Package 'TFBSTools'

November 13, 2026

Version 1.51.0

Date 2025-07-22

Title Software Package for Transcription Factor Binding Site (TFBS) Analysis

Description TFBSTools is a package for the analysis and manipulation of transcription factor binding sites. It includes matrices conversion between Position Frequency Matirx (PFM), Position Weight Matirx (PWM) and Information Content Matrix (ICM). It can also scan putative TFBS from sequence/alignment, query JASPAR database and provides a wrapper of de novo motif discovery software.

VignetteBuilder knitr

Imports Biobase(>= 2.28), Biostrings(>= 2.36.4), pwalign, BiocGenerics(>= 0.14.0), BiocParallel(>= 1.2.21), BSgenome(>= 1.36.3), caTools(>= 1.17.1), DirichletMultinomial(>= 1.10.0), Seqinfo, GenomicRanges(>= 1.20.6), gtools(>= 3.5.0), grid, IRanges(>= 2.2.7), methods, DBI (>= 0.6), RSQLite(>= 1.0.0), rtracklayer(>= 1.28.10), seqLogo(>= 1.34.0), S4Vectors(>= 0.9.25), TFMPvalue(>= 0.0.5), XML(>= 3.98-1.3), XVector(>= 0.8.0), parallel

Depends R (>= 3.2.2)

Suggests BiocStyle(>= 1.7.7), JASPAR2014(>= 1.4.0), knitr(>= 1.11), testthat, JASPAR2016(>= 1.0.0), JASPAR2018(>= 1.0.0), rmarkdown

License GPL-2

URL https://github.com/ge11232002/TFBSTools

BugReports https://github.com/ge11232002/TFBSTools/issues

Type Package

biocViews MotifAnnotation, GeneRegulation, MotifDiscovery, Transcription, Alignment

NeedsCompilation yes

LazyData yes

Collate AllGenerics.r AllClasses.r show-methods.r util-methods.r XMatrix-methods.r XMatrixList-methods.r SiteSet-methods.r coercion-methods.r DB-methods.r JASPAR.R PairwiseAlignment-methods.r ICM-methods.r PWM-methods.r

Motif-methods.r PFM-methods.r Wrappers-methods.r
DirichletMixture.r TFFM.R TFFMseqLogo.R IO.R Enrichment.R

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

git_branch devel

git_last_commit f7b3785

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-11-12

Author Ge Tan [aut, cre]

Maintainer Ge Tan <ge_tan@live.com>

Contents

TFBSTools-package . . . 3
calConservation-methods . . . 3
deleteMatrixHavingID . . . 4
dmmEM-methods . . . 5
getEmissionProb . . . 6
getMatrixByID . . . 7
getMatrixSet . . . 8
getPosProb . . . 10
IUPAC2Matrix . . . 11
MA0004.1 . . . 12
makeFlatFileDir . . . 12
MotifSet . . . 13
parseMEMEOutput . . . 14
permuteMatrix-methods . . . 15
PFMSimilarity-methods . . . 16
PWMSimilarity-methods . . . 17
readJASPARMatrix . . . 18
readXMLTFFM . . . 19
rPWMDmm-methods . . . 19
runMEME . . . 21
sampleRanges . . . 22
searchAln . . . 23
searchPairBSgenome-methods . . . 26
searchSeq . . . 27
seqLogo . . . 29
shannon.entropy . . . 30
SitePairSet . . . 31
SitePairSetList-class . . . 32
SiteSet . . . 33
SiteSetList . . . 34
TFFM . . . 35
toGRangesList-methods . . . 37
toICM . . . 37
toPWM . . . 39
writeGFF3-methods . . . 41
XMatrix . . . 41
XMatrixList . . . 44

Index **46**

`TFBSTools-package` *TFBS software package.*

Description

TFBS includes a set of tools for transcription factor binding site detection and analysis as well as database interface functions for JASPAR, etc.

Author(s)

Ge Tan

`calConservation-methods`
calConservation method

Description

Calculate the conservation score for a pairwise alignment given a smooth window size.

Usage

```
calConservation(aln1, aln2, windowSize=51L, which="1")
```

Arguments

`aln1`	A `DNAString` object , a `DNAStringSet` or a `character` object, which contains the pairwise alignments. When the last two objects have a length of 2, the argument `aln2` can be missing.
`aln2`	A `DNAString`, a `character` object or missing.
`windowSize`	The size of the sliding window (in nucleotides) for calculating local conservation in the alignment. This should be an odd value.
`which`	The conservation profile of Which sequence in the alignments is computed. It can be "1" or "2".

Value

A `numeric` vector with the same length of alignment is returned.

Author(s)

Ge Tan

See Also

`searchAln`

deleteMatrixHavingID *JASPAR database operations*

Description

The functions to initialize, store matrix or delete matrix in JASPAR database.

Usage

```
  ## S4 method for signature 'character'
deleteMatrixHavingID(x, IDs)
  ## S4 method for signature 'SQLiteConnection'
deleteMatrixHavingID(x, IDs)
  ## S4 method for signature 'JASPAR2014'
deleteMatrixHavingID(x, IDs)
  ## S4 method for signature 'character,PFMatrixList'
storeMatrix(x, pfmList)
  ## S4 method for signature 'SQLiteConnection,PFMatrixList'
storeMatrix(x, pfmList)
  ## S4 method for signature 'JASPAR2014,PFMatrixList'
storeMatrix(x, pfmList)
  ## S4 method for signature 'character,PFMatrix'
storeMatrix(x, pfmList)
  ## S4 method for signature 'SQLiteConnection,PFMatrix'
storeMatrix(x, pfmList)
  ## S4 method for signature 'JASPAR2014,PFMatrix'
storeMatrix(x, pfmList)
  ## S4 method for signature 'SQLiteConnection'
initializeJASPARDB(x, version=c("2014", "2016", "2018", "2020", "2022"))
  ## S4 method for signature 'character'
initializeJASPARDB(x, version=c("2014", "2016", "2018", "2020", "2022"))
  ## S4 method for signature 'JASPAR2014'
initializeJASPARDB(x, version)
  ## S4 method for signature 'JASPAR2016'
initializeJASPARDB(x, version)
  ## S4 method for signature 'JASPAR2018'
initializeJASPARDB(x, version)
  ## S4 method for signature 'JASPAR2020'
initializeJASPARDB(x, version)
```

Arguments

x	A character vector of length 1 for the path of JASPAR SQLite file, or a SQLiteConnection object.
IDs	JASPAR stable IDs.
pfmList	The PFMatrixList object, or pfm object.
version	Which version of JASPAR to create. So far, it supports 2014, 2016 and 2018.

Value

If the operation works, a "success" will be returned.

