Package 'motifStack'

April 24, 2025

Type Package

Version 1.52.0

Title Plot stacked logos for single or multiple DNA, RNA and amino acid sequence

Author Jianhong Ou, Michael Brodsky, Scot Wolfe and Lihua Julie Zhu

Maintainer Jianhong Ou <jou@morgridge.org>

Depends R (>= 2.15.1), methods, grid

Imports ade4, Biostrings, ggplot2, grDevices, graphics, htmlwidgets, stats, stats4, utils, XML, TFBSTools

Suggests Cairo, grImport, grImport2, BiocGenerics, MotifDb, RColorBrewer, BiocStyle, knitr, RUnit, rmarkdown, JASPAR2020

biocViews SequenceMatching, Visualization, Sequencing, Microarray, Alignment, ChIPchip, ChIPSeq, MotifAnnotation, DataImport

Description The motifStack package is designed for graphic representation of multiple motifs with different similarity scores. It works with both DNA/RNA sequence motif and amino acid sequence motif. In addition, it provides the flexibility for users to customize the graphic parameters such as the font type and symbol colors.

License GPL (>= 2)

Lazyload yes

VignetteBuilder knitr

RoxygenNote 7.3.1

Encoding UTF-8

git_url https://git.bioconductor.org/packages/motifStack

git_branch RELEASE_3_21

git_last_commit 6cf9408

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-04-23

Contents

motifStack-package . . . 3
AAmotifAlignment . . . 3
browseMotifs . . . 4
browseMotifs-shiny . . . 5
calF . . . 6
calI . . . 6
clusterMotifs . . . 7
colorset . . . 7
compare2profiles . . . 8
compareProfiles . . . 9
DNAmotifAlignment . . . 9
DNAmotifToRNAmotif . . . 10
dpGlobal . . . 11
dpLocal . . . 11
GeomMotif . . . 12
geom_motif . . . 13
getALLRscoreFromCounts . . . 15
getDistance . . . 15
getRankedUniqueMotifs . . . 16
getScore . . . 17
highlightCol . . . 17
importMatrix . . . 18
marker-class . . . 19
matalign . . . 19
mergeMotifs . . . 20
motifCircos . . . 21
motifCloud . . . 24
motifGrob . . . 26
motifHclust . . . 27
motifPiles . . . 28
motifSig-class . . . 31
motifSignature . . . 32
motifStack . . . 33
ouNode-class . . . 34
pcm-class . . . 35
pfm-class . . . 37
pfm2pwm . . . 39
plotAffinityLogo . . . 40
plotMotifLogo . . . 41
plotMotifLogoA . . . 42
plotMotifLogoStack . . . 43
plotMotifLogoStackWithTree . . . 44
plotMotifOverMotif . . . 45
plotMotifStackWithPhylog . . . 46
plotMotifStackWithRadialPhylog . . . 48
plotXaxis . . . 51

plotYaxis 51
psam-class 52
pssm-class 53
readPCM 55
reorderUPGMAtree 56
traceBackGlobal 57
traceBackLocal 57

Index **58**

`motifStack-package` *Plot stacked logos for single or multiple DNA, RNA and amino acid sequence*

Description

motifStack is a package that is able to draw amino acid sequence as easy as to draw DNA/RNA sequence. motifStack provides the flexibility for users to select the font type and symbol colors. motifStack is designed for graphical representation of multiple motifs.

Author(s)

Jianhong Ou and Lihua Julie Zhu

Maintainer: Jianhong Ou <jianhong.ou@umassmed.edu>

`AAmotifAlignment` *align AA motifs*

Description

align AA motifs for plotting motifs stack

Usage

```
AAmotifAlignment(pcms, threshold = 0.4, minimalConsensus = 0)
```

Arguments

`pcms`	a list of position frequency matrices, pfms must be a list of class pcm
`threshold`	information content cutoff threshold for useful postions
`minimalConsensus`	minimal length of consensus for alignment

Value

a list of aligned motifs

Examples

```
pcms<-importMatrix(system.file("extdata", "prot.meme", package="motifStack"),
                   format="meme", to="pfm")
motifs<-AAmotifAlignment(pcms)
```

`browseMotifs` *browse motifs*

Description

browse motifs in a web browser

Usage

```
browseMotifs(
  pfms,
  phylog,
  layout = c("tree", "cluster", "radialPhylog"),
  nodeRadius = 2.5,
  baseWidth = 12,
  baseHeight = 30,
  xaxis = TRUE,
  yaxis = TRUE,
  width = NULL,
  height = NULL,
  ...
)
```

Arguments

`pfms`	a list of pfm
`phylog`	layout type. see GraphvizLayouts
`layout`	layout type. Could be tree, cluster or radialPhylog.
`nodeRadius`	node radius, default 2.5px.
`baseWidth, baseHeight`	width and height of each alphabet of the motif logo.
`xaxis, yaxis`	plot x-axis or y-axis or not in the motifs.
`width`	width of the figure
`height`	height of the figure
`...`	parameters not used

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
 library("MotifDb")
 matrix.fly <- query(MotifDb, "Dmelanogaster")
 motifs <- as.list(matrix.fly)
 motifs <- motifs[grepl("Dmelanogaster-FlyFactorSurvey-",
                        names(motifs), fixed=TRUE)]
 names(motifs) <- gsub("Dmelanogaster_FlyFactorSurvey_", "",
                       gsub("_FBgn[0-9]+$", "",
                            gsub("[^a-zA-Z0-9]","_",
                                 gsub("(_[0-9]+)+$", "", names(motifs)))))
 motifs <- motifs[unique(names(motifs))]
 pfms <- sample(motifs, 10)
 pfms <- mapply(pfms, names(pfms), FUN=function(.ele, .name){
                 new("pfm",mat=.ele, name=.name)})
 browseMotifs(pfms)
}
```

`browseMotifs-shiny` *Shiny bindings for browseMotifs*

Description

Output and render functions for using browseMotifs within Shiny applications and interactive Rmd documents.

Usage

```
browseMotifsOutput(outputId, width = "100%", height = "400px")

renderbrowseMotifs(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	output variable to read from
`width, height`	Must be a valid CSS unit (like `'100%'`, `'400px'`, `'auto'`) or a number, which will be coerced to a string and have `'px'` appended.
`expr`	An expression that generates a browseMotifs
`env`	The environment in which to evaluate `expr`.
`quoted`	Is `expr` a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

calF *calculate frequence*

Description

calculate frequence

Usage

```
calF(count, P = rep(1/length(count), length(count)), pseudo = 1)
```

Arguments

count	position counts
P	background probility
pseudo	pseudocount

Value

numeric(1)

calI *calculate I'*

Description

calculate I'

Usage

```
calI(freq1, freq2, P)
```

Arguments

freq1	position frequence for matrix 1 position j
freq2	position frequence for matrix 2 position j
P	background of profile1

Value

numeric(1)

clusterMotifs *cluster motifs*

Description

A help function to do matalign and motifHclust in one function.

Usage

```
clusterMotifs(motifs, ...)
```

Arguments

motifs	A list of pcms of pfms.
...	parameter to be passed to matalign function.

Value

An object of hclust.

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
 fp <- system.file("extdata", package="motifStack")
 fs <- dir(fp, "pcm$")
 pcms <- importMatrix(file.path(fp, fs), format="pcm")
 hc <- clusterMotifs(pcms)
}
```

colorset *retrieve color setting for logo*

Description

retrieve color setting for logo

Usage

```
colorset(alphabet = "DNA", colorScheme = "auto")
```

Arguments

alphabet	character, 'DNA', 'RNA' or 'AA'
colorScheme	'auto', 'charge', 'chemistry', 'classic' or 'hydrophobicity' for AA, 'auto', 'base-pairing', or 'blindnessSafe' for DNA ro RNA

Value

A character vector of color scheme

Examples

```
col <- colorset("AA", "hydrophobicity")
```

`compare2profiles` *compare two profiles*

Description

compare two pcm object

Usage

```
compare2profiles(
  pcm1,
  pcm2,
  method = c("Smith-Waterman", "Needleman-Wunsch"),
  pseudo = 1
)
```

Arguments

`pcm1`, `pcm2`	object of pcm
`method`	Alignment method. "Smith-Waterman" or "Needleman-Wunsch". Default is "Smith-Waterma"
`pseudo`	pseudocount

Value

a list with names: motif1, motif2, alignmentScore, startPos1, startPos2, endPos1, endPos2, alignmentLength.

compareProfiles *comapre w pcm*

Description

compare two pcm objects

Usage

```
compareProfiles(
  pcm1,
  pcm2,
  method = c("Smith-Waterman", "Needleman-Wunsch"),
  pseudo = 1,
  revcomp = TRUE
)
```

Arguments

pcm1, pcm2	object of pcm
method	Alignment method. "Smith-Waterman" or "Needleman-Wunsch". Default is "Smith-Waterma"
pseudo	pseudocount
revcomp	Check reverseComplement or not.

Value

a list with names: motif1, motif2, alignmentScore, startPos1, startPos2, endPos1, endPos2, alignmentLength.

DNAmotifAlignment *align DNA motifs*

Description

align DNA motifs for plotting motifs stack

Usage

