether scores and p-values should be compressed.
digits	The number of decimal places to retain for compression.
...	Further arguments.

### Objects from the Class

Objects can be created by calls of the form `new("BindScore", functionCall, score, pvalue, peaks, cutoff, nullDist, names, ...)`. Objects of this class are typically created (and returned) by functions that perform peak calling on ChIP-seq data. Usually there should be no need to create them directly.

**Slots**

**functionCall**: Object of class "call" storing the function call used to initiate the analysis.

**score**: Object of class "list". The binding site score. One numeric vector per chromosome.

**pvalue**: Object of class "list". The (adjusted) p-values corresponding to the scores in slot **score**.

**peaks**: Object of class "list" giving the location of significant peaks in the binding site score. These correspond to the location of predicted binding sites.

**cutoff**: Object of class "numeric" with entries 'pvalue' and 'score' giving the significance threshold used for peak calling in terms of p-value and score.

**nullDist**: Object of class "numeric" providing the parameters of the null distribution used to determine p-values.

**start**: Object of class "integer" indicating the index corresponding to the first entry in **score** (assumed to be the same for all chromosomes).

**Methods**

**as.data.frame** signature(x = "BindScore"): Convert results into a data.frame giving the location, score and p-value of significant peaks.

[ signature(x = "BindScore", i = "ANY", j = "missing", drop = "missing"): Restrict results to a subset of chromosomes. Chromosomes can either be identified by name or by numerical index.

[[ signature(x = "BindScore", i = "ANY", j = "missing"): Restrict results to a single chromosome. Note that x[["chr1"]] is identical to x["chr1"].

[[ signature(x = "BindScore", i = "ANY", j = "numeric"): subset results to restrict them to a region on a single chromosome.

**binding** signature(x = "BindScore"): Returns length of binding site used during analysis.

**chrLength** signature(x = "BindScore", subset = "ANY"): Returns length of all chromosomes represented in x.

**cutoff<-** signature(x = "BindScore"): Sets the significance cut-off. Argument type=c("score", "pvalue") determines which cut-off is to be set, the other is adjusted accordingly. This recalculates the significance of peaks in the binding site score and may be slow.

**cutoff** signature(x = "BindScore"): Returns significance threshold used for analysis.

**head** signature(x = "BindScore"): Returns the first n peaks. Argument by = c("score", "position") determines whether results are sorted by score or by genomic location.

**lapply** signature(X = "BindScore"): Applies a function to results for each chromosome.

**length<-** signature(x = "BindScore"): Reduces the number of chromosomes for which results are stored, i.e., length(x) <- 3 only retains the first three chromosomes.

**length** signature(x = "BindScore"): Returns the number of binding sites identified by the analysis.

**max** signature(x = "BindScore"): Returns maximum score.

**min** signature(x = "BindScore"): Returns minimum score.

**names<-** signature(x = "BindScore", value = "ANY"): Sets the chromosome names.

**names** signature(x = "BindScore"): Returns the chromosome names.

**nullDist**<- signature(x = "BindScore"): Sets the parameters of the null distribution adjusting the significance cut-off in the process and predicts binding sites using the new null distribution.

**peaks** signature(x = "BindScore"): Returns list of predicted binding sites.

**range** signature(x = "BindScore"): Range of binding site scores.

**score** signature(x = "BindScore"): Returns list of binding site scores.

**support** signature(x = "BindScore"): Returns length of support region used during analysis.

**tail** signature(x = "BindScore"): Returns the last *n* peaks. Argument `by = c("score", "position")` determines whether results are sorted by score or by genomic location.

### Author(s)

Peter Humburg

### References

~put references to the literature/web site here ~

### See Also

[ReadCounts](#) for the data structure used as input for the analysis and [callBindingSites](#)

### Examples

```
showClass("BindScore")

set.seed(1)

## determine binding site locations
b <- sample(1:1e6, 5000)

## sample read locations
fwd <- unlist(lapply(b, function(x) sample((x-83):(x-73), 20, replace=TRUE)))
rev <- unlist(lapply(b, function(x) sample((x+73):(x+83), 20, replace=TRUE)))

## add some background noise
fwd <- c(fwd, sample(1:(1e6-25), 50000))
rev <- c(rev, sample(25:1e6, 50000))

## create data.frame with read positions as input to strandPileup
reads <- data.frame(chromosome="chr1", position=c(fwd, rev),
length=25, strand=factor(rep(c("+", "-"), times=c(150000, 150000))))

## create object of class ReadCounts
readPile <- strandPileup(reads, chrLen=1e6, extend=1, plot=FALSE)

## predict binding site locations
## the artificial dataset is very small so predictions may not be very reliable
bindScore <- simpleNucCall(readPile, bind=147, support=20, plot=FALSE, compress=FALSE)

## number of binding sites found
length(bindScore)
```

```
## the first few predictions, by score
head(bindScore)

## score and p-value cut-off used
cutoff(bindScore)
```

---

ChIPseqR-package    *Identifying Protein Binding Sites in High-Throughput Sequencing Data*

---

## Description

ChIPseqR provides a set of functions for the analysis of ChIP-seq data. Protein binding sites are located by identifying a characteristic pattern of peaks in read counts on both DNA strands.

## Details

Package:	ChIPseqR
Type:	Package
Version:	1.5.2
Date:	2011-03-24
License:	GPL (>=2)
LazyLoad:	yes

The easiest way to obtain binding site predictions for nucleosomes is to use [simpleNucCall](#). This provides a simple interface to [callBindingSites](#). This function operates on [AlignedRead](#) objects and provides useful defaults for nucleosome analysis. Parameters can be adjusted to detect the presence of other DNA binding proteins, e.g. transcription factors. If more fine control is desired the following steps will produce binding site predictions:

[strandPileup](#): Turn mapped reads into read counts along the genome.

[startScore](#): Score potential binding sites.

[getCutoff](#): Determine cutoff required to achieve desired false discovery rate.

[pickPeak](#): Find all peaks in the binding site score that exceed the significance threshold determined by [getCutoff](#). These are the predicted binding sites.

## Author(s)

Peter Humburg

Maintainer: Peter Humburg <Peter.Humburg@well.ox.ac.uk>

## References

Humburg, P., Helliwell, C., Bulger, D. & Stone, G. ChIPseqR: Analysis of ChIP-seq Experiments. BMC Bioinformatics 12, 39+ (2011).

