

# Package ‘DiffLogo’

December 31, 2024

**Type** Package

**Title** DiffLogo: A comparative visualisation of biooligomer motifs

**Version** 2.31.0

**Date** 2015-02-12

**Author** c(

```
person(`` Martin", `` Nettling", role = c(`` aut", `` cre"), email = `` martin.nettling@informatik.uni-halle.de"),
person(`` Hendrik", `` Treutler", role = c(`` aut", `` cre"), email = `` hendrik.treutler@ipb-halle.de"),
person(`` Jan", `` Grau", role = c(`` aut", `` ctb"), email = `` grau@informatik.uni-halle.de"),
person(`` Andrey", `` Lando", role = c(`` aut", `` ctb"), email = `` dronte@autosome.ru"),
person(`` Jens", `` Keilwagen", role = c(`` aut", `` ctb"), email = `` jens.keilwagen@julius-kuehn.de"),
person(`` Stefan", `` Posch", role = `` aut", email = `` posch@informatik.uni-halle.de"),
person(`` Ivo", `` Grosse", role = `` aut", email = `` grosse@informatik.uni-halle.de"))
```

**Depends** R (>= 3.4), stats, cba

**Imports** grDevices, graphics, utils, tools

**Suggests** knitr, testthat, seqLogo, MotifDb

**Maintainer** Hendrik Treutler<hendrik.treutler@gmail.com>

**Description** DiffLogo is an easy-to-use tool to visualize motif differences.

**License** GPL (>= 2)

**URL** <https://github.com/mgledi/DiffLogo/>

**BugReports** <https://github.com/mgledi/DiffLogo/issues>

**biocViews** Software, SequenceMatching, MultipleComparison, MotifAnnotation, Visualization, Alignment

**Collate** 'alphabet.R' 'baseDistrs.R' 'diffSeqLogo.R' 'preconditions.R' 'seqLogo.R' 'stackHeights.R' 'utilities.R' 'diffSeqLogoSupport.R' 'pwmAlignment.R'

**RoxygenNote** 6.1.1

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

**git\_branch** devel  
**git\_last\_commit** 33af0b4  
**git\_last\_commit\_date** 2024-10-29  
**Repository** Bioconductor 3.21  
**Date/Publication** 2024-12-31

## Contents

alignPwmSets . . . . .	3
Alphabet . . . . .	4
ASN . . . . .	4
baseDistributionPwm . . . . .	5
calculatePvalue . . . . .	5
createDiffLogoObject . . . . .	6
differenceOfICs . . . . .	8
diffLogo . . . . .	9
diffLogoFromPwm . . . . .	10
diffLogoTable . . . . .	11
diffLogoTableConfiguration . . . . .	12
DNA . . . . .	13
drawDiffLogoTable . . . . .	14
enrichDiffLogoObjectWithPvalues . . . . .	15
enrichDiffLogoTableWithPvalues . . . . .	16
extendPwmsFromAlignmentVector . . . . .	17
FULL_ALPHABET . . . . .	18
getAlphabetFromCharacters . . . . .	18
getAlphabetFromSequences . . . . .	19
getPwmFromAlignment . . . . .	19
getPwmFromAlignmentFile . . . . .	20
getPwmFromFastaFile . . . . .	21
getPwmFromFile . . . . .	21
getPwmFromHomerFile . . . . .	22
getPwmFromPfmOrJasparFile . . . . .	22
getPwmFromPwmFile . . . . .	23
getSequencesFromAlignmentFile . . . . .	23
getSequencesFromFastaFile . . . . .	24
informationContent . . . . .	25
localPwmAlignment . . . . .	25
lossOfAbsICDifferences . . . . .	26
multipleLocalPwmsAlignment . . . . .	27
normalizedDifferenceOfProbabilities . . . . .	28
normalizePWM . . . . .	29
prepareDiffLogoTable . . . . .	30
probabilities . . . . .	31
pwmDivergence . . . . .	31
pwmsDistanceMatrix . . . . .	32

*alignPwmSets* 3

reverseAlignmentVector . . . . .	33
RNA . . . . .	33
seqLogo . . . . .	34
shannonDivergence . . . . .	35
sumOfAbsICDifferences . . . . .	36
sumOfAbsProbabilityDifferences . . . . .	37
sumProbabilities . . . . .	38
switchDirection . . . . .	38
twoSetsAveragePwmDivergenceFromAlignmentVector . . . . .	39

**Index** 40

---

*alignPwmSets*      *Multiple PWMs alignment*

---

### **Description**

Align two sets of pwms

### **Usage**

```
alignPwmSets(left_pwms_set, left_alignment, right_pwms_set,  
             right_alignment, try_reverse_complement)
```

### **Arguments**

`left_pwms_set` list of pwms(matrixes)  
`left_alignment` alignment of left\_pwms\_set.  
`right_pwms_set` list of pwms;  
`right_alignment`  
                  alignment of right\_pwms\_set.  
`try_reverse_complement`  
                  if true(default), also try reverse complement.

### **Value**

list - alignment of concatenation of left\_pwms\_set and right\_pwms\_set

### **Author(s)**

Lando Andrey

---

Alphabet	<i>built alphabet</i>
----------	-----------------------

---

**Description**

builds an object of class Alphabet from the given set of symbols and colors

**Usage**

```
Alphabet(chars, cols, supportReverseComplement)
```

**Arguments**

chars	set of symbols
cols	set of colors; one for each symbol
supportReverseComplement	boolean whether the alphabet supports reverse complementation (like DNA/RNA) or not (like ASN)

**Value**

the Alphabet object

**Author(s)**

Martin Nettling

**Examples**

```
DNA = Alphabet(c("A", "C", "G", "T"), c("green4", "blue", "orange", "red"), TRUE)
```

---

ASN	<i>ASN alphabet</i>
-----	---------------------

---

**Description**

the amino acid alphabet (20 symbols), i.e. A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y

**Usage**

```
ASN
```

**Format**

An object of class Alphabet of length 4.