```
DNAmotifAlignment(
  pfms,
  threshold = 0.4,
  minimalConsensus = 0,
  rcpostfix = "(RC)",
  revcomp = rep(TRUE, length(pfms))
)
```

Arguments

pfms	a list of position frequency matrices, pfms must be a list of class pfm or psam
threshold	information content cutoff threshold for useful postions
minimalConsensus	minimal length of consensus for alignment
rcpostfix	the postfix for reverse complements
revcomp	a logical vector to indicates whether the reverse complemet should be involved into alignment

Value

a list of aligned motifs

Examples

```
pcms<-readPCM(file.path(find.package("motifStack"), "extdata"),"pcm$")
motifs<-lapply(pcms,pcm2pfm)
motifs<-DNAmotifAlignment(motifs)
```

DNAmotifToRNAmotif *convert DNA motif into RNA motif*

Description

convert DNA motif into RNA motif

Usage

```
DNAmotifToRNAmotif(pfm)
```

Arguments

pfm	An object of "pcm" or "pfm"

Value

An object of "pcm" or "pfm" of RNA motif

Examples

```
motifs<-importMatrix(dir(file.path(find.package("motifStack"),
                         "extdata"),"pcm$", full.names = TRUE))
rnaMotifs <- DNAmotifToRNAmotif(motifs)
```

dpGlobal *Global alignment version*

Description

Global alignment version

Usage

```
dpGlobal(score, m, n)
```

Arguments

score	ALLR scores, m x n matrix
m, n	matrix width

Value

score matrix

dpLocal *Dynamic programming function, local version*

Description

Dynamic programming function, local version

Usage

```
dpLocal(score, m, n)
```

Arguments

score	ALLR scores, m x n matrix
m, n	matrix width

Value

score matrix

GeomMotif *GeomMotif object*

Description

GeomMotif object is a ggproto object.

Usage

```
GeomMotif
```

Format

The format is: Classes 'GeoMotif', 'Geom', 'ggproto', 'gg' <ggproto object: Class GeoMotif, Geom, gg> aesthetics: function default_aes: uneval draw_group: function draw_key: function draw_layer: function draw_panel: function extra_params: na.rm handle_na: function non_missing_aes: optional_aes: parameters: function required_aes: xmin ymin xmax ymax motif setup_data: function use_defaults: function super: <ggproto object: Class Geom, gg>

See Also

geom_motif

Examples

```
pcm <- read.table(file.path(find.package("motifStack"),
                            "extdata", "bin_SOLEXA.pcm"))
pcm <- pcm[,3:ncol(pcm)]
rownames(pcm) <- c("A","C","G","T")
motif <- new("pcm", mat=as.matrix(pcm), name="bin_SOLEXA")

df <- data.frame(xmin=c(.25, .25), ymin=c(.25, .75), xmax=c(.75, .75), ymax=c(.5, 1))
df$motif <- list(pcm2pfm(motif), pcm2pfm(motif))

library(ggplot2)

ggplot(df, aes(xmin=xmin, ymin=ymin, xmax=xmax, ymax=ymax, motif=motif)) +
geom_motif() + theme_bw() + ylim(0, 1) + xlim(0, 1)
```

geom_motif *geom_motif*

Description

geom_motif uses the locations of the four corners (xmin, xmax, ymin and ymax) to plot motifs.

Usage

```
geom_motif(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  ic.scale = TRUE,
  use.xy = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer.
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to layer().
ic.scale	logical If TRUE, the height of each column is proportional to its information content. Otherwise, all columns have the same height.
use.xy	logical If TRUE, the required aesthethics will be x, y, width, height, and motif. Otherwise, xmin, ymin, xmax, ymax and motif.
show.legend	Not used.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them.

Value

a layer that contains GeomMotif object.

Aesthetics

geom_motif() understands the following aesthetics (required aesthetics are in bold):

- xmin
- xmax
- ymin
- ymax
- motif
- angle
- fontfamily
- fontface

OR

- x
- y
- width
- height
- motif
- angle
- fontfamily
- fontface

Author(s)

Jianhong Ou

Examples

```
pcm <- read.table(file.path(find.package("motifStack"),
                            "extdata", "bin_SOLEXA.pcm"))
pcm <- pcm[,3:ncol(pcm)]
rownames(pcm) <- c("A","C","G","T")
motif <- new("pcm", mat=as.matrix(pcm), name="bin_SOLEXA")

df <- data.frame(xmin=c(.25, .25), ymin=c(.25, .75), xmax=c(.75, .75), ymax=c(.5, 1))
df$motif <- list(pcm2pfm(motif), pcm2pfm(motif))

library(ggplot2)
ggplot(df, aes(xmin=xmin, ymin=ymin, xmax=xmax, ymax=ymax, motif=motif)) +
geom_motif() + theme_bw() + ylim(0, 1) + xlim(0, 1)
```

getALLRscoreFromCounts *calculate ALLR from counts*

Description

calculate ALLR from counts

Usage

```
getALLRscoreFromCounts(count1, count2, P1, P2, pseudo)
```

Arguments

count1, count2	count in position j for matrix 1 or 2
P1, P2	background for matrix 1 or 2
pseudo	pseudocount

Value

numeric(1) of ALLR

getDistance *Calculate distances between two profiles*

Description

Calculate distances between two profiles

Usage

```
getDistance(hsp, count1, count2, P1, P2, pseudo)
```

Arguments

hsp	output of traceBack function
count1, count2	motif profile 1 or 2
P1, P2	background of profile 1 or 2
pseudo	pseudocount

Value

full distance and aligned distance.

`getRankedUniqueMotifs` *get the unique motif in each category grouped by distance*

Description

to get the unique motif in a given category, eg by species.

Usage

```
getRankedUniqueMotifs(phylog, attr)
```

Arguments

`phylog`	an object of class phylog
`attr`	attribute used for category of motifs

Value

return a list:

`uni.rank`	unique motif ranks
`uni.length`	length of unique motif grouped by distance
`uni.list`	unique motif names grouped by distance

Author(s)

Jianhong Ou

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
     library("MotifDb")
     matrix.fly <- query(MotifDb, "Dmelanogaster")
     matrix.human <- query(MotifDb, "Hsapiens")
     pfms <- c(as.list(matrix.fly), as.list(matrix.human))
     pfms <- pfms[sample(seq_along(pfms), 100)]
     hc <- clusterMotifs(pfms)
     library(ade4)
     phylog <- ade4::hclust2phylog(hc)
     leaves <- names(phylog$leaves)
     attr <- gsub("^(.*?)_.*$", "\1", leaves)
     getRankedUniqueMotifs(phylog, attr)
 }
```

getScore *Calculate pair_wise position score*

Description

Calculate pair_wise position score

Usage

```
getScore(pcm1, pcm2, pseudo = 1)
```

Arguments

pcm1, pcm2	object of pcm
pseudo	pseudocount

Value

A score matrix with nrow of ncol of pcm1 and ncol of ncol of pcm2.

highlightCol *add alpha transparency value to a color*

Description

An alpha transparency value can be specified to a color, in order to get better color for background.

Usage

```
highlightCol(col, alpha = 0.5)
```

Arguments

col	vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].
alpha	a value in [0, 1]

Value

a vector of colors in hexadecimal string of the form "#rrggbbaa".

Author(s)

Jianhong Ou

Examples

```
highlightCol(1:5, 0.3)
highlightCol(c("red", "green", "blue"), 0.3)
```

`importMatrix` *import motifs from local files*

Description

Import the motifs into pcm-class or pfm-class from files exported from Transfac, CisBP, and JASPAR.

Usage

```
importMatrix(
  filenames,
 format = c("auto", "pfm", "cm", "pcm", "meme", "transfac", "jaspar", "scpd", "cisbp",
    "psam", "xmatrix"),
  to = c("auto", "pcm", "pfm", "pssm", "psam")
)
```

Arguments

`filenames`	filename, an XMatrixList object, or an XMatrix object to be imported.
`format`	file format
`to`	import to pcm-class or pfm-class

Value

a list of object pcm-class or pfm-class

Author(s)

Jianhong Ou

Examples

```
path <- system.file("extdata", package = "motifStack")
importMatrix(dir(path, "*.pcm", full.names = TRUE))
```

marker-class *Class* marker

Description

An object of class "marker" represents a marker in a motif

Usage

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

Arguments

x	A marker object
name	slot name of marker object

Objects from the Class

Objects can be created by calls of the form new("marker", type, start, stop, label, gp).

Examples

```
new("marker", type="rect", start=c(2, 4), gp=gpar(lty=3))
```

matalign *Matrix Aligner*

Description

Matrix Aligner is modified from Matalign-v4a. Matalign-v4a is a program to compare two positional specific matrices. The author of Matalign-v4a is Ting Wang and Gary Stormo.

Usage

```
matalign(
  pcms,
  method = c("Smith-Waterman", "Needleman-Wunsch"),
  pseudo = 1,
  revcomp = TRUE,
  ...
)
```

Arguments

pcms	A list of pcm
method	Alignment method. "Smith-Waterman" or "Needleman-Wunsch". Default is "Smith-Waterma"
pseudo	pseudocount
revcomp	Check reverseComplement or not.
...	Not use.

Value

A data frame with alignment information. The column names are motif1, motif2, alignmentScore, startPos1, startPos2, endPos1, endPos2, alignmentLength.

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
  fp <- system.file("extdata", package="motifStack")
  fs <- dir(fp, "pcm$")
  pcms <- importMatrix(file.path(fp, fs), format="pcm")
  matalign(pcms)
 }
```

---

mergeMotifs *merge multiple motifs*

Description

merge multiple motifs by calculate mean of each position

Usage

```
mergeMotifs(..., bgNoise = NA)
```

Arguments

...	pcm or pfm objects
bgNoise	if it is not NA, test will using a background by Dirichlet(1)-distributed random frequencies with weight bg.noise. The value of bgNoise should be a number in the range of 0 to 1, eg. 0.05

Value

a pfm object

Author(s)

Jianhong Ou

Examples

```
pcms<-readPCM(file.path(find.package("motifStack"), "extdata"),"pcm$")
mergeMotifs(pcms)
```

`motifCircos` *plot sequence logo stacks with a radial phylogenic tree and multiple color rings*

Description

plot sequence logo stacks with a radial phylogenic tree and multiple color rings. The difference from plotMotifStackWithRadialPhylog is that it has more color setting and one more group of pfms.

