

# Package ‘PWMErich’

May 16, 2024

**Type** Package

**Title** PWM enrichment analysis

**Version** 4.40.0

**Date** 2015-09-25

**Author** Robert Stojnic, Diego Diez

**Maintainer** Diego Diez <diego10ruiz@gmail.com>

**LazyLoad** yes

**Description** A toolkit of high-level functions for DNA motif scanning and enrichment analysis built upon Biostrings. The main functionality is PWM enrichment analysis of already known PWMs (e.g. from databases such as MotifDb), but the package also implements high-level functions for PWM scanning and visualisation. The package does not perform “de novo” motif discovery, but is instead focused on using motifs that are either experimentally derived or computationally constructed by other tools.

**License** LGPL (>= 2)

**Depends** R (>= 3.5.0), methods, BiocGenerics, Biostrings

**Imports** grid, seqLogo, gdata, evd, S4Vectors

**Suggests** MotifDb, BSgenome, BSgenome.Dmelanogaster.UCSC.dm3, PWMErich.Dmelanogaster.background, testthat, gtools, parallel, PWMErich.Hsapiens.background, PWMErich.Mmusculus.background, BiocStyle, knitr

**biocViews** MotifAnnotation, SequenceMatching, Software

**VignetteBuilder** knitr

**RoxygenNote** 7.1.1

**Encoding** UTF-8

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

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

**git\_last\_commit** fc31a1e

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

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-15

## Contents

PWMEnrich-package . . . . .	4
.inputParamMotifs . . . . .	4
.inputParamSequences . . . . .	5
.inputPFMfromMatrixOrPWM . . . . .	5
.normalize.bg.seq . . . . .	6
.normargPfm . . . . .	6
.normargPriorParams . . . . .	6
affinitySequenceSet . . . . .	7
as.data.frame,MotifEnrichmentReport-method . . . . .	7
cloverPvalue1seq . . . . .	8
cloverScore . . . . .	9
colMedians . . . . .	9
colSds . . . . .	10
concatenateSequences . . . . .	10
cutoffZscore . . . . .	11
cutoffZscoreSequenceSet . . . . .	11
divideRows . . . . .	12
DNAStringSetToList . . . . .	12
empiricalPvalue . . . . .	12
empiricalPvalueSequenceSet . . . . .	13
getBackgroundFrequencies . . . . .	14
getPromoters . . . . .	15
gevPerSequence . . . . .	15
groupReport,MotifEnrichmentResults-method . . . . .	16
keepFinite . . . . .	17
logNormPval . . . . .	18
logNormPvalSequenceSet . . . . .	18
makeBackground . . . . .	19
makePriors . . . . .	20
makePWMCutoffBackground . . . . .	21
makePWMEmpiricalBackground . . . . .	22
makePWMGEVBackground . . . . .	23
makePWMLognBackground . . . . .	24
makePWMPvalCutoffBackground . . . . .	25
makePWMPvalCutoffBackgroundFromSeq . . . . .	26
makeStartEndPos . . . . .	27
matrixShuffleZscorePerSequence . . . . .	27
maxAligned . . . . .	28
motifDiffEnrichment . . . . .	29
motifEcdf . . . . .	30
motifEnrichment . . . . .	31

MotifEnrichmentReport-class . . . . .	35
MotifEnrichmentResults-class . . . . .	35
motifIC . . . . .	35
motifPrAUC . . . . .	36
motifRankingForGroup,MotifEnrichmentResults-method . . . . .	37
motifRankingForSequence,MotifEnrichmentResults-method . . . . .	38
motifRecoveryAUC . . . . .	39
motifScores . . . . .	40
motifScoresBigMemory . . . . .	41
motifSimilarity . . . . .	42
names,MotifEnrichmentReport . . . . .	43
names,MotifEnrichmentResults . . . . .	44
names,PWM . . . . .	44
names,PWMCutoffBackground . . . . .	45
names,PWMEmpiricalBackground . . . . .	45
names,PWMGEVBackground . . . . .	46
names,PWMLognBackground . . . . .	47
PFMtoPWM . . . . .	47
plot,MotifEnrichmentReport,missing-method . . . . .	48
plot,PWM,missing-method . . . . .	49
plotMotifScores . . . . .	50
plotMultipleMotifs . . . . .	52
plotPFM . . . . .	53
plotTopMotifsGroup,MotifEnrichmentResults-method . . . . .	53
plotTopMotifsSequence,MotifEnrichmentResults-method . . . . .	54
PWM-class . . . . .	55
PWMCutoffBackground-class . . . . .	55
PWMEmpiricalBackground-class . . . . .	56
PWMGEVBackground-class . . . . .	56
PWMLognBackground-class . . . . .	57
PWMUnscaled . . . . .	57
rankingProcessAndReturn . . . . .	58
readJASPAR . . . . .	59
readMotifs . . . . .	60
readTRANSFAC . . . . .	60
registerCoresPWMErich . . . . .	61
reverseComplement,PWM-method . . . . .	62
scanWithPWM . . . . .	62
seqLogoGrid . . . . .	64
sequenceReport,MotifEnrichmentResults-method . . . . .	65
show,MotifEnrichmentReport-method . . . . .	66
show,MotifEnrichmentResults-method . . . . .	66
show,PWM-method . . . . .	67
show,PWMCutoffBackground-method . . . . .	67
show,PWMEmpiricalBackground-method . . . . .	68
show,PWMGEVBackground-method . . . . .	68
show,PWMLognBackground-method . . . . .	69
toPWM . . . . .	69

tryAllMotifAlignments . . . . .	70
useBigMemoryPWMEnrich . . . . .	71
[,PWMCutoffBackground-method . . . . .	71
[,PWMEmpiricalBackground-method . . . . .	72
[,PWMGEVBackground-method . . . . .	72
[,PWMLognBackground-method . . . . .	73

<b>Index</b>	<b>74</b>
--------------	-----------

---

PWMEnrich-package	<i>PWMEnrich: PWM enrichment analysis</i>
-------------------	---

---

### Description

A toolkit of high-level functions for DNA motif scanning and enrichment analysis built upon Biostrings. The main functionality is PWM enrichment analysis of already known PWMs (e.g. from databases such as MotifDb), but the package also implements high-level functions for PWM scanning and visualisation. The package does not perform "de novo" motif discovery, but is instead focused on using motifs that are either experimentally derived or computationally constructed by other tools.

---

<i>.inputParamMotifs</i>	<i>Normalizes the motifs input argument for multiple functions</i>
--------------------------	--

---

### Description

Normalizes the motifs input argument for multiple functions

### Usage

```
.inputParamMotifs(motifs)
```

### Arguments

motifs	a list of motifs either as frequency matrices (PFM) or as PWM objects. If PFMs are specified they are converted to PWMs using uniform background.
--------	---

---

.inputParamSequences *Normalize the sequences input argument*

---

### **Description**

Normalize the sequences input argument

### **Usage**

.inputParamSequences(sequences)

### **Arguments**

sequences          a set of sequences to be scanned, a list of DNASTring or other scannable objects

---

.inputPFMfromMatrixOrPWM  
*Check the frequency matrix input parameter for motifSimilarity*

---

### **Description**

Check the frequency matrix input parameter for motifSimilarity

### **Usage**

.inputPFMfromMatrixOrPWM(m)

### **Arguments**

m                    either a PWM object or a matrix

### **Value**

corresponding PFM

---

`.normalize.bg.seq`      *check consistency of bg.seq input parameter*

---

**Description**

check consistency of bg.seq input parameter

**Usage**

`.normalize.bg.seq(bg.seq)`

**Arguments**

`bg.seq`              a set of background sequences, either a list of DNAStrng object or DNAS-trngSet object

---

`.normargPfm`              *Input parameter normalization for PWMUnscaled*

---

**Description**

This function is from Biostrings package. A Position Frequency Matrix (PFM) is also represented as an ordinary matrix. Unlike a PWM, it must be of type integer (it will typically be the result of `consensusMatrix()`).

**Usage**

`.normargPfm(x)`

**Arguments**

`x`                      a frequency matrix

---

`.normargPriorParams`      *Input parameter normalization function for PWMUnscaled*

---

**Description**

This function is from Biostrings package

**Usage**

`.normargPriorParams(prior.params)`

**Arguments**

`prior.params`      Typical 'prior.params' vector: `c(A=0.25, C=0.25, G=0.25, T=0.25)`

---

affinitySequenceSet    *Calculate total affinity over a set of sequences*

---

**Description**

Calculate total affinity over a set of sequences

**Usage**

```
affinitySequenceSet(scores, seq.len, pwm.len)
```

**Arguments**

scores	affinity scores for individual sequences
seq.len	lengths of sequences
pwm.len	lengths of PWMs

---

as.data.frame, MotifEnrichmentReport-method  
*Convert a MotifEnrichmentReport into a data.frame object*

---

**Description**

Convert a MotifEnrichmentReport into a data.frame object

**Usage**

```
## S4 method for signature 'MotifEnrichmentReport'  
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

**Arguments**

x	the MotifEnrichmentReport object
row.names	unused
optional	unused
...	unused

---

cloverPvalue1seq      *Calculate the Clover P-value as described in the Clover paper*

---

### Description

This function only take one background sequence as input, it also just calculates the P-value so it is more efficient.

### Usage

```
cloverPvalue1seq(  
  scores,  
  seq.len,  
  pwm.len,  
  bg.fwd,  
  bg.rev,  
  B = 1000,  
  verbose = TRUE,  
  clover = NULL  
)
```

### Arguments

scores	the affinity scores for individual sequences
seq.len	lengths of sequences
pwm.len	lengths of PWMs
bg.fwd	the raw score of forward strand
bg.rev	the raw scores of reverse strand
B	the number of random replicates
verbose	if to give verbose progress reports
clover	the clover scores if already calculated

### Value

P-value



---

cloverScore	<i>Calculate the Clover score using the recursive formula from Frith et al</i>
-------------	--

---

**Description**

Calculate the Clover score using the recursive formula from Frith et al

**Usage**

```
cloverScore(scores, lr3 = FALSE, verbose = FALSE)
```

**Arguments**

scores	a matrix of average odds scores, where columns are motifs, and rows sequences
lr3	if to return a matrix of LR3 scores, where columns correspond to motifs, and rows to subset sizes
verbose	if to produce verbose output of progress

**Value**

the LR4 score, which is the mean of LR3 scores over subset sizes

---

colMedians	<i>Calculate medians of columns</i>
------------	-------------------------------------

---

**Description**

Calculate medians of columns

**Usage**

```
colMedians(x)
```

**Arguments**

x	a matrix
---	----------

---

colSds	<i>Calculate standard deviations of columns</i>
--------	---

---

**Description**

Calculate standard deviations of columns

**Usage**

```
colSds(x)
```

**Arguments**

x                    a matrix

---

concatenateSequences	<i>Concatenata DNA sequences into a single character object</i>
----------------------	---

---

**Description**

Concatenata DNA sequences into a single character object

**Usage**

```
concatenateSequences(sequences)
```

**Arguments**

sequences            either a list of DNAStrng objects, or a DNAStrngSet

**Value**

a single character string

---

cutoffZscore	<i>Z-score calculation for cutoff hits</i>
--------------	--

---

**Description**

The Z-score is calculated separately for each sequence

**Usage**

```
cutoffZscore(scores, seq.len, pwm.len, bg.P)
```

**Arguments**

scores	the hit counts for the sequences
seq.len	the length distribution of sequences
pwm.len	the length distribution of the PWMs
bg.P	background probabilities of observing a motif hit at nucleotide resolution (scaled to sequence length, not 2 * length)

**Value**

Z-score

---

cutoffZscoreSequenceSet	<i>Z-score calculation for cutoff hits for group of sequences</i>
-------------------------	---

---

**Description**

The Z-score is calculated as if the sequence came for one very long sequence

**Usage**

```
cutoffZscoreSequenceSet(scores, seq.len, pwm.len, bg.P)
```

**Arguments**

scores	the hit counts for the sequences
seq.len	the length distribution of sequences
pwm.len	the length distribution of the PWMs
bg.P	background probabilities of observing a motif hit at nucleotide resolution

**Value**

Z-score

---

divideRows	<i>Divide each row of a matrix with a vector</i>
------------	--

---

**Description**

Divide each row of a matrix with a vector

**Usage**