Examples

```
initializeJASPARDB("jaspar.sqlite", version="2014")
data("MA0043")
storeMatrix("jaspar.sqlite", MA0043)
deleteMatrixHavingID("jaspar.sqlite","MA0043.1")
file.remove("jaspar.sqlite")
```

---

dmmEM-methods *dmmEM method*

Description

This function trains the Dirichlet multinomial mixture models parameters for a set of profile matrices.

Usage

```
dmmEM(x, K=6, alg=c("C", "R"))
```

Arguments

x	x can be a matrix, PFMatrixList or JASPAR2014 to be trained.
K	The maximal number of components to test in the mixture model when alg is "C". Then an optimal number of components between 1 and K will be chosen based on the fitness of the model. The fixed number of components to use when alg is "R". The default is 6.
alg	The algorithm to use. "C" uses the implementation from DirichletMultinomial package which has more advanced feature and performance. "R" uses our own implemention in R.

Details

When using the implementation from DirichletMultinomial package, the final number of components can be 1:K. An internal selection will be made based on the maximum likelihood.

When using the implementation of R, the number of component is fixed to K.

Value

A list of trainned alpha0, pmix and likelihood during the training.

Methods

signature(x = "ANY")

signature(x = "matrix")

signature(x = "PFMatrixList")

Author(s)

Ge Tan

See Also

rPWMDmm

Examples

```
data(MA0003.2)
data(MA0004.1)
pfmList <- PFMatrixList(pfm1=MA0003.2, pfm2=MA0004.1, use.names=TRUE)
dmmParameters <- dmmEM(pfmList, K=6, alg="C")
```

---

getEmissionProb *Get the emission distribution parameters.*

Description

This function accesses the emission distribution parameters of the TFFM.

Usage

```
getEmissionProb(tffm)
```

Arguments

tffm	A TFFMFirst object or a TFFMDetail object.

Details

This function accesses the emission distribution parameters for each position of the TFFM. It returns the probability of emitting certain nucleotide based on the nucleotide on the previous site.

Value

A matrix of numeric with dimensions of 16 * ncol(tffm).

Author(s)

Ge Tan

See Also

getPosProb

Examples

```
xmlFirst <- file.path(system.file("extdata", package="TFBSTools"),
                      "tffm_first_order.xml")
tffmFirst <- readXMLTFFM(xmlFirst, type="First")
getEmissionProb(tffmFirst)

xmlDetail <- file.path(system.file("extdata", package="TFBSTools"),
                       "tffm_detailed.xml")
tffmDetail <- readXMLTFFM(xmlDetail, type="Detail")
getEmissionProb(tffmDetail)
```

---

getMatrixByID *Basic JASPAR database search functionis* getMatrixByID, getMatrixByName

---

Description

This method fetches matrix data under the given ID or name from the database and returns a XMatrix object.

Usage

```
  ## S4 method for signature 'character'
getMatrixByID(x, ID)
  ## S4 method for signature 'SQLiteConnection'
getMatrixByID(x, ID)
  ## S4 method for signature 'JASPAR2014'
getMatrixByID(x, ID)
  ## S4 method for signature 'character'
getMatrixByName(x, name)
  ## S4 method for signature 'SQLiteConnection'
getMatrixByName(x, name)
  ## S4 method for signature 'JASPAR2014'
getMatrixByName(x, name)
```

Arguments

x	character(1) for the path of JASPAR SQLite file, a SQLiteConnection object, a JASPAR2014, or a JASPAR2016object.
ID	character() of JASPAR stable ID(s). See more details below.
name	character() of JASPAR stable name(s).

Details

For getMatrixByID, ID is a string which refers to the stable JASPAR ID (usually something like "MA0001") with or without version numbers. "MA0001" will give the latest version on MA0001, while "MA0001.2" will give the second version, if existing.

For getMatrixByName, according to the current JASPAR data model, name is not necessarily a unique identifier. Also, names change over time. In the case where there are several matrices with the same name in the database, the function fetches the first one and prints a warning. You've been warned. Some matrices have multiple versions. The function will return the latest version. For specific versions, use getMatrixByID(ID.version)

Value

A PFMMatrix object is returned when input ID or name is length 1. Otherwise, `PFMMatrixList` is returned.

Author(s)

Ge Tan

See Also

`getMatrixSet`

Examples

```
library(JASPAR2014)
db <- file.path(system.file("extdata", package="JASPAR2014"),
                "JASPAR2014.sqlite")

## character and ID
pfm <- getMatrixByID(db, ID="MA0003")

## character and IDs
pfmList <- getMatrixByID(db, ID=c("MA0003", "MA0004"))

## character and name
pfm <- getMatrixByName(db, name="TFAP2A")

##
## character and name
pfmList <- getMatrixByName(db, name=c("TFAP2A", "Arnt"))

## JASPAR2014 and ID
pfm <- getMatrixByID(JASPAR2014, ID="MA0003")
```

`getMatrixSet` *Advanced JASPAR database search functions* `get_MatrixSet`

Description

This function fetches matrix data for all matrices in the database matching criteria defined by the named arguments and returns a PFMatrixList object

Usage

```
  ## S4 method for signature 'character'
getMatrixSet(x, opts)
  ## S4 method for signature 'SQLiteConnection'
getMatrixSet(x, opts)
  ## S4 method for signature 'JASPAR2014'
getMatrixSet(x, opts)
```

Arguments

x	a character vector of length 1 for the path of JASPAR SQLite file, a SQLiteConnection object, or a JASPAR2014 object.
opts	a search options list. See more details below.

Details

The search options include three categories:

(1) Database basic criterias:

all=c(TRUE, FALSE)

ID: a unique identifier for each model. CORE matrices always have a "MAnnnnIDs.Version".

name: The name of the transcription factor. As far as possible, the name is based on the standardized Entrez gene symbols. In the case the model describes a transcription factor hetero-dimer, two names are concatenated, such as RXR-VDR. In a few cases, different splice forms of the same gene have different binding specificity: in this case the splice form information is added to the name, based on the relevant literature.

collection=c("CORE", "CNE", "PHYLOFACTS", "SPLICE", "POLII", "FAM", "PBM", "PBM_HOMEO", "PBM_HLH", "UNVALIDATED"

all_versions=c(FALSE,TRUE): We constantly update the profiles in JASPAR. Some profiles may have multiple versions. By default, only the latest version will be returned.

species: The species source for the sequences, in Latin (Homo sapiens) or NCBI tax IDs (9606).

matrixtype=c("PFM", "PWM", "ICM")

(2) Tags based criterias:

class: Structural class of the transcription factor, based on the TFCaT system. Examples: "Zipper-Type"", "Helix-Turn-Helix", etc.

type: Methodology used for matrix construction: "SELEX", "ChIP-seq", "PBM", etc.

tax_group: Group of species, currently consisting of "plants", "vertebrates", "insects", "urochordat", "nematodes", "fungi".

family: Structural sub-class of the transcription factor, based on the TFCaT system.