Usage

```
motifCircos(
  phylog,
  pfms = NULL,
  pfms2 = NULL,
  R = 2.5,
  r.tree = 1,
  col.tree.bg = NULL,
  col.tree.bg.alpha = 1,
  cnodes = 0,
  labels.nodes = names(phylog$nodes),
  clabel.nodes = 0,
  r.leaves = NA,
  cleaves = 1,
  labels.leaves = names(phylog$leaves),
  clabel.leaves = 1,
  col.leaves = rep("black", length(labels.leaves)),
  col.leaves.bg = NULL,
  col.leaves.bg.alpha = 1,
  r.pfms = NA,
  r.pfms2 = NA,
  r.rings = 0,
  col.rings = list(),
  col.inner.label.circle = NULL,
  inner.label.circle.width = 0.02,
  col.outer.label.circle = NULL,
  outer.label.circle.width = 0.02,
  draw.box = FALSE,
  clockwise = FALSE,
  init.angle = if (clockwise) 90 else 0,
  angle = 360,
  pfmNameSpliter = ";",
```

```
  rcpostfix = "(RC)",
  motifScale = c("linear", "logarithmic", "none"),
  ic.scale = TRUE,
  plotIndex = FALSE,
  IndexCol = "black",
  IndexCex = 0.8,
  groupDistance = NA,
  groupDistanceLineCol = "red",
  plotAxis = FALSE
)
```

Arguments

phylog	an object of class phylog
pfms	a list of objects of class pfm
pfms2	a list of objects of class pfm
R	radius of canvas
r.tree	half width of the tree
col.tree.bg	a vector of colors for tree background
col.tree.bg.alpha	a alpha value [0, 1] of colors for tree background
cnodes	a character size for plotting the points that represent the nodes, used with par("cex")*cnodes. If zero, no points are drawn
labels.nodes	a vector of strings of characters for the nodes labels
clabel.nodes	a character size for the nodes labels, used with par("cex")*clabel.nodes. If zero, no nodes labels are drawn
r.leaves	width of the leaves
cleaves	a character size for plotting the points that represent the leaves, used with par("cex")*cleaves. If zero, no points are drawn
labels.leaves	a vector of strings of characters for the leaves labels
clabel.leaves	a character size for the leaves labels, used with par("cex")*clavel.leaves
col.leaves	a vector of colors for leaves labels
col.leaves.bg	a vector of colors for background of leaves labels
col.leaves.bg.alpha	alpha value [0, 1] for the colors of backgroud of leaves labels
r.pfms	width of the pfms
r.pfms2	width of the pfms2
r.rings	a vector of width of color rings
col.rings	a list of color rings
col.inner.label.circle	a vector of colors for inner cirlce of pfms
inner.label.circle.width	width for inner circle of pfms

col.outer.label.circle
: a vector of colors for outer circle of pfms

outer.label.circle.width
: width for outer circle of pfms

draw.box
: if TRUE draws a box around the current plot with the function box()

clockwise
: a logical value indicating if slices are drawn clockwise or counter clockwise

init.angle
: number specifying the starting angle (in degrees) for the slices. Defaults to 0 (i.e., '3 o'clock') unless clockwise is true where init.angle defaults to 90 (degrees), (i.e., '12 o'clock')

angle
: number specifying the angle (in degrees) for phylogenic tree. Defaults 360

pfmNameSpliter
: spliter when name of pfms/pfms2 contain multiple node of labels.leaves

rcpostfix
: the postfix for reverse complements

motifScale
: the scale of logo size

ic.scale
: logical. If TRUE, the height of each column is proportional to its information content. Otherwise, all columns have the same height.

plotIndex
: logical. If TRUE, will plot index number in the motifLogo which can help user to describe the motifLogo

IndexCol
: The color of the index number when plotIndex is TRUE.

IndexCex
: The cex of the index number when plotIndex is TRUE.

groupDistance
: show groupDistance on the draw

groupDistanceLineCol
: groupDistance line color, default: red

plotAxis
: logical. If TRUE, will plot distance axis.

Value

none

Author(s)

Jianhong Ou

See Also

plotMotifStackWithRadialPhylog

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
    library("MotifDb")
    matrix.fly <- query(MotifDb, "Dmelanogaster")
    motifs <- as.list(matrix.fly)
    motifs <- motifs[grepl("Dmelanogaster-FlyFactorSurvey-",
                           names(motifs), fixed=TRUE)]
    names(motifs) <- gsub("Dmelanogaster_FlyFactorSurvey_", "",
                gsub("_FBgn[0-9]+$", "",
```

```
                gsub("[^a-zA-Z0-9]","_",
                   gsub("(_[0-9]+)+$", "", names(motifs)))))
  motifs <- motifs[unique(names(motifs))]
  pfms <- sample(motifs, 50)
  hc <- clusterMotifs(pfms)
  library(ade4)
  phylog <- ade4::hclust2phylog(hc)
  leaves <- names(phylog$leaves)
  pfms <- pfms[leaves]
  pfms <- mapply(pfms, names(pfms), FUN=function(.ele, .name){
              new("pfm",mat=.ele, name=.name)})
  pfms <- DNAmotifAlignment(pfms, minimalConsensus=3)
  library(RColorBrewer)
  color <- brewer.pal(12, "Set3")
  motifCircos(phylog, pfms, cleaves = 0.5, clabel.leaves = 0.7,
                  col.tree.bg=rep(color, each=5),
                  col.leaves=rep(color, each=5),
                   r.rings=c(0.02, 0.03, 0.04),
                   col.rings=list(sample(colors(), 50),
                                  sample(colors(), 50),
                                  sample(colors(), 50)))
}
```

`motifCloud` *plot a DNA sequence logo cloud*

Description

Plot a DNA sequence logo cloud

Usage

```
motifCloud(
  motifSig,
  rcpostfix = "(RC)",
  layout = c("rectangles", "cloud", "tree"),
  scale = c(6, 0.5),
  rot.per = 0.1,
  draw.box = TRUE,
  draw.freq = TRUE,
  box.col = "gray",
  freq.col = "gray",
  group.col = NULL,
  groups = NULL,
  draw.legend = FALSE,
  font = "sans",
  ic.scale = TRUE
)
```

Arguments

motifSig	an object of class motifSig
rcpostfix	postfix for reverse-complement motif names, default: (RC)
layout	layout of the logo cloud, rectangles, cloud or tree
scale	A vector of length 2 indicating the range of the size of the sequence logo.
rot.per	proportion sequence logo with 90 degree rotation. Only work for "cloud" layout
draw.box	draw box for each sequence logo or not
draw.freq	label frequency of each signature or not
box.col	color of box for each sequence logo
freq.col	color of frequency label
group.col	color setting for groups
groups	a named vectors of motif groups
draw.legend	draw group color legend or not
font	font of logo
ic.scale	logical If TRUE, the height of each column is proportional to its information content. Otherwise, all columns have the same height.

Value

none

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
  library("MotifDb")
  matrix.fly <- query(MotifDb, "Dmelanogaster")
  motifs <- as.list(matrix.fly)
  motifs <- motifs[grepl("Dmelanogaster-FlyFactorSurvey-",
                         names(motifs), fixed=TRUE)]
  names(motifs) <- gsub("Dmelanogaster_FlyFactorSurvey_", "",
              gsub("_FBgn[0-9]+$", "",
                gsub("[^a-zA-Z0-9]","_",
                   gsub("(_[0-9]+)+$", "", names(motifs)))))
  motifs <- motifs[unique(names(motifs))]
  pfms <- sample(motifs, 50)
  hc <- clusterMotifs(pfms)
  library(ade4)
  phylog <- ade4::hclust2phylog(hc)
  leaves <- names(phylog$leaves)
  pfms <- pfms[leaves]
  pfms <- mapply(pfms, names(pfms), FUN=function(.ele, .name){
              new("pfm",mat=.ele, name=.name)})
  motifSig <- motifSignature(pfms, phylog, cutoffPval=0.0001)
  motifCloud(motifSig)
}
```

motifGrob *Motif Grob*

Description

This function create a motif grob.

Usage

```
motifGrob(
  pfm,
  x = unit(0.5, "npc"),
  y = unit(0.5, "npc"),
  width = unit(1, "npc"),
  height = unit(1, "npc"),
  angle = 0,
  ic.scale = TRUE,
  default.units = "native",
  name = NULL,
  gp = gpar(fontfamily = "sans", fontface = "bold")
)
```

Arguments

pfm	an object of pfm
x	A numeric vector or unit object specifying x-values.
y	A numeric vector or unit object specifying y-values.
width	A numeric vector or unit object specifying width.
height	A numeric vector or unit object specifying height.
angle	A numeric value indicating the angle of rotation of the motif. Positive values indicate the amount of rotation, in degrees, anticlockwise from the positive x-axis.
ic.scale	logical If TRUE, the height of each column is proportional to its information content. Otherwise, all columns have the same height.
default.units	A string indicating the default units to use if x, y, width, or height are only given as numeric vectors.
name	A character value to uniquely identify the motifGrob once it has been pushed onto the grob tree.
gp	A gpar object, typically the output from a call to the function gpar. The list will be used as parameter of plotMotifLogoA.