```
divideRows(m, v)
```

**Arguments**

m	matrix to be divided
v	the vector to use for division

---

DNAStrngSetToList	<i>Convert DNAStrngSet to list of DNAStrng objects</i>
-------------------	--

---

**Description**

as.list doesn't seem to always work for DNAStrngSets, so implementing this ourselves.

**Usage**

```
DNAStrngSetToList(x)
```

**Arguments**

x	an object of class DNAStrngSet
---	--------------------------------

---

empiricalPvalue	<i>Calculate the empirical P-value by affinity of cutoff.</i>
-----------------	---

---

**Description**

This is the new backend function for empirical P-values for either affinity or cutoff. The function only works on single sequences.

**Usage**

```

empiricalPvalue(
  scores,
  seq.len,
  pwm.len,
  bg.fwd,
  bg.rev,
  cutoff = NULL,
  B = 10000,
  verbose = FALSE,
  exact.length = FALSE
)

```

**Arguments**

scores	the scores obtained for the sequence
seq.len	the length of the sequence, if a single value will take a single sequence of given length. If a vector of values, will take sequences of given lengths and joint them together
pwm.len	the lengths of PWMs
bg.fwd	raw odds scores for the forward strand of background
bg.rev	raw odds scores for the reverse strand of background
cutoff	if not NULL, will use hit count above this cutoff. The cutoff should be specified in log2.
B	the number of random replicates
verbose	if to give verbose progress reports
exact.length	if to take into consideration that the actual sequence lengths differ for different PWMs. For very long sequences (i.e. seq.len » pwm.len) this make very little difference, however the run time with exact.length is much longer.

---

empiricalPvalueSequenceSet

*Empirical P-value for a set of sequences*

---

**Description**

Calculate empirical P-value for a set of sequences, using either affinity or cutoff. When cutoff is used, the score is a number of motif hits above a certain log-odds cutoff.

**Usage**

```
empiricalPvalueSequenceSet(
  scores,
  seq.len,
  pwm.len,
  bg.fwd,
  bg.rev,
  cutoff = NULL,
  B = 10000,
  verbose = FALSE
)
```

**Arguments**

scores	a matrix of scores, rows for sequences, columns for PWMs
seq.len	the lengths of sequences
pwm.len	the lengths of PWMs
bg.fwd	raw odds scores for the forward strand of background
bg.rev	raw odds scores for the reverse strand of background
cutoff	if not NULL, will use hit count above this cutoff. The cutoff should be specified in log2.
B	the number of random replicates
verbose	if to give verbose progress reports

---

```
getBackgroundFrequencies
```

*Get the four nucleotides background frequencies*

---

**Description**

Estimate the background frequencies of A,C,G,T on a set of promoters from an organism

**Usage**

```
getBackgroundFrequencies(organism = "dm3", pseudo.count = 1, quick = FALSE)
```

**Arguments**

organism	either a name of the organisms for which the background should be compiled (supported names are "dm3", "mm9" and "hg19"), a BSgenome object, DNASTringSet, or list of DNASTring objects
pseudo.count	the number to which the frequencies sum up to, by default 1
quick	if to preform fitting on a reduced set of 100 promoters. This will not give as good results but is much quicker than fitting to all the promoters (~10k). Usage of this parameter is recommended only for testing and rough estimates.

**Author(s)**

Robert Stojnic, Diego Diez

**Examples**

```
## Not run:
  getBackgroundFrequencies("dm3")

## End(Not run)
```

---

getPromoters	<i>Get the promoter sequences either for a named organism such as "dm3" or a BSgenome object</i>
--------------	--

---

**Description**

Get the promoter sequences either for a named organism such as "dm3" or a BSgenome object

**Usage**

```
getPromoters(organismOrGenome)
```

**Arguments**

organismOrGenome  
either organism name, e.g. "dm3", or BSgenome object

**Value**

a list of: promoters - DNASTringSet of (unique) promoters; organism - name of species; version - genome version

---

gevPerSequence	<i>Apply GEV background normalization per every sequence</i>
----------------	--

---

**Description**

Apply GEV background normalization per every sequence

**Usage**

```
gevPerSequence(scores, seq.len, pwm.len, bg.loc, bg.scale, bg.shape)
```

**Arguments**

<code>scores</code>	affinity scores for the PWMs, can contain scores for more than one sequence (as rows), P-values are extracted separately
<code>seq.len</code>	the length distribution of the sequences
<code>pwm.len</code>	the lengths of PWMs
<code>bg.loc</code>	list of linear regression for location parameter
<code>bg.scale</code>	list of linear regression for scale parameter
<code>bg.shape</code>	list of linear regression for shape parameter

---

`groupReport, MotifEnrichmentResults-method`

*Generate a motif enrichment report for the whole group of sequences together*

---

**Description**

Generate a motif enrichment report for the whole group of sequences together

**Usage**

```
## S4 method for signature 'MotifEnrichmentResults'
groupReport(obj, top = 0.05, bg = TRUE, by.top.motifs = FALSE, ...)
```

**Arguments**

<code>obj</code>	a <code>MotifEnrichmentResults</code> object
<code>top</code>	what proportion of top motifs should be examined in each individual sequence (by default 0.05, i.e. 5%)
<code>bg</code>	if to use background corrected P-values to do the ranking (if available)
<code>by.top.motifs</code>	if to rank by the proportion of sequences where the motif is within 'top' percentage of motifs
<code>...</code>	unused

**Value**

a `MotifEnrichmentReport` object containing a table with the following columns:

- `'rank'` - The rank of the PWM's enrichment in the whole group of sequences together
- `'target'` - The name of the PWM's target gene, transcript or protein complex.
- `'id'` - The unique identifier of the PWM (if set during PWM creation).
- `'raw.score'` - The raw score before P-value calculation
- `'p.value'` - The P-value of motif enrichment (if available)
- `'top.motif.prop'` - The proportion (between 0 and 1) of sequences where the motif is within top proportion of enrichment motifs.



**Examples**

```
if(requireNamespace("PWMErich.Dmelanogaster.background")){
  ###
  # load the pre-compiled lognormal background
  data(PWMLogn.dm3.MotifDb.Dmel, package = "PWMErich.Dmelanogaster.background")

  # scan two sequences for motif enrichment
  sequences = list(DNAString("GAAGTATCAAGTGACCAGTAAGTCCCAGATGA"),
    DNAString("AGGTAGATAGAACAGTAGGCAATGAAGCCGATG"))

  res = motifEnrichment(sequences, PWMLogn.dm3.MotifDb.Dmel)

  # produce a report for all sequences taken together
  r.default = groupReport(res)

  # produce a report where the last column takes top 1% motifs
  r = groupReport(res, top=0.01)

  # view the results
  r

  # plot the top 10 most enriched motifs
  plot(r[1:10])
}
```

---

keepFinite

*Replace all infinite values by 0*

---

**Description**

Replace all infinite values by 0

**Usage**

keepFinite(x)

**Arguments**

x                    a vector of values

---

logNormPval	<i>Calculate the P-value from lognormal distribution with background of equal length</i>
-------------	--

---

**Description**

Calculate the P-value from lognormal distribution with background of equal length

**Usage**

```
logNormPval(scores, seq.len, pwm.len, bg.mean, bg.sd, bg.len, log = FALSE)
```

**Arguments**

scores	affinity scores for the PWMs, can contain scores for more than one sequence (as rows), P-values are extracted separately
seq.len	the length distribution of the sequences
pwm.len	the leggths of PWMs
bg.mean	the mean values from the background for PWMs
bg.sd	the sd values from the background
bg.len	the length distribution of the background (we currently support only constant length)
log	if to produce log p-values

---

logNormPvalSequenceSet	<i>Lognormal P-value for a set of sequences</i>
------------------------	---

---

**Description**

Lognormal P-value for a set of sequences

**Usage**

```
logNormPvalSequenceSet(scores, seq.len, pwm.len, bg.mean, bg.sd, bg.len)
```

**Arguments**

scores	a matrix of per-sequence affinity scores
seq.len	lengths of sequences
pwm.len	lengths of pwms
bg.mean	mean background at length of bg.len
bg.sd	standard deviation of background at length of bg.len
bg.len	the length for which mean and sd are calculated

**Value**

P-value

---

makeBackground	<i>Make a background for a set of position frequency matrices</i>
----------------	---

---

**Description**

This is a convenience front-end function to compile new backgrounds for a set of PFMs. Currently only supports *D. melanogaster*, but in the future should support other common organisms as well.

**Usage**

```
makeBackground(
  motifs,
  organism = "dm3",
  type = "logn",
  quick = FALSE,
  bg.seq = NULL,
  ...
)
```

**Arguments**

motifs	a list of position frequency matrices (4xL matrices)
organism	either a name of the organisms for which the background should be compiled (currently supported names are "dm3", "mm9" and "hg19"), or a BSgenome object (see BSgenome package).
type	the type of background to be compiled. Possible types are: <ul style="list-style-type: none"> <li>"logn" - estimate a lognormal background</li> <li>"cutoff" - estimate a Z-score background with fixed log-odds cutoff (in log2)</li> <li>"pval" - estimate a Z-score background with a fixed P-value cutoff. Note that this may require a lot of memory since the P-value of motif hits is first estimated from the empirical distribution.</li> <li>"empirical" - create an empirical P-value background. Note that this may require a lot of memory (up to 10GB in default "slow" mode (quick=FALSE) for 126 JASPAR motifs and 1000 <i>D. melanogaster</i> promoters).</li> <li>"GEV" - estimate a generalized extreme value (GEV) distribution background by fitting linear regression to distribution parameters in log space</li> </ul>
quick	if to preform fitting on a reduced set of 100 promoters. This will not give as good results but is much quicker than fitting to all the promoters (~10k). Usage of this parameter is recommended only for testing and rough estimates.
bg.seq	a set of background sequences to use. This parameter overrides the "organism" and "quick" parameters.
...	other named parameters that backend function makePWM***Background functions take.

**Author(s)**

Robert Stojnic, Diego Diez

**Examples**

```
# load in the two example de-novo motifs
motifs = readMotifs(system.file(package = "PWMErich", dir = "extdata", file = "example.transfac"),
  remove.acc = TRUE)

## Not run:
# construct lognormal background
bg.logn = makeBackground(motifs, organism="dm3", type="logn")

# alternatively, any BSgenome object can also be used
if(requireNamespace("BSgenome.Dmelanogaster.UCSC.dm3"))
  bg.logn = makeBackground(motifs, organism=Dmelanogaster, type="logn")

# construct a Z-score of hits with P-value background
bg.pval = makeBackground(motifs, organism="dm3", type="pval", p.value=1e-3)

# now we can use them to scan for enrichment in sequences (in this case there is a consensus
# Tin binding site).
motifEnrichment(DNAString("TGCATCAAGTGTAGTG"), bg.logn)
motifEnrichment(DNAString("TGCATCAAGTGTAGTG"), bg.pval)

## End(Not run)
```

---

makePriors

*Make priors from background sequences*

---

**Description**

These priors serve both as background nucleotide frequencies and pseudo-counts for PWMs.

**Usage**

```
makePriors(bg.seq, bg.pseudo.count)
```

**Arguments**

bg.seq            a set of background sequences  
bg.pseudo.count    the total pseudocount shared between nucleotides

**Examples**

```
# some example sequences
sequences = list(DNAString("AAAGAGAGTGACCGATGAC"), DNAString("ACGATGAGGATGAC"))
# make priors with pseudo-count of 1 shared between them
makePriors(sequences, 1)
```

---

```
makePWMCutoffBackground
    Make a cutoff background
```

---

**Description**

Make a background based on number of motifs hits above a certain threshold.

**Usage**

```
makePWMCutoffBackground(
  bg.seq,
  motifs,
  cutoff = log2(exp(4)),
  bg.pseudo.count = 1,
  bg.source = "",
  verbose = TRUE
)
```

**Arguments**

bg.seq	a set of background sequences, either a list of DNAString object or DNAS-tringSet object
motifs	a set of motifs, either a list of frequency matrices, or a list of PWM objects. If frequency matrices are given, the background distribution is fitted from bg.seq.
cutoff	the cutoff at which the background should be made, i.e. at which a motif hit is called significant
bg.pseudo.count	the pseudo count which is shared between nucleotides when frequency matrices are given
bg.source	a free-form textual description of how the background was generated
verbose	if to produce verbose output

**Examples**