Acc: A representative protein accession number in Genbank for the transcription factor. Human takes precedence if several exists.

medline: relevant publication reporting the sites used in the mode building.

Pazar_tf_id: PAZAR database id.

(3) Further criterias:

min_ic (minimum total information content of the matrix)

length (minimum sites length)

sites (minimum average sites number per base)

When all is TRUE, it will get all the matrices and has higher priority over other options. Then ID has the second highest priority, and will ignore all the followiing options. The rest options are combined in search with AND, while multiple elements under one options have the logical operator OR.

Value

A PFMatrixList object.

Author(s)

Ge Tan

See Also

getMatrixByID, getMatrixByName

Examples

```
library(JASPAR2014)
db <- file.path(system.file("extdata", package="JASPAR2014"),
                "JASPAR2014.sqlite")
opts <- list()
opts[["species"]] <- 9606
opts[["type"]] <- "SELEX"
opts[["all_versions"]] <- FALSE
siteList <- getMatrixSet(db, opts)
siteList2 <- getMatrixSet(JASPAR2014, opts)
```

---

getPosProb *Get the emission probabilities of nucleotides*

Description

Get the emission probabilities of ACGT at each position of TFFM.

Usage

```
getPosProb(tffm)
```

Arguments

tffm	A TFFMFirst object or a TFFMDetail object.

Details

This function calculates the probabilities of emitting nucleotides ACGT at each position of TFFM.

Value

A matrix of numeric with dimensions of 4 * ncol(tffm).

Author(s)

Ge Tan

See Also

getEmissionProb

Examples

```
xmlFirst <- file.path(system.file("extdata", package="TFBSTools"),
                      "tffm_first_order.xml")
tffmFirst <- readXMLTFFM(xmlFirst, type="First")
getPosProb(tffmFirst)

xmlDetail <- file.path(system.file("extdata", package="TFBSTools"),
                       "tffm_detailed.xml")
tffmDetail <- readXMLTFFM(xmlDetail, type="Detail")
getPosProb(tffmDetail)
```

IUPAC2Matrix *IUPAC2Matrix*

Description

Convert a IUPAC string into a Postion Weight Matirx

Usage

```
IUPAC2Matrix(x)
```

Arguments

x	The IUPAC string.

Details

The mapping between IUPAC Extended Genetic Alphabet and the DNA bases letters are from `IUPAC_CODE_MAP` in Biostrings package.

Value

A matrix with position weight.

Author(s)

Ge Tan

Examples

```
x <- "RMGNV"
IUPAC2Matrix(x)
```

MA0004.1 *Some example PFM matrices.*

Description

Some example PFM matrices from JASPAR 2014.

Usage

```
data(MA0004.1)
data(MA0003.2)
data(MA0048)
data(MA0043)
```

Format

The format is: `PFMatrix` object.

Details

Some examples PFM matrices from JASPAR 2014.

Value

The `PFMatrix` object.

Source

http://jaspar.genereg.net/

Examples

```
data(MA0004.1)
data(MA0003.2)
data(MA0048)
data(MA0043)
```

makeFlatFileDir *Generate "FlatFileDir" directory*

Description

On JASPAR web service, "FlatFileDir" includes all the *.pfm and a matrix_list.txt file

Usage

```
makeFlatFileDir(JASPAR)
```

Arguments

JASPAR	A JASPAR object. Now it can be JASPAR2014 or JASPAR2016.

Details

The matrix_list.txt file contains each pfm per line. Each line has the ID, total information content, name, class and tags of one pfm.

Value

The generated files are under "FlatFileDir" directory.

Author(s)

Ge Tan

Examples

```
library(JASPAR2014)
makeFlatFileDir(JASPAR2014)
unlink("FlatFileDir", recursive = TRUE)
```

MotifSet *Class* "MotifSet"

Description

This MotifSet object is a container for storing the generated motifs from Motif identification softwares, such as MEME.

Usage

```
## Constructor
MotifSet(motifList=GRangesList(), motifEvalues=numeric(),
         subjectSeqs=DNAStringSet())
```

Arguments

motifList	A GRangesList. Each GRanges store the starts, ends, strand, seqnames and scores information of one motif sites sequences.
motifEvalues	A numeric vector of the E values generated from MEME for each motif.
subjectSeqs	A DNAStringSet object. It stores the original sequences which are scanned by the software.

Value

A MotifSet object is returned.

Methods

[`signature(x = "MotifSet")`: Getter

consensusMatrix `signature(x = "MotifSet")`(x, as.prob = FALSE, shift = 0L, width = NULL, ...): Calculate the consensus matrix. Other arguments, please check the `consensusMatrix` in `Biostrings` package.

length `signature(x = "MotifSet")`: Returns the number of motifs.

sitesSeq `signature(x = "MotifSet")`(x, n=10L, type="none"): Gets the sites sequences.

`n` is the number of bases to include from flanking region.

`type` controls "all", "left", "right" or "none" flanking sequences are included.

Author(s)

Ge Tan

See Also

`runMEME`

Examples

```
## Not run:
  motifSet <- runMEME(file.path(system.file("extdata", package="TFBSTools"),
                      "crp0.s"),
                      binary="/usr/local/Cellar/meme/4.10.1/bin/meme",
                      arguments=list("-nmotifs"=3))
  sitesSeq(motifSet, type="all")
  sitesSeq(motifSet, type="none")
  consensusMatrix(motifSet)

## End(Not run)
```

---

parseMEMEOutput *parseMEMEOutput*

Description

Parse the output file from "MEME".

Usage

```
parseMEMEOutput(x)
```

Arguments

x	`character`(1): the filename of the "MEME" output.

Value

A list of motifs and evalues is returned.

Author(s)

Ge Tan

See Also

runMEME

Examples

```
memeOutput <- file.path(system.file("extdata", package="TFBSTools"),
                        "meme.output")
parseMEMEOutput(memeOutput)
```

---

permuteMatrix-methods *permuteMatrix method*

Description

This method simply shuffles the columns in matrices. This can either be done by just shuffling columns within each selected matrix, or by shuffling columns almong all selected matrices.

Usage

```
permuteMatrix(x, type="intra")
```

Arguments

x	A matrix which meets the PFM standard, PFMatrix object, or PFMatrixList obejct.
type	The type of shuffling. It can be "intra" or "inter", which shuffle within each matrix, or between all the matrix.

Value

A object with shuffled matrix.