## See Also

[ShortRead](#)

## Examples

```
## See 'simpleNucCall' for examples of how to obtain nucleosome predictions.
```

---

RLEBindScore-class *Run-length Encoded Binding Site Scores*

---

## Description

This class provides a memory efficient representation of binding site scores.

## Objects from the Class

Objects can be created by calls of the form `BindScore(functionCall, score, pvalue, peaks, cutoff, nullDist, names, start, digits, compress=TRUE)` or through calls to `callBindingSites`.

## Slots

**functionCall:** Object of class "call" storing the function call used to initiate the analysis.

**score:** Object of class "list". The binding site score. One run-length encoded numeric vector per chromosome.

**pvalue:** Object of class "list". The (adjusted and run-length encoded) p-values corresponding to the scores in slot `score`.

**peaks:** Object of class "list" giving the location of significant peaks in the binding site score. These correspond to the location of predicted binding sites.

**cutoff:** Object of class "numeric" with entries 'pvalue' and 'score' giving the significance threshold used for peak calling in terms of p-value and score.

**nullDist:** Object of class "numeric" providing the parameters of the null distribution used to determine p-values.

**start:** Object of class "integer" indicating the index corresponding to the first entry in `score` (assumed to be the same for all chromosomes).

## Extends

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

## Methods

**decompress** signature (`x = "RLEBindScore"`): conversion to [BindScore](#) object.

## Author(s)

Peter Humburg

## See Also

[BindScore](#), [Rle](#)

**Examples**

```

showClass("RLEBindScore")

set.seed(1)

## determine binding site locations
b <- sample(1:1e6, 5000)

## sample read locations
fwd <- unlist(lapply(b, function(x) sample((x-83):(x-73), 20, replace=TRUE)))
rev <- unlist(lapply(b, function(x) sample((x+73):(x+83), 20, replace=TRUE)))

## add some background noise
fwd <- c(fwd, sample(1:(1e6-25), 50000))
rev <- c(rev, sample(25:1e6, 50000))

## create data.frame with read positions as input to strandPileup
reads <- data.frame(chromosome="chr1", position=c(fwd, rev),
length=25, strand=factor(rep(c("+", "-"), times=c(150000, 150000))))

## create object of class ReadCounts
readPile <- strandPileup(reads, chrLen=1e6, extend=1, plot=FALSE)

## predict binding site locations
## the artificial dataset is very small so predictions may not be very reliable
bindScore <- simpleNucCall(readPile, bind=147, support=20, plot=FALSE, compress=TRUE)

## number of binding sites found
length(bindScore)

## the first few predictions, by score
head(bindScore)

## score and p-value cut-off used
cutoff(bindScore)

```

---

RLEReadCounts-class

*Run-length Encoded Read Counts*


---

**Description**

This class provides a memory efficient representation of strand specific read counts.

**Objects from the Class**

Objects can be created by calls of the form `ReadCounts(counts, names, compress = TRUE)` or by calls to `strandPileup`.

**Slots**

**counts:** Object of class "list" with one component per chromosome, containing a read counts encoded in an object of class `RleList`.

**Extends**

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

**Methods**

**chrLength** signature (x = "RLEReadCounts"): Returns length of all chromosomes represented in x.

**decompress** signature (x = "RLEReadCounts"): Expands read counts and returns object of class [ReadCounts](#).

**nreads** signature (x = "RLEReadCounts"): Returns the number of reads on each chromosome, split by strand (if `byStrand` is TRUE)

**plot** signature (x = "RLEReadCounts", y = "missing"): Generates plots of read counts.

**Author(s)**

Peter Humburg

**See Also**

[ReadCounts](#), [RleList](#)

**Examples**

```
showClass("RLEReadCounts")

## generate some very simple artificial read data
set.seed(1)
fwd <- sample(c(50:70, 250:270), 30, replace=TRUE)
rev <- sample(c(197:217, 347:417), 30, replace=TRUE)
## create data.frame with read positions as input to strandPileup
reads <- data.frame(chromosome="chr1", position=c(fwd, rev),
length=25, strand=factor(rep(c("+", "-"), times=c(30, 30))))

## create object of class ReadCounts
readPile <- strandPileup(reads, chrLen=500, extend=1, plot=FALSE, compress=TRUE)

names(readPile)
length(readPile)
sapply(readPile, sum)
```

---

ReadCounts-class    *Class "ReadCounts"*

---

**Description**

Represents counts of (possibly extended) reads for each strand of the genome.

**Usage**

```
ReadCounts(counts=list(), names=NULL, compress=TRUE)
```

**Arguments**

<code>counts</code>	A list of read counts. Each component is a two column matrix of strand specific read counts for a chromosome.
<code>names</code>	Character vector of chromosome names. If this is <code>NULL</code> the names of <code>counts</code> are used instead.
<code>compress</code>	Logical indicating whether read counts should be compressed.

**Objects from the Class**

Objects can be created by calls of the form `ReadCounts(counts, names, compress=FALSE)` or by calls to `strandPileup`.

**Slots**

`counts`: Object of class "list" with one component per chromosome, containing a matrix of read counts (one column per strand).

**Methods**

`[<- signature(x = "ReadCounts", i = "ANY", j = "missing")`: Replace read counts for chromosomes indicated by `i`.

`[ signature(x = "ReadCounts", i = "ANY", j = "missing", drop = "missing")`: Returns list of read counts for chromosomes indicated by `i`.

`[[<- signature(x = "ReadCounts", i = "ANY", j = "missing")`: Replace read counts for chromosome `i`.

`[[ signature(x = "ReadCounts", i = "ANY", j = "missing")`: Returns read counts for chromosome `i`.

`$<- signature(x = "ReadCounts")`: Replace read counts for chromosome `i` (by name).

`$ signature(x = "ReadCounts")`: Returns read counts for chromosome `i` (by name).