```
## Not run:
if(requireNamespace("PWMErich.Dmelanogaster.background")){
  data(MotifDb.Dmel.PFM, package = "PWMErich.Dmelanogaster.background")

  # make background for MotifDb motifs using 2Kb promoters of all D. melanogaster transcripts
```

```

# using a cutoff of 5
if(requireNamespace("BSgenome.Dmelanogaster.UCSC.dm3"))
  makePWMCutoffBackground(Dmelanogaster$upstream2000, MotifDb.Dmel.PFM, cutoff=log2(exp(5)))
}

## End(Not run)

```

---

```
makePWMEmpiricalBackground
```

*Make an empirical P-value background*

---

### Description

Make a background appropriate for empirical P-value calculation. The provided set of background sequences is concatenated into a single long sequence which is then scanned with the motifs and raw scores are saved. This object can be very large.

### Usage

```

makePWMEmpiricalBackground(
  bg.seq,
  motifs,
  bg.pseudo.count = 1,
  bg.source = "",
  verbose = TRUE,
  ...
)

```

### Arguments

<code>bg.seq</code>	a set of background sequences, either a list of <code>DNAStrng</code> object or <code>DNAStringSet</code> object
<code>motifs</code>	a set of motifs, either a list of frequency matrices, or a list of PWM objects. If frequency matrices are given, the background distribution is fitted from <code>bg.seq</code> .
<code>bg.pseudo.count</code>	the pseudo count which is shared between nucleotides when frequency matrices are given
<code>bg.source</code>	a free-form textual description of how the background was generated
<code>verbose</code>	if to produce verbose output
<code>...</code>	currently unused (this is for convenience for <code>makeBackground</code> function)

### Details

For reliable P-value calculation the size of the background set needs to be at least  $\text{seq.len} / \text{min.P.value}$ . For instance, to get P-values at a resolution of 0.001 for a single sequence of 500bp, we would need a background of at least  $500/0.001 = 50\text{kb}$ . This ensures that we can make 1000 independent 500bp samples from this background to properly estimate the P-value. For a group of sequences, we would take `seq.len` to be the total length of all sequences in a group.

**Examples**

```
## Not run:
if(requireNamespace("PWMErich.Dmelanogaster.background")){
  data(MotifDb.Dmel.PFM, package = "PWMErich.Dmelanogaster.background")

  # make empirical background by saving raw scores for each bp in the sequence. This can be
  # very large in memory!
  if(requireNamespace("BSgenome.Dmelanogaster.UCSC.dm3"))
    makePWMEmpiricalBackground(Dmelanogaster$upstream2000[1:100], MotifDb.Dmel.PFM)
}

## End(Not run)
```

---

makePWMGEVBackground *Make a GEV background distribution*

---

**Description**

Construct a lognormal background distribution for a set of sequences. Sequences concatenated are binned in 'bg.len' chunks and lognormal distribution fitted to them.

**Usage**

```
makePWMGEVBackground(
  bg.seq,
  motifs,
  bg.pseudo.count = 1,
  bg.len = seq(200, 2000, 200),
  bg.source = "",
  verbose = TRUE,
  fit.log = TRUE
)
```

**Arguments**

bg.seq	a set of background sequences, either a list of DNASTring object or DNASTringSet object
motifs	a set of motifs, either a list of frequency matrices, or a list of PWM objects. If frequency matrices are given, the background distribution is fitted from bg.seq.
bg.pseudo.count	the pseudo count which is shared between nucleotides when frequency matrices are given
bg.len	the length range of background chunks
bg.source	a free-form textual description of how the background was generated
verbose	if to produce verbose output
fit.log	if to fit log odds (instead of odds)

**Examples**

```
## Not run:
if(requireNamespace("PWMLognBackground")){
  data(MotifDb.Dmel.PFM, package = "PWMLognBackground")

  # make background for MotifDb motifs using 2kb promoters of all D. melanogaster transcripts
  if(requireNamespace("BSgenome.Dmelanogaster.UCSC.dm3"))
    makePWMLognBackground(Dmelanogaster$upstream2000, MotifDb.Dmel.PFM)
}

## End(Not run)
```

---

makePWMLognBackground *Make a lognormal background distribution*

---

**Description**

Construct a lognormal background distribution for a set of sequences. Sequences concatenated are binned in 'bg.len' chunks and lognormal distribution fitted to them.

**Usage**

```
makePWMLognBackground(
  bg.seq,
  motifs,
  bg.pseudo.count = 1,
  bg.len = 250,
  bg.len.sizes = 2^(0:4),
  bg.source = "",
  verbose = TRUE,
  algorithm = "default"
)
```

**Arguments**

bg.seq	a set of background sequences, either a list of DNASTring object or DNASTringSet object
motifs	a set of motifs, either a list of frequency matrices, or a list of PWM objects. If frequency matrices are given, the background distribution is fitted from bg.seq.
bg.pseudo.count	the pseudo count which is shared between nucleotides when frequency matrices are given
bg.len	background sequences will be split into tiles of this length (default: 250bp)
bg.len.sizes	background tiles will be joined into bigger tiles containing this much smaller tiles. The default is 2^(0:4), which with bg.len translates into 250bp, 500bp, 1000bp, 1500bp, 2000bp, 4000bp. Note this is only used in the "human" algorithm.



bg.source        a free-form textual description of how the background was generated  
 verbose         if to produce verbose output  
 algorithm       type of algorithm to use, valid values are: "default" and "human".

### Examples

```
## Not run:
if(requireNamespace("PWMErich.Dmelanogaster.background")){
  data(MotifDb.Dmel.PFM, package = "PWMErich.Dmelanogaster.background")

  # make background for MotifDb motifs using 2kb promoters of all D. melanogaster transcripts
  if(requireNamespace("BSgenome.Dmelanogaster.UCSC.dm3"))
    makePWMLognBackground(Dmelanogaster$upstream2000, MotifDb.Dmel.PFM)
}

## End(Not run)
```

---

makePWMPvalCutoffBackground

*Construct a cutoff background from empirical background*

---

### Description

This function takes already calculated empirical background distribution and chooses cutoff for each motif based on P-value cutoff for individual sites.

### Usage

```
makePWMPvalCutoffBackground(bg.p, p.value = 0.001, bg.source = "")
```

### Arguments

bg.p            an object of class PWMEmpiricalBackground  
 p.value        the P-value used to find cutoffs for each of the motifs  
 bg.source      textual description of background source

### Value

an object of type PWMCutoffBackground

### Examples

```
## Not run:
if(requireNamespace("PWMErich.Dmelanogaster.background")){
  data(MotifDb.Dmel.PFM, package = "PWMErich.Dmelanogaster.background")

  # make empirical background - here we use only 100 sequences for illustrative purposes
  if(requireNamespace("BSgenome.Dmelanogaster.UCSC.dm3"))
```

```

bg.p = makePWMEmpiricalBackground(Dmelanogaster$upstream2000[1:100], MotifDb.Dmel.PFM)

# use the empirical background to pick a threshold and make cutoff background
makePWMPvalCutoffBackground(bg.p, 0.001)
}

## End(Not run)

```

---

```
makePWMPvalCutoffBackgroundFromSeq
```

*Construct a P-value cutoff background from a set of sequences*

---

### Description

This function creates a P-value cutoff background for motif enrichment.

### Usage

```

makePWMPvalCutoffBackgroundFromSeq(
  bg.seq,
  motifs,
  p.value = 0.001,
  bg.pseudo.count = 1,
  bg.source = "",
  verbose = TRUE
)

```

### Arguments

bg.seq	a set of background sequences, either a list of DNAStrng object or DNAS-trngSet object
motifs	a set of motifs, either a list of frequency matrices, or a list of PWM objects. If frequency matrices are given, the background distribution is fitted from bg.seq.
p.value	the P-value used to find cutoffs for each of the motifs
bg.pseudo.count	the pseudo count which is shared between nucleotides when frequency matrices are given
bg.source	textual description of background source
verbose	if to print verbose output

### Value

an object of type PWMCutoffBackground

**Examples**

```
## Not run:
if(requireNamespace("PWMErich.Dmelanogaster.background")){
  data(MotifDb.Dmel.PFM, package = "PWMErich.Dmelanogaster.background")

  # use the empirical background to pick a threshold and make cutoff background
  makePWMPvalCutoffBackground(Dmelanogaster$upstream2000, 0.001)
}

## End(Not run)
```

---

makeStartEndPos	<i>Divide total.len into fragments of length len by providing start,end positions</i>
-----------------	---

---

**Description**

Divide total.len into fragments of length len by providing start,end positions

**Usage**

```
makeStartEndPos(total.len, len)
```

**Arguments**

total.len	total available length to be subdivided
len	size of the individual chunk

**Value**

a data.frame containing paired up start,end positions

---

matrixShuffleZscorePerSequence	<i>Obtain z-score for motif column shuffling</i>
--------------------------------	--

---

**Description**

All PWMs are shuffled at the same time. This function would be too slow to produce empirical P-values, thus we return a z-score from a small number of shuffles.

**Usage**

```
matrixShuffleZscorePerSequence(scores, sequences, pwms, cutoff = NULL, B = 30)
```

**Arguments**

scores	a set of already calculated scores
sequences	either one sequence or a list/set of sequences (objects of type DNASTring or DNASTringSet)
pwms	a list of PWMs
cutoff	if NULL, will use affinity, otherwise will use number of hits over this log2 odds cutoff
B	number of replicates, i.e. PWM column shuffles

**Details**

The z-scores are calculated for each sequence individually.

---

maxAligned	<i>Returned the aligned motif parts</i>
------------	---

---

**Description**

This function takes the offset of first motif relative to second and chops off the end of both motifs that are not aligned. It returns a list containing only the columns that align.

**Usage**

```
maxAligned(m1, m2, offset)
```

**Arguments**

m1	frequency matrix of first motif
m2	frequency matrix of second motif
offset	a number of nucleotides by which the first motif is offsetted compared to the second

**Value**

a list of column-trimmed motifs m1, m2

---

motifDiffEnrichment     *Differential motif enrichment*

---

### Description

Test for differential enrichment between two groups of sequences

### Usage

```
motifDiffEnrichment(  
  sequences1,  
  sequences2,  
  pwms,  
  score = "autodetect",  
  bg = "autodetect",  
  cutoff = log2(exp(4)),  
  verbose = TRUE,  
  res1 = NULL,  
  res2 = NULL  
)
```

### Arguments

- |            |  |
|------------|--|
| sequences1 | First set of sequences. Can be either a single sequence (an object of class DNASTring), or a list of DNASTring objects, or a DNASTringSet object.  |
| sequences2 | Second set of sequences. Can be either a single sequence (an object of class DNASTring), or a list of DNASTring objects, or a DNASTringSet object.   |
| pwms       | this parameter can take multiple values depending on the scoring scheme and background correction used. When the method parameter is set to "autodetect", the following default algorithms are going to be used: <ul style="list-style-type: none"><li>• if pwms is a list containing either frequency matrices or a list of PWM objects then the "affinity" algorithm is selected. If frequency matrices are given, they are converted to PWMs using uniform background. For best performance, convert frequency matrices to PWMs before calling this function using realistic genomic background.</li><li>• Otherwise, appropriate scoring scheme and background correction are selected based on the class of the object (see below).</li></ul> |
| score      | this parameter determines which scoring scheme to use. Following scheme as available: <ul style="list-style-type: none"><li>• "autodetect" - default value. Scoring method is determined based on the type of pwms parameter.</li><li>• "affinity" - use threshold-free affinity scores without a background. The pwms parameter can either be a list of frequency matrices, PWM objects, or a PWMLognBackground object.</li></ul>   |

	<ul style="list-style-type: none"> <li>• "cutoff" - use number of motif hits above a score cutoff as a measure of enrichment. No background correction is performed. The pwms parameter can either be a list of frequency matrices, PWM objects, or a PWMCutofBackground object.</li> </ul>
bg	<p>this parameter determines which background correction to use, if any.</p> <ul style="list-style-type: none"> <li>• "autodetect" - default value. Background correction is determined based on the type of the pwms parameter.</li> <li>• "logn" - use a lognormal distribution background pre-computed for a set of PWMs. This requires pwms to be of class PWMLognBackground.</li> <li>• "z" - use a z-score for the number of significant motif hits compared to background number of hits. This requires pwms to be of class PWMCutofBackground.</li> <li>• "none" - no background correction</li> </ul>
cutoff	the score cutoff for a significant motif hit if scoring scheme "cutoff" is selected.
verbose	if to produce verbose output
res1	the output of motifEnrichment if already calculated for sequences1
res2	the output of motifEnrichment if already calculated for sequences2

### Details

This function calls motifEnrichment on two groups of sequences and calculates the difference statistics when possible.

### Examples