Author(s)

Ge Tan

Examples

```
data("MA0043")
pfmSubject <- MA0043
data("MA0048")
pfmQuery <- MA0048
#opts = list()
#opts[["class"]] = "Ig-fold"
#pfmList = getMatrixSet(JASPAR2014, opts)
pfmList <- PFMatrixList(pfmSubject, pfmQuery)
foo = permuteMatrix(pfmQuery)
foo1 = permuteMatrix(pfmList, type="intra")
foo2 = permuteMatrix(pfmList, type="inter")
```

PFMSimilarity-methods *PFMSimilarity method*

Description

Given a PFMatrix or a normal matrix, align it with another set of PFMatrix to assess the similarity.

Usage

```
PFMSimilarity(pfmSubject, pfmQuery, openPenalty=3, extPenalty=0.01)
```

Arguments

pfmSubject	A matrix, PFMatrix or PFMatrixList object, which is compared with query matrix.
pfmQuery	A matrix, PFMatrix or IUPAC character object.
openPenalty	The gap open penalty used in the modified Needleman-Wunsch algorithm. By default, it is 3.
extPenalty	The gap extension penalty used in the modified Needleman-Wunsch algorithm. By default, it is 0.01.

Value

For each pfmSubject, an absolute score and a relative percentage score is returned. The maximum absolute score is 2*the width of the smaller matrix in the comparison pair.

Author(s)

Ge Tan

References

Sandelin, A., H glund, A., Lenhard, B., & Wasserman, W. W. (2003). Integrated analysis of yeast regulatory sequences for biologically linked clusters of genes. Functional & Integrative Genomics, 3(3), 125-134. doi:10.1007/s10142-003-0086-6

Examples

```
library(Biostrings)
library(JASPAR2016)
## Example matrix from JASPAR database
profileMatrix <- matrix(as.integer(
  c(13, 13,  3,  1, 54,  1,  1,  1,  0,  3,  2,  5,
    13, 39,  5, 53,  0,  1, 50,  1,  0, 37,  0, 17,
    17,  2, 37,  0,  0, 52,  3,  0, 53,  8, 37, 12,
    11,  0,  9,  0,  0,  0,  0, 52,  1,  6, 15, 20)),
  nrow=4, byrow=TRUE, dimnames=list(DNA_BASES))
pfmQuery <- PFMatrix(profileMatrix=profileMatrix)
pfmSubjects <- getMatrixSet(JASPAR2016,
                            opts=list(ID=c("MA0500", "MA0499", "MA0521",
                                           "MA0697", "MA0048", "MA0751",
                                           "MA0832")))
PFMSimilarity(pfmSubjects, pfmQuery)
```

PWMSimilarity-methods *PWMSimilarity method*

Description

This function measures the similarity of two PWM matrix in three measurements: "normalised Euclidean distance", "Pearson correlation" and "Kullback Leibler divergence".

Usage

```
PWMSimilarity(pwmSubject, pwmQuery, method=c("Euclidean", "Pearson", "KL"))
```

Arguments

pwmSubject	A matrix or PWMatrix or PWMatrixList object in "prob" type.
pwmQuery	A matrix or PWMatrix object.
method	The method can be "Euclidean", "Pearson", "KL".

Details

When pwmSubject and pwmQuery have different number of columns, the smaller PWM will be shifted from the start position of larger PWM and compare all the possible alignments. Only the smallest distance, divergence or largest correlation will be reported.

Value

A numeric value is returned.

Methods

```
signature(pwmSubject = "matrix", pwmQuery = "matrix")
signature(pwmSubject = "matrix", pwmQuery = "PWMatrix")
signature(pwmSubject = "PWMatrix", pwmQuery = "matrix")
signature(pwmSubject = "PWMatrix", pwmQuery = "PWMatrix")
signature(pwmSubject = "PWMatrixList", pwmQuery = "matrix")
signature(pwmSubject = "PWMatrixList", pwmQuery = "PWMatrix")
signature(pwmSubject = "PWMatrixList", pwmQuery = "PWMatrixList")
```

References

Linhart, C., Halperin, Y., & Shamir, R. (2008). Transcription factor and microRNA motif discovery: The Amadeus platform and a compendium of metazoan target sets. Genome Research, 18(7), 1180-1189. doi:10.1101/gr.076117.108

See Also

PFMSimilarity

Examples

```
data(MA0003.2)
data(MA0004.1)
pwm1 = toPWM(MA0003.2, type="prob")
pwm2 = toPWM(MA0004.1, type="prob")
PWMSimilarity(pwm1, pwm2, method="Euclidean")
```

`readJASPARMatrix` *Read JASPAR format matrix*

Description

Read a JASPAR format matrix file with 'individual' matrix or 'all' matrices in one file.

Usage

```
readJASPARMatrix(fn, matrixClass=c("PFM", "PWM", "PWMProb"))
```

Arguments

`fn`	`character(1)`: The filename of JASPAR format matrix file.
`matrixClass`	`character(1)`: the matrices in the files are 'PFM', 'PWM' or 'PFM' in probability.

Details

An example of 'individual' format matrix file is available at http://jaspar.genereg.net/html/DOWNLOAD/JASPAR_CORE/pfm/individual/MA0001.1.pfm

An exmaple of 'all' format matrix file is available at http://jaspar.genereg.net/html/DOWNLOAD/JASPAR_CORE/pfm/nonredundant/pfm_all.txt

Value

A `PFMatrixList` or `PWMatrixList` object is returned, depending on the matrix class.

Author(s)

Ge Tan

See Also

`makeFlatFileDir`

Examples

```
fn <- file.path(system.file("extdata", package="TFBSTools"),
                "MA0001.1.pfm")
readJASPARMatrix(fn, matrixClass="PFM")
fn <- file.path(system.file("extdata", package="TFBSTools"),
                "pfm_all_example.txt")
readJASPARMatrix(fn, matrixClass="PFM")
```

readXMLTFFM *readXMLTFFM*

Description

Read the ouput xml files from Puython module "TFFM" into R.

Usage

```
readXMLTFFM(fn, type=c("First", "Detail"))
```

Arguments

fn	The path of xml file.
type	The type of xml file. It can be one of the two types of xml files, "First" or "Detail".

Value

A TFFMFirst object or a TFFMDetail object is returned.

Author(s)

Ge Tan

See Also

TFFM

Examples

```
xmlFirst <- file.path(system.file("extdata", package="TFBSTools"),
                      "tffm_first_order.xml")
tffmFirst <- readXMLTFFM(xmlFirst, type="First")
```

rPWMDmm-methods *rPWMDmm method*

Description

This function samples matrices from trainned Dirichlet mixture model based on selected matrices.

Usage

```
rPWMDmm(x, alpha0, pmix, N=1, W=6)
```

Arguments

x	x can be a matrix, PFMatrixList. The count matrix on which the sampling is based.
alpha0	The trained Dirichlet mixture parameters.
pmix	The trained mixing proportions of the components.
N	The number of matrices to sample.
W	The desired width of matrice from the sampling.