**Author(s)**

Martin Nettling

**Examples**

ASN

---

baseDistributionPwm     *Generates a PWM*

---

**Description**

Generates a PWM consisting of only the uniform distribution or the given base\_distribution (if defined).

**Usage**

```
baseDistributionPwm(pwm_length, alphabet_length,  
                  base_distribution = NULL)
```

**Arguments**

pwm\_length        the number of positions  
alphabet\_length        the alphabet size  
base\_distribution     optional base distribution for each PWM position

**Value**

a PWM

---

calculatePvalue        *p-value that two PWM-positions are from the same distribution*

---

**Description**

Calculates the p-value for the null-hypothesis that two given probability vectors p1, p2 calculated from n1/n2 observations arise from the same distribution

**Usage**

```
calculatePvalue(p1, p2, n1, n2, stackHeight = shannonDivergence,  
                  numberOfPermutations = 100, plotGammaDistributionFit = FALSE)
```

**Arguments**

**p1** first probability vector with one probability for each symbol of the alphabet  
**p2** second probability vector with one probability for each symbol of the alphabet  
**n1** number of observations for the calculation of p1  
**n2** number of observations for the calculation of p2  
**stackHeight** function for the calculation of a divergence measure for two probability vectors  
**numberOfPermutations** the number of permutations to perform for the calculation of stackHeights  
**plotGammaDistributionFit** if TRUE the fit of a gamma distribution to the sampled stackHeights is plotted

**Value**

a numeric p-value

**Author(s)**

Hendrik Treutler

**Examples**

```

p1 <- c(0.2, 0.3, 0.1, 0.4)
p2 <- c(0.2, 0.1, 0.3, 0.4)
n1 <- 100
n2 <- 200
numberOfPermutations = 100
plotGammaDistributionFit = TRUE

pValue <- calculatePvalue(p1 = p1, p2 = p2, n1 = n1, n2 = n2, stackHeight = shannonDivergence, numberOfPermutations

```

---

createDiffLogoObject *DiffLogo object*

---

**Description**

Creates a DiffLogo object

**Usage**

```

createDiffLogoObject(pwm1, pwm2, stackHeight = shannonDivergence,
  baseDistribution = normalizedDifferenceOfProbabilities,
  alphabet = DNA, align_pwm = FALSE,
  unaligned_penalty = divergencePenaltyForUnaligned,
  try_reverse_complement = TRUE, base_distribution = NULL,
  length_normalization = FALSE, unaligned_from_left = 0,
  unaligned_from_right = 0)

```

**Arguments**

pwm1	representation of the first position weight matrix (PWM) of type pwm, data.frame, or matrix
pwm2	representation of the second position weight matrix (PWM) of type pwm, data.frame, or matrix
stackHeight	function for the height of a stack at position i
baseDistribution	function for the heights of the individual bases
alphabet	of type Alphabet
align_pwm	if True, will align and extend pwms.
unaligned_penalty	is a function for localPwmAlignment.
try_reverse_complement	if True, alignment will try reverse complement pwms
base_distribution	is a vector of length nrow(pwm) that is added to unaligned columns of pwms for comparing. If NULL, uniform distribution is used
length_normalization	If true, will minimize the average divergence between PWMs. Otherwise will minimize the sum of divergences between positions. In both cases unaligned positions are compared to base_distribution and are counted when computing the alignment length.
unaligned_from_left	the number of unaligned positions on the left
unaligned_from_right	the number of unaligned positions on the right

**Value**

DiffLogo object

**Author(s)**

Martin Nettling

**Examples**

```

motif_folder= "extdata/pwm"
motif_names = c("HepG2","MCF7","HUVEC","ProgFib")
motifs = list()
for (name in motif_names) {
  fileName = paste(motif_folder,"/",name,".pwm",sep="")
  file = system.file(fileName, package = "DiffLogo")
  motifs[[name]] = getPwmFromPwmFile(file)
}

pwm1 = motifs[[motif_names[[1]]]]

```

```

pwm2 = motifs[[motif_names[[2]]]]

diffLogoObj = createDiffLogoObject(pwm1 = pwm1, pwm2 = pwm2)
diffLogo(diffLogoObj)

```

---

differenceOfICs	<i>normalized information content differences</i>
-----------------	---

---

### Description

information content differences normalized by the sum of absolute information content differences for the given pair of probability vectors

### Usage

```
differenceOfICs(p1, p2)
```

### Arguments

p1	probability vector representing the first symbol distribution
p2	probability vector representing the second symbol distribution

### Value

a vector with one result for each symbol

### Author(s)

Martin Nettling

### Examples

```

motif_folder= "extdata/pwm"
motif_names = c("HepG2","MCF7","HUVEC","ProgFib")
motifs = list()
for (name in motif_names) {
  fileName = paste(motif_folder,"/",name,".pwm",sep="")
  file = system.file(fileName, package = "DiffLogo")
  motifs[[name]] = getPwmFromPwmFile(file)
}

pwm1 = motifs[[motif_names[[1]]]]
pwm2 = motifs[[motif_names[[2]]]]

diffLogoFromPwm(pwm1 = pwm1, pwm2 = pwm2, baseDistribution = differenceOfICs)

```



---

`diffLogo`*Draw DiffLogo*

---

**Description**

Draws the difference of two sequence logos.