```
if(requireNamespace("PWMLogn.Dmelanogaster.background")){
  # load the background file for drosophila and lognormal correction
  data(PWMLogn.dm3.MotifDb.Dmel, package = "PWMLogn.Dmelanogaster.background")

  # get the differential enrichment
  diff = motifDiffEnrichment(DNAString("TGCATCAAGTGTGTAGTGTGAGATTAGT"),
    DNAString("TGAACGAGTAGGACGATGAGAGATTGATG"), PWMLogn.dm3.MotifDb.Dmel, verbose=FALSE)

  # motifs differentially enriched in the first sequence (with lognormal background correction)
  head(sort(diff$group.bg, decreasing=TRUE))

  # motifs differentially enriched in the second sequence (with lognormal background correction)
  head(sort(diff$group.bg))
}
```

---

motifEcdf

*Calculate the empirical distribution score distribution for a set of motifs*

---

### Description

Calculate the empirical distribution score distribution for a set of motifs

**Usage**

```
motifEcdf(
  motifs,
  organism = NULL,
  bg.seq = NULL,
  quick = FALSE,
  pseudo.count = 1
)
```

**Arguments**

motifs	a set of motifs, either a list of frequency matrices, or a list of PWM objects. If frequency matrices are given, the background distribution is fitted from bg.seq.
organism	either a name of the organisms for which the background should be compiled (supported names are "dm3", "mm9" and "hg19"), or a BSgenome object (see BSgenome package).
bg.seq	a set of background sequence (either this or organism needs to be specified!). Can be a DNASTring or DNASTringSet object.
quick	if to do the fitting only on a small subset of the data (only in combination with organism). Useful only for code testing!
pseudo.count	the pseudo count which is shared between nucleotides when frequency matrices are given

**Value**

a list of ecdf objects (see help page for ecdf for usage).

---

motifEnrichment	<i>Motif enrichment</i>
-----------------	-------------------------

---

**Description**

Calculate motif enrichment using one of available scoring algorithms and background corrections.

**Usage**

```
motifEnrichment(
  sequences,
  pwms,
  score = "autodetect",
  bg = "autodetect",
  cutoff = NULL,
  verbose = TRUE,
  motif.shuffles = 30,
  B = 1000,
  group.only = FALSE
)
```

**Arguments**

sequences	the sequences to be scanned for enrichment. Can be either a single sequence (an object of class DNASTring), or a list of DNASTring objects, or a DNASTringSet object.
pwms	<p>this parameter can take multiple values depending on the scoring scheme and background correction used. When the method parameter is set to "autodetect", the following default algorithms are going to be used:</p> <ul style="list-style-type: none"> <li>• if pwms is a list containing either frequency matrices or a list of PWM objects then the "affinity" algorithm is selected. If frequency matrices are given, they are converted to PWMs using uniform background. For best performance, convert frequency matrices to PWMs before calling this function using realistic genomic background.</li> <li>• Otherwise, appropriate scoring scheme and background correction are selected based on the class of the object (see below).</li> </ul>
score	<p>this parameter determines which scoring scheme to use. Following scheme as available:</p> <ul style="list-style-type: none"> <li>• "autodetect" - default value. Scoring method is determined based on the type of pwms parameter.</li> <li>• "affinity" - use threshold-free affinity score. The pwms parameter can either be a list of frequency matrices, PWM objects, or a PWMLognBackground object.</li> <li>• "cutoff" - use number of motif hits above a score cutoff. The pwms parameter can either be a list of frequency matrices, PWM objects, or a PWMCutoffBackground object.</li> <li>• "clover" - use the Clover algorithm (Frith et al, 2004). The Clover score of a single sequence is identical to the affinity score, while for a group of sequences is an average of products of affinities over all sequence subsets.</li> </ul>
bg	<p>this parameter determines how the raw score is compared to the background distribution.</p> <ul style="list-style-type: none"> <li>• "autodetect" - default value. Background correction is determined based on the type of the pwms parameter.</li> <li>• "logn" - use a lognormal distribution background pre-computed for a set of PWMs. This requires pwms to be of class PWMLognBackground.</li> <li>• "z" - use a z-score for the number of significant motif hits compared to background number of hits. This requires pwms to be of class PWMCutoffBackground.</li> <li>• "pval" - use empirical P-value based on a set of background sequences. This requires pwms to be of class PWMEmpiricalBackground. Note that PWMEmpiricalBackground objects tend to be very large so that the empirical P-value can be calculated in reasonable time.</li> <li>• "ms" - shuffle columns of motif matrices and use that as basis for P-value calculation. Note that since the sequences need to rescanned with all of the new shuffled motifs this can be very slow. Also, this also works only on *individual* sequences, not groups.</li> <li>• "none" - no background correction</li> </ul>
cutoff	the score cutoff for a significant motif hit if scoring scheme "cutoff" is selected.



verbose	if to print verbose output
motif.shuffles	number of times to shuffle motifs if using "ms" background correction
B	number of replicates when calculating empirical P-value
group.only	if to return statistics only for the group of sequences, not individual sequences. In the case of empirical background the P-values for individual sequences are not calculated (thus saving time), for other backgrounds they are calculated but not returned.

## Details

This function provides an interface to all algorithms available in PWMEnrich to find motif enrichment in a single or a group of sequences with/without background correction.

Since for all algorithms the first step involves calculating raw scores without background correction, the output always contains the scores without background correction together with (optional) background-corrected scores.

Unless otherwise specified the scores are returned both separately for each sequence (without/with background) and for the whole group of sequences (without/with background).

To use a background correction you need to supply a set of PWMs with precompiled background distribution parameters (see function [makeBackground](#)). When such an object is supplied as the `pwm` parameter, the scoring scheme and background correction are automatically determined.

There are additional packages with already pre-computed background (e.g. see package `PWMEnrich.Dmelanogaster.background`).

Please refer to (Stojnic & Adryan, 2012) for more details on the algorithms.

## Value

a `MotifEnrichmentResults` object containing a subset following elements:

- "score" - scoring scheme used
- "bg" - background correction used
- "params" - any additional parameters
- "sequences" - the set of sequences used
- "pwms" - the set of pwms used
- "sequence.nobg" - per-sequence scores without any background correction. For "affinity" and "clover" a matrix of mean affinity scores; for "cutoff" number of significant hits above a cutoff
- "sequence.bg" - per-sequence scores after background correction. For "logn" and "pval" the P-value (smaller is better); for "z" and "ms" background corrections the z-scores (bigger is better).
- "group.nobg" - aggregate scores for the whole group of sequences without background correction. For "affinity" and "clover" the mean affinity over all sequences in the set; for "cutoff" the total number of hits in all sequences.
- "group.bg" - aggregate scores for the whole group of sequences with background correction. For "logn" and "pval", the P-value for the whole group (smaller is better); for "z" and "ms" the z-score for the whole set (bigger is better).

- "sequence.norm" - (only for "logn") the length-normalized scores for each of the sequences. Currently only implemented for "logn", where it returns the values normalized from LogN(0,1) distribution
- "group.norm" - (only for "logn") similar to sequence.norm, but for the whole group of sequences

## References

- R. Stojnic & B. Adryan: Identification of functional DNA motifs using a binding affinity lognormal background distribution, submitted.
- MC Frith et al: Detection of functional DNA motifs via statistical over-representation, Nucleic Acid Research (2004).

## Examples

```

if(requireNamespace("PWMErich.Dmelanogaster.background")){
  ###
  # load the pre-compiled lognormal background
  data(PWMLogn.dm3.MotifDb.Dmel, package = "PWMErich.Dmelanogaster.background")

  # scan two sequences for motif enrichment
  sequences = list(DNAString("GAAGTATCAAGTGACCAGTAGATTGAAGTAGACCAGTC"),
    DNAString("AGGTAGATAGAACAGTAGGCAATGGGGGAAATTGAGAGTC"))
  res = motifEnrichment(sequences, PWMLogn.dm3.MotifDb.Dmel)

  # most enriched in both sequences (lognormal background P-value)
  head(motifRankingForGroup(res))

  # most enriched in both sequences (raw affinity, no background)
  head(motifRankingForGroup(res, bg=FALSE))

  # most enriched in the first sequence (lognormal background P-value)
  head(motifRankingForSequence(res, 1))

  # most enriched in the first sequence (raw affinity, no background)
  head(motifRankingForSequence(res, 1, bg=FALSE))

  ###
  # Load the pre-compiled background for hit-based motif counts with cutoff of P-value = 0.001
  data(PWMPvalueCutoff1e3.dm3.MotifDb.Dmel, package = "PWMErich.Dmelanogaster.background")

  res.count = motifEnrichment(sequences, PWMPvalueCutoff1e3.dm3.MotifDb.Dmel)

  # Enrichment in the whole group, z-score for the number of motif hits
  head(motifRankingForGroup(res))

  # First sequence, sorted by number of motif hits with P-value < 0.001
  head(motifRankingForSequence(res, 1, bg=FALSE))
}

```

---

 MotifEnrichmentReport-class

*A report class with formatted results of motif enrichment*


---

### Description

The columns stored in this object will depend on the type of the report (either for group of sequences, or individual sequences).

### Slots

**d:** a DataFrame object that contains the main tabular report data

**pwms:** a list of PWM objects corresponding to rows of d

---

MotifEnrichmentResults-class

*A wrapper class for results of motifEnrichment() that should make it easier to access the results.*


---

### Description

Note that this is only a wrapper around a list which is the return value in PWMEnrich 1.3 and as such it provides the same interface as a list (for backward compatibility), with some additional methods.

### Slots

**res:** a list of old results with elements such as: sequence.bg, sequence.nobg, group.bg, group.nobg

---

motifIC

*Information content for a PWM or PFM*


---

### Description

Information content for a PWM or PFM

### Usage

```
motifIC(
  motif,
  prior.params = c(A = 0.25, C = 0.25, G = 0.25, T = 0.25),
  bycol = FALSE
)
```

**Arguments**

motif	a matrix of frequencies, or a PWM object
prior.params	the prior parameters to use when a matrix is given (ignored if motif is already a PWM)
bycol	if to return values separately for each column

**Value**

information content in bits (i.e.  $\log_2$ )

**Examples**

```
if(requireNamespace("PWMErich.Dmelanogaster.background")){
  data(MotifDb.Dmel, package = "PWMErich.Dmelanogaster.background")
  data(MotifDb.Dmel.PFM, package = "PWMErich.Dmelanogaster.background")

  # the nucleotide distribution is taken from the PWM (in this case genomic background)
  motifIC(MotifDb.Dmel[["ttk"]])
  # information content with default uniform background because the input is a matrix,
  # not PWM object
  motifIC(MotifDb.Dmel.PFM[["ttk"]])
}
```

---

motifPrAUC	<i>Calculate PR-AUC for motifs ranked according to some scoring scheme</i>
------------	--

---

**Description**

Note that this function assumes that smaller values are better!

**Usage**

```
motifPrAUC(seq.res)
```

**Arguments**

seq.res	a matrix where each column represents a PWM and each row a result for a different sequence.
---------	---

---

motifRankingForGroup,MotifEnrichmentResults-method

*Get a ranking of motifs by their enrichment in the whole set of sequences*

---

## Description

Get a ranking of motifs by their enrichment in the whole set of sequences

## Usage