Value

An gTree object.

Author(s)

Jianhong Ou

Examples

```
pcm<-matrix(runif(40,0,100),nrow=4,ncol=10)
pfm<-pcm2pfm(pcm)
rownames(pfm)<-c("A","C","G","T")
motif <- new("pfm", mat=pfm, name="bin_SOLEXA")
motifGrob(motif)
```

`motifHclust` *Hierarchical Clustering motifs*

Description

functions to perfom clustering of output of matalign

Usage

```
motifHclust(align, ...)
```

Arguments

`align`	output of matalign, used to generate distance matrix.
`...`	parameter to pass to the hclust.

Value

An object of hclust.

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
 fp <- system.file("extdata", package="motifStack")
 fs <- dir(fp, "pcm$")
 pcms <- importMatrix(file.path(fp, fs), format="pcm")
 align <- matalign(pcms)
 hc <- motifHclust(align, method="average")
}
```

motifPiles *plot sequence logo stacks with a linear phylogenic tree and multiple color sets*

Description

plot sequence logo stacks with a linear phylogenic tree and multiple color sets.

Usage

```
motifPiles(
  phylog,
  pfms = NULL,
  pfms2 = NULL,
  r.tree = 0.45,
  col.tree = NULL,
  cnodes = 0,
  labels.nodes = names(phylog$nodes),
  clabel.nodes = 0,
  cleaves = 0.2,
  labels.leaves = names(phylog$leaves),
  clabel.leaves = 1,
  col.leaves = rep("black", length(labels.leaves)),
  col.leaves.bg = NULL,
  col.leaves.bg.alpha = 1,
  r.pfms = NA,
  r.pfms2 = NA,
  motifScale = c("logarithmic", "linear", "none"),
  col.pfms = NULL,
  col.pfms.width = 0.02,
  col.pfms2 = NULL,
  col.pfms2.width = 0.02,
  r.anno = 0,
  col.anno = list(),
  pfmNameSpliter = ";",
  rcpostfix = "(RC)",
  ic.scale = TRUE,
  plotIndex = FALSE,
  IndexCol = "black",
  IndexCex = 0.8,
  groupDistance = NA,
  groupDistanceLineCol = "red"
)
```

Arguments

phylog	an object of class phylog

pfms	a list of objects of class pfm
pfms2	a list of objects of class pfm
r.tree	width of the tree
col.tree	a vector of colors for tree
cnodes	a character size for plotting the points that represent the nodes, used with par("cex")*cnodes. If zero, no points are drawn
labels.nodes	a vector of strings of characters for the nodes labels
clabel.nodes	a character size for the nodes labels, used with par("cex")*clabel.nodes. If zero, no nodes labels are drawn
cleaves	a character size for plotting the points that represent the leaves, used with par("cex")*cleaves. If zero, no points are drawn
labels.leaves	a vector of strings of characters for the leaves labels
clabel.leaves	a character size for the leaves labels, used with par("cex")*clavel.leaves
col.leaves	a vector of colors for leaves labels
col.leaves.bg	a vector of colors for background of leaves labels
col.leaves.bg.alpha	alpha value [0, 1] for the colors of backgroud of leaves labels
r.pfms	width of the pfms
r.pfms2	width of the pfms2
motifScale	the scale of logo size
col.pfms	a vector of colors for inner pile of pfms
col.pfms.width	width for inner pile of pfms
col.pfms2	a vector of colors for outer pile of pfms
col.pfms2.width	width for outer pile of pfms
r.anno	a vector of width of color sets
col.anno	a list of color sets
pfmNameSpliter	spliter when name of pfms/pfms2 contain multiple node of labels.leaves
rcpostfix	the postfix for reverse complements
ic.scale	logical. If TRUE, the height of each column is proportional to its information content. Otherwise, all columns have the same height.
plotIndex	logical. If TRUE, will plot index number in the motifLogo which can help user to describe the motifLogo
IndexCol	The color of the index number when plotIndex is TRUE.
IndexCex	The cex of the index number when plotIndex is TRUE.
groupDistance	show groupDistance on the draw
groupDistanceLineCol	groupDistance line color, default: red

Value

none

Author(s)

Jianhong Ou

See Also

motifCircos

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
    library("MotifDb")
    matrix.fly <- query(MotifDb, "Dmelanogaster")
    motifs <- as.list(matrix.fly)
    motifs <- motifs[grepl("Dmelanogaster-FlyFactorSurvey-",
                           names(motifs), fixed=TRUE)]
    names(motifs) <- gsub("Dmelanogaster_FlyFactorSurvey_", "",
                gsub("_FBgn[0-9]+$", "",
                  gsub("[^a-zA-Z0-9]","_",
                     gsub("(_[0-9]+)+$", "", names(motifs)))))
    motifs <- motifs[unique(names(motifs))]
    pfms <- sample(motifs, 50)
    hc <- clusterMotifs(pfms)
    library(ade4)
    phylog <- ade4::hclust2phylog(hc)
    leaves <- names(phylog$leaves)
    pfms <- pfms[leaves]
    pfms <- mapply(pfms, names(pfms), FUN=function(.ele, .name){
                 new("pfm",mat=.ele, name=.name)})
    pfms <- DNAmotifAlignment(pfms, minimalConsensus=3)
    library(RColorBrewer)
    color <- brewer.pal(12, "Set3")
    motifPiles(phylog, pfms, cleaves = 0.5, clabel.leaves = 0.7,
             col.leaves=rep(color, each=5),
             col.leaves.bg = sample(colors(), 50),
             col.tree=rep(color, each=5),
             r.anno=c(0.02, 0.03, 0.04),
             col.anno=list(sample(colors(), 50),
                           sample(colors(), 50),
                           sample(colors(), 50)))
  }
```

motifSig-class *Class* "motifSig"

Description

An object of class "motifSig" represents the output of function motifSignature

methods for motifSig objects.

Usage

```
signatures(object)

frequence(object)

nodelist(object)

sigColor(object)

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

Arguments

object	An object of class motifSig.
x	A motifSig object
name	slot name of motifSig object

Objects from the Class

Objects can be created by calls of the form new("motifSig", signature, freq, nodelist, gpcol).

Methods

signatures signature(object = "motifSig") return the signatures of motifSig

frequence signature(object = "motifSig") return the frequency of motifSig

nodelist signature(object = "motifSig") return the nodelist of motifSig

sigColor signature(object = "motifSig") return the group color sets of motifSig

$, $<- Get or set the slot of motifSig

motifSignature *get signatures from motifs*

Description

extract signatures from multiple motifs by distance calculated from STAMP

Usage

```
motifSignature(
  pfms,
  phylog,
  cutoffPval,
  groupDistance,
  rcpostfix = "(RC)",
  min.freq = 2,
  trim = 0.2,
  families = list(),
  sort = TRUE
)
```

Arguments

pfms	a list of objects of class pfm
phylog	an object of class phylog
cutoffPval	pvalue for motifs to merge.
groupDistance	maxmal distance of motifs in the same group
rcpostfix	postfix for reverse-complement motif names, default: (RC)
min.freq	signatures with frequency below min.freq will not be plotted
trim	minimal information content for each position of signature
families	for each family, the motif number in one signature should only count as 1
sort	sort the signatures by frequency or not.

Value

an Object of class motifSig

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
  library("MotifDb")
  matrix.fly <- query(MotifDb, "Dmelanogaster")
  motifs <- as.list(matrix.fly)
  motifs <- motifs[grepl("Dmelanogaster-FlyFactorSurvey-",
                         names(motifs), fixed=TRUE)]
```

```
  names(motifs) <- gsub("Dmelanogaster_FlyFactorSurvey_", "",
              gsub("_FBgn[0-9]+$", "",
                gsub("[^a-zA-Z0-9]","_",
                   gsub("(_[0-9]+)+$", "", names(motifs)))))
  motifs <- motifs[unique(names(motifs))]
  pfms <- sample(motifs, 50)
  hc <- clusterMotifs(pfms)
  library(ade4)
  phylog <- ade4::hclust2phylog(hc)
  leaves <- names(phylog$leaves)
  pfms <- pfms[leaves]
  pfms <- mapply(pfms, names(pfms), FUN=function(.ele, .name){
               new("pfm",mat=.ele, name=.name)})
  motifSig <- motifSignature(pfms, phylog, cutoffPval=0.0001)
}
```

`motifStack` *plot a DNA sequence logo stack*

Description

Plot a DNA sequence logo stack

Usage

```
motifStack(
  pfms,
  layout = c("stack", "treeview", "phylog", "radialPhylog"),
  reorder = TRUE,
  ...
)
```

Arguments

`pfms`	a list of objects of class pfm
`layout`	layout of the logo stack, stack, treeview or radialPhylog
`reorder`	logical(1). Default TRUE. Set to FALSE will do alignment but keep the order of the pfms. This parameter only work for stack layout.
`...`	any parameters could to pass to plotMotifLogoStack, plotMotifLogoStackWithTree, plotMotifStackWithPhylog or plotMotifStackWithRadialPhylog. And the 'revcomp' parameter for DNAmotifAlignment.