Details

This feature enables the users to generate random Position Frequency Matrices (PFMs) from selected profiles.

We assume that each column in the profile is independent and described by a mixture of Dirichlet multinomials in which the letters are drawn from a multinomial and the multinomial parameters are drawn from a mixture of Dirichlets. Within this model each column has its own set of multinomial parameters but the higher level parameters – those of the mixture prior is assumed to be common to all Jaspar matrices. We can therefore use a maximum likelihood approach to learn these from the observed column counts of all Jaspar matrices. The maximum likelihood approach automatically ensures that matrices receive a weight relative to the number of counts it contains.

Drawing samples from the prior distribution will generate PWMs with the same statistical properties as the Jaspar matrices as a whole. PWMs with statistical properties like those of the selected profiles can be obtained by drawing from a posterior distribution which is proportional to the prior times a multinomial likelihood term with counts taken from one of the columns of the selected profiles.

Each 4-dimensional column is sampled by the following three-step procedure: 1. draw the mixture component according to the distribution of mixing proportions, 2. draw an input column randomly from the concatenated selected profiles and 3. draw the probability vector over nucleotides from a 4-dimensional Dirichlet distribution. The parameter vector alpha of the Dirichlet is equal to the sum of the count (of the drawn input) and the parameters of the Dirichlet prior (of the drawn component).

Draws from a Dirichlet can be obtained in the following way from Gamma distributed samples: (X1,X2,X3,X4) = (Y1/V,Y2/V,Y3/V,Y4/V) ~ Dir(a1,a2,a3,a4) where V = sum(Yi) ~ Gamma(shape = sum(ai), scale = 1).

Value

A list of matrices from the sampling.

Methods

```
signature(x = "PFMatrix")
signature(x = "matrix")
signature(x = "PFMatrixList")
```

Note

This code is based on the Matlab code original written by Ole Winther, binf.ku.dk, June 2006.

Author(s)

Ge Tan

References

L. Devroye, "Non-Uniform Random Variate Generation", Springer-Verlag, 1986

Kimura, T., Tokuda, T., Nakada, Y., Nokajima, T., Matsumoto, T., & Doucet, A. (2011). Expectation-maximization algorithms for inference in Dirichlet processes mixture. Pattern Analysis and Applications, 16(1), 55-67. doi:10.1007/s10044-011-0256-4

See Also

dmmEM

Examples

```
data(MA0003.2)
data(MA0004.1)
pfmList <- PFMatrixList(pfm1=MA0003.2, pfm2=MA0004.1, use.names=TRUE)
dmmParameters <- dmmEM(pfmList, 6)
rPWMDmm(MA0003.2, dmmParameters$alpha0, dmmParameters$pmix, N=1, W=6)
```

---

runMEME *Wrapper function for MEME*

Description

This function builds position frequency matrices using an external program **MEME** written by *Bailey* and *Elkan*.

Usage

```
  ## S4 method for signature 'character'
runMEME(x, binary="meme", seqtype="DNA",
                                arguments=list(), tmpdir=tempdir())
  ## S4 method for signature 'DNAStringSet'
runMEME(x, binary="meme", seqtype="DNA",
                                   arguments=list(), tmpdir=tempdir())
```

Arguments

x	A character(1) vector of the path of fasta file or a XStringSet.
binary	character(1): the path of MEME binary. By default, we assume the meme is in the *PATH*.
seqtype	The sequence type. "AA" and "DNA" are allowed.
arguments	A list: the addtional arguments for meme. This list takes the parameter of MEME as names of the elements, and the values of the parameters as the elements. For examples, arguments=list("-nmotifs"=3).
tmpdir	A character(1) vector to change the defult R's temp directory.

Value

A MotifSet object is returned.

Note

This wrapper is tested on "MEME" 4.10.1 and 4.12.0.

Author(s)

Ge Tan

References

Bailey, T. L., Boden, M., Buske, F. A., Frith, M., Grant, C. E., Clementi, L., et al. (2009). MEME SUITE: tools for motif discovery and searching. Nucleic acids research, 37(Web Server issue), W202-8. doi:10.1093/nar/gkp335

http://meme-suite.org/

See Also

MotifSet

Examples

```
## Not run:
  motifSet <- runMEME(file.path(system.file("extdata", package="TFBSTools"),
                      "crp0.s"),
                  binary="/usr/local/Cellar/meme/4.10.1/bin/meme",
                  arguments=list("-nmotifs"=3))
  ## Get the site sequences
  sitesSeq(motifSet, type="all")
  sitesSeq(motifSet, type="none")

  ## Get the consensu matrix, then it can be used as a PFMatrix
  consensusMatrix(motifSet)

## End(Not run)
```

---

sampleRanges *sampleRanges*

Description

Sample ranges with same widths of input rannges from a set of subject ranges.

Usage

```
sampleRanges(inputGRanges, subjectGRanges, ignore.strand=TRUE)
```

Arguments

inputGRanges	The input GRanges.
subjectGRanges	The subject GRanges.
ignore.strand	When set to TRUE, the strand information is ignoreed during the sampling. Otherwise, the input ranges on positvie strand will only sample from subject ranges on positvie strand.

Value

A GRanges object with the same length and widths of inputGRanges.

Author(s)

Ge Tan

Examples

```
library(GenomicRanges)
inputGRanges <- GRanges(seqnames=c("chr1", "chr2"),
                        range=IRanges(start=c(2L, 10L), end=c(6L, 15L)),
                         strand=c("+", "-"))

subjectGRanges <- GRanges(
                           seqnames=c("chr1", "chr1", "chr1", "chr1",
                                      "chr2", "chr2"),
                           ranges=IRanges(start=c(20L, 20L, 30L, 30L, 7L, 25L),
                                          end=c(50L, 50L, 32L, 32L,9L, 55L)),
                           strand=c("+","-", "+", "-", "+","-"))
set.seed(16)
sampleRanges(inputGRanges, subjectGRanges, ignore.strand=TRUE)
sampleRanges(inputGRanges, subjectGRanges, ignore.strand=FALSE)
```

---

searchAln *searchAln method*

Description

Scans a pairwise alignment of nucleotide sequences with the pattern represented by the PWMatrix. It reports only those hits that are overlapped in the alignment of of the two sequences and exceed a specified threshold score in both, AND are found in regions of the alignment above the specified conservation cutoff value.

Usage

```
searchAln(pwm, aln1, aln2, seqname1="Unknown1", seqname2="Unknown2",
          min.score="80%", windowSize=51L,
          cutoff=0.7, strand="*", type="any", conservation=NULL,
          mc.cores=1L)
```

Arguments

pwm	A PWMatrix object or a PWMatrixList object.
aln1	A DNAString, character, DNAStringSet or Axt object can be used to represent the pairwise alignment. When the last two objects are used and have a length of 2, the argument aln2 can be missing.
aln2	A DNAString, character. It can be missing when aln1 is DNAStringSet or Axt object.