**callBindingSites** `signature(data = "ReadCounts")`: Predict bindingsites from read counts.

**chrLength** `signature(x = "ReadCounts", subset = "ANY")`: Returns length of all chromosomes represented in `x`.

**lapply** `signature(X = "ReadCounts")`: Apply function to read counts for each chromosome.

**length<-** `signature(x = "ReadCounts")`: Change the number of chromosomes represented by `x` to value.

**length** `signature(x = "ReadCounts")`: Number of chromosomes represented by `x`.

**names<-** `signature(x = "ReadCounts", value = "ANY")`: Change names of chromosomes.

**names** `signature(x = "ReadCounts")`: Chromosome names.

**nreads** `signature(x = "ReadCounts", byStrand = "Logical", subset = "ANY")`: Returns the number of reads on each chromosome, split by strand (if `byStrand` is `TRUE`).

**sapply** `signature(X = "ReadCounts")`: Apply function to read counts for each chromosome.

**Author(s)**

Peter Humburg

**See Also**

[BindScore](#), [strandPileup](#), [compress](#), [decompress](#)

**Examples**

```
showClass("ReadCounts")

## generate some very simple artificial read data
set.seed(1)
fwd <- sample(c(50:70, 250:270), 30, replace=TRUE)
rev <- sample(c(197:217, 347:417), 30, replace=TRUE)
## create data.frame with read positions as input to strandPileup
reads <- data.frame(chromosome="chr1", position=c(fwd, rev),
length=25, strand=factor(rep(c("+", "-"), times=c(30, 30))))

## create object of class ReadCounts
readPile <- strandPileup(reads, chrLen=500, extend=1, plot=FALSE, compress=FALSE)

names(readPile)
length(readPile)
sapply(readPile, sum)
```

---

accessors

*Access slots of S4 classes*

---

**Description**

Accessor functions for S4 classes in package "ChIPseqR".

**Usage**

```
## S4 method for signature 'ANY'
binding(x, ...)
## S4 method for signature 'ANY'
cutoff(x, ...)
## S4 replacement method for signature 'ANY'
cutoff(x, ...) <- value
## S4 method for signature 'ANY'
nullDist(x, ...)
## S4 replacement method for signature 'ANY'
nullDist(x, ...) <- value
## S4 method for signature 'ANY'
peaks(x, ...)
## S4 method for signature 'ANY'
pvalue(x, ...)
## S4 method for signature 'ANY'
support(x, ...)
```

**Arguments**

x	An S4 object.
...	Further arguments, ignored by default method.
value	New value for slot.

**Details**

These methods allow safe read (and in some cases write) access to slots of S4 classes and should be used for this purpose rather than modifying slots manually.

**Value**

The current value of the interrogated slot.

**Methods**

x = "ANY" Default method for accessor function.

**Note**

This page documents the generics and their default behaviour. See the help page of each class for class specific implementations.

**Author(s)**

Peter Humburg

**See Also**

[ReadCounts](#), [BindScore](#)

---

alignFeature      *Read counts relative to annotated features*

---

**Description**

Creates a set of (strand specific) read counts centred at the genomic features provided.

**Usage**

```
alignFeature(data, anno, offset = 1000)
```

**Arguments**

data	List with read counts as returned by <a href="#">strandPileup</a> .
anno	Data frame with annotation data in GFF format.
offset	Half width of window around start point of annotated features.

**Value**

List with one component for each feature in anno.

**Author(s)**

Peter Humburg

**References**The GFF file format specification: [http://www.sanger.ac.uk/Software/formats/GFF/GFF\\_Spec.shtml](http://www.sanger.ac.uk/Software/formats/GFF/GFF_Spec.shtml)**Examples**

```

set.seed(1)

## determine binding site locations
b <- sample(1:8.5e5, 500)

## sample read locations
fwd <- unlist(lapply(b, function(x) sample((x-83):(x-73), 20, replace=TRUE)))
rev <- unlist(lapply(b, function(x) sample((x+73):(x+83), 20, replace=TRUE)))

## add some background noise
fwd <- c(fwd, sample(1:(1e6-25), 5000))
rev <- c(rev, sample(25:1e6, 5000))

## create data.frame with read positions as input to strandPileup
reads <- data.frame(chromosome="chr1", position=c(fwd, rev),
length=25, strand=factor(rep(c("+", "-"), times=c(15000, 15000))))

## create object of class ReadCounts
readPile <- strandPileup(reads, chrLen=1e6, extend=1, plot=FALSE)

## convert binding site locations into GFF format
gff <- data.frame(chromosome="chr1", source="test", feature="binding", start=b-73, end=b+
score=".", strand=".", frame=".")

## align read counts relative to binding site location
aligned <- alignFeature(readPile, gff, offset=500)

```

---

callBindingSites-methods

*Predict protein binding sites from high-throughput sequencing data*


---

**Description**

Methods for function `callBindingSites` in Package ‘ChIPseqR’. These methods are used to identify protein binding sites from ChIP-seq data.

**Usage**

```

## S4 method for signature 'ANY'
callBindingSites(data, chrLen, plot=TRUE, verbose=TRUE, ..., plotTo)
## S4 method for signature 'character'
callBindingSites(data, type, minQual=70, ...)
## S4 method for signature 'matrix'
callBindingSites(data, chrName="chr", ...)

```

```
## S4 method for signature 'ReadCounts'
callBindingSites(data, bind, support, background, bgCutoff=0.9, supCutoff=0.9,
fdr = 0.05, extend=1, tailCut=0.95, piLambda=0.5, adapt=FALSE, corSummary=median,
digits = 16, plot=TRUE, verbose=TRUE, ask=FALSE, plotTo, ...)
```