Value

return a list contains pfms and phylog

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
  library("MotifDb")
  matrix.fly <- query(MotifDb, "Dmelanogaster")
  motifs <- as.list(matrix.fly)
  motifs <- motifs[grepl("Dmelanogaster-FlyFactorSurvey-",
                         names(motifs), fixed=TRUE)]
  names(motifs) <- gsub("Dmelanogaster_FlyFactorSurvey_", "",
              gsub("_FBgn[0-9]+$", "",
                gsub("[^a-zA-Z0-9]","_",
                   gsub("(_[0-9]+)+$", "", names(motifs)))))
  motifs <- motifs[unique(names(motifs))]
  pfms <- sample(motifs, 50)
  pfms <- mapply(pfms, names(pfms), FUN=function(.ele, .name){
              new("pfm",mat=.ele, name=.name)})
  motifStack(pfms, "radialPhylog")

  ## AA motifs
  pcms<-importMatrix(system.file("extdata", "prot.meme",
                                 package="motifStack"),
                format="meme", to="pfm")
  motifStack(pcms[1:5])
  motifStack(pcms[1:5], reorder=FALSE)
}
```

---

ouNode-class *Class* ouNode

Description

An object of class "ouNode" represents a motif node in a cluster tree

Usage

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

Arguments

x	A ouNode object
name	slot name of ouNode object

Objects from the Class

Objects can be created by calls of the form `new("ouNode", left, right, parent, distl, distr, sizel, sizer)`.

Examples

```
new("ouNode", left="A", right="B", parent="Root", distl=1, distr=2, sizel=1, sizer=1)
```

---

pcm-class *Class* "pcm"

Description

An object of class "pcm" represents the position count matrix of a DNA/RNA/amino-acid sequence motif. The entry stores a matrix, which in row i, column j gives the counts of observing nucleotide/or amino acid i in position j of the motif.

methods for pcm objects.

Usage

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

## S4 method for signature 'pcm,ANY'
plot(x, y = "missing", ...)

trimMotif(x, t)

matrixReverseComplement(x)

addBlank(x, n, b)

getIC(x, p)

pcm2pfm(x, background)

pcm2pssm(x, background)

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

## S4 method for signature 'pcm'
format(x, ...)
```

Arguments

x	An object of class pcm. For getIC, if parameter p is followed, x should be an object of matrix. For pcm2pfm, x also could be an object of matrix.
name	slot name of pcm object.
y	Not use.

`...`	Further potential arguments passed to `plotMotifLogo`.
`t`	numeric value of information content threshold for trimming.
`n`	how many spaces should be added.
`b`	logical value to indicate where the space should be added.
`p`	p is the background frequency.
`background`	a `"numeric"` vector. The background frequency.
`row.names, optional`	see as.data.frame

Objects from the Class

Objects can be created by calls of the form `new("pcm", mat, name, alphabet, color, background)`.

Methods

addBlank `signature(x="pcm",n="numeric", b="logical")` add space into the position count matrix for alignment. b is a bool value, if TRUE, add space to the 3' end, else add space to the 5' end. n indicates how many spaces should be added.

coerce `signature(from = "pcm", to = "matrix")`: convert object pcm to matrix

getIC `signature(x = "pcm",)` Calculate information content profile for position frequency matrix.

matrixReverseComplement `signature(x = "pcm")` get the reverse complement of position frequency matrix.

plot `signature(x = "pcm")` Plots the sequence logo of the position count matrix.

trimMotif `signature(x = "pcm", t= "numeric")` trim motif by information content.

$, $<- Get or set the slot of pcm-class

as.data.frame convert pcm-class to a data.frame

format return the name_pcm of pcm-class

Examples

```
pcm <- read.table(file.path(find.package("motifStack"), "extdata", "bin_SOLEXA.pcm"))
pcm <- pcm[,3:ncol(pcm)]
rownames(pcm) <- c("A","C","G","T")
motif <- new("pcm", mat=as.matrix(pcm), name="bin_SOLEXA")
plot(motif)
pcm2pfm(pcm)
pcm2pssm(pcm)

pcm <- read.table(file.path(find.package("motifStack"), "extdata", "bin_SOLEXA.pcm"))
pcm <- pcm[,3:ncol(pcm)]
rownames(pcm) <- c("A","C","G","T")
motif <- new("pcm", mat=as.matrix(pcm), name="bin_SOLEXA")
getIC(motif)
matrixReverseComplement(motif)
as(motif,"matrix")
```

```
pcm2pfm(motif)
as.data.frame(motif)
format(motif)
```

---

pfm-class *Class* "pfm"

Description

An object of class "pfm" represents the position frequency matrix of a DNA/RNA/amino-acid sequence motif. The entry stores a matrix, which in row i, column j gives the frequency of observing nucleotide/or amino acid i in position j of the motif.

methods for pfm objects.

Usage

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

## S4 method for signature 'pfm,ANY'
plot(x, y = "missing", ...)

## S4 method for signature 'pfm,ANY'
getIC(x, p = "missing")

## S4 method for signature 'pfm,numeric'
trimMotif(x, t)

## S4 method for signature 'pfm'
matrixReverseComplement(x)

## S4 method for signature 'pfm,numeric,logical'
addBlank(x, n, b)

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

## S4 method for signature 'pfm'
format(x, ...)
```

Arguments

x	An object of class pfm. For getIC, if parameter p is followed, x should be an object of matrix.
name	Slot name.
y	Not use.

...	Further potential arguments passed to plotMotifLogo.
p	p is the background frequency.
t	numeric value of information content threshold for trimming.
n	how many spaces should be added.
b	logical value to indicate where the space should be added.
row.names, optional	see as.data.frame

Objects from the Class

Objects can be created by calls of the form new("pfm", mat, name, alphabet, color, background).

Methods

addBlank signature(x="pfm",n="numeric", b="logical") add space into the position frequency matrix for alignment. b is a bool value, if TRUE, add space to the 3' end, else add space to the 5' end. n indicates how many spaces should be added.

getIC signature(x = "pfm",) Calculate information content profile for position frequency matrix.

getIC signature(x = "matrix", p = "numeric") Calculate information content profile for matrix. p is the background frequency

matrixReverseComplement signature(x = "pfm") get the reverse complement of position frequency matrix.

plot signature(x = "pfm") Plots the sequence logo of the position frequency matrix.

trimMotif signature(x = "pfm", t= "numeric") trim motif by information content.

$, $<- Get or set the slot of pfm-class

as.data.frame convert pfm-class to a data.frame

format return the name_pfm of pfm-class

Examples

```
pcm <- read.table(file.path(find.package("motifStack"), "extdata", "bin_SOLEXA.pcm"))
pcm <- pcm[,3:ncol(pcm)]
rownames(pcm) <- c("A","C","G","T")
motif <- pcm2pfm(pcm)
motif <- new("pfm", mat=motif, name="bin_SOLEXA")
plot(motif)


pcm <- read.table(file.path(find.package("motifStack"), "extdata", "bin_SOLEXA.pcm"))
pcm <- pcm[,3:ncol(pcm)]
rownames(pcm) <- c("A","C","G","T")
motif <- pcm2pfm(pcm)
motif <- new("pfm", mat=motif, name="bin_SOLEXA")
getIC(motif)
matrixReverseComplement(motif)
addBlank(motif, 1, FALSE)
```

```
addBlank(motif, 3, TRUE)
as(motif,"matrix")
as.data.frame(motif)
format(motif)
```

---

pfm2pwm *convert pfm object to PWM*

Description

convert pfm object to PWM

Usage

```
pfm2pwm(x, N = 10000)
```

Arguments

x	an object of pfm or pcm or matrix
N	Total number of event counts used for pfm generation.

Value

A numeric matrix representing the Position Weight Matrix for PWM.

Author(s)

Jianhong Ou

See Also

PWM

Examples

```
library("MotifDb")
matrix.fly <- query(MotifDb, "Dmelanogaster")
pfm2pwm(matrix.fly[[1]])
```

plotAffinityLogo *plot affinity logo*

Description

plot affinity logo

Usage

```
plotAffinityLogo(
  psam,
  motifName,
  font = "sans",
  fontface = "bold",
  colset = c("#00811B", "#2000C7", "#FFB32C", "#D00001"),
  alpha = 0.5,
  newpage = TRUE,
  draw = TRUE
)
```

Arguments

psam	a position-specific affinity matrix
motifName	motif name
font	font of logo
fontface	fontface of logo
colset	color setting for each logo letter
alpha	Alpha channel for transparency of low affinity letters.
newpage	plot in a new canvas or not.
draw	Vector (logical(1)). TRUE to plot. FALSE, return a gList

Value

none

References

Barrett C. Foat, Alexandre V. Morozov, Harmen J. Bussemaker; Statistical mechanical modeling of genome-wide transcription factor occupancy data by MatrixREDUCE, Bioinformatics, Volume 22, Issue 14, 15 July 2006, Pages e141-e149, https://doi.org/10.1093/bioinformatics/btl223

Examples

```
psam <- importMatrix(file.path(find.package("motifStack"), "extdata", "PSAM.mxr"),
                     format="psam")[[1]]
plotAffinityLogo(psam)
```

plotMotifLogo *plot sequence logo*

Description

plot amino acid or DNA sequence logo

Usage