```
## S4 method for signature 'MotifEnrichmentResults'  
motifRankingForGroup(  
  obj,  
  bg = TRUE,  
  id = FALSE,  
  order = FALSE,  
  rank = FALSE,  
  unique = FALSE,  
  ...  
)
```

## Arguments

obj	a MotifEnrichmentResults object
bg	if to use background corrected P-values to do the ranking (if available)
id	if to show PWM IDs instead of target TF names
order	if to output the ordering of PWMs instead of actual P-values or raw values
rank	if the output should be rank of a PWM instead of actual P-values or raw values
unique	if TRUE, only the best rank is taken for each TF (only when id = FALSE, order = FALSE)
...	currently unused

## Value

a vector of P-values or raw enrichments sorted such that the first motif is most enriched

## Examples

```
if(requireNamespace("PWMEnrich.Dmelanogaster.background")){  
  ###  
  # load the pre-compiled lognormal background  
  data(PWMLogn.dm3.MotifDb.Dmel, package = "PWMEnrich.Dmelanogaster.background")  
  
  # scan two sequences for motif enrichment  
  sequences = list(DNAString("GAAGTATCAAGTGACCAGTAAGTCCCAGATGA"),
```

```

DNAStr("AGGTAGATAGAACAGTAGGCAATGAAGCCGATG")
res = motifEnrichment(sequences, PWMLogn.dm3.MotifDb.Dmel)

# most enriched in both sequences (sorted by lognormal background P-value)
head(motifRankingForGroup(res))

# Return a non-redundant set of TFs
head(motifRankingForGroup(res, unique=TRUE))

# sorted by raw affinity instead of P-value
head(motifRankingForGroup(res, bg=FALSE))

# show IDs instead of target TF names
head(motifRankingForGroup(res, id=TRUE))

# output the rank instead of P-value
head(motifRankingForGroup(res, rank=TRUE))
}

```

---

`motifRankingForSequence,MotifEnrichmentResults-method`

*Get a ranking of motifs by their enrichment in one specific sequence*

---

## Description

Get a ranking of motifs by their enrichment in one specific sequence

## Usage

```

## S4 method for signature 'MotifEnrichmentResults'
motifRankingForSequence(
  obj,
  seq.id,
  bg = TRUE,
  id = FALSE,
  order = FALSE,
  rank = FALSE,
  unique = FALSE,
  ...
)

```

## Arguments

<code>obj</code>	a <code>MotifEnrichmentResults</code> object
<code>seq.id</code>	either the sequence number or sequence name
<code>bg</code>	if to use background corrected P-values to do the ranking (if available)
<code>id</code>	if to show PWM IDs instead of target TF names

order	if to output the ordering of PWMs instead of actual P-values or raw values
rank	if the output should be rank of a PWM instead of actual P-values or raw values
unique	if TRUE, only the best rank is taken for each TF (only when id = FALSE, order = FALSE)
...	currently unused

**Value**

a vector of P-values or raw enrichments sorted such that the first motif is most enriched

**Examples**

```

if(requireNamespace("PWMLogn.Dmelanogaster.background")){
  ###
  # load the pre-compiled lognormal background
  data(PWMLogn.dm3.MotifDb.Dmel, package = "PWMLogn.Dmelanogaster.background")

  # scan two sequences for motif enrichment
  sequences = list(DNAString("GAAGTATCAAGTGACCAGTAAGTCCCAGATGA"),
    DNAString("AGGTAGATAGAACAGTAGGCAATGAAGCCGATG"))
  res = motifEnrichment(sequences, PWMLogn.dm3.MotifDb.Dmel)

  # most enriched in the second sequences (sorted by lognormal background P-value)
  head(motifRankingForSequence(res, 2))

  # return unique TFs enriched in sequence 2
  head(motifRankingForSequence(res, 2, unique=TRUE))

  # sorted by raw affinity instead of P-value
  head(motifRankingForSequence(res, 2, bg=FALSE))

  # show IDs instead of target TF names
  head(motifRankingForSequence(res, 2, id=TRUE))

  # output the rank instead of P-value
  head(motifRankingForSequence(res, 2, rank=TRUE))
}

```

---

motifRecoveryAUC	<i>Calculate Recovery-AUC for motifs ranked according to some scoring scheme</i>
------------------	--

---

**Description**

Note that this function assumes that smaller values are better!

**Usage**

```
motifRecoveryAUC(seq.res)
```

**Arguments**

seq.res            a matrix where each column represents a PWM and each row a result for a different sequence.

---

motifScores            *Motif affinity or number of hits over a threshold*

---

**Description**

Scan a number of sequences either to find overall affinity, or a number of hits over a score threshold.

**Usage**

```
motifScores(
  sequences,
  motifs,
  raw.scores = FALSE,
  verbose = TRUE,
  cutoff = NULL
)
```

**Arguments**

sequences            a set of sequences to be scanned, a list of DNASTring or other scannable objects

motifs                a list of motifs either as frequency matrices (PFM) or as PWM objects. If PFMs are specified they are converted to PWMs using uniform background.

raw.scores            if to return raw scores (odds) for each position in the sequence. Note that scores for forward and reverse strand are concatenated into a single long vector of scores (twice the length of the sequence)

verbose                if to print verbose output

cutoff                 if not NULL, will count number of matches with score above value specified (instead of returning the average affinity). Can either be one value, or a vector of values for each of the motifs.

**Value**

if raw.scores=FALSE, returns a matrix of mean scores (after cutoff if any), where columns are motifs. The returned values are either mean odd scores (not log-odd), or number of hits above a threshold; otherwise if raw.scores=TRUE, returns a list of raw score values (before cutoff)



**Examples**

```

if(requireNamespace("PWMEnrich.Dmelanogaster.background")){
  data(MotifDb.Dmel, package = "PWMEnrich.Dmelanogaster.background")

  # affinity scores
  affinity = motifScores(DNAString("CGTAGGATAAAGTAACTAGTTGATGATGAAAG"), MotifDb.Dmel)

  # motif hit count with Patser score of 4
  counts = motifScores(DNAString("CGTAGGATAAAGTAACTAGTTGATGATGAAAG"), MotifDb.Dmel,
    cutoff=log2(exp(4)))

  print(affinity)
  print(counts)

  # scanning multiple sequences
  sequences = list(DNAString("CGTAGGATAAAGTAACTAGTTGATGATGAAAG"),
    DNAString("TGAGACGAAGGGGATGAGATGCCGAAGAGTGAAA"))
  affinity2 = motifScores(sequences, MotifDb.Dmel)
  print(affinity2)
}

```

---

`motifScoresBigMemory` *This is a memory intensive version of `motifScore()` which is about 2 times faster*

---

**Description**

The parameters and functionality are the same as [motifScores](#). Please refer to documentation of this function for detailed explanation of functionality.

**Usage**

```

motifScoresBigMemory(
  sequences,
  motifs,
  raw.scores = FALSE,
  verbose = TRUE,
  cutoff = NULL,
  seq.all = NULL
)

```

**Arguments**

<code>sequences</code>	set of input sequences
<code>motifs</code>	set of input PWMs or PFMs
<code>raw.scores</code>	if to return scores for each base-pair
<code>verbose</code>	if to produce verbose output

cutoff	the cutoff for calling binding sites (in base 2 log).
seq.all	already concatenated sequences if already available (used to internally speed up things)

### Details

This function is not meant to be called directly, but is indirectly called by motifScores() once a global parameters useBigMemory is set.

### See Also

[motifScores](#)

---

motifSimilarity	<i>Calculates similarity between two PFMs.</i>
-----------------	--

---

### Description

This function calculates the normalized motif correlation as a measure of motif frequency matrix similarity.

### Usage

```
motifSimilarity(m1, m2, trim = 0.4, self.sim = FALSE)
```

### Arguments

m1	matrix with four rows representing the frequency matrix of first motif
m2	matrix with four rows representing the frequency matrix of second motif
trim	bases with information content smaller than this value will be trimmed off both motif ends
self.sim	if to calculate self similarity (i.e. without including offset=0 in alignment)

### Details

This score is essentially a normalized version of the sum of column correlations as proposed by Petrokovski (1996). The sum is normalized by the average motif length of m1 and m2, i.e.  $(ncol(m1)+ncol(m2))/2$ . Thus, for two identical motifs this score is going to be 1. For unrelated motifs the score is going to be typically around 0.

Motifs need to aligned for this score to be calculated. The current implementation tries all possible ungapped alignment with a minimal of two basepair matching, and the maximal score over all alignments is returned.

Motif 1 is aligned both to Motif 2 and its reverse complement. Thus, the motif similarities are the same if the reverse complement of any of the two motifs is given.

**References**

Petrokovski S. Searching databases of conserved sequence regions by aligning protein multiple-alignments. *Nucleic Acids Res* 1996;24:3836-3845.

**Examples**

```
if(requireNamespace("PWMErich.Dmelanogaster.background")){
  data(MotifDb.Dmel.PFM, package = "PWMErich.Dmelanogaster.background")

  # calculate the similarity of tin and vnd motifs (which are almost identical)
  motifSimilarity(MotifDb.Dmel.PFM[["tin"]], MotifDb.Dmel.PFM[["vnd"]])

  # similarity of two unrelated motifs
  motifSimilarity(MotifDb.Dmel.PFM[["tin"]], MotifDb.Dmel.PFM[["ttk"]])
}
```

---

```
names,MotifEnrichmentReport
      Names of variables
```

---

**Description**

Columns stored in the motif enrichment report

**Usage**

```
## S4 method for signature 'MotifEnrichmentReport'
names(x)

## S4 method for signature 'MotifEnrichmentReport'
x$name

## S4 method for signature 'MotifEnrichmentReport'
x[i, j, ..., drop = TRUE]
```

**Arguments**

x	the MotifEnrichmentReport object
name	the variable name
i	the row selector
j	unused
...	unused
drop	unused (always FALSE)

**Value**

the names of the variables

---

names,MotifEnrichmentResults  
*Names of variables*

---

**Description**

Name of different pieces of information associated with MotifEnrichmentResults

**Usage**

```
## S4 method for signature 'MotifEnrichmentResults'
names(x)
```

```
## S4 method for signature 'MotifEnrichmentResults'
x$name
```

**Arguments**

x                    the MotifEnrichmentResults object  
name                 the variable name

**Value**

the names of the variables

---

names,PWM                 *Names of variables*

---

**Description**

Name of different pieces of information associated with PWM  
Returns the motif length, i.e. the number of columns in the PWM.

**Usage**

```
## S4 method for signature 'PWM'
names(x)
```

```
## S4 method for signature 'PWM'
x$name
```

```
## S4 method for signature 'PWM'
length(x)
```

**Arguments**

x                    the PWM object  
 name                the variable name

**Value**

the names of the variables

---

names,PWMCutoffBackground  
*Names of variables*

---

**Description**

Name of different pieces of information associated with PWMCutoffBackground

**Usage**

```
## S4 method for signature 'PWMCutoffBackground'
names(x)

## S4 method for signature 'PWMCutoffBackground'
x$name
```

**Arguments**

x                    the PWMCutoffBackground object  
 name                the variable name

**Value**

the names of the variables

---

names,PWMEmpiricalBackground  
*Names of variables*

---

**Description**

Name of different pieces of information associated with PWMEmpiricalBackground

**Usage**

```
## S4 method for signature 'PWMEmpiricalBackground'  
names(x)
```

```
## S4 method for signature 'PWMEmpiricalBackground'  
x$name
```

**Arguments**

x	the PWMEmpiricalBackground object
name	the variable name

**Value**

the names of the variables

---

names,PWMGEVBackground

*Names of variables*

---

**Description**

Name of different pieces of information associated with PWMGEVBackground

**Usage**

```
## S4 method for signature 'PWMGEVBackground'  
names(x)
```

```
## S4 method for signature 'PWMGEVBackground'  
x$name
```

**Arguments**

x	the PWMGEVBackground object
name	the variable name

**Value**

the names of the variables

---

names,PWMLognBackground  
*Names of variables*

---

**Description**

Name of different pieces of information associated with PWMLognBackground

**Usage**

```
## S4 method for signature 'PWMLognBackground'  
names(x)
```

```
## S4 method for signature 'PWMLognBackground'  
x$name
```

**Arguments**

x	the PWMLognBackground object
name	the variable name

**Value**

the names of the variables

---

PFMtoPWM *Convert frequencies into motifs using PWMUnscaled*

---

**Description**

Note that this function is deprecated and replaced by toPWM().

**Usage**