### Arguments

data	Either an object containing information about mapped reads or a list. See below for details.
bind	Length of binding region to use (see Details).
support	Length of support region to use (see Details).
background	Length of background window. If this is missing it will be set to 10*(bind+2*support).
chrLen	Numeric vector indicating the length of all chromosomes. Only needed when data is an <a href="#">AlignedRead</a> object. <a href="#">readBfaToC</a> may be used to supply this information.
bgCutoff	Numeric value between 0.5 and 1. This determines how much estimates of the background read density are allowed to vary for adjacent windows. Set to 1 to disable cutoff.
supCutoff	Numeric value between 0.5 and 1. This determines how much estimates of the support region read density are allowed to vary for forward and reverse strand. Set to 1 to disable cutoff.
fdr	Target false discovery rate.
extend	Numeric value indicating how far mapped reads should be extended when calculating read counts.
type	Format of alignment file (see <a href="#">readAligned</a> for details).
minQual	Minimum alignment quality to use. All reads with lower alignment quality are discarded.
tailCut	Truncation point used to exclude outliers when estimating null distribution.
chrName	Name to use for the single chromosome.
piLambda	If <code>adapt=TRUE</code> this parameter is used to estimate the proportion of scores not related to binding sites.
adapt	Logical indicating whether an adaptive false discovery rate should be used. If this is <code>FALSE</code> (the default) the usual Benjamini-Hochberg procedure is used to control the FDR.
corSummary	Function used to summarise cross-correlation across chromosomes. See the Details section on binding and support region.
compress	Logical indicating whether the return value should be compressed.
digits	Number of decimal places to retain for binding site score for compression.
plot	Logical. If <code>plot=TRUE</code> (the default) some diagnostic plots are produced during the analysis.
verbose	Logical. If <code>verbose=TRUE</code> (the default) status messages are printed to indicate progress.
ask	Logical. Setting this to <code>TRUE</code> causes the system to wait for user input before displaying a new plot. See <a href="#">devAskNewPage</a> .
plotTo	Character string giving the name of a file that should be used to store plots generated during the analysis. If this is not missing a pdf file with the given name will be created.
...	Additional arguments. Most methods pass them on to the <code>ReadCounts</code> method.

## Details

The length of binding and support regions can either be given as a single value or as a range of possible values (by providing the minimum and maximum). In the latter case the cross-correlation between read counts on forward and reverse strand will be used to determine a value within that range. Note that this may lead sub-optimal choices of binding and support region length.

## Value

An object of class `BindScore` if `compress = FALSE`, otherwise an object of class `RLEBindScore`

## Methods

**data = "ANY"** Default method to handle all forms of input not explicitly handled by their own method. In particular this will be used for objects of class `AlignedRead` and `data.frame` but it will handle class for which a `strandPileup` method is available.

**data = "character"** Allows to use a file name referring to a file of mapped sequence reads as input.

**data = "matrix"** Uses a matrix of read counts (for a single chromosome) as input.

**data = "ReadCounts"** This methods implements the peak calling algorithm. Other methods will typically reformat their input and pass it on to this method.

## See Also

`simpleNucCall` for an interface with nucleosome specific defaults. This function uses `strandPileup`, `startScore`, `getCutoff` and `pickPeak`. See the help pages of these functions for additional detail on the individual steps involved. See `getBindLen` for details on the estimation of binding and support region length.

## Examples

```
set.seed(1)

## determine binding site locations
b <- sample(1:1e6, 5000)

## sample read locations
fwd <- unlist(lapply(b, function(x) sample((x-83):(x-73), 20, replace=TRUE)))
rev <- unlist(lapply(b, function(x) sample((x+73):(x+83), 20, replace=TRUE)))

## add some background noise
fwd <- c(fwd, sample(1:(1e6-25), 50000))
rev <- c(rev, sample(25:1e6, 50000))

## create data.frame with read positions as input to strandPileup
reads <- data.frame(chromosome="chr1", position=c(fwd, rev),
length=25, strand=factor(rep(c("+", "-"), times=c(150000, 150000))))

## create object of class ReadCounts
readPile <- strandPileup(reads, chrLen=1e6, extend=1, plot=FALSE)

## predict binding site locations
## the artificial dataset is very small so predictions may not be very reliable
bindScore <- callBindingSites(readPile, bind=147, support=20, background=2000, plot=FALSE)
```

---

compress-BindScore *Compress BindScore Objects*

---

## Description

Generates a compressed representation of binding site scores.

## Usage

```
## S4 method for signature 'BindScore'  
compress(x, digits=16)
```

## Arguments

<code>x</code>	An object of class <a href="#">BindScore</a> .
<code>digits</code>	Integer indicating the number of decimal places to retain.

## Details

Binding site scores are compressed by first rounding them to `digits` decimal places and then applying run-length encoding.

## Value

An object of class [RLEBindScore](#).

## Note

Compression reduces the precision of binding site scores and may affect results, especially for small values of `digits`.

## Author(s)

Peter Humburg

## See Also

[Rle](#), [RleList](#), [compress-BindScore](#)

---

`compress-ReadCounts`*Compress ReadCount Objects*

---

**Description**

Generates a compressed representation of read counts.

**Usage**

```
## S4 method for signature 'ReadCounts'  
compress(x)
```

**Arguments**

`x` An object of class `ReadCounts`

**Details**

Run-length encoding is used to obtain a compressed representation of read counts.

**Value**

An object of class `RLEReadCounts`

**Author(s)**

Peter Humburg

**See Also**

`Rle`, `RleList`, `compress-BindScore`

---

`compress-methods`*Methods for Function compress in Package 'ChIPseqR'*

---

**Description**

Methods to obtain compressed versions of data structures.

**Methods**

```
signature(x = "BindScore") Converts x into an object of class RLEBindScore.  
signature(x = "ReadCounts") Converts x into an object of class RLEReadCounts.  
signature(x = "RLEBindScore") Returns (the already compressed) x.  
signature(x = "RLEReadCounts") Returns (the already compressed) x.
```

---

decompress-methods *Methods for Function decompress in Package 'ChIPseqR'*

---

### Description

Methods to extract compressed data structures.

### Methods

`signature(x = "ANY")` The default method simply returns `x`.  
`signature(x = "Rle")` Restores the atomic vector encoded in `x`.  
`signature(x = "RLEBindScore")` Returns an object of [BindScore](#).  
`signature(x = "RleList")` Extracts the components of `x` and restores them to atomic vectors.  
`signature(x = "RLEReadCounts")` Returns an object of [ReadCounts](#).

---

decompress *Extract Read Count and Binding Site Score Representations*

---

### Description

These methods extract read count and binding site scores from compressed representations.

### Usage

```
## S4 method for signature 'RLEReadCounts'  
decompress(x)  
## S4 method for signature 'RLEBindScore'  
decompress(x)
```

### Arguments

`x` An object of class [RLEBindScore](#) or [RLEReadCounts](#).

### Value

An object of class [BindScore](#) or [ReadCounts](#) respectively.

### Author(s)

Peter Humburg

### See Also

[compress](#)

---

exportBindSequence *Export sequence of predicted binding sites*

---

### Description

Extracts sequence of predicted binding sites from reference genome and exports them in FASTA format.