`seqname1`, `seqname2`
: A `chracter` object for the name of sequence. "Unknown1" and "Unknown2" are used by default. These two arguments are ignored when aln1 is `Axt`, or the seqnames are available from aln1.

`min.score`
: The minimum score for the hit. Can be given an character string in the format of "80%" or as a single absolute value. When it is percentage value, it means the percentage of the maximal possible from the PWM.

`windowSize`
: The size of the sliding window (in nucleotides) for calculating local conservation in the alignment. This should be an odd value.

`cutoff`
: The conservation cutoff can be from 0 (0% identity) to 1 (100% identity). The regions which have lower conservation than the cutoff will be discarded from the results of the pattern searching. The conservation is calculated by comparing the alignments within the `windowSize`: 1 for match and 0 for mismatch and gap.

`strand`
: When searching the alignment, we can search the positive strand or negative strand. While strand is "*", it will search both strands and return the results based on the positvie strand coordinate.

`type`
: This argument can be "any" or "all". When it is "any", one motif will be kept if the maximal conservation value of the motif is larger than the cutoff. When it is "all", one motif will be kept if the minimal conservation value of the motif is larger than the cutoff.

`conservation`
: A vector of conservation profile. If not supplied, the conservation profile will be computed internally on the fly.

`mc.cores`
: The number of cpu threads to use when searching `Axt`. 1L is assigned by default.

Details

In brief, given a pairwise alignment of two sequences, first of all, we remove the gaps ("-", "-", "."). Then we scan both ungapped sequences with the pwm and return the hits that above `min.score`. Since we only want to keep the conserved hits, we choose the pair of motifs that overlap most in the alignment. Finally, the pair of motifs have to be conserved above the threshold `cutoff`.

In the returned `SitePairSet`, the coordinates of start, end are based on the ungapped sequences, instead of the original alignment. This is due to we are more concerned about the actual location of motif in the genome rather than in the alignment.

Value

A `SitePairSet` object is returned when pwm is a `PWMatrix`, while a `SitePairSetList` is returned when pwm is a `PWMatrixList`.

Author(s)

Ge Tan

See Also

`searchSeq`

Examples

```
data(MA0003.2)
data(MA0004.1)
pwm1 <- toPWM(MA0003.2)
pwm2 <- toPWM(MA0004.1)
pwmList <- PWMatrixList(pwm1=pwm1, pwm2=pwm2)
# Two character objects
aln1 <- "ACCACATTGCCTCAGGGCAGGTAAGTTGATC---AAAGG---AAACGCAAAGTTTTCAAG"
aln2 <- "GTTTCACTACATTGCTTCAGGGCAGTAAATATATAAATATATAAAAATATAATTTTCATC"
aln <- c(aln1=aln1, aln2=aln2)
library(Biostrings)
alnDNAStringSet <- DNAStringSet(c(aln1=aln1, aln2=aln2))

# PWMatrix, character, character
## Only scan the positive strand of the alignments
sitePairSet <- searchAln(pwm1, aln1, aln2, seqname1="aln1", seqname2="aln2",
                         min.score="70%", cutoff=0.5,
                         strand="+", type="any")
## Only scan the negative strand of the alignments
sitePairSet <- searchAln(pwm1, aln1, aln2, seqname1="aln1", seqname2="aln2",
                         min.score="70%", cutoff=0.5,
                         strand="-", type="any")
## Scan the both strands of the alignments
sitePairSet <- searchAln(pwm1, aln1, aln2, seqname1="aln1", seqname2="aln2",
                         min.score="70%", cutoff=0.5,
                         strand="*", type="any")
## Convert the SitePairSet object into other R objects
as(sitePairSet, "data.frame")
as.data.frame(sitePairSet)
as(sitePairSet, "DataFrame")
as(sitePairSet, "GRanges")
writeGFF3(sitePairSet)
writeGFF2(sitePairSet)

# PWMatrix, character, missing
sitePairSet <- searchAln(pwm1, aln,
                         min.score="70%", cutoff=0.5,
                         strand="*", type="any")

# PWMatrix, DNAString, DNAString
sitePairSet <- searchAln(pwm1, DNAString(aln1), DNAString(aln2),
                         seqname1="aln1", seqname2="aln2",
                         min.score="70%", cutoff=0.5,
                         strand="*", type="any")

# PWMatrix, DNAStringSet, missing
sitePairSet <- searchAln(pwm1, alnDNAStringSet,
                         min.score="70%", cutoff=0.5,
                         strand="*", type="any")

# PWMatrixList, character, character
sitePairSetList <- searchAln(pwmList, aln1, aln2,
                         seqname1="aln1", seqname2="aln2",
                         min.score="70%", cutoff=0.5,
                         strand="*", type="any")
## elementLenths of each pwm hits
```

```
 elementNROWS(sitePairSetList)

## output
writeGFF2(sitePairSetList)
writeGFF3(sitePairSetList)
as(sitePairSetList, "DataFrame")
as(sitePairSetList, "data.frame")
as.data.frame(sitePairSetList)
as(sitePairSetList, "GRanges")

# PWMatrix, Axt, missing
#library(CNEr)
#axtFilesHg19DanRer7 <- file.path(system.file("extdata", package="TFBSTools"),
#                                 "hg19.danRer7.net.axt")
#axtHg19DanRer7 <- readAxt(axtFilesHg19DanRer7)
#sitePairSetList <-  searchAln(pwm1, axtHg19DanRer7, min.score="80%",
#                         windowSize=51L, cutoff=0.7, strand="*",
#                         type="any", conservation=NULL, mc.cores=1)
## We may want to coordinates of motif in the genome
# GRangesTFBS <- toGRangesList(sitePairSetList, axtHg19DanRer7)
```

`searchPairBSgenome-methods`

searchPairBSgenome method

Description

Given a chain file for liftover from one genome to another, it searches two `BSgenome` with a `PWMatrix`, and only reports the hits that are presents in two genomes with equivalent positions.

Usage

```
searchPairBSgenome(pwm, BSgenome1, BSgenome2, chr1, chr2,
                   min.score="80%", strand="*", chain)
```

Arguments

`pwm`	A PWMatrix object or a PWMatrixList object.
`BSgenome1, BSgenome2`	A `BSgenome` class.
`chr1, chr2`	A `character` object, specifying the chromosomes you want to search.
`min.score`	The minimum score for the hit. Can be given an character string in th format of "80%" or as a single absolute value.
`strand`	When searching the alignment, we can search the positive "+" strand or negative "-" strand. While strand is "*", it will search both strands and return the results based on the positvie strand coordinate.
`chain`	A `Chain` object. It can be generated by `import.chain` from package `rtracklayer`. Please provide the chain from BSgenome1 to BSgenome2.