**Usage**

```
diffLogo(diffLogoObj, ymin = 0, ymax = 0, sparse = FALSE,  
diffLogoConfiguration = list())
```

**Arguments**

<code>diffLogoObj</code>	a DiffLogoObject created by the function <code>createDiffLogoObject</code>
<code>ymin</code>	minimum value on the y-axis
<code>ymax</code>	maximum value on the y-axis
<code>sparse</code>	if TRUE margins are reduced and tickmarks are removed from the logo
<code>diffLogoConfiguration</code>	list of configuration parameters (see function <code>diffLogoTableConfiguration(...)</code> )

**Value**

none (draws difference logo)

**Author(s)**

Martin Nettling

**Examples**

```
motif_folder= "extdata/pwm"  
motif_names = c("HepG2", "MCF7", "HUVEC", "ProgFib")  
motifs = list()  
for (name in motif_names) {  
  fileName = paste(motif_folder, "/", name, ".pwm", sep="")  
  file = system.file(fileName, package = "DiffLogo")  
  motifs[[name]] = getPwmFromPwmFile(file)  
}  
  
pwm1 = motifs[[motif_names[[1]]]]  
pwm2 = motifs[[motif_names[[2]]]]  
  
diffLogoObj = createDiffLogoObject(pwm1 = pwm1, pwm2 = pwm2)  
diffLogo(diffLogoObj)
```

---

diffLogoFromPwm      *Draw DiffLogo from PWM*

---

### Description

Draws the difference of two sequence logos.

### Usage

```
diffLogoFromPwm(pwm1, pwm2, ymin = 0, ymax = 0,
  stackHeight = shannonDivergence,
  baseDistribution = normalizedDifferenceOfProbabilities,
  sparse = FALSE, alphabet = DNA, align_pwm = FALSE,
  unaligned_penalty = divergencePenaltyForUnaligned,
  try_reverse_complement = TRUE, base_distribution = NULL,
  length_normalization = FALSE)
```

### Arguments

pwm1	representation of the first position weight matrix (PWM) of type pwm, data.frame, or matrix
pwm2	representation of the second position weight matrix (PWM) of type pwm, data.frame, or matrix
ymin	minimum value on the y-axis
ymax	maximum value on the y-axis
stackHeight	function for the height of a stack at position i
baseDistribution	function for the heights of the individual bases
sparse	if TRUE margins are reduced and tickmarks are removed from the logo
alphabet	of type Alphabet
align_pwm	if true, DiffLogo will align pwms before plotting
unaligned_penalty	is a function for localPwmAlignment.
try_reverse_complement	if True, alignment will try reverse complement pwms
base_distribution	is a vector of length nrow(pwm) that is added to unaligned columns of pwms for comparing. If NULL, uniform distribution is used
length_normalization	If true, will minimize the average divergence between PWMs. Otherwise will minimize the sum of divergences between positions. In both cases unaligned positions are compared to base_distribution and are counted when computing the alignment length.

**Value**

none (draws difference logo)

**Author(s)**

Martin Nettling

**Examples**

```

motif_folder= "extdata/pwm"
motif_names = c("HepG2","MCF7","HUVEC","ProgFib")
motifs = list()
for (name in motif_names) {
  fileName = paste(motif_folder,"/",name,".pwm",sep="")
  file = system.file(fileName, package = "DiffLogo")
  motifs[[name]] = getPwmFromPwmFile(file)
}

pwm1 = motifs[[motif_names[[1]]]]
pwm2 = motifs[[motif_names[[2]]]]

diffLogoFromPwm(pwm1 = pwm1, pwm2 = pwm2)

```

---

diffLogoTable

*Draw DiffLogo-table*


---

**Description**

Draws a table of DiffLogos.

**Usage**

```
diffLogoTable(PWMs, sampleSizes = NULL, alphabet = DNA,
  configuration = list(), ...)
```

**Arguments**

PWMs	a list/vector of position weight matrices (PWMs) each of type pwm, data.frame, or matrix
sampleSizes	the number of sequences behind each PWM
alphabet	the alphabet of the given PWMs
configuration	list of (probably part of) of configuration options. See diffLogoTableConfiguration.
...	set of parameters passed to the function 'axis' for plotting

**Value**

none (draws table of difference logos)

**Author(s)**

Martin Nettling

**Examples**

```

motif_folder= "extdata/pwm"
motif_names = c("HepG2","MCF7","HUVEC","ProgFib")
motifs = list()
for (name in motif_names) {
  fileName = paste(motif_folder,"/",name,".pwm",sep="")
  file = system.file(fileName, package = "DiffLogo")
  motifs[[name]] = getPwmFromPwmFile(file)
}

diffLogoTable(motifs)

```

---

diffLogoTableConfiguration

*Configuration object for diffLogoTable*

---

**Description**

Default configuration list for diffLogoTable

**Usage**

```

diffLogoTableConfiguration(alphabet, stackHeight = shannonDivergence,
  baseDistribution = normalizedDifferenceOfProbabilities,
  uniformYaxis = TRUE, sparse = TRUE, showSequenceLogosTop = TRUE,
  enableClustering = TRUE, treeHeight = 0.5, margin = 0.02,
  ratio = 1, align_pwm = FALSE, multiple_align_pwm = TRUE,
  unaligned_penalty = divergencePenaltyForUnaligned,
  try_reverse_complement = TRUE, length_normalization = FALSE,
  numberOfPermutations = 100)

```

**Arguments**

alphabet	used alphabet of type Alphabet
stackHeight	function for the height of a stack at position i
baseDistribution	function for the heights of the individual bases
uniformYaxis	if TRUE each DiffLogo is plotted with the same scaling of the y-axis
sparse	if TRUE margins are reduced and tickmarks are removed from the logo
showSequenceLogosTop	if TRUE the classical sequence logos are drawn above each column of the table

<code>enableClustering</code>	if TRUE the motifs are reordered, so that similar motifs have a small vertical and horizontal distance in the table
<code>treeHeight</code>	the height of the plotted cluster tree above the columns of the table; set equal to zero to omit the cluster tree
<code>margin</code>	the space reserved for labels
<code>ratio</code>	the ratio of the plot; this is needed to determine the margin sizes correctly
<code>align_pwm</code>	if True, will align and extend pwms in each cell of <code>diffLogoTable</code> independently.
<code>multiple_align_pwm</code>	if True, will align and extend pwms in the <code>diffLogoTable</code> jointly.
<code>unaligned_penalty</code>	is a function for <code>localPwmAlignment</code> .
<code>try_reverse_complement</code>	if True, alignment will try reverse complement pwms
<code>length_normalization</code>	if True, divergence between pwms is divided by length of pwms.
<code>numberOfPermutations</code>	number of permutations for the permutation test for the calculation of p-values

**Value**

list of parameters

**Author(s)**

Lando Andrey

**Examples**

```
diffLogoTableConfiguration(DNA)
```

---

DNA

*DNA alphabet*

---

**Description**

the DNA alphabet, i.e. A, C, G, T

**Usage**

DNA

**Format**

An object of class `Alphabet` of length 4.