### Usage

```
exportBindSequence(prediction, reference, bind, overlap = FALSE, file = "")
```

### Arguments

prediction	Object of class <a href="#">BindScore</a> .
reference	Reference genome sequence (as <a href="#">XStringSet</a> object).
bind	Length of binding site to assume for sequence extraction. This may be missing in which case the value is derived from 'prediction'.
overlap	Logical indicating whether overlapping predictions should be allowed.
file	Name of output file.

### Value

An [XStringViews](#) object containing the sequences. If a file name is provided this is returned invisibly.

### Author(s)

Peter Humburg

### References

Package Biostrings

### See Also

[XStringViews](#), [XStringSet](#), [BindScore](#)

---

getBindCor	<i>Calculate cross-correlation between read counts</i>
------------	--

---

**Description**

This function calculates the cross-correlation between read counts on forward and reverse strand.

**Usage**

```
getBindCor(data, max.lag, summary, plot = TRUE, ...)
```

**Arguments**

<code>data</code>	An object of class <a href="#">ReadCounts</a>
<code>max.lag</code>	Maximum lag to use in cross-correlation calculation.
<code>summary</code>	Function to use to summarise cross-correlation across chromosomes.
<code>plot</code>	Logical indicating whether to plot cross-correlation.
<code>...</code>	Further arguments, currently ignored.

**Details**

Function [fttcor](#) in package “timsac” is used to carry out the computation.

**Value**

The (summarised) cross-correlation. If `summary` is missing a list of cross-correlations for each chromosome is returned.

**Author(s)**

Peter Humburg

**See Also**

[fttcor](#), [ReadCounts](#), [getBindLen](#)

---

getBindLen	<i>Estimate length of binding and support region</i>
------------	--

---

**Description**

The cross-correlation between forward and reverse strand read counts is used to estimate the distance between peaks on both strands. This is then used to derive suitable values for the length of binding and support regions.

**Usage**

```
getBindLen(data, bind, support, summary = median, verbose = FALSE, plot = TRUE,
```

**Arguments**

data	An object of class <code>ReadCounts</code> .
bind	Either known length of binding region or minimum and maximum of binding region length to consider.
support	Either known length of support region or minimum and maximum of support region length to consider.
summary	Function to use to summarise cross-correlation across chromosomes.
verbose	Logical indicating whether progress messages should be printed.
plot	Logical indicating whether cross-correlation should be plotted.
...	Further arguments to <code>getBindCor</code> .

**Details**

This assumes that the first peak in cross-correlation corresponds to the length of the binding site. Note that this is not accurate. The peak is closer to  $\text{bind} + 2 * m$  where  $m$  is the median of the read distribution in the support region ('read distribution in the support region' means the read density as a function of distance to binding site start/end). Consequently this method will overestimate the length of the binding site. If either bind or support are of length 1 this is assumed to be the known value and a more accurate estimate for the remaining parameter is used.

**Value**

A numeric vector giving the estimated lengths of binding and support regions.

**Author(s)**

Peter Humburg

---

getCutoff

*Determine significance threshold for binding site scores*

---

**Description**

Given a vector of observed binding site scores and a desired false discovery rate, this function returns the lowest score that should be considered significant to achieve the given false discovery rate.

**Usage**

```
getCutoff(score, alpha = 0.05, tailCut = 0.95, adapt = FALSE, lambda, plot = TRUE)
```

**Arguments**

score	Numeric vector with binding site scores.
alpha	Desired false discovery rate.
tailCut	Truncation point used to exclude outliers when fitting the null distribution.
adapt	Logical indicating whether an adaptive false discovery rate should be used.

lambda	If <code>adapt</code> is TRUE this is used in estimating the proportion of scores that is unrelated to binding sites.
plot	If this is TRUE (the default) a plot of the observed score distribution and estimated null distribution is generated.
returnPval	Indicates whether or not the corrected p-values for all scores should be returned.

**Value**

A list with components

cutoff	A numeric vector with the score and nominal false discovery rate corresponding to the determined cutoff.
h0	A numeric vector with the mean and standard deviation of the estimated null distribution.
pvalue	If <code>returnPval</code> is TRUE, the p-values corresponding to the scores in <code>score</code> . Note that all missing values are removed.
pi0	If <code>adapt</code> is TRUE, the estimated proportion of scores not related to binding sites.

**Note**

This function is used by [callBindingSites](#) to determine the significance threshold for peak-calling.

**Author(s)**

Peter Humburg

**References**

For the adaptive false discovery rate procedure used if `adapt=TRUE` see JD Storey, JE Taylor and D Siegmund. Strong control, conservative point estimation and simultaneous conservative consistency of false discovery rates: a unified approach. *Journal of the Royal Statistical Society B*, 66(1):187-205, 2004.

**See Also**

[callBindingSites](#)

---

pickPeak

*Identify peaks above a given threshold*

---

**Description**

Given a vector of scores and a threshold, this function finds all peaks that exceed the threshold.

**Usage**

```
pickPeak(score, threshold, offset = 0, sub = FALSE)
```

**Arguments**

score	Numeric vector.
threshold	All values in <code>score</code> below this value are ignored.
offset	Offset to add to the determined peak locations.
sub	Logical. If this is <code>FALSE</code> (the default) for each region that exceeds the threshold only the global maximum is returned. Otherwise local maxima are returned as well.

**Value**

	If <code>sub = FALSE</code> a numeric vector giving the location of all peaks. Otherwise a list with components
peaks	The same peak locations that are returned for <code>sub = FALSE</code> .
subPeaks	A list with one component for each entry in 'peaks' giving the location of local maxima.

**Note**

This function is used by `callBindingSites` for peak-calling.

**Author(s)**

Peter Humburg

**See Also**

`callBindingSites`, `startScore`, `getCutoff`

---

plot-BindScore      *Diagnostic Plots for Binding Site Scores*

---

**Description**

Generates plots to assess the fit of the estimated null distribution.

**Usage**

```
## S4 method for signature 'BindScore,missing'
plot(x, npoints = 10000, type=c("density", "qqplot"), ...)
```

**Arguments**

x	An object of class <code>BindScore</code> .
npoints	Maximum number of points to plot in a QQ-plot.
type	Character string indicating the plot type.
...	Further arguments (currently ignored).