Value

A `SitePairSet` object is returned when pwm is a `PWMatrix`, while a `SitePairSetList` is returned when pwm is a `PWMatrixList`.

Author(s)

Ge Tan

See Also

searchAln

Examples

```
## Not run:
  library(rtracklayer)
  library(JASPAR2014)
  library(BSgenome.Hsapiens.UCSC.hg19)
  library(BSgenome.Mmusculus.UCSC.mm10)
  data("MA0004.1")
  pfm <- MA0004.1
  pwm <- toPWM(pfm)
  chain <- import.chain("Downloads/hg19ToMm10.over.chain")
  sitepairset <- searchPairBSgenome(pwm, BSgenome.Hsapiens.UCSC.hg19,
                                    BSgenome.Mmusculus.UCSC.mm10,
                                    chr1="chr1", chr2="chr1",
                                    min.score="90%", strand="+",
                                    chain=chain)

## End(Not run)
```

---

searchSeq *searchSeq method*

Description

It scans a nucleotide sequence with the pattern represented by a PWMatrix and identifies putative transcription factor binding sites.

Usage

```
searchSeq(x, subject, seqname="Unknown", strand="*", min.score="80%",
          mc.cores=1L)
```

Arguments

x	PWMatrix or PWMatrixList object.
subject	A DNAStringSet, DNAString, XStringViews or MaskedDNAString object that will be scanned.
seqname	This is sequence name of the target sequence. If subject is a DNAStringSet, the names of the DNAStringSet object will be used.
strand	When searching the sequence, we can search the positive strand or negative strand. While strand is "*", it will search both strands and return the results based on the positvie strand coordinate.

min.score	The minimum score for the hit. Can be given an character string in the format of "80%" or as a single absolute value between 0 and 1. When it is percentage value, it represents the quantile between the minimal and the maximal possible value from the PWM.
mc.cores	integer(1): The number of cores to use. It is only used when 'x' is a PWMatrixList object and not available on windows platform.

Value

A SiteSet object is returned when x is a PWMatrix object. A SiteSetList object is returned when x is a PWMatrixList or subject is a DNAStringSet.

Author(s)

Ge Tan

References

Wasserman, W. W., & Sandelin, A. (2004). Applied bioinformatics for the identification of regulatory elements. Nature Publishing Group, 5(4), 276-287. doi:10.1038/nrg1315

See Also

searchAln, matchPWM

Examples

```
data(MA0003.2)
data(MA0004.1)
pwm1 <- toPWM(MA0003.2)
pwm2 <- toPWM(MA0004.1)
pwmList <- PWMatrixList(pwm1=pwm1, pwm2=pwm2)
seq1 <- "GAATTCTCTCTTGTTGTAGCATTGCCTCAGGGCACACGTGCAAAATG"
seq2 <- "GTTTCACCATTGCCTCAGGGCATAAATATATAAAAAAATATAATTTTCATC"

# PWMatrix, character
## Only scan the positive strand of the input sequence
siteset <- searchSeq(pwm1, seq1, seqname="seq1", strand="+", min.score="80%")
siteset <- searchSeq(pwm1, seq1, seqname="seq1", strand="+", min.score=0.8)
## Only scan the negative strand of the input sequence
siteset <- searchSeq(pwm1, seq1, seqname="seq1", strand="-", min.score="80%")
## Scan both strands of the input sequences
siteset <- searchSeq(pwm1, seq1, seqname="seq1", strand="*", min.score="80%")
## Convert the SiteSet object into other R objects
as(siteset, "data.frame")
as(siteset, "DataFrame")
as(siteset, "GRanges")
writeGFF3(siteset)
writeGFF2(siteset)

# PWMatrixList, character
sitesetList <- searchSeq(pwmList, seq1, seqname="seq1", strand="*",
                         min.score="80%")
sitesetList <- searchSeq(pwmList, seq1, seqname="seq1", strand="*",
                         min.score="80%", mc.cores=1L)
```

```
## Convert the SiteSteList object into other R objects
as(sitesetList, "data.frame")
as(sitesetList, "DataFrame")
as(sitesetList, "GRanges")
writeGFF3(sitesetList)
writeGFF2(sitesetList)

# PWMatrix, DNAStringSet
library(Biostrings)
seqs <- DNAStringSet(c(seq1=seq1, seq2=seq2))
sitesetList <- searchSeq(pwm1, seqs, min.score="80%")

# PWMatrixList, DNAStringSet
sitesetList <- searchSeq(pwmList, seqs, min.score="80%")
```

`seqLogo` *Plot a sequence logo*

Description

This function takes a `ICMatrix` or `TFFM` object and plot the sequence logo.

Usage

```
seqLogo(x, ic.scale = TRUE, xaxis = TRUE, yaxis = TRUE,
         xfontsize = 15, yfontsize = 15)
```

Arguments

`x`	x is a valid `ICMatrix` object or `TFFM` object.
`ic.scale`	A logical value. If TRUE, the total height of one column is proportional to the information content at that position. Otherwise, all the columns will have the same height. Ignored for `TFFM` object.
`xaxis`	A logical value. If TRUE, the x-axis will be plotted. Ignored for `TFFM` object.
`yaxis`	A logical value. If TRUE, the y-axis will be plotted. Ignored for `TFFM` object.
`xfontsize`	A numeric value. The font size for x-axis.
`yfontsize`	A numeric value. The font size for y-axis.

Details

A sequence logo is a graphical representation of the matrix model, based on the information content of each position. The information content ranges from 0 (no base preference) to 2 (only 1 base used). If `ic.scale` is TRUE, the height of the logo at certain site is proportinal to the information content value. And each stacked base (A, C, G, T)'s height is also proportional to the information content of each base at that position, and sorted based on the character size.

For a `TFFM` object, a novel graphical representation is used for capturing the dinucleotide dependencies on the TFFM. For the upper part of the sequence logo, we represent the nucleotide probabilities

at position p for each possible nucleotide at position p-1. Hence, each column represents a position within a TFBS and each row the nucleotide probabilities found at that position. Each row assumes a specific nucleotide has been emitted by the previous hidden state. The intersection between a column corresponding to position p and row corresponding to nucleotide n gives the probabilities of getting each nucleotide at position p if n has been seen at position p-1. The opacity to represent the sequence logo is proportional to the probablity of possible row to be used by the TFFM.

Value

No return value.

Note

This function is based on the function `seqLogo` from the Bioconductor package `seqLogo`, especially for the plotting code of TFFM.