**Author(s)**

Martin Nettling

**Examples**

DNA

---

drawDiffLogoTable      *Draws a table of DiffLogos*

---

**Description**

Draws a table of DiffLogos.

**Usage**

```
drawDiffLogoTable(diffLogoTableObj, ...)
```

**Arguments**

```
diffLogoTableObj      the diffLogoTable-Object created by function prepareDiffLogoTable(...)
...                    optional parameters for function axis
```

**Value**

none (draws difference logo)

**Examples**

```
motif_folder= "extdata/pwm"
motif_names = c("HepG2", "MCF7", "HUVEC", "ProgFib")
motifs = list()
for (name in motif_names) {
  fileName = paste(motif_folder, "/", name, ".pwm", sep="")
  file = system.file(fileName, package = "DiffLogo")
  motifs[[name]] = getPwmFromPwmFile(file)
}

diffLogoTableObj = prepareDiffLogoTable(motifs)
drawDiffLogoTable(diffLogoTableObj)
```

---

`enrichDiffLogoObjectWithPvalues`*Enriches a difflogo object with p-values*

---

**Description**

Enriches a difflogo object with p-values which quantifies the probability that two PWM-positions are from the same distribution

**Usage**

```
enrichDiffLogoObjectWithPvalues(diffLogoObj, n1, n2,  
  stackHeight = shannonDivergence, numberOfPermutations = 100)
```

**Arguments**

<code>diffLogoObj</code>	matrix of difflogo objects
<code>n1</code>	the number of sequences behind the first pwm behind the given difflogo object
<code>n2</code>	the number of sequences behind the second pwm behind the given difflogo object
<code>stackHeight</code>	function for the calculation of a divergence measure for two probability vectors
<code>numberOfPermutations</code>	the number of permutations to perform for the calculation of stackHeights

**Value**

enriched difflogo object

**Examples**

```
motif_folder= "extdata/pwm"  
motif_names = c("HepG2", "MCF7", "HUVEC", "ProgFib")  
motifs = list()  
for (name in motif_names) {  
  fileName = paste(motif_folder, "/", name, ".pwm", sep="")  
  file = system.file(fileName, package = "DiffLogo")  
  motifs[[name]] = getPwmFromPwmFile(file)  
}  
  
pwm1 = motifs[[motif_names[[1]]]]  
pwm2 = motifs[[motif_names[[2]]]]  
n1 <- 100  
n2 <- 100  
diffLogoObj = createDiffLogoObject(pwm1 = pwm1, pwm2 = pwm2)  
diffLogoObj = enrichDiffLogoObjectWithPvalues(diffLogoObj, n1, n2)
```

---

`enrichDiffLogoTableWithPvalues`*Enriches a matrix of difflogo objects with p-values*

---

**Description**

Enriches a matrix of difflogo objects with p-values which quantifies the probability that two PWM-positions are from the same distribution

**Usage**

```
enrichDiffLogoTableWithPvalues(diffLogoObjMatrix, sampleSizes,  
  stackHeight = shannonDivergence, numberOfPermutations = 100)
```

**Arguments**

<code>diffLogoObjMatrix</code>	matrix of difflogo objects
<code>sampleSizes</code>	number of sequences behind the pwms behind the given difflogo objects
<code>stackHeight</code>	function for the calculation of a divergence measure for two probability vectors
<code>numberOfPermutations</code>	the number of permutations to perform for the calculation of stackHeights

**Value**

matrix of difflogo objects enriched with p-values

**Author(s)**

Martin Nettling

**Examples**

```
motif_folder= "extdata/pwm"  
motif_names = c("HepG2", "MCF7", "HUVEC", "ProgFib")  
motifs = list()  
for (name in motif_names) {  
  fileName = paste(motif_folder, "/", name, ".pwm", sep="")  
  file = system.file(fileName, package = "DiffLogo")  
  motifs[[name]] = getPwmFromPwmFile(file)  
}  
sampleSizes <- c(100, 150, 200, 250)  
names(sampleSizes) <- motif_names  
  
diffLogoTableObj = prepareDiffLogoTable(motifs);  
diffLogoTableObj$diffLogoObjMatrix = enrichDiffLogoTableWithPvalues(diffLogoTableObj$diffLogoObjMatrix, sampleS
```



---

`extendPwmsFromAlignmentVector`*Extend pwms with respect to alignment*

---

**Description**

Extends pwms by adding `base_distribution` to both sides, so that they keep aligned, but have equal length.

**Usage**

```
extendPwmsFromAlignmentVector(pwms, alignment_vector,  
  base_distribution = NULL)
```

**Arguments**

`pwms` is a list of matrixes

`alignment_vector` is a list of shifts (`$shift`) and orientations (`$direction`)

`base_distribution` is a vector of length `nrow(pwm)` that is added to unaligned columns of pwms for comparing. If `NULL`, uniform distribution is used

**Value**

extended pwms

**Author(s)**

Lando Andrey

**Examples**

```
file1 = system.file("extdata/homer/Max.motif", package = "DiffLogo")  
file2 = system.file("extdata/homer/c-Myc.motif", package = "DiffLogo")  
pwm1 = getPwmFromFile(file1)  
pwm2 = getPwmFromFile(file2)  
  
pwms <- list(pwm1, pwm2)  
multiple_pwms_alignment = multipleLocalPwmsAlignment(pwms)  
aligned_pwms = extendPwmsFromAlignmentVector(pwms, multiple_pwms_alignment$alignment$vector)
```

FULL\_ALPHABET      *Complete character alphabet*

---

**Description**

the alphabet of all 26 characters

**Usage**

FULL\_ALPHABET

**Format**

An object of class Alphabet of length 4.