**Details**

Type 'density' produces a histogram of binding site scores with overlaid null distribution. Type 'qqplot' produces a normal QQ-plot comparing the observed binding site scores to the null distribution.

**Value**

Called for its side effect.

**Author(s)**

Peter Humburg

---

plot,ReadCounts,missing-method

*Diagnostic Plots for Read Counts*

---

**Description**

Produces plots to assess the distribution of reads, either for an entire chromosome or within a (small) window.

**Usage**

```
## S4 method for signature 'ReadCounts,missing'
plot(x, chr, center, score, width=2000, type=c("hilbert", "window"), ...)
## S4 method for signature 'RLEReadCounts,missing'
plot(x, chr, center, score, width=2000, type=c("hilbert", "window"), ...)
```

**Arguments**

x	Object of class <a href="#">ReadCounts</a> or <a href="#">RLEReadCounts</a> .
chr	Index or name of chromosome for which read counts should be plotted.
center	For type 'window', the center coordinate of the window to plot.
score	For type 'window', an object of type <a href="#">BindScore</a> (or <a href="#">BindScore</a> ) that should be used to include information about the score and predicted binding sites into the plot.
width	For type 'window', the width of the window.
type	Character string indicating the type of plot that should be produced (see details).
...	Further arguments (see details).

**Details**

Type 'window' produces a plot that shows read counts as vertical bars. If `score` is not missing, it is used to plot the score and predicted binding sites (if any) as well. Any additional arguments are passed on to [.plotWindow](#).

Type 'hilbert' produces a Hilbert curve plot of read counts for an entire chromosome. Additional arguments are passed on to [.plotReads](#).

**Value**

Called for its side effect.

**Author(s)**

Peter Humburg

**See Also**

[hilbertImage](#)

---

pos2gff

*Convert genome coordinates into GFF format*

---

**Description**

Provides facility to export the location of genomic features to a GFF formatted file.

**Usage**

```
pos2gff(pos, method, feature, len, strand, score, name)
```

**Arguments**

pos	Named list with one component per chromosome giving the start position of features on that chromosome.
method	Entry for method field in GFF file. Recycled as necessary
feature	Entry for feature field in GFF file. Recycled as necessary
len	Length of fetures. This is used to calculate matching end positions for each start position given in pos
strand	Entry for feature field in GFF file. Recycled as necessary
score	Entry for feature field in GFF file. Recycled as necessary
name	Entry for feature field in GFF file. Recycled as necessary

**Value**

A `data.frame` with columns 'chromosome', 'method', 'feature', 'start', 'end', 'score', 'strand'. Writing this data frame to a text file produces a GFF formatted file.

**Author(s)**

Peter Humburg

**References**

The GFF specification: [http://www.sanger.ac.uk/Software/formats/GFF/GFF\\_Spec.shtml](http://www.sanger.ac.uk/Software/formats/GFF/GFF_Spec.shtml)

**Examples**

```
pos <- list(chr1=c(10, 50, 60), chr2=c(22, 200, 500))
pos2gff(pos, "test", "foo", 25, c("+", "+", "-", "+", "-", "-"), 0, "test")
```

---

`simpleNucCall`*Predict nucleosome positions from high-throughput sequencing data*

---

### Description

This function provides a simplified interface to `callBindingSites` with defaults suitable for the detection of nucleosomes.

### Usage

```
simpleNucCall(data, bind=128, support=17, background=2000, chrLen, ...)
```

### Arguments

<code>data</code>	Either an object of class <code>AlignedRead</code> or a list. See below for details.
<code>bind</code>	Length of binding region to use (see Details).
<code>support</code>	Length of support region to use (see Details).
<code>background</code>	Length of background window. If this is missing it will be set to $10 * (\text{bind} + 2 * \text{support})$ .
<code>chrLen</code>	Numeric vector indicating the length of all chromosomes. Only needed when <code>data</code> is an <code>AlignedRead</code> object. <code>readBfaToc</code> may be used to supply this information.
<code>...</code>	Further arguments to <code>callBindingSites</code>

### Value

A list with components

<code>binding</code>	A <code>data.frame</code> with columns 'chromosome', 'position', 'score' and 'pvalue' indicating the centre of predicted binding sites together with their score and associated p-value.
<code>score</code>	A list with all calculated scores. One numeric vector per chromosome.
<code>pval</code>	A list with all corrected p-values. One numeric vector per chromosome.

### Author(s)

Peter Humburg

### References

~put references to the literature/web site here ~

### See Also

`callBindingSites` for additional parameters.

**Examples**

```
## generate some simple artificial read data
set.seed(1)

## determine binding site locations
b <- sample(1:1e6, 5000)

## sample read locations
fwd <- unlist(lapply(b, function(x) sample((x-83):(x-73), 20, replace=TRUE)))
rev <- unlist(lapply(b, function(x) sample((x+73):(x+83), 20, replace=TRUE)))

## add some background noise
fwd <- c(fwd, sample(1:(1e6-25), 50000))
rev <- c(rev, sample(25:1e6, 50000))

## create data.frame with read positions as input to strandPileup
reads <- data.frame(chromosome="chr1", position=c(fwd, rev),
length=25, strand=factor(rep(c("+", "-"), times=c(150000, 150000))))

## create object of class ReadCounts
readPile <- strandPileup(reads, chrLen=1e6, extend=1, plot=FALSE)

## predict binding site locations
bindScore <- simpleNucCall(readPile, bind=147, support=20, plot=FALSE)
```

---

startScore	<i>Score potential protein binding sites</i>
------------	--

---

**Description**

For each position in the genome this function computes a score indicating the likelihood that a protein binding site starts at that position.

**Usage**

```
startScore(data, b, support, background, bgCutoff, supCutoff)
```

**Arguments**

data	A two column matrix with read counts. The two columns correspond to reads on the forward and reverse strand respectively.
b	Length of binding region.
support	Length of support region.
background	Length of background window.
bgCutoff	Cutoff for the change in read rates between adjacent windows (see Details).
supCutoff	Cutoff for the change in read rates between support regions on forward and reverse strand (see Details).