```
plotMotifLogo(
  pfm,
  motifName,
  p = rep(0.25, 4),
  font = "sans",
  fontface = "bold",
  colset = c("#00811B", "#2000C7", "#FFB32C", "#D00001"),
  xaxis = TRUE,
  yaxis = TRUE,
  xlab = "position",
  ylab = "bits",
  xlcex = 1.2,
  ylcex = 1.2,
  ncex = 1.2,
  ic.scale = TRUE,
  newpage = TRUE,
  margins = c(4.1, 4.1, 2.1, 0.1),
  draw = TRUE,
  ...
)
```

Arguments

pfm	a position frequency matrices
motifName	motif name
p	background possibility
font	font of logo
fontface	fontface of logo
colset	color setting for each logo letter
xaxis	draw x-axis or not. If a vector of character or numeric is provided, the function will try to plot the x-axis by setting the labels as the vectors.
yaxis	draw y-axis or not
xlab	x-label, do nothing if set xlab as NA
ylab	y-label, do nothing if set ylab as NA
xlcex	cex value for x-label

ylcex	cex value for y-label
ncex	cex value for motif name
ic.scale	logical If TRUE, the height of each column is proportional to its information content. Otherwise, all columns have the same height. It will also can be set as FALSE followed by a numeric vectors. The format is c(FALSE, scale). If it is FALSE followed by a number (eg c(FALSE, 100)), the y axis labels will be re-scaled by 100.
newpage	logical If TRUE, plot it in a new page.
margins	A numeric vector interpreted in the same way as par(mar) in base graphics.
draw	Vector (logical(1)). TRUE to plot. FALSE, return a gList
...	Not used.

Value

none

Examples

```
pcm<-matrix(runif(40,0,100),nrow=4,ncol=10)
pfm<-pcm2pfm(pcm)
rownames(pfm)<-c("A","C","G","T")
plotMotifLogo(pfm)
```

---

plotMotifLogoA *plot sequence logo without plot.new*

Description

plot amino acid or DNA sequence logo in a given canvas

Usage

```
plotMotifLogoA(
  pfm,
  font = "sans",
  fontface = "bold",
  ic.scale = TRUE,
  draw = TRUE
)
```

Arguments

pfm	an object of pfm
font	font of logo
fontface	fontface of logo
ic.scale	logical If TRUE, the height of each column is proportional to its information content. Otherwise, all columns have the same height.
draw	Vector (logical(1)). TRUE to plot. FALSE, return a gList

Value

none

Examples

```
pcm<-matrix(runif(40,0,100),nrow=4,ncol=10)
pfm<-pcm2pfm(pcm)
rownames(pfm)<-c("A","C","G","T")
motif <- new("pfm", mat=pfm, name="bin_SOLEXA")
plotMotifLogoA(motif)
```

---

plotMotifLogoStack *plot sequence logos stack*

Description

plot sequence logos stack

Usage

```
plotMotifLogoStack(pfms, ...)
```

Arguments

pfms	a list of position frequency matrices, pfms must be a list of class pfm
...	other parameters can be passed to plotMotifLogo function

Value

none

Examples

```
pcm1<-matrix(c(0,50,0,50,
         100,0,0,0,
         0,100,0,0,
         0,0,100,0,
         0,0,0,100,
         50,50,0,0,
         0,0,50,50), nrow=4)
pcm2<-matrix(c(50,50,0,0,
         0,100,0,0,
         0,50,50,0,
         0,0,0,100,
         50,50,0,0,
         0,0,50,50), nrow=4)
rownames(pcm1)<-c("A","C","G","T")
rownames(pcm2)<-c("A","C","G","T")
pfms<-list(p1=new("pfm",mat=pcm2pfm(pcm1),name="m1"),
       p2=new("pfm",mat=pcm2pfm(pcm2),name="m2"))
pfms<-DNAmotifAlignment(pfms)
plotMotifLogoStack(pfms)
```

`plotMotifLogoStackWithTree` *plot sequence logos stack with hierarchical cluster tree*

Description

plot sequence logos stack with hierarchical cluster tree

Usage

```
plotMotifLogoStackWithTree(pfms, hc, treewidth = 1/8, trueDist = FALSE, ...)
```

Arguments

`pfms`	a list of position frequency matrices, pfms must be a list of class pfm
`hc`	an object of the type produced by stats::hclust
`treewidth`	the width to show tree
`trueDist`	logical flags to use hclust height or not.
`...`	other parameters can be passed to plotMotifLogo function

Value

none

Examples

```
#####Input#####
pcms<-readPCM(file.path(find.package("motifStack"), "extdata"),"pcm$")

#####Clustering#####
hc <- clusterMotifs(pcms)

##reorder the motifs for plotMotifLogoStack
motifs<-pcms[hc$order]
motifs <- lapply(motifs, pcm2pfm)
##do alignment
motifs<-DNAmotifAlignment(motifs)
##plot stacks
plotMotifLogoStack(motifs, ncex=1.0)
plotMotifLogoStackWithTree(motifs, hc=hc)
```

`plotMotifOverMotif` *plot motif over another motif*

Description

plot motif over another motif to emphesize the difference.

Usage

```
plotMotifOverMotif(
  motif,
  backgroundMotif,
  bgNoise = NA,
  font = "sans",
  textgp = gpar()
)
```

Arguments

`motif`	an object of `pcm` or `pfm`
`backgroundMotif`	an object of `pcm` or `pfm`
`bgNoise`	if it is not NA, test will using a background by Dirichlet(1)-distributed random frequencies with weight bg.noise. The value of bgNoise should be a number in the range of 0 to 1, eg. 0.05
`font`	font for logo symbol
`textgp`	text parameter

Value

none

Examples

```
pcms <- readPCM(file.path(find.package("motifStack"), "extdata"),"pcm$")
len <- sapply(pcms, function(.ele) ncol(.ele$mat))
pcms <- pcms[len==7]
plotMotifOverMotif(pcms[[1]], pcms[[2]], bgNoise=0.05)
```

`plotMotifStackWithPhylog` *plot sequence logo stacks with a ape4-style phylogenic tree*

Description

plot sequence logo stacks with a ape4-style phylogenic tree

Usage

```
plotMotifStackWithPhylog(
  phylog,
  pfms = NULL,
  f.phylog = 0.3,
  f.logo = NULL,
  cleaves = 1,
  cnodes = 0,
  labels.leaves = names(phylog$leaves),
  clabel.leaves = 1,
  labels.nodes = names(phylog$nodes),
  clabel.nodes = 0,
  font = "sans",
  ic.scale = TRUE,
  ...
)
```

Arguments

`phylog`	an object of class phylog
`pfms`	a list of objects of class pfm
`f.phylog`	a size coefficient for tree size (a parameter to draw the tree in proportion to leaves label)
`f.logo`	a size coefficient for the motif
`cleaves`	a character size for plotting the points that represent the leaves, used with par("cex")*cleaves. If zero, no points are drawn
`cnodes`	a character size for plotting the points that represent the nodes, used with par("cex")*cnodes. If zero, no points are drawn
`labels.leaves`	a vector of strings of characters for the leaves labels

clabel.leaves	a character size for the leaves labels, used with par("cex")*clavel.leaves
labels.nodes	a vector of strings of characters for the nodes labels
clabel.nodes	a character size for the nodes labels, used with par("cex")*clabel.nodes. If zero, no nodes labels are drawn
font	font of logo
ic.scale	logical If TRUE, the height of each column is proportional to its information content. Otherwise, all columns have the same height.
...	not used.

Value

none

See Also

plot.phylog

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
  library("MotifDb")
  matrix.fly <- query(MotifDb, "Dmelanogaster")
  motifs <- as.list(matrix.fly)
  motifs <- motifs[grepl("Dmelanogaster-FlyFactorSurvey-", names(motifs), fixed=TRUE)]
  names(motifs) <- gsub("Dmelanogaster_FlyFactorSurvey_", "",
              gsub("_FBgn[0-9]+$", "",
                gsub("[^a-zA-Z0-9]","_",
                   gsub("(_[0-9]+)+$", "", names(motifs)))))
  motifs <- motifs[unique(names(motifs))]
  pfms <- sample(motifs, 50)
  hc <- clusterMotifs(pfms)
  library(ade4)
  phylog <- ade4::hclust2phylog(hc)
  leaves <- names(phylog$leaves)
  pfms <- pfms[leaves]
  pfms <- mapply(pfms, names(pfms), FUN=function(.ele, .name){
               new("pfm",mat=.ele, name=.name)})
  pfms <- DNAmotifAlignment(pfms, minimalConsensus=3)
  plotMotifStackWithPhylog(phylog, pfms, f.phylog=0.3,
                           cleaves = 0.5, clabel.leaves = 0.7)
}
```

plotMotifStackWithRadialPhylog *plot sequence logo stacks with a radial phylogenic tree*