**Author(s)**

Hendrik Treutler

**Examples**

FULL\_ALPHABET

---

getAlphabetFromCharacters  
*returns the alphabet which fits to the given characters*

---

**Description**

returns the alphabet which fits to the given characters

**Usage**

getAlphabetFromCharacters(characters)

**Arguments**

characters      a character vector of characters

**Value**

an alphabet of type Alphabet

**Examples**

```
alphabet = getAlphabetFromSequences(c("A", "A", "C", "C", "G", "G", "T", "T"))
```

---

getAlphabetFromSequences  
*returns the alphabet which fits to the given sequences*

---

**Description**

returns the alphabet which fits to the given sequences

**Usage**

```
getAlphabetFromSequences(sequences)
```

**Arguments**

sequences      a character vector of sequences

**Value**

an alphabet of type Alphabet

**Examples**

```
alphabet = getAlphabetFromSequences("AACCGGTT")
```

---

getPwmFromAlignment      *Create PWM from alignment*

---

**Description**

Creates a matrix-representation of a PWM from a set of sequences

**Usage**

```
getPwmFromAlignment(alignment, alphabet = NULL, pseudoCount = 0)
```

**Arguments**

alignment      a vector or list of sequences each with equal length  
alphabet      of type Alphabet  
pseudoCount      the number of pseudo-observations for each character in the alphabet

**Value**

PWM as matrix

**Author(s)**

Hendrik Treutler

**Examples**

```
motif_folder= "extdata/alignments"  
motif_name = "calamodulin_1"  
fileName = paste(motif_folder,"/",motif_name,".txt",sep="")  
file = system.file(fileName, package = "DiffLogo")  
motif = getPwmFromAlignment(readLines(file), ASN, 1)  
seqLogo(pwm = motif, alphabet=ASN)
```

---

getPwmFromAlignmentFile

*generates a pwm from an alignment file*

---

**Description**

generates a pwm from an alignment file

**Usage**

```
getPwmFromAlignmentFile(filename, alphabet = NULL)
```

**Arguments**

filename	the alignment file
alphabet	the desired alphabet of type Alphabet

**Value**

a pwm

**Examples**

```
fileName = "extdata/alignments/calamodulin_1.txt"  
file = system.file(fileName, package = "DiffLogo")  
pwm = getPwmFromAlignmentFile(file)
```

---

getPwmFromFastaFile     *generates a pwm from a FASTA file*

---

**Description**

generates a pwm from a FASTA file

**Usage**

```
getPwmFromFastaFile(filename, alphabet = NULL)
```

**Arguments**

filename	the FASTA file
alphabet	the desired alphabet of type Alphabet

**Value**

a pwm

**Examples**

```
fileName = "extdata/alignments/F-box_bacteria.seq.fa"  
file = system.file(fileName, package = "DiffLogo")  
pwm = getPwmFromFastaFile(file)
```

---

getPwmFromFile     *generates a pwm from a file of different formats*

---

**Description**

Generates a pwm from a file of different formats. Supported formats are FASTA files (.fa, .fasta), alignment files (.txt, .text, .al, .alignment), PWM files (.pwm), JASPAR / Position Frequency Matrix files (.pfm), and homer files (.motif).

**Usage**

```
getPwmFromFile(filename)
```

**Arguments**

filename	the file
----------	----------

**Value**

a pwm

**Examples**

```
fileName = "extdata/pwm/H1-hESC.pwm"  
file = system.file(fileName, package = "DiffLogo")  
pwm = getPwmFromFile(file)
```

---

getPwmFromHomerFile     *generates a pwm from a homer file*

---

**Description**

generates a pwm from a homer file

**Usage**

```
getPwmFromHomerFile(filename)
```

**Arguments**

filename            the homer file

**Value**

a pwm

**Examples**

```
fileName = "extdata/homer/CTCF_Zf_CD4.motif"  
file = system.file(fileName, package = "DiffLogo")  
pwm = getPwmFromHomerFile(file)
```

---

getPwmFromPfmOrJasparFile  
                          *generates a pwm from a jaspar file*

---

**Description**

generates a pwm from a jaspar file

**Usage**

```
getPwmFromPfmOrJasparFile(filename)
```

**Arguments**

filename            the jaspar file

**Value**

a pwm

**Examples**

```
fileName = "extdata/pfm/ctcf_jaspar.pfm"  
file = system.file(fileName, package = "DiffLogo")  
pwm = getPwmFromPfmOrJasparFile(file)
```

---

`getPwmFromPwmFile`      *generates a pwm from a pwm file*

---

**Description**

generates a pwm from a pwm file

**Usage**

```
getPwmFromPwmFile(filename)
```

**Arguments**

filename      the pwm file

**Value**

a pwm

**Examples**

```
fileName = "extdata/pwm/H1-hESC.pwm"  
file = system.file(fileName, package = "DiffLogo")  
pwm = getPwmFromPwmFile(file)
```

---

`getSequencesFromAlignmentFile`  
*extracts the sequences from an alignment file*

---

**Description**

extracts the sequences from an alignment file

**Usage**

```
getSequencesFromAlignmentFile(filename)
```

**Arguments**

filename            the alignment file

**Value**

a vector of sequences

**Examples**

```
fileName = "extdata/alignments/calamodulin_1.txt"
file = system.file(fileName, package = "DiffLogo")
sequences = getSequencesFromAlignmentFile(file)
```

---

getSequencesFromFastaFile

*extracts the sequences from a FASTA file*

---

**Description**

extracts the sequences from a FASTA file

**Usage**

```
getSequencesFromFastaFile(filename)
```

**Arguments**

filename            the FASTA file

**Value**

a vector of sequences

**Examples**

```
fileName = "extdata/alignments/F-box_bacteria.seq.fa"
file = system.file(fileName, package = "DiffLogo")
sequences = getSequencesFromFastaFile(file)
```