**Details**

Robust estimates of read rates in background windows and support regions are obtained by limiting the difference between related estimates. Consider a forward support region of length 10 containing 20 reads. The maximum likelihood estimate for the rate parameter of the (assumed) underlying Poisson distribution is  $\hat{\lambda} = \frac{20}{10} = 2$ . If there are 50 reads in the reverse support region a robust estimate of the rate parameter is obtained as `max(50/10, qpois(supCutoff, lambda=lambda_hat))`

**Value**

Numeric vector with binding site scores.

**Note**

Instead of calling this function directly use [callBindingSites](#).

**Author(s)**

Peter Humburg

**See Also**

[callBindingSites](#)

---

strandPileup	<i>Strand specific read counts</i>
--------------	------------------------------------

---

**Description**

Given a set of aligned reads this function computes the number of reads starting at each position in the genome.

**Usage**

```
## S4 method for signature 'AlignedRead'
strandPileup(aligned, chrLen, extend, coords=c("leftmost", "fiveprime"),
compress = TRUE, plot = TRUE, ask = FALSE, ...)
## S4 method for signature 'data.frame'
strandPileup(aligned, chrLen, extend, coords=c("leftmost", "fiveprime"),
compress = TRUE, plot = TRUE, ask = FALSE, ...)
```

**Arguments**

<code>aligned</code>	An object containing information about aligned reads (see Details).
<code>chrLen</code>	A numeric vector giving the length of each chromosome.
<code>extend</code>	A numeric value indicating how far reads should be extended.
<code>coords</code>	A character value indicating the coordinate system to use. See <a href="#">coverage</a> for details.
<code>compress</code>	Logical indicating whether read counts should be compressed.
<code>plot</code>	If this is TRUE (the default) read coverage is plotted for all chromosomes.

ask Logical. Setting this to TRUE causes the system to wait for user input before displaying a new plot. See [devAskNewPage](#).

... Further arguments to [coverage](#).

### Details

The method for `data.frame` requires the column names to follow a strict naming scheme. Required columns are

**‘chromosome’** A factor with chromosome names.

**‘strand’** A factor with levels “-” and “+” indicating which strand the read mapped to.

**‘start’ or ‘position’** Start position of read on chromosome.

**‘end’ or ‘length’** End position of read or length of read respectively.

### Value

An object of class [ReadCounts](#).

### Author(s)

Peter Humburg

### See Also

[coverage](#), [AlignedRead](#), [callBindingSites](#)

### Examples

```
## generate some very simple artificial read data
set.seed(1)
fwd <- sample(c(50:70, 250:270), 30, replace=TRUE)
rev <- sample(c(197:217, 347:417), 30, replace=TRUE)
## create data.frame with read positions as input to strandPileup
reads <- data.frame(chromosome="chr1", position=c(fwd, rev),
length=25, strand=factor(rep(c("+", "-"), times=c(30, 30))))

readPile <- strandPileup(reads, chrLen=500, extend=1, plot=FALSE)
```

---

windowCounts

*Summarize read counts in a sliding window*

---

### Description

Read counts are summarized in a sliding window of variable size with variable overlap between windows.

### Usage

```
windowCounts(reads, window = 1000, shift = 500, method = sum)
```

**Arguments**

reads	Numeric vector of read counts.
window	Width of window.
shift	Distance between consecutive window start positions.
method	Function used to produce a summary for each window. It should accept a single numeric vector as argument.

**Value**

If `method` returns a single value a vector of all window summaries is returned, otherwise the return value is a list with one component for each window.

**Author(s)**

Peter Humburg

**Examples**

```
## generate some very simple artificial read data
set.seed(1)
fwd <- sample(c(50:70, 250:270), 30, replace=TRUE)
rev <- sample(c(197:217, 347:417), 30, replace=TRUE)
## create data.frame with read positions as input to strandPileup
reads <- data.frame(chromosome="chr1", position=c(fwd, rev),
length=25, strand=factor(rep(c("+", "-"), times=c(30, 30))))

## create object of class ReadCounts
readPile <- strandPileup(reads, chrLen=501, extend=1, plot=FALSE, compress=FALSE)

## get number of reads in sliding window
wdwCount <- windowCounts(apply(readPile[[1]], 1, sum), window=10, shift=5)
```

# Index

- \*Topic **classes**
  - BindScore-class, 1
  - ReadCounts-class, 8
  - RLEBindScore-class, 6
  - RLEReadCounts-class, 7
- \*Topic **hplot**
  - plot, ReadCounts, missing-method, 23
  - plot-BindScore, 22
- \*Topic **htest**
  - callBindingSites-methods, 12
  - getCutoff, 20
  - simpleNucCall, 25
- \*Topic **methods**
  - callBindingSites-methods, 12
  - compress-methods, 16
  - decompress-methods, 17
- \*Topic **misc**
  - alignFeature, 11
  - pickPeak, 21
  - windowCounts, 28
- \*Topic **models**
  - callBindingSites-methods, 12
  - simpleNucCall, 25
  - startScore, 26
- \*Topic **package**
  - ChIPseqR-package, 5
- \*Topic **utilities**
  - accessors, 10
  - compress-BindScore, 15
  - compress-ReadCounts, 16
  - decompress, 17
  - exportBindSequence, 18
  - getBindCor, 19
  - getBindLen, 19
  - pos2gff, 24
  - strandPileup, 27
- .plotReads, 23
- .plotWindow, 23
- [, BindScore, ANY, missing, missing-method (BindScore-class), 1
- [, ReadCounts, ANY, missing, missing-method (ReadCounts-class), 8
- [<-, ReadCounts, ANY, missing, ANY-method (ReadCounts-class), 8
- [<-, ReadCounts, ANY, missing-method (ReadCounts-class), 8
- [[, BindScore, ANY, missing-method (BindScore-class), 1
- [[, BindScore, ANY, numeric-method (BindScore-class), 1
- [[, ReadCounts, ANY, missing-method (ReadCounts-class), 8
- [[<-, ReadCounts, ANY, missing, ANY-method (ReadCounts-class), 8
- [[<-, ReadCounts, ANY, missing-method (ReadCounts-class), 8
- \$, ReadCounts-method (ReadCounts-class), 8
- \$<-, ReadCounts, ANY-method (ReadCounts-class), 8
- \$<-, ReadCounts-method (ReadCounts-class), 8
- accessors, 10
- AlignedRead, 5, 13, 14, 25, 28
- alignFeature, 11
- as.data.frame, BindScore-method (BindScore-class), 1
- binding (accessors), 10
- binding, ANY-method (accessors), 10
- binding, BindScore-method (BindScore-class), 1
- binding-methods (accessors), 10
- BindScore, 6, 10, 11, 14, 15, 17, 18, 22, 23
- BindScore (BindScore-class), 1
- BindScore-class, 1
- callBindingSites, 4–6, 21, 22, 25, 27, 28
- callBindingSites (callBindingSites-methods), 12
- callBindingSites, ANY-method (callBindingSites-methods), 12