```
PFMtoPWM(  
  motifs,  
  id = names(motifs),  
  name = names(motifs),  
  seq.count = NULL,  
  ...  
)
```

**Arguments**

motifs	a list of motifs represented as matrices of frequencies (PFM)
id	the set of IDs for the motifs (defaults to names of the 'motifs' list)
name	the set of names for the motifs (defaults to names of the 'motifs' list)
seq.count	if frequencies in the motifs are normalized to 1, provides a vector of sequence counts (e.g. for MotifDb motifs)
...	other parameters to PWMUnscaled

**Examples**

```
## Not run:
if (requireNamespace("PWMEnrich.Dmelanogaster.background")) {
  data(MotifDb.Dmel.PFM, package = "PWMEnrich.Dmelanogaster.background")

  # convert to PWM with uniform background
  PFMtoPWM(MotifDb.Dmel.PFM)

  # get background for drosophila (quick mode on a reduced dataset)
  prior = getBackgroundFrequencies("dm3", quick=TRUE)

  # convert with genomic background
  PFMtoPWM(MotifDb.Dmel.PFM, prior.params=prior)
}

## End(Not run)
```

---

`plot,MotifEnrichmentReport,missing-method`  
*Plot the motif enrichment report*

---

**Description**

Plots a graphical version of the motif enrichment report. Note that all values are plotted, if you want to plot only a subset of a report, first select this subset (see examples).

**Usage**

```
## S4 method for signature 'MotifEnrichmentReport,missing'
plot(
  x,
  y,
  fontsize = 14,
  id.fontsize = fontsize,
  header.fontsize = fontsize,
  widths = NULL,
  ...
)
```



**Arguments**

x	a MotifEnrichmentReport object
y	unused
fontsize	font size to use in the plot
id.fontsize	font size to use for the motif IDs
header.fontsize	font size of the header
widths	the relative widths of columns
...	unused if(requireNamespace("PWMErich.Dmelanogaster.background")) ### # load the pre-compiled lognormal background data(PWMLogn.dm3.MotifDb.Dmel, package = "PWMErich.Dmelanogaster.background") # scan two sequences for motif enrichment sequences = list(DNAString("GAAGTATCAAGTGACCAGTA DNAString("AGGTAGATAGAACAGTAGGCAATGAAGCCGATG")) res = motifEnrichment(sequences, PWMLogn.dm3.MotifDb.Dmel) # produce a report for all sequences taken together r = groupReport(res) # plot the top 10 most enriched motifs plot(r[1:10])

---

plot,PWM,missing-method

*Plotting for the PWM class*


---

**Description**

This function produces a sequence logo (via package seqLogo).

**Usage**

```
## S4 method for signature 'PWM,missing'
plot(x, y, ...)
```

**Arguments**

x	the PWM object
y	unused
...	other parameters to pass to seqLogo's plot function

**Examples**

```
if(requireNamespace("PWMErich.Dmelanogaster.background")){
  data(MotifDb.Dmel, package = "PWMErich.Dmelanogaster.background")

  # plot the tinman motif from MotifDb
  plot(MotifDb.Dmel[["tin"]])
}
```

---

plotMotifScores      *Plot the raw motifs scores as returned by motifScores()*

---

### Description

This function visualises the motif scores for one or more sequences. Sequences are drawn as lines, and scores are plotted as triangles at both sides of the line (corresponding to the two strands). The width of the base of the triangle corresponds to motif width and the height to the motif  $\log(\text{score})$  that is positive and greater than the `cutoff` parameter (if specified). All scores have the same y-axis, so the heights of bars are comparable between sequences and motifs.

### Usage

```
plotMotifScores(
  scores,
  sel.motifs = NULL,
  seq.names = NULL,
  cols = NULL,
  cutoff = NULL,
  log.fun = log2,
  main = "",
  legend.space = 0.3,
  max.score = NULL,
  trans = 0.5,
  text.cex = 0.9,
  legend.cex = 0.9,
  motif.names = NULL,
  seq.len.spacing = 8,
  shape = "rectangle"
)
```

### Arguments

<code>scores</code>	the list of motifs scores. Each element of the list is a matrix of scores for one sequences. The columns in the matrix correspond to different motifs. Each column contains the odds (not log-odds!) scores over both strands. For example, for a sequence of length 5, scores for a 3 bp motifs could be: <code>c(0.1, 1, 4, NA, NA, 1, 0.3, 2, NA, NA)</code> . The first 3 numbers are odds scores starting at first three bases, and the second lot of 3 numbers is the scores starting at the same positions but with the reverse complement of the motif. The last two values are NA on both strands because we do not support partial motif hits.
<code>sel.motifs</code>	a vector of motif names. Use this parameter to show the motif hits to only a subset of motifs for which the scores are available.
<code>seq.names</code>	a vector of sequence names to show in the graph. If none specified, the sequences will be named Sequence 1, Sequence 2, ...

cols	a vector of colours to use to colour code motif hits. If none are specified, the current palette will be used.
cutoff	either a single value, or a vector of values. The values are PWM cutoffs after log.fun (see below). Only motif scores above these cutoffs will be shown. If a single values is specified, it will be used for all PWMs, otherwise the vector needs to specify one cutoff per PWM.
log.fun	the logarithm function to use to calculate log-odds. By default log2 is used for consistency with Biostrings.
main	the main title
legend.space	the proportion of horizontal space to reserve for the legend. The default is 30%.
max.score	the maximal log-odds score used to scale all other scores. By default this values is automatically determined, but it can also be set manually to make multiple plots comparable.
trans	the level of transparency. By default 50% transparency to be able to see overlapping binding sites
text.cex	the scaling factor for sequence names
legend.cex	the scaling factor for the legend
motif.names	optional vector of motif names to show instead of those present as column names in scores
seq.len.spacing	the spacing (in bp units) between the end of the sequence line and the text showing the length in bp
shape	the shape to use to draw motif occurrences, valid values are "rectangle" (default), "line" and "triangle"

### Examples

```

if(requireNamespace("PWMErich.Dmelanogaster.background")){
  ###
  # Load Drosophila PWMs
  data(MotifDb.Dmel, package = "PWMErich.Dmelanogaster.background")

  # two sequences of interest
  sequences = list(DNAString("GAAGTATCAAGTGACCAGGTGAAGTCCCAGATGA"),
    DNAString("AGGTAGATAGAACAGTAGGCAATGAAGCCGATG"))

  # select the tinman and snail motifs
  pwms = MotifDb.Dmel[c("tin", "sna")]

  # get the raw score that will be plotted
  scores = motifScores(sequences, pwms, raw.scores=TRUE)

  # plot the scores in both sequences, green for tin and blue for sna
  plotMotifScores(scores, cols=c("green", "blue"))
}

```

---

plotMultipleMotifs      *Plot multiple motifs in a single plot*

---

### Description

Individual motif logos are plotted on a rows x cols grid. This function is a convenience interface for the seqLogoGrid function that deals with viewpoint placement in a matrix-like grid layout.

### Usage

```
plotMultipleMotifs(  
  pwms,  
  titles = names(pwms),  
  rows = ceiling(sqrt(length(pwms))),  
  cols = ceiling(sqrt(length(pwms))),  
  xmargin.scale = 0.4,  
  ymargin.scale = 0.4,  
  ...  
)
```

### Arguments

pwms	a list of PWM objects or frequency matrices
titles	a character vector of titles for each of the plots
rows	number of rows in the grid
cols	number of cols in the grid
xmargin.scale	the scaling parameter for the X-axis margin. Useful when plotting more than one logo on a page
ymargin.scale	the scaling parameter for the Y-axis margin. Useful when plotting more than one logo on a page
...	other parameters passed to seqLogoGrid()

### Details

By default will try to make a square grid plot that would fit all the motifs and use list names as captions.

---

plotPFM	<i>Plot a PFM (not PWM) using seqLogo</i>
---------	---

---

**Description**

Plot a PFM (not PWM) using seqLogo

**Usage**

```
plotPFM(pfm, ...)
```

**Arguments**

pfm	a matrix where rows are the four nucleotides
...	additional parameters for plot()

---

plotTopMotifsGroup, MotifEnrichmentResults-method	<i>Plot the top N enrichment motifs in a group of sequences</i>
---	---

---

**Description**

Plot the top N enrichment motifs in a group of sequences

**Usage**

```
## S4 method for signature 'MotifEnrichmentResults'
plotTopMotifsGroup(obj, n, bg = TRUE, id = FALSE, ...)
```

**Arguments**

obj	a MotifEnrichmentResults object
n	the number of top ranked motifs to plot
bg	if to use background corrected P-values to do the ranking (if available)
id	if to show PWM IDs instead of target TF names
...	other parameters passed to plotMultipleMotifs()

**Examples**

```

if(requireNamespace("PWMErich.Dmelanogaster.background")){
  ###
  # load the pre-compiled lognormal background
  data(PWMLogn.dm3.MotifDb.Dmel, package = "PWMErich.Dmelanogaster.background")

  # scan two sequences for motif enrichment
  sequences = list(DNAString("GAAGTATCAAGTGACCAGTAAGTCCCAGATGA"),
    DNAString("AGGTAGATAGAACAGTAGGCAATGAAGCCGATG"))

  res = motifEnrichment(sequences, PWMLogn.dm3.MotifDb.Dmel)

  # plot the top 4 motifs in a 2x2 grid
  plotTopMotifsGroup(res, 4)

  # plot top 3 motifs in a single row
  plotTopMotifsGroup(res, 3, row=1, cols=3)
}

```

---

`plotTopMotifsSequence,MotifEnrichmentResults-method`

*Plot the top N enrichment motifs in a single sequence*

---

**Description**

Plot the top N enrichment motifs in a single sequence

**Usage**

```

## S4 method for signature 'MotifEnrichmentResults'
plotTopMotifsSequence(obj, seq.id, n, bg = TRUE, id = FALSE, ...)

```

**Arguments**

<code>obj</code>	a <code>MotifEnrichmentResults</code> object
<code>seq.id</code>	either the sequence number or sequence name
<code>n</code>	the number of top ranked motifs to plot
<code>bg</code>	if to use background corrected P-values to do the ranking (if available)
<code>id</code>	if to show PWM IDs instead of target TF names
<code>...</code>	other parameters passed to <code>plotMultipleMotifs()</code>

**Examples**

```

if(requireNamespace("PWMErich.Dmelanogaster.background")){
  ###
  # load the pre-compiled lognormal background
  data(PWMLogn.dm3.MotifDb.Dmel, package = "PWMErich.Dmelanogaster.background")

  # scan two sequences for motif enrichment
  sequences = list(DNAString("GAAGTATCAAGTGACCCAGTAAGTCCCAGATGA"),
    DNAString("AGGTAGATAGAACAGTAGGCAATGAAGCCGATG"))

  res = motifEnrichment(sequences, PWMLogn.dm3.MotifDb.Dmel)