---

informationContent	<i>information content</i>
--------------------	----------------------------

---

**Description**

the information content for the given probability vector

**Usage**

```
informationContent(p)
```

**Arguments**

`p` probability vector representing the symbol distribution

**Value**

an object consisting of height a ylab

**Author(s)**

Martin Nettling

**Examples**

```
motif_folder= "extdata/pwm"  
motif_name = "HepG2"  
fileName = paste(motif_folder,"/",motif_name,".pwm",sep="")  
file = system.file(fileName, package = "DiffLogo")  
motif = getPwmFromPwmFile(file)  
seqLogo(pwm = motif, stackHeight = informationContent)
```

---

localPwmAlignment	<i>Align pwms</i>
-------------------	-------------------

---

**Description**

Finds best local alignment for two PWMs.

**Usage**

```
localPwmAlignment(pwm_left, pwm_right, divergence = shannonDivergence,  
unaligned_penalty = divergencePenaltyForUnaligned,  
try_reverse_complement = TRUE, base_distribution = NULL,  
length_normalization = FALSE)
```

**Arguments**

pwm_left	first PWM, a matrix of type matrix
pwm_right	first PWM, a matrix of type matrix
divergence	is a measure of difference between two pwm columns. Smaller is more similar. If you want to use non-uniform background distribution, provide your own function.
unaligned_penalty	distance for unaligned columns at edges of matrixes. See divergencePenalty-ForUnaligned as an example for providing your own function
try_reverse_complement	If false the alignment will not be performed on reverse complements. If true, the input pwms should have column order of ACTG/ACGU.
base_distribution	is a vector of length nrow(pwm) that is added to unaligned columns of pwms for comparing. If NULL, uniform distribution is used
length_normalization	If true, will minimize the average divergence between PWMs. Otherwise will minimize the sum of divergences between positions. In both cases unaligned positions are compared to base_distribution and are counted when computing the alignment length.

**Value**

list of length two containing the alignment and the divergence

**Author(s)**

Lando Andrey

---

lossOfAbsICDifferences

*the change of information content*

---

**Description**

the change of information content for the given probability vectors

**Usage**

```
lossOfAbsICDifferences(p1, p2)
```

**Arguments**

p1	probability vector representing the first symbol distribution
p2	probability vector representing the second symbol distribution

**Value**

an object consisting of height and ylab

**Author(s)**

Martin Nettling

**Examples**

```

motif_folder= "extdata/pwm"
motif_names = c("HepG2","MCF7","HUVEC","ProgFib")
motifs = list()
for (name in motif_names) {
  fileName = paste(motif_folder,"/",name,".pwm",sep="")
  file = system.file(fileName, package = "DiffLogo")
  motifs[[name]] = getPwmFromPwmFile(file)
}

pwm1 = motifs[[motif_names[[1]]]]
pwm2 = motifs[[motif_names[[2]]]]

diffLogoFromPwm(pwm1 = pwm1, pwm2 = pwm2, stackHeight = lossOfAbsICDifferences)

```

---

multipleLocalPwmsAlignment

*Multiple PWMs alignment*

---

**Description**

Creates a multiple alignment of pwms

**Usage**

```

multipleLocalPwmsAlignment(pwms, divergence = shannonDivergence,
  unaligned_penalty = divergencePenaltyForUnaligned,
  try_reverse_complement = TRUE, base_distribution = NULL,
  length_normalization = FALSE)

```

**Arguments**

pwms	list of pwms
divergence	Divergence measure.
unaligned_penalty	is a function for localPwmAlignment.
try_reverse_complement	if True, alignment will try reverse complement pwms

base\_distribution

is a vector of length `nrow(pwm)` that is added to unaligned columns of pwms for comparing. If NULL, uniform distribution is used

length\_normalization

If true, will minimize the average divergence between PWMs. Otherwise will minimize the sum of divergences between positions. In both cases unaligned positions are compared to `base_distribution` and are counted when computing the alignment length.

### Value

list

### Author(s)

Lando Andrey

### Examples

```
file1 = system.file("extdata/homer/Max.motif", package = "DiffLogo")
file2 = system.file("extdata/homer/c-Myc.motif", package = "DiffLogo")
pwm1 = getPwmFromFile(file1)
pwm2 = getPwmFromFile(file2)
```

```
multiple_pwms_alignment = multipleLocalPwmsAlignment(list(pwm1, pwm2))
```

---

normalizedDifferenceOfProbabilities

*normalized probability differences*

---

### Description

probability differences normalized by the sum of absolute probability differences for the given pair of probability vectors

### Usage

```
normalizedDifferenceOfProbabilities(p1, p2)
```

### Arguments

p1 probability vector representing the first symbol distribution  
p2 probability vector representing the second symbol distribution

### Value

a vector with one result for each symbol

**Author(s)**

Martin Nettling

**Examples**

```
motif_folder= "extdata/pwm"
motif_names = c("HepG2","MCF7","HUVEC","ProgFib")
motifs = list()
for (name in motif_names) {
  fileName = paste(motif_folder,"/",name,".pwm",sep="")
  file = system.file(fileName, package = "DiffLogo")
  motifs[[name]] = getPwmFromPwmFile(file)
}

pwm1 = motifs[[motif_names[[1]]]]
pwm2 = motifs[[motif_names[[2]]]]

diffLogoFromPwm(pwm1 = pwm1, pwm2 = pwm2, baseDistribution = normalizedDifferenceOfProbabilities)
```

---

`normalizePWM`*normalizes the given pwm*

---

**Description**

normalizes the given pwm to column-sums of 1.0

**Usage**`normalizePWM(pwm)`**Arguments**`pwm`            a pwm**Value**

a normalized pwm

**Examples**

```
pwm = matrix(1:40, nrow = 4, dimnames = list(c("A","C","G","T"), 1:10))
pwm = normalizePWM(pwm)
```

---

prepareDiffLogoTable *Prepare a table of difflogo objects*

---

### Description

Prepares a DiffLogoTable and generates an object that contains the hierarchical clustering and a matrix of prepared difference logos.