Author(s)

Ge Tan

References

T D Schneider, R. M. S. (1990). Sequence logos: a new way to display consensus sequences. Nucleic acids research, 18(20), 6097.

Mathelier, A., and Wasserman, W.W. (2013). The next generation of transcription factor binding site prediction. PLoS Comput. Biol. 9, e1003214.

See Also

`toICM`, `ICMatrix`,

Examples

```
## ICMatrix
data(MA0003.2)
icm = toICM(MA0003.2)
seqLogo(icm, ic.scale = TRUE)

## TFFM
xmlFirst <- file.path(system.file("extdata", package="TFBSTools"),
                      "tffm_first_order.xml")
tffmFirst <- readXMLTFFM(xmlFirst, type="First")
seqLogo(tffmFirst)
```

---

shannon.entropy *Calculate the Shannon entropy*

Description

This function calculates the Shannon entropy for a discrete random variable with finite n values sample.

Usage

```
shannon.entropy(p)
```

Arguments

p	A numeric object with non-negative values.

Details

The entropy is calculated by H(x) = -sum_i^n(P(x_i)log_b(P(x_i))).

Value

A numeric value of entropy is returned.

Author(s)

Ge Tan

Examples

```
x <- c(1, 1, 1, 1)
shannon.entropy(x)
x <- c(1, 0, 0, 0)
shannon.entropy(x)
```

SitePairSet *Class* "SitePairSet"

Description

The SitePairSet object is a container for storing two SiteSet objects. Usually it is used to hold the results returned by searchAln.

Usage

```
## Constructor
SitePairSet(siteset1, siteset2)
```

Arguments

siteset1, siteset2	Each SiteSet object is from one sequence in the pairwise alignment.

Value

A SitePairSet object.

Methods

siteset1 signature(x = "SitePairSet"): Gets the first SiteSet object.

siteset2 signature(x = "SitePairSet"): Gets the second SiteSet object.

Author(s)

Ge Tan

See Also

SitePairSet, searchAln

SitePairSetList-class *Class* "SitePairSetList"

Description

The SitePairSetList class is a container for storing a collection of SitePairSet objects. Basically it is a SimpleList and is designed for manipulating the set of SitePairSet objects as a whole.

Usage

```
## Constructors:
  SitePairSetList(..., use.names=TRUE)
```

Arguments

...	The SitePairSet objects are supplied in .... A list of SitePairSet objects is also acceptable.
use.names	A logical value. When TRUE, the names of the SitePairSet will be kept.

Value

A SitePairSetList object.

Author(s)

Ge Tan

See Also

SitePairSet,

SiteSet *Class* "SiteSet"

Description

The SiteSet object is a container for storing a set of putative transcription factor binding sites on a nucleotide sequence (start, end, strand, score, pattern as a PWMatrix, etc.)

Usage

```
## Constructors:
SiteSet(views, score, strand="*", seqname="Unknown", sitesource="TFBS",
        primary="TF binding site", pattern)
```

Arguments

views	Object of class "XStringViews": It holds the start, end and the nucleotide sequence information of the transcription factor binding sites.
score	Object of class "numeric": A vector of PWM score for each putative binding site based on the PWM matrix..
strand	Object of class "character": The binding site is from the positive ("+"), negative ("-") or unknown("*") strand.
seqname	Object of class "character": The seqname of the sequence which contains these binding sites.
sitesource	Object of class "character": Currently it is set to "TFBS"
primary	Object of class "character": Currently it is set to "TF binding site"
pattern	Object of class "PWMatrix": The PWMatrix object which is used to search the binding sites.

Details

The score retuned in SiteSet is the absolute score of each putative TFBS scanned by the corresponding PWM. The way of calculating the score is shown on the refernce, Page 281.

Methods

[`signature(x = "SiteSet")`: Getter function.

length `signature(x = "SiteSet")`: The number of binding sites in this SiteSet.

pattern `signature(x = "SiteSet")`: Returns the PWMatrix used.

relScore `signature(x = "SiteSet")`: Gets relative score (between 0.0 to 1.0) with respect of the score range of the associated pattern (PWMatrix).

score `signature(x = "SiteSet")`: Returns the score of each site.

seqname `signature(x = "SiteSet")`: Returns the sequence name of the sequence which contains these sites.

strand `signature(x = "SiteSet")`: Returns the strand information.

views `signature(x = "SiteSet")`: Returns the views object.

start `signature(x = "SiteSet")`: Returns the start coordinates.

end `signature(x = "SiteSet")`: Returns the end coordinates.

pvalues `signature(x = "SiteSet")` (x, type=c("TFMPvalue", "sampling")): Calculates the empirical p-values for the scores with two methods: the exact method from TFMPaluve package or implementation of sampling in this package. The background probability for sampling is based on the PWM matrix in the `SiteSet` object.

Author(s)

Ge Tan

References

Wasserman, W. W., & Sandelin, A. (2004). Applied bioinformatics for the identification of regulatory elements. Nature Publishing Group, 5(4), 276-287. doi:10.1038/nrg1315

See Also

`searchSeq`, `searchAln`, `PWMatrix`, `SiteSetList`, `SitePairSet`

Examples

```
data(MA0003.2)
pwm <- toPWM(MA0003.2)
siteset <- searchSeq(pwm, "GAATTCTCTCTTGTTGTAGTCTCTTGACAAAATG",
                     min.score="60%")
writeGFF3(siteset, scoreType="absolute")
as(siteset, "data.frame")
as(siteset, "DataFrame")
as(siteset, "GRanges")

relScore(siteset)
pvalues(siteset, type="TFMPvalue")
pvalues(siteset, type="sampling")
```

SiteSetList *Class* "SiteSetList"

Description

The SiteSetList class is a container for storing a collection of SiteSet objects. Basically it is a SimpleList and is designed for manipulating the set of SiteSet objects as a whole.

Usage

```
## Constructors:
  SiteSetList(..., use.names=TRUE)
```

Arguments

`...`	The SiteSet objects are supplied in .... A list of SiteSet objects is also acceptable.
`use.names`	A logical value. When TRUE, the names of the `SiteSet` will be kept.

Value

A SiteSetList object.

Methods

pvalues signature(x = "SiteSetList") (x, type=c("TFMPvalue", "sampling")): Calculates the empirical p-values for the scores.

Author(s)

Ge Tan

See Also

SiteSet, searchSeq, searchAln

Examples

```
data(MA0003.2)
data(MA0004.1)
pwmList <- PWMatrixList(MA0003.2=toPWM(MA0003.2), MA0004.1=toPWM(MA0004.1))
sitesetList <- searchSeq(pwmList, "GAATTCTCTCTTGTTGTAGTCTCTTGACAAAATG",
                         min.score="50%")

## elementNROWS of