- callBindingSites, character-method  
(*callBindingSites-methods*), 12
- callBindingSites, matrix-method  
(*callBindingSites-methods*), 12
- callBindingSites, ReadCounts-method  
(*callBindingSites-methods*), 12
- callBindingSites-methods, 12
- ChIPseqR (*ChIPseqR-package*), 5
- ChIPseqR-package, 5
- chrLength (*ReadCounts-class*), 8
- chrLength, BindScore-method  
(*BindScore-class*), 1
- chrLength, ReadCounts-method  
(*ReadCounts-class*), 8
- chrLength, RLEReadCounts-method  
(*RLEReadCounts-class*), 7
- compress, 10, 17
- compress (*compress-methods*), 16
- compress, BindScore-method  
(*compress-BindScore*), 15
- compress, ReadCounts-method  
(*compress-ReadCounts*), 16
- compress, RLEBindScore-method  
(*compress-methods*), 16
- compress, RLEReadCounts-method  
(*compress-methods*), 16
- compress-BindScore, 15, 16
- compress-BindScore, 15
- compress-methods, 16
- compress-ReadCounts, 16
- coverage, 27, 28
- cutoff (*acessors*), 10
- cutoff, ANY-method (*acessors*), 10
- cutoff, BindScore-method  
(*BindScore-class*), 1
- cutoff-methods (*acessors*), 10
- cutoff<- (*acessors*), 10
- cutoff<-, ANY-method (*acessors*), 10
- cutoff<-, BindScore-method  
(*BindScore-class*), 1
- cutoff<-methods (*acessors*), 10
  
- decompress, 10, 17
- decompress, ANY-method  
(*decompress-methods*), 17
- decompress, Rle-method  
(*decompress-methods*), 17
- decompress, RLEBindScore-method  
(*decompress*), 17
- decompress, RleList-method  
(*decompress-methods*), 17
- decompress, RLEReadCounts-method  
(*decompress*), 17
- decompress-methods, 17
- devAskNewPage, 13, 28
- exportBindSequence, 18
  
- fftcor, 19
- fttcor, 19
  
- getBindCor, 19, 20
- getBindLen, 14, 19, 19
- getCutoff, 5, 14, 20, 22
  
- head, BindScore-method  
(*BindScore-class*), 1
- hilbertImage, 24
  
- lapply, BindScore-method  
(*BindScore-class*), 1
- lapply, ReadCounts-method  
(*ReadCounts-class*), 8
- length, BindScore-method  
(*BindScore-class*), 1
- length, ReadCounts-method  
(*ReadCounts-class*), 8
- length<-, BindScore-method  
(*BindScore-class*), 1
- length<-, ReadCounts-method  
(*ReadCounts-class*), 8
  
- max, BindScore-method  
(*BindScore-class*), 1
- min, BindScore-method  
(*BindScore-class*), 1
  
- names, BindScore-method  
(*BindScore-class*), 1
- names, ReadCounts-method  
(*ReadCounts-class*), 8
- names<-, BindScore, ANY-method  
(*BindScore-class*), 1
- names<-, ReadCounts, ANY-method  
(*ReadCounts-class*), 8
- nreads (*ReadCounts-class*), 8
- nreads, ReadCounts-method  
(*ReadCounts-class*), 8
- nreads, RLEReadCounts-method  
(*RLEReadCounts-class*), 7
- nullDist (*acessors*), 10
- nullDist, ANY-method (*acessors*), 10

- nullDist, BindScore-method  
(*BindScore-class*), 1
  - nullDist-methods (*accessors*), 10
  - nullDist<- (*accessors*), 10
  - nullDist<- ,ANY-method (*accessors*),  
10
  - nullDist<- ,BindScore-method  
(*BindScore-class*), 1
  - nullDist<-methods (*accessors*), 10
- peaks (*accessors*), 10
- peaks, ANY-method (*accessors*), 10
- peaks, BindScore-method  
(*BindScore-class*), 1
- peaks-methods (*accessors*), 10
- pickPeak, 5, 14, 21
- plot, BindScore, missing-method  
(*plot-BindScore*), 22
- plot, ReadCounts, missing-method,  
23
- plot, RLEReadCounts, missing-method  
(*plot, ReadCounts, missing-method*),  
23
- plot-BindScore, 22
- pos2gff, 24
- pvalue (*accessors*), 10
- pvalue, ANY-method (*accessors*), 10
- pvalue-methods (*accessors*), 10
- range, BindScore-method  
(*BindScore-class*), 1
- readAligned, 13
- readBfaToc, 13, 25
- ReadCounts, 4, 8, 11, 16, 17, 19, 20, 23, 28
- ReadCounts (*ReadCounts-class*), 8
- ReadCounts-class, 8
- Rle, 6, 15, 16
- RLEBindScore, 14–17
- RLEBindScore-class, 6
- RleList, 7, 8, 15, 16
- RLEReadCounts, 16, 17, 23
- RLEReadCounts-class, 7
- sapply, ReadCounts-method  
(*ReadCounts-class*), 8
- score (*accessors*), 10
- score, BindScore-method  
(*BindScore-class*), 1
- ShortRead, 5
- simpleNucCall, 5, 14, 25
- startScore, 5, 14, 22, 26
- strandPileup, 5, 7, 9–11, 14, 27
- strandPileup, AlignedRead-method  
(*strandPileup*), 27
- strandPileup, data.frame-method  
(*strandPileup*), 27
- strandPileup-methods  
(*strandPileup*), 27
- support (*accessors*), 10
- support, ANY-method (*accessors*), 10
- support, BindScore-method  
(*BindScore-class*), 1
- support-methods (*accessors*), 10
- tail, BindScore-method  
(*BindScore-class*), 1
- windowCounts, 28
- XStringSet, 18
- XStringViews, 18