### Usage

```
prepareDiffLogoTable(PWMs, alphabet = DNA, configuration = list())
```

### Arguments

**PWMs** a list/vector of position weight matrices (PWMs) each of type pwm, data.frame, or matrix

**alphabet** the alphabet of the given PWMs

**configuration** list of (probably part of) of configuration options. See diffLogoTableConfiguration.

### Value

matrix of difference logos

### Author(s)

Martin Nettling

### Examples

```
motif_folder= "extdata/pwm"
motif_names = c("HepG2", "MCF7", "HUVEC", "ProgFib")
motifs = list()
for (name in motif_names) {
  fileName = paste(motif_folder, "/", name, ".pwm", sep="")
  file = system.file(fileName, package = "DiffLogo")
  motifs[[name]] = getPwmFromPwmFile(file)
}
sampleSizes <- c(100, 150, 200, 250)

diffLogoTableObj = prepareDiffLogoTable(motifs);
```

---

probabilities            *probabilities*

---

**Description**

the given probabilities

**Usage**

probabilities(p)

**Arguments**

p                      probability vector representing the symbol distribution

**Value**

the given vector

**Author(s)**

Martin Nettling

**Examples**

```
motif_folder= "extdata/pwm"
motif_name = "HepG2"
fileName = paste(motif_folder,"/",motif_name,".pwm",sep="")
file = system.file(fileName, package = "DiffLogo")
motif = getPwmFromPwmFile(file)
seqLogo(pwm = motif, baseDistribution = probabilities)
```

---

pwmDivergence            *PWM divergence*

---

**Description**

Counts PWM divergence as sum of divergencies of their columns.

**Usage**

pwmDivergence(pwm\_left, pwm\_right, divergence = shannonDivergence)

**Arguments**

pwm_left	is a PWM representation in type of matrix
pwm_right	is a PWM representation in type of matrix. The result is symmetric on pwm_left and pwm_right
divergence	is a Divergence function on columns.

**Value**

float - sum of divergences

---

pwmsDistanceMatrix     *Multiple PWMs alignment*

---

**Description**

Creates a distance matrix for pwms

**Usage**

```
pwmsDistanceMatrix(pwms, diagonal_value = 0,
  bottom_default_value = NULL, divergence = shannonDivergence,
  unaligned_penalty = divergencePenaltyForUnaligned,
  try_reverse_complement = TRUE, base_distribution = NULL,
  length_normalization = FALSE)
```

**Arguments**

pwms	list of pwms
diagonal_value	value to put on diagonal.
bottom_default_value	value to put on bottom triangle. Set to NULL to get symmetric distance matrix.
divergence	divergence measure.
unaligned_penalty	is a function for localPwmAlignment.
try_reverse_complement	if True, alignment will try reverse complement pwms
base_distribution	is a vector of length nrow(pwm) that is added to unaligned columns of pwms for comparing. If NULL, uniform distribution is used
length_normalization	is a vector of length nrow(pwm) that is added to unaligned columns of pwms for comparing. If NULL, uniform distribution is used

**Value**

list



**Author(s)**

Lando Andrey

---

`reverseAlignmentVector`  
*Reverse for alignment vector*

---

**Description**

Returns alignment vector as if all pwm were reverted.

**Usage**

```
reverseAlignmentVector(alignment_vector, pwms)
```

**Arguments**

`alignment_vector` list of list which \$shift and \$orientation  
`pwms` list of matrixes.

**Value**

list - reversed alignment vector

**Author(s)**

Lando Andrey

---

`RNA` *RNA alphabet*

---

**Description**

the RNA alphabet, i.e. A, C, G, U

**Usage**

```
RNA
```

**Format**

An object of class Alphabet of length 4.

**Author(s)**

Martin Nettling

**Examples**

RNA

---

seqLogo

*Draw sequence logo*

---

**Description**

Draws the classic sequence logo.

**Usage**

```
seqLogo(pwm, sparse = FALSE, drawLines = 0.5,  
        stackHeight = informationContent, baseDistribution = probabilities,  
        alphabet = DNA, main = NULL)
```

**Arguments**

pwm	representation of a position weight matrix (PWM) of type pwm, data.frame, or matrix
sparse	if TRUE margins are reduced and tickmarks are removed from the logo
drawLines	distance between background lines
stackHeight	function for the height of a stack at position i
baseDistribution	function for the heights of the individual bases
alphabet	of type Alphabet
main	the main title for the plot

**Value**

none (draws sequence logo)

**Author(s)**

Martin Nettling

**Examples**

```
motif_folder= "extdata/pwm"  
motif_name = "HepG2"  
fileName = paste(motif_folder,"/",motif_name,".pwm",sep="")  
file = system.file(fileName, package = "DiffLogo")  
motif = getPwmFromPwmFile(file)  
seqLogo(pwm = motif)
```

---

shannonDivergence	<i>shannon divergence</i>
-------------------	---------------------------

---

**Description**

the shannon divergence for the given pair of probability vectors

**Usage**

```
shannonDivergence(p1, p2)
```

**Arguments**

p1	probability vector representing the first symbol distribution
p2	probability vector representing the second symbol distribution

**Value**

an object consisting of height and ylab

**Author(s)**

Martin Nettling

**Examples**

```
motif_folder= "extdata/pwm"
motif_names = c("HepG2", "MCF7", "HUVEC", "ProgFib")
motifs = list()
for (name in motif_names) {
  fileName = paste(motif_folder, "/", name, ".pwm", sep="")
  file = system.file(fileName, package = "DiffLogo")
  motifs[[name]] = getPwmFromPwmFile(file)
}

pwm1 = motifs[[motif_names[[1]]]]
pwm2 = motifs[[motif_names[[2]]]]

diffLogoFromPwm(pwm1 = pwm1, pwm2 = pwm2, stackHeight = shannonDivergence)
```