  # plot the top 4 motifs in a 2x2 grid
  plotTopMotifsSequence(res, 1, 4)

  # plot top 3 motifs in a single row
  plotTopMotifsSequence(res, 1, 3, row=1, cols=3)
}

```

---

PWM-class

*A class that represents a Position Weight Matrix (PWM)*


---

**Description**

A class that represents a Position Weight Matrix (PWM)

**Slots**

**id:** a systematic ID given to this PWM, could include the source, version, etc

**name:** the name of the transcription factor (TF) to which the PWM corresponds to

**pfm:** Position Frequency Matrix (PFM) from which the PWM is derived

**prior.params:** Defines prior frequencies of the four bases (A,C,G,T), a named vector. These will be added to individual values for the PFM and at the same time used as background probabilities

**pwm:** Final Position Weight Matrix (PWM) constructed using prior.params with logarithm base 2

---

PWMCutoffBackground-class

*Hit count background distribution for a set of PWMs*


---

**Description**

Hit count background distribution for a set of PWMs

**Slots**

bg.source: textual description of where the background distribution is derived from  
 bg.cutoff: the cutoff score used to find significant motif hits (in log2 odds), either a single value or a vector of values  
 bg.P: the density of significant motif hits per nucleotide in background  
 pwms: the pwms for which the background has been compiled

---

 PWMEmpiricalBackground-class

*Background for calculating empirical P-values*

---

**Description**

This object contains raw scores for one very long sequence, thus it can be very large.

**Slots**

bg.source: textual description of where the background distribution is derived from  
 bg.fwd: affinity scores (odds) for the forward strand. PWMs as columns  
 bg.rev: affinity scores (odds) for the reverse strand. PWMs as columns  
 pwms: the pwms for which the background has been compiled

---

 PWMGEVBackground-class

*Generalized Extreme Values (GEV) background for P-values*

---

**Description**

The three parameters of the GEV distribution are fitted by doing linear regression on log of sequence length.

**Slots**

bg.source: textual description of where the background distribution is derived from  
 bg.loc: linear regression model for estimating the location parameter based on log(L), list of lm objects of PWMs  
 bg.scale: linear regression model for estimating the scale parameter based on log(L), list of lm objects of PWMs  
 bg.shape: linear regression model for estimating the shape parameter based on log(L), list of lm objects of PWMs  
 pwms: the pwms for which the background has been compiled



---

 PWMLognBackground-class

*Lognormal background distribution for a set of PWMs*


---

### Description

Lognormal background distribution for a set of PWMs

### Slots

`bg.source`: textual description of where the background distribution is derived from

`bg.len`: the length to which the background is normalized to. This is a vector of values, can have a different value for each motif.

`bg.mean`: the mean value of the lognormal distribution at `bg.len`

`bg.sd`: the standard deviation of the lognormal distribution at `bg.len`

`pwms`: the pwms for which the background has been compiled

---

 PWMUnscaled

*Create a PWM from PFM*


---

### Description

The PWM function from Biostrings without unit scaling

### Usage

```
PWMUnscaled(
  x,
  id = "",
  name = "",
  type = c("log2probratio", "prob"),
  prior.params = c(A = 0.25, C = 0.25, G = 0.25, T = 0.25),
  pseudo.count = prior.params,
  unit.scale = FALSE,
  seq.count = NULL
)
```

### Arguments

<code>x</code>	the integer count matrix representing the motif, rows as nucleotides
<code>id</code>	a systematic ID given to this PWM, could include the source, version, etc
<code>name</code>	the name of the transcription factor (TF) to which the PWM corresponds to

type	the type of PWM calculation, either as log2-odds, or posterior probability (frequency matrix)
prior.params	the pseudocounts for each of the nucleotides
pseudo.count	the pseudo-count values if different from priors
unit.scale	if to unit.scale the pwm (default is no unit scaling)
seq.count	if x is a normalised PFM (i.e. with probabilities instead of sequence counts), then this sequence count will be used to convert x into a count matrix

### Details

By default the Biostrings package scales the log-odds score so it is within 0 and 1. In this function we take a more traditional approach with no unit scaling and offer unit scaling as an additional parameter.

See ?PWM from Biostrings for more information on input arguments.

### Value

a new PWM object representing the PWM

### Examples

```
if(requireNamespace("PWMEnrich.Dmelanogaster.background")){
  data(MotifDb.Dmel.PFM, package = "PWMEnrich.Dmelanogaster.background")

  ttk = MotifDb.Dmel.PFM[["ttk"]]

  # make a PWM with uniform background
  PWMUnscaled(ttk, id="ttk-JASPAR", name="ttk")

  # custom background
  PWMUnscaled(ttk, id="ttk-JASPAR", name="ttk",
    prior.params=c("A" = 0.2, "C" = 0.3, "G" = 0.3, "T" = 0.2))

  # get background for drosophila (quick mode on a reduced dataset)
  prior = getBackgroundFrequencies("dm3", quick=TRUE)

  # convert using genomic background
  PWMUnscaled(ttk, id="ttk-JASPAR", name="ttk", prior.params=prior)
}
```

---

rankingProcessAndReturn

*A helper function for motifRankingForGroup and motifRankingForSequence with the common code*

---

**Description**

A helper function for `motifRankingForGroup` and `motifRankingForSequence` with the common code

**Usage**

```
rankingProcessAndReturn(res, r, id, order, rank, unique, decreasing)
```

**Arguments**

<code>res</code>	the list of results from <code>MotifEnrichmentResults</code> object
<code>r</code>	the vector of raw results that needs to be processed
<code>id</code>	if to return IDs instead of names
<code>order</code>	if to return the ordering of motifs
<code>rank</code>	if to return the rank of motifs
<code>unique</code>	if to remove duplicates
<code>decreasing</code>	specifies the sorting order

---

<code>readJASPAR</code>	<i>Read motifs in JASPAR format</i>
-------------------------	-------------------------------------

---

**Description**

Read motifs in JASPAR format

**Usage**

```
readJASPAR(file, remove.ids = FALSE)
```

**Arguments**

<code>file</code>	the filename
<code>remove.ids</code>	if to strip JASPAR ID's from motif names, e.g. "MA0211.1 bap" would become just "bap"

**Value**

a list of matrices representing motifs (with four nucleotides as rows)

---

readMotifs	<i>Read in motifs in JASPAR or TRANSFAC format</i>
------------	--

---

**Description**

The format is autodetected based on file format. If the autodetection fail then the file cannot be read.

**Usage**

```
readMotifs(file, remove.acc = FALSE)
```

**Arguments**

file	the filename
remove.acc	if to remove accession numbers. If TRUE, the AC entry in TRANSFAC files is ignored, and the accession is stripped from JASPAR, e.g. motif with name "MA0211.1 bap" would become just "bap". If FALSE, both the AC and ID are used to generate the TRANSFAC name and the original motif names are preserved in JASPAR files.

**Value**

a list of 4xL matrices representing motifs (four nucleotides as rows)

**Examples**

```
# read in example TRANSFAC motifs without accession codes (just IDs)
readMotifs(system.file(package = "PWMErich", dir = "extdata", file = "example.transfac"),
  remove.acc = TRUE)

# read in the JASPAR insects motifs provided as example
readMotifs(system.file(package = "PWMErich", dir = "extdata", file = "jaspar-insecta.jaspar"),
  remove.acc = TRUE)
```

---

readTRANSFAC	<i>Read in motifs in TRANSFAC format</i>
--------------	--

---

**Description**

Read in motifs in TRANSFAC format

**Usage**

```
readTRANSFAC(file, remove.acc = TRUE)
```

**Arguments**

file            the filename  
remove.acc     if to ignore transfac accession numbers

**Value**

a list of matrices representing motifs (with four nucleotides as rows)

---

registerCoresPWMErich

*Register than PWMErich can use parallel CPU cores*

---

**Description**

Certain functions (like motif scanning) can be parallelized in PWMErich. This function registers a number of parallel cores (via core package parallel) to be used in code that can be parallelized. After this function is called, all further PWMErich function calls will run in parallel if possible.

**Usage**

```
registerCoresPWMErich(numCores = NA)
```

**Arguments**

numCores        number of cores to use (default to take all cores), or NULL if no parallel execution is to be used

**Details**

By default parallel execution is turned off. To turn it off after using it, call this function by passing NULL.

**Examples**

```
## Not run:  
registerCoresPWMErich(4) # use 4 CPU cores in PWMErich  
registerCoresPWMErich() # use maximal number of CPUs  
registerCoresPWMErich(NULL) # do not use parallel execution  
  
## End(Not run)
```

---

reverseComplement, PWM-method  
*Reverse complement for the PWM object*

---

### Description

Finds the reverse complement of the PWM

### Usage

```
## S4 method for signature 'PWM'  
reverseComplement(x, ...)
```

### Arguments

x	an object of type PWM
...	unused

### Value

an object of type PWM that is reverse complement of x

### Examples

```
if(requireNamespace("PWMEnrich.Dmelanogaster.background")){  
  data(MotifDb.Dmel.PFM, package = "PWMEnrich.Dmelanogaster.background")  
  
  reverseComplement(MotifDb.Dmel.PFM[["ttk"]]) # reverse complement of the ttk PWM  
}
```

---

scanWithPWM *Scan the whole sequence on both strands*

---

### Description

The whole sequence is scanned with a PWM and scores returned beginning at each position. Partial motif matches are not done, thus the last #[length of motif]-1 scores are NA.

**Usage**

```
scanWithPWM(
  pwm,
  dna,
  pwm.rev = NULL,
  odds.score = FALSE,
  both.strands = FALSE,
  strand.fun = "mean"
)
```

**Arguments**

pwm	PWM object
dna	a DNASTring or other sequence from Biostrings
pwm.rev	the reverse complement for a pwm (if it is already pre-computed)
odds.score	if to return raw scores in odds (not logodds) space
both.strands	if to return results on both strands
strand.fun	which function to use to summarise values over two strands (default is "mean")

**Details**

The function returns either an odds average (*\*not\** log-odds average), maximal score on each strand, or scores on both strands.

The function by default returns the score in log2 following the package Biostrings.

**Value**

a vector representing scores starting at each position, or a matrix with score in the two strands

**Examples**

```
if(requireNamespace("PWMErich.Dmelanogaster.background")){
  data(MotifDb.Dmel, package = "PWMErich.Dmelanogaster.background")

  ttk = MotifDb.Dmel[["ttk"]]

  # odds average over the two strands expressed as log2-odds
  scanWithPWM(ttk, DNASTring("CGTAGGATAAAGTAACT"))

  # log2-odds scores on both strands
  scanWithPWM(ttk, DNASTring("CGTAGGATAAAGTAACT"), both.strands=TRUE)
}
```

seqLogoGrid

*Draw a motif logo on an existing viewport***Description**

This function comes from the seqLogo package. It has been modified to remove some unnecessary code as suggested by W Huber (<https://stat.ethz.ch/pipermail/bioconductor/2010-September/035267.html>).

**Usage**

```
seqLogoGrid(
  pwm,
  ic.scale = TRUE,
  xaxis = TRUE,
  yaxis = TRUE,
  xfontsize = 10,
  yfontsize = 10,
  xmargin.scale = 1,
  ymargin.scale = 1,
  title = "",
  titlefontsize = 15
)
```

**Arguments**

pwm	numeric	The 4xW position weight matrix.
ic.scale	logical	If TRUE, the height of each column is proportional to its information content. Otherwise, all columns have the same height.
xaxis	logical	If TRUE, an X-axis will be plotted.
yaxis	logical	If TRUE, a Y-axis will be plotted.
xfontsize	numeric	Font size to be used for the X-axis.
yfontsize	numeric	Font size to be used for the Y-axis.
xmargin.scale		the scaling parameter for the X-axis margin. Useful when plotting more than one logo on a page
ymargin.scale		the scaling parameter for the Y-axis margin. Useful when plotting more than one logo on a page
title		to be shown on the top
titlefontsize		the fontsize of the title

**Details**

Use this function for more advanced plotting where the viewports are directly set up and maintained (see package grid).



---

sequenceReport,MotifEnrichmentResults-method  
*Generate a motif enrichment report for a single sequence*

---

## Description

Generate a motif enrichment report for a single sequence

## Usage

```
## S4 method for signature 'MotifEnrichmentResults'  
sequenceReport(obj, seq.id, bg = TRUE, ...)
```

## Arguments

obj	a MotifEnrichmentResults object
seq.id	the sequence index or name
bg	if to use background corrected P-values to do the ranking (if available)
...	unused

## Value

a MotifEnrichmentReport object containing a table with the following columns:

- 'rank' - The rank of the PWM's enrichment in the sequence
- 'target' - The name of the PWM's target gene, transcript or protein complex.
- 'id' - The unique identifier of the PWM (if set during PWM creation).
- 'raw.score' - The raw score before P-value calculation
- 'p.value' - The P-value of motif enrichment (if available)

## Examples

```
if(requireNamespace("PWMEnrich.Dmelanogaster.background")){  
  ###  
  # load the pre-compiled lognormal background  
  data(PWMLogn.dm3.MotifDb.Dmel, package = "PWMEnrich.Dmelanogaster.background")  
  
  # scan two sequences for motif enrichment  
  sequences = list(DNAString("GAAGTATCAAGTGACCAGTAAGTCCCAGATGA"),  
    DNAString("AGGTAGATAGAACAGTAGGCAATGAAGCCGATG"))  
  
  res = motifEnrichment(sequences, PWMLogn.dm3.MotifDb.Dmel)  
  