Description

plot sequence logo stacks with a radial phylogenic tree

Usage

```
plotMotifStackWithRadialPhylog(
  phylog,
  pfms = NULL,
  circle = 0.75,
  circle.motif = NA,
  cleaves = 1,
  cnodes = 0,
  labels.leaves = names(phylog$leaves),
  clabel.leaves = 1,
  labels.nodes = names(phylog$nodes),
  clabel.nodes = 0,
  draw.box = FALSE,
  col.leaves = rep("black", length(labels.leaves)),
  col.leaves.bg = NULL,
  col.leaves.bg.alpha = 1,
  col.bg = NULL,
  col.bg.alpha = 1,
  col.inner.label.circle = NULL,
  inner.label.circle.width = "default",
  col.outer.label.circle = NULL,
  outer.label.circle.width = "default",
  clockwise = FALSE,
  init.angle = if (clockwise) 90 else 0,
  angle = 360,
  pfmNameSpliter = ";",
  rcpostfix = "(RC)",
  motifScale = c("linear", "logarithmic"),
  ic.scale = TRUE,
  plotIndex = FALSE,
  IndexCol = "black",
  IndexCex = 0.8,
  groupDistance = NA,
  groupDistanceLineCol = "red",
  plotAxis = FALSE,
  font = "sans",
  ...
)
```

Arguments

phylog	an object of class phylog
pfms	a list of objects of class pfm
circle	a size coefficient for the outer circle of the labels. Please note this is the position of inner.label.cirle.
circle.motif	a size coefficient for the motif circle
cleaves	a character size for plotting the points that represent the leaves, used with par("cex")*cleaves. If zero, no points are drawn
cnodes	a character size for plotting the points that represent the nodes, used with par("cex")*cnodes. If zero, no points are drawn
labels.leaves	a vector of strings of characters for the leaves labels
clabel.leaves	a character size for the leaves labels, used with par("cex")*clabel.leaves
labels.nodes	a vector of strings of characters for the nodes labels
clabel.nodes	a character size for the nodes labels, used with par("cex")*clabel.nodes. If zero, no nodes labels are drawn
draw.box	if TRUE draws a box around the current plot with the function box()
col.leaves	a vector of colors for leaves labels
col.leaves.bg	a vector of colors for background of leaves labels
col.leaves.bg.alpha	alpha value [0, 1] for the colors of backgroud of leaves labels
col.bg	a vector of colors for tree background
col.bg.alpha	a alpha value [0, 1] of colors for tree background
col.inner.label.circle	a vector of colors for inner cirlce of pfms
inner.label.circle.width	width for inner circle of pfms
col.outer.label.circle	a vector of colors for outer circle of pfms
outer.label.circle.width	width for outer circle of pfms
clockwise	a logical value indicating if slices are drawn clockwise or counter clockwise
init.angle	number specifying the starting angle (in degrees) for the slices. Defaults to 0 (i.e., '3 o'clock') unless clockwise is true where init.angle defaults to 90 (degrees), (i.e., '12 o'clock')
angle	number specifying the angle (in degrees) for phylogenic tree. Defaults 360
pfmNameSpliter	spliter when name of pfms contain multiple node of labels.leaves
rcpostfix	the postfix for reverse complements
motifScale	the scale of logo size
ic.scale	logical. If TRUE, the height of each column is proportional to its information content. Otherwise, all columns have the same height.

plotIndex	logical. If TRUE, will plot index number in the motifLogo which can help user to describe the motifLogo
IndexCol	The color of the index number when plotIndex is TRUE.
IndexCex	The cex of the index number when plotIndex is TRUE.
groupDistance	show groupDistance on the draw
groupDistanceLineCol	groupDistance line color, default: red
plotAxis	logical. If TRUE, will plot distance axis.
font	font of logo
...	not used.

Value

none

See Also

plot.phylog

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
  library("MotifDb")
  matrix.fly <- query(MotifDb, "Dmelanogaster")
  motifs <- as.list(matrix.fly)
  motifs <- motifs[grepl("Dmelanogaster-FlyFactorSurvey-",
                         names(motifs), fixed=TRUE)]
  names(motifs) <- gsub("Dmelanogaster_FlyFactorSurvey_", "",
              gsub("_FBgn[0-9]+$", "",
                gsub("[^a-zA-Z0-9]","_",
                   gsub("(_[0-9]+)+$", "", names(motifs)))))
  motifs <- motifs[unique(names(motifs))]
  pfms <- sample(motifs, 50)
  hc <- clusterMotifs(pfms)
  library(ade4)
  phylog <- ade4::hclust2phylog(hc)
  leaves <- names(phylog$leaves)
  pfms <- pfms[leaves]
  pfms <- mapply(pfms, names(pfms), FUN=function(.ele, .name){
              new("pfm",mat=.ele, name=.name)})
  pfms <- DNAmotifAlignment(pfms, minimalConsensus=3)
  library(RColorBrewer)
  color <- brewer.pal(12, "Set3")
  plotMotifStackWithRadialPhylog(phylog, pfms, circle=0.9,
              cleaves = 0.5, clabel.leaves = 0.7,
              col.bg=rep(color, each=5), col.leaves=rep(color, each=5))
}
```

plotXaxis *plot x-axis*

Description

plot x-axis for the sequence logo

Usage

```
plotXaxis(pfm, p = rep(0.25, 4), label = NULL)
```

Arguments

pfm	position frequency matrices
p	background possibility
label	x-axis labels

Value

none

plotYaxis *plot y-axis*

Description

plot y-axis for the sequence logo

Usage

```
plotYaxis(ymax, ic.scale)
```

Arguments

ymax	max value of y axix
ic.scale	Use IC scale or not. See plotMotifLogo for help.

Value

none

psam-class *Class* "psam"

Description

An object of class "psam" represents the position specific affinity matrix (PSAM) of a DNA/RNA/amino-acid sequence motif. The entry stores a matrix, which in row i, column j gives the affinity of observing nucleotide/or amino acid i in position j of the motif.

methods for psam objects.

Usage

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

## S4 method for signature 'psam,ANY'
plot(x, y = "missing", ...)

## S4 method for signature 'psam'
matrixReverseComplement(x)

## S4 method for signature 'psam,numeric,logical'
addBlank(x, n, b)

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

## S4 method for signature 'psam'
format(x, ...)
```

Arguments

x	An object of class psam.
name	Slot name.
y	Not use.
...	Further potential arguments passed to plotAffinityLogo.
n	how many spaces should be added.
b	logical value to indicate where the space should be added.
row.names, optional	see as.data.frame

Objects from the Class

Objects can be created by calls of the form new("psam", mat, name, alphabet, color).

Methods

addBlank `signature(x="psam",n="numeric", b="logical")` add space into the position specific affinity matrix for alignment. b is a bool value, if TRUE, add space to the 3' end, else add space to the 5' end. n indicates how many spaces should be added.

matrixReverseComplement `signature(x = "psam")` get the reverse complement of position specific affinity matrix.

plot `signature(x = "psam")` Plots the affinity logo of the position specific affinity matrix.

$, $<- Get or set the slot of psam-class

as.data.frame convert psam-class to a data.frame

format return the name_pfm of psam-class

Examples

```
motif <- importMatrix(file.path(find.package("motifStack"), "extdata", "PSAM.mxr"),
         format="psam")[[1]]
plot(motif)


motif <- importMatrix(file.path(find.package("motifStack"), "extdata", "PSAM.mxr"),
                      format="psam")[[1]]
matrixReverseComplement(motif)
addBlank(motif, 1, FALSE)
addBlank(motif, 3, TRUE)
as(motif,"matrix")
as.data.frame(motif)
format(motif)
```

---

pssm-class *Class* "pssm"

Description

An object of class "pssm" represents the position specific score matrix of a DNA/RNA/amino-acid sequence motif. The entry stores a matrix, which in row i, column j gives the log-odds probability of nucleotide/or amino acid i in position j of the motif.

methods for pssm objects.

Usage

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

## S4 method for signature 'pssm,ANY'
plot(x, y = "missing", ...)
```

```
## S4 method for signature 'pssm'
matrixReverseComplement(x)

## S4 method for signature 'pssm,numeric,logical'
addBlank(x, n, b)

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

## S4 method for signature 'pssm'
format(x, ...)
```

Arguments

x	An object of class pssm. For getIC, if parameter p is followed, x should be an object of matrix.
name	Slot name.
y	Not use.
...	Further potential arguments passed to plotMotifLogo.
n	how many spaces should be added.
b	logical value to indicate where the space should be added.
row.names, optional	see as.data.frame

Objects from the Class

Objects can be created by calls of the form new("pssm", mat, name, alphabet, color, background).

Methods

addBlank signature(x="pssm",n="numeric", b="logical") add space into the position frequency matrix for alignment. b is a bool value, if TRUE, add space to the 3' end, else add space to the 5' end. n indicates how many spaces should be added.

matrixReverseComplement signature(x = "pssm") get the reverse complement of position frequency matrix.

plot signature(x = "pssm") Plots the sequence logo of the position frequency matrix.

$, $<- Get or set the slot of pssm-class

as.data.frame convert pssm-class to a data.frame

format return the name_pssm of pssm-class

Examples

```
pcm <- read.table(file.path(find.package("motifStack"),
                            "extdata", "bin_SOLEXA.pcm"))
pcm <- pcm[,3:ncol(pcm)]
rownames(pcm) <- c("A","C","G","T")
```

```
motif <- pcm2pssm(pcm)
motif <- new("pssm", mat=motif, name="bin_SOLEXA")
plot(motif)


pcm <- read.table(file.path(find.package("motifStack"),
                  "extdata", "bin_SOLEXA.pcm"))
pcm <- pcm[,3:ncol(pcm)]
rownames(pcm) <- c("A","C","G","T")
motif <- pcm2pssm(pcm)
motif <- new("pssm", mat=motif, name="bin_SOLEXA")
matrixReverseComplement(motif)
addBlank(motif, 1, FALSE)
addBlank(motif, 3, TRUE)
as(motif,"matrix")
as.data.frame(motif)
format(motif)
```

---

readPCM *read pcm from a path*

Description

read position count matrix from a path

Usage