---

sumOfAbsICDifferences *sum of absolute information content differences*

---

### Description

the sum of absolute information content differences for the given pair of probability vectors

### Usage

```
sumOfAbsICDifferences(p1, p2)
```

### Arguments

p1                    probability vector representing the first symbol distribution  
p2                    probability vector representing the second symbol distribution

### Value

an object consisting of height and ylab

### Author(s)

Martin Nettling

### Examples

```
motif_folder= "extdata/pwm"  
motif_names = c("HepG2", "MCF7", "HUVEC", "ProgFib")  
motifs = list()  
for (name in motif_names) {  
  fileName = paste(motif_folder, "/", name, ".pwm", sep="")  
  file = system.file(fileName, package = "DiffLogo")  
  motifs[[name]] = getPwmFromPwmFile(file)  
}  
  
pwm1 = motifs[[motif_names[[1]]]]  
pwm2 = motifs[[motif_names[[2]]]]  
  
diffLogoFromPwm(pwm1 = pwm1, pwm2 = pwm2, stackHeight = sumOfAbsICDifferences)
```

---

sumOfAbsProbabilityDifferences  
*sum of absolute probability differences*

---

**Description**

the sum of absolute probability differences for the given pair of probability vectors

**Usage**

```
sumOfAbsProbabilityDifferences(p1, p2)
```

**Arguments**

p1                    probability vector representing the first symbol distribution  
p2                    probability vector representing the second symbol distribution

**Value**

an object consisting of height and ylab

**Author(s)**

Martin Nettling

**Examples**

```
motif_folder= "extdata/pwm"
motif_names = c("HepG2","MCF7","HUVEC","ProgFib")
motifs = list()
for (name in motif_names) {
  fileName = paste(motif_folder,"/",name,".pwm",sep="")
  file = system.file(fileName, package = "DiffLogo")
  motifs[[name]] = getPwmFromPwmFile(file)
}

pwm1 = motifs[[motif_names[[1]]]]
pwm2 = motifs[[motif_names[[2]]]]

diffLogoFromPwm(pwm1 = pwm1, pwm2 = pwm2, stackHeight = sumOfAbsProbabilityDifferences)
```

---

sumProbabilities	<i>sum of probabilities, i.e. 1.0</i>
------------------	---------------------------------------

---

**Description**

the sum of probabilities for the given probability vector, i.e. 1.0

**Usage**

```
sumProbabilities(p)
```

**Arguments**

p                    probability vector representing the symbol distribution

**Value**

an object consisting of height and ylab

**Author(s)**

Martin Nettling

**Examples**

```
motif_folder= "extdata/pwm"  
motif_name = "HepG2"  
fileName = paste(motif_folder, "/", motif_name, ".pwm", sep="")  
file = system.file(fileName, package = "DiffLogo")  
motif = getPwmFromPwmFile(file)  
seqLogo(pwm = motif, stackHeight = sumProbabilities)
```

---

switchDirection	<i>Switches between 'forward' and 'reverse'</i>
-----------------	---

---

**Description**

Switches between 'forward' and 'reverse'

**Usage**

```
switchDirection(direction)
```

**Arguments**

direction            either 'forward' or 'reverse'

**Value**

either 'reverse' or 'forward'

---

twoSetsAveragePwmDivergenceFromAlignmentVector

*Average divergence between two sets.*

---

**Description**

Computes average pwm divergence from alignment vector for two datasets. This equals to average divergence between all pairs where one pwm comes from left set, and other comes from right

**Usage**

```
twoSetsAveragePwmDivergenceFromAlignmentVector(left_pwms_list,  
left_pwms_alignment, right_pwms_list, right_pwms_alignment,  
divergence = shannonDivergence)
```

**Arguments**

left\_pwms\_list is a list of matrixes  
left\_pwms\_alignment is a list of shifts (\$shift) and orientations (\$direction)  
right\_pwms\_list is a list of matrixes  
right\_pwms\_alignment is a list of shifts (\$shift) and orientations (\$direction)  
divergence divergence measure.

**Value**

float

**Author(s)**

Lando Andrey

# Index

## \* datasets

- ASN, 4
- DNA, 13
- FULL\_ALPHABET, 18
- RNA, 33

alignPwmSets, 3

Alphabet, 4

ASN, 4

baseDistributionPwm, 5

calculatePvalue, 5

createDiffLogoObject, 6

differenceOfICs, 8

diffLogo, 9

diffLogoFromPwm, 10

diffLogoTable, 11

diffLogoTableConfiguration, 12

DNA, 13

drawDiffLogoTable, 14

enrichDiffLogoObjectWithPvalues, 15

enrichDiffLogoTableWithPvalues, 16

extendPwmsFromAlignmentVector, 17

FULL\_ALPHABET, 18

getAlphabetFromCharacters, 18

getAlphabetFromSequences, 19

getPwmFromAlignment, 19

getPwmFromAlignmentFile, 20

getPwmFromFastaFile, 21

getPwmFromFile, 21

getPwmFromHomerFile, 22

getPwmFromPfmOrJasparFile, 22

getPwmFromPwmFile, 23

getSequencesFromAlignmentFile, 23

getSequencesFromFastaFile, 24

informationContent, 25

localPwmAlignment, 25

lossOfAbsICDifferences, 26

multipleLocalPwmsAlignment, 27

normalizedDifferenceOfProbabilities,  
28

normalizePWM, 29

prepareDiffLogoTable, 30

probabilities, 31

pwmDivergence, 31

pwmsDistanceMatrix, 32

reverseAlignmentVector, 33

RNA, 33

seqLogo, 34

shannonDivergence, 35

sumOfAbsICDifferences, 36

sumOfAbsProbabilityDifferences, 37

sumProbabilities, 38

switchDirection, 38

twoSetsAveragePwmDivergenceFromAlignmentVector,  
39