  # reports for the two sequences  
  r1 = sequenceReport(res, 1)  
  r2 = sequenceReport(res, 2)
```

```
# view the results
r1
r2

# plot the top 10 most enriched motifs in the first, and then second sequence
plot(r1[1:10])
plot(r2[1:10])

}
```

---

```
show,MotifEnrichmentReport-method
    show method for MotifEnrichmentReport
```

---

### Description

show method for MotifEnrichmentReport

### Usage

```
## S4 method for signature 'MotifEnrichmentReport'
show(object)
```

### Arguments

object            the MotifEnrichmentReport object

---

```
show,MotifEnrichmentResults-method
    show method for MotifEnrichmentResults
```

---

### Description

show method for MotifEnrichmentResults

### Usage

```
## S4 method for signature 'MotifEnrichmentResults'
show(object)
```

### Arguments

object            the MotifEnrichmentResults object

---

show, PWM-method	<i>show method for PWM</i>
------------------	----------------------------

---

### **Description**

show method for PWM

### **Usage**

```
## S4 method for signature 'PWM'  
show(object)
```

### **Arguments**

object            the PWM object

---

show, PWMCutoffBackground-method	<i>show method for PWMCutoffBackground</i>
----------------------------------	--

---

### **Description**

show method for PWMCutoffBackground

### **Usage**

```
## S4 method for signature 'PWMCutoffBackground'  
show(object)
```

### **Arguments**

object            the PWMCutoffBackground object

---

show,PWMEmpiricalBackground-method  
*show method for PWMEmpiricalBackground*

---

**Description**

show method for PWMEmpiricalBackground

**Usage**

```
## S4 method for signature 'PWMEmpiricalBackground'  
show(object)
```

**Arguments**

object            the PWMEmpiricalBackground object

---

show,PWMGEVBackground-method  
*show method for PWMGEVBackground*

---

**Description**

show method for PWMGEVBackground

**Usage**

```
## S4 method for signature 'PWMGEVBackground'  
show(object)
```

**Arguments**

object            the PWMGEVBackground object

---

```
show,PWMLognBackground-method
      show method for PWMLognBackground
```

---

**Description**

show method for PWMLognBackground

**Usage**

```
## S4 method for signature 'PWMLognBackground'
show(object)
```

**Arguments**

object                    the PWMLognBackground object

---

```
toPWM                      Convert motifs into PWMs
```

---

**Description**

Convert motifs into PWMs

**Usage**

```
toPWM(
  motifs,
  ids = names(motifs),
  targets = names(motifs),
  seq.count = 50,
  prior = c(A = 0.25, C = 0.25, G = 0.25, T = 0.25),
  ...
)
```

**Arguments**

motifs                    a list of motifs either as position probability matrices (PPM) or frequency matrices (PFMs)

ids                        the set of IDs for the motifs (defaults to names of the 'motifs' list)

targets                   the set of target TF names for the motifs (defaults to names of the 'motifs' list)

seq.count                provides a vector of sequence counts for probability matrices (PPMs). Default is 50.

prior                     frequencies of the four letters in the genome. Default is uniform background.

...                        other parameters to PWMUnscaled

## Examples

```
## Not run:
if (requireNamespace("PWMEnrich.Dmelanogaster.background")) {
  data(MotifDb.Dmel.PFM, package = "PWMEnrich.Dmelanogaster.background")

  toPWM(MotifDb.Dmel.PFM) # convert to PWM with uniform background

  # get background for drosophila (quick mode on a reduced dataset)
  prior = getBackgroundFrequencies("dm3", quick=TRUE)
  toPWM(MotifDb.Dmel.PFM, prior=prior) # convert with genomic background
}

## End(Not run)
```

---

tryAllMotifAlignments *Try all motif alignments and return max score*

---

## Description

This function tries all offsets of motif1 compared to motif2 and returns the maximal (unnormalized) correlation score.

## Usage

```
tryAllMotifAlignments(m1, m2, min.align = 2, exclude.zero = FALSE)
```

## Arguments

m1	frequency matrix of motif 1
m2	frequency matrix of motif 2
min.align	minimal number of basepairs that need to align
exclude.zero	if to exclude offset=0, useful for calculating self-similarity

## Details

The correlation score is essentially the sum of correlations of individual aligned columns as described in Pietrokovski (1996).

## Value

single maximal score

## References

Pietrokovski S. Searching databases of conserved sequence regions by aligning protein multiple-alignments. *Nucleic Acids Res* 1996;24:3836-3845.

---

useBigMemoryPWMErich *If to use a faster implementation of motif scanning that requires about 5 to 10 times more memory*

---

**Description**

If to use a faster implementation of motif scanning that requires about 5 to 10 times more memory

**Usage**

```
useBigMemoryPWMErich(useBigMemory = FALSE)
```

**Arguments**

useBigMemory a boolean value denoting if to use big memory implementation

**Examples**

```
## Not run:
useBigMemoryPWMErich(TRUE) # switch to big memory implementation globally
useBigMemoryPWMErich(FALSE) # switch back to default implementation

## End(Not run)
```

---

[,PWMCutoffBackground-method  
*Get the background for a subset of PWMs*

---

**Description**

Get the background for a subset of PWMs

**Usage**

```
## S4 method for signature 'PWMCutoffBackground'
x[i, j, ..., drop = TRUE]
```

**Arguments**

x	the PWMCutoffBackground object
i	the indices of PWMs
j	unused
...	unused
drop	unused

---

```
[,PWMEmpiricalBackground-method
    Get the background for a subset of PWMs
```

---

**Description**

Get the background for a subset of PWMs

**Usage**

```
## S4 method for signature 'PWMEmpiricalBackground'
x[i, j, ..., drop = TRUE]
```

**Arguments**

x	the PWMEmpiricalBackground object
i	the indices of PWMs
j	unused
...	unused
drop	unused

---

```
[,PWMGEVBackground-method
    Get the background for a subset of PWMs
```

---

**Description**

Get the background for a subset of PWMs

**Usage**

```
## S4 method for signature 'PWMGEVBackground'
x[i, j, ..., drop = TRUE]
```

**Arguments**

x	the PWMGEVBackground object
i	the indices of PWMs
j	unused
...	unused
drop	unused



---

`[,PWMLognBackground-method`*Get the background for a subset of PWMs*

---

**Description**

Get the background for a subset of PWMs

**Usage**

```
## S4 method for signature 'PWMLognBackground'  
x[i, j, ..., drop = TRUE]
```

**Arguments**

x	the PWMLognBackground object
i	the indices of PWMs
j	unused
...	unused
drop	unused

# Index

- \* **internal**
  - PWMErich-package, 4
  - .inputPFMfromMatrixOrPWM, 5
  - .inputParamMotifs, 4
  - .inputParamSequences, 5
  - .normalize.bg.seq, 6
  - .normargPfm, 6
  - .normargPriorParams, 6
  - [,MotifEnrichmentReport-method
    - (names,MotifEnrichmentReport), 43
  - [,PWMCutoffBackground-method, 71
  - [,PWMEmpiricalBackground-method, 72
  - [,PWMGEVBackground-method, 72
  - [,PWMLognBackground-method, 73
  - \$,MotifEnrichmentReport-method
    - (names,MotifEnrichmentReport), 43
  - \$,MotifEnrichmentResults-method
    - (names,MotifEnrichmentResults), 44
  - \$,PWM-method (names,PWM), 44
  - \$,PWMCutoffBackground-method
    - (names,PWMCutoffBackground), 45
  - \$,PWMEmpiricalBackground-method
    - (names,PWMEmpiricalBackground), 45
  - \$,PWMGEVBackground-method
    - (names,PWMGEVBackground), 46
  - \$,PWMLognBackground-method
    - (names,PWMLognBackground), 47
- affinitySequenceSet, 7
- as.data.frame
  - (as.data.frame,MotifEnrichmentReport-method), 7
- as.data.frame,MotifEnrichmentReport-method, 7
- cloverPvalue1seq, 8
- cloverScore, 9
- colMedians, 9
- colSds, 10
- concatenateSequences, 10
- cutoffZscore, 11
- cutoffZscoreSequenceSet, 11
- divideRows, 12
- DNAStrngSetToList, 12
- empiricalPvalue, 12
- empiricalPvalueSequenceSet, 13
- getBackgroundFrequencies, 14
- getPromoters, 15
- gevPerSequence, 15
- groupReport
  - (groupReport,MotifEnrichmentResults-method), 16
- groupReport,MotifEnrichmentResults-method, 16
- keepFinite, 17
- length,PWM-method (names,PWM), 44
- logNormPval, 18
- logNormPvalSequenceSet, 18
- makeBackground, 19, 33
- makePriors, 20
- makePWMCutoffBackground, 21
- makePWMEmpiricalBackground, 22
- makePWMGEVBackground, 23
- makePWMLognBackground, 24
- makePWMPvalCutoffBackground, 25
- makePWMPvalCutoffBackgroundFromSeq, 26
- makeStartEndPos, 27
- matrixShuffleZscorePerSequence, 27
- maxAligned, 28
- motifDiffEnrichment, 29
- motifEcdf, 30

- motifEnrichment, 31
- MotifEnrichmentReport-class, 35
- MotifEnrichmentResults-class, 35
- motifIC, 35
- motifPrAUC, 36
- motifRankingForGroup
  - (motifRankingForGroup,MotifEnrichmentResults-method), 37
- motifRankingForGroup,MotifEnrichmentResults-method, (plotTopMotifsSequence,MotifEnrichmentResults-method), 37
- motifRankingForSequence
  - (motifRankingForSequence,MotifEnrichmentResults-method), 38
- motifRankingForSequence,MotifEnrichmentResults-method, (plotTopMotifsSequence,MotifEnrichmentResults-method), 38
- motifRecoveryAUC, 39
- motifScores, 40, 41, 42
- motifScoresBigMemory, 41
- motifSimilarity, 42
  
- names,MotifEnrichmentReport, 43
- names,MotifEnrichmentReport-method
  - (names,MotifEnrichmentReport), 43
- names,MotifEnrichmentResults, 44
- names,MotifEnrichmentResults-method
  - (names,MotifEnrichmentResults), 44
- names,PWM, 44
- names,PWM-method (names,PWM), 44
- names,PWMCutoffBackground, 45
- names,PWMCutoffBackground-method
  - (names,PWMCutoffBackground), 45
- names,PWMEmpiricalBackground, 45
- names,PWMEmpiricalBackground-method
  - (names,PWMEmpiricalBackground), 45
- names,PWMGEVBackground, 46
- names,PWMGEVBackground-method
  - (names,PWMGEVBackground), 46
- names,PWMLognBackground, 47
- names,PWMLognBackground-method
  - (names,PWMLognBackground), 47
  
- PFMtoPWM, 47
- plot,MotifEnrichmentReport,missing-method, 48
- plot,PWM,missing-method, 49
- plotMotifScores, 50
- plotMultipleMotifs, 52
- plotPFM, 53
- plotTopMotifsGroup
  - (plotTopMotifsGroup,MotifEnrichmentResults-method), 53
- plotTopMotifsGroup,MotifEnrichmentResults-method, plotTopMotifsSequence, 54
- plotTopMotifsSequence,MotifEnrichmentResults-method, PWM-class, 55
- PWMCutoffBackground-class, 55
- PWMEmpiricalBackground-class, 56
- PWMErich (PWMErich-package), 4
- PWMErich-package, 4
- PWMGEVBackground-class, 56
- PWMLognBackground-class, 57
- PWMUnscaled, 57
  
- rankingProcessAndReturn, 58
- readJASPAR, 59
- readMotifs, 60
- readTRANSFAC, 60
- registerCoresPWMErich, 61
- reverseComplement, PWM-method, 62
  
- scanWithPWM, 62
- seqLogoGrid, 64
- sequenceReport
  - (sequenceReport,MotifEnrichmentResults-method), 65
- sequenceReport,MotifEnrichmentResults-method, 65
- show,MotifEnrichmentReport-method, 66
- show,MotifEnrichmentResults-method, 66
- show,PWM-method, 67
- show,PWMCutoffBackground-method, 67
- show,PWMEmpiricalBackground-method, 68
- show,PWMGEVBackground-method, 68
- show,PWMLognBackground-method, 69
  
- toPWM, 69
- tryAllMotifAlignments, 70
- useBigMemoryPWMErich, 71