```
readPCM(path = ".", pattern = NULL)
```

Arguments

path	a character vector of full path names
pattern	an optional regular expression

Value

A list of pcm objects

Examples

```
pcms<-readPCM(file.path(find.package("motifStack"), "extdata"),"pcm$")
```

reorderUPGMAtree *re-order UPGMA tree*

Description

re-order the UPGMA tree by adjacent motif distance

Usage

```
reorderUPGMAtree(phylog, motifs, rcpostfix = "(RC)")
```

Arguments

phylog	an object of phylog
motifs	a list of objects of pfm
rcpostfix	the postfix for reverse complements

Value

an object of phylog

Author(s)

Jianhong Ou

Examples

```
if(interactive() || Sys.getenv("USER")=="jianhongou"){
  library("MotifDb")
  matrix.fly <- query(MotifDb, "Dmelanogaster")
  motifs <- as.list(matrix.fly)
  motifs <- motifs[grepl("Dmelanogaster-FlyFactorSurvey-", names(motifs), fixed=TRUE)]
  names(motifs) <- gsub("Dmelanogaster_FlyFactorSurvey_", "",
              gsub("_FBgn[0-9]+$", "",
                gsub("[^a-zA-Z0-9]","_",
                   gsub("(_[0-9]+)+$", "", names(motifs)))))
  motifs <- motifs[unique(names(motifs))]
  pfms <- sample(motifs, 50)
  hc <- clusterMotifs(pfms)
  library(ade4)
  phylog <- ade4::hclust2phylog(hc)
  pfms <- mapply(pfms, names(pfms), FUN=function(.ele, .name){
              new("pfm",mat=.ele, name=.name)})
  reorderUPGMAtree(phylog, pfms)
}
```

traceBackGlobal *traceback global*

Description

traceback global

Usage

```
traceBackGlobal(dpScore, score, m, n)
```

Arguments

dpScore	Dynamic programming score
score	ALLR scores
m, n	matrix width

Value

a data.frame

traceBackLocal *traceback local*

Description

traceback local

Usage

```
traceBackLocal(dpScore, score, m, n)
```

Arguments

dpScore	Dynamic programming score matrix
score	ALLR scores, m x n matrix
m, n	matrix width

Value

a data.frame

Index

∗ **classes**
  marker-class, 19
  motifSig-class, 31
  ouNode-class, 34
  pcm-class, 35
  pfm-class, 37
  psam-class, 52
  pssm-class, 53
∗ **datasets**
  GeomMotif, 12
∗ **misc**
  getRankedUniqueMotifs, 16
  highlightCol, 17
  importMatrix, 18
  mergeMotifs, 20
  motifCircos, 21
  motifPiles, 28
  pfm2pwm, 39
  reorderUPGMAtree, 56
∗ **package**
  motifStack-package, 3
∗ **plot**
  browseMotifs, 4
$,marker-method *(marker-class)*, 19
$,motifSig-method *(motifSig-class)*, 31
$,ouNode-method *(ouNode-class)*, 34
$,pcm-method *(pcm-class)*, 35
$,pfm-method *(pfm-class)*, 37
$,psam-method *(psam-class)*, 52
$,pssm-method *(pssm-class)*, 53
$<-,marker-method *(marker-class)*, 19
$<-,motifSig-method *(motifSig-class)*, 31
$<-,ouNode-method *(ouNode-class)*, 34
$<-,pcm-method *(pcm-class)*, 35
$<-,pfm-method *(pfm-class)*, 37
$<-,psam-method *(psam-class)*, 52
$<-,pssm-method *(pssm-class)*, 53

AAmotifAlignment, 3
addBlank *(pcm-class)*, 35
addBlank,pcm,numeric,logical-method *(pcm-class)*, 35
addBlank,pfm,numeric,logical-method *(pfm-class)*, 37
addBlank,psam,numeric,logical-method *(psam-class)*, 52
addBlank,pssm,numeric,logical-method *(pssm-class)*, 53
as *(pcm-class)*, 35
as.data.frame,pfm-method *(pfm-class)*, 37
as.data.frame,psam-method *(psam-class)*, 52
as.data.frame,pssm-method *(pssm-class)*, 53

browseMotifs, 4
browseMotifs-shiny, 5
browseMotifsOutput *(browseMotifs-shiny)*, 5

calF, 6
calI, 6
clusterMotifs, 7
coerce *(pfm-class)*, 37
coerce,pcm,matrix-method *(pcm-class)*, 35
coerce,pfm,matrix-method *(pfm-class)*, 37
coerce,psam,matrix-method *(psam-class)*, 52
coerce,pssm,matrix-method *(pssm-class)*, 53
colors, *17*
colorset, 7
compare2profiles, 8
compareProfiles, 9

DNAmotifAlignment, 9, *33*
DNAmotifToRNAmotif, 10
dpGlobal, 11
dpLocal, 11

format,pcm-method *(pcm-class)*, 35

format,pfm-method (pfm-class), 37
format,psam-method (psam-class), 52
format,pssm-method (pssm-class), 53
frequence (motifSig-class), 31
frequence,motifSig-method (motifSig-class), 31

geom_motif, 13
GeomMotif, 12
getALLRscoreFromCounts, 15
getDistance, 15
getIC (pcm-class), 35
getIC,matrix,matrix-method (pfm-class), 37
getIC,matrix,numeric-method (pfm-class), 37
getIC,pcm,ANY-method (pcm-class), 35
getIC,pfm,ANY-method (pfm-class), 37
getRankedUniqueMotifs, 16
getScore, 17
GraphvizLayouts, *4*

hclust, *27*
highlightCol, 17

importMatrix, 18

marker (marker-class), 19
marker-class, 19
matalign, *7*, 19
matrixReverseComplement (pcm-class), 35
matrixReverseComplement,pcm-method (pcm-class), 35
matrixReverseComplement,pfm-method (pfm-class), 37
matrixReverseComplement,psam-method (psam-class), 52
matrixReverseComplement,pssm-method (pssm-class), 53
mergeMotifs, 20
motifCircos, 21, *30*
motifCloud, 24
motifGrob, 26
motifHclust, 27
motifPiles, 28
motifSig, *25*, *31*, *32*
motifSig (motifSig-class), 31
motifSig-class, 31
motifSignature, *31*, 32
motifStack, 33
motifStack-package, 3

nodelist (motifSig-class), 31
nodelist,motifSig-method (motifSig-class), 31

ouNode (ouNode-class), 34
ouNode-class, 34

palette, *17*
pcm, *20*, *39*, *45*, *55*
pcm (pcm-class), 35
pcm-class, *18*, 35
pcm2pfm (pcm-class), 35
pcm2pfm,data.frame,ANY-method (pcm-class), 35
pcm2pfm,data.frame,numeric-method (pcm-class), 35
pcm2pfm,list,ANY-method (pcm-class), 35
pcm2pfm,list,numeric-method (pcm-class), 35
pcm2pfm,matrix,ANY-method (pcm-class), 35
pcm2pfm,matrix,numeric-method (pcm-class), 35
pcm2pfm,pcm,ANY-method (pcm-class), 35
pcm2pssm (pcm-class), 35
pcm2pssm,data.frame,ANY-method (pcm-class), 35
pcm2pssm,data.frame,numeric-method (pcm-class), 35
pcm2pssm,list,ANY-method (pcm-class), 35
pcm2pssm,list,numeric-method (pcm-class), 35
pcm2pssm,matrix,ANY-method (pcm-class), 35
pcm2pssm,matrix,numeric-method (pcm-class), 35
pcm2pssm,pcm,ANY-method (pcm-class), 35
pfm, *4*, *20*, *33*, *39*, *45*
pfm (pfm-class), 37
pfm-class, *18*, 37
pfm2pwm, 39
plot (pcm-class), 35
plot,pcm,ANY-method (pcm-class), 35
plot,pfm,ANY-method (pfm-class), 37
plot,psam,ANY-method (psam-class), 52
plot,pssm,ANY-method (pssm-class), 53

plot.phylog, *47*, *50*
plotAffinityLogo, 40
plotMotifLogo, 41
plotMotifLogoA, 42
plotMotifLogoStack, *33*, 43
plotMotifLogoStackWithTree, *33*, 44
plotMotifOverMotif, 45
plotMotifStackWithPhylog, *33*, 46
plotMotifStackWithRadialPhylog, *23*, *33*, 48
plotXaxis, 51
plotYaxis, 51
psam *(*psam-class*)*, 52
psam-class, 52
pssm *(*pssm-class*)*, 53
pssm-class, 53
PWM, *39*

readPCM, 55
renderbrowseMotifs *(*browseMotifs-shiny*)*, 5
reorderUPGMAtree, 56
rgb, *17*

sigColor *(*motifSig-class*)*, 31
sigColor,motifSig-method *(*motifSig-class*)*, 31
signatures *(*motifSig-class*)*, 31
signatures,motifSig-method *(*motifSig-class*)*, 31

traceBackGlobal, 57
traceBackLocal, 57
trimMotif *(*pcm-class*)*, 35
trimMotif,pcm,numeric-method *(*pcm-class*)*, 35
trimMotif,pfm,numeric-method *(*pfm-class*)*, 37

XMatrix, *18*
XMatrixList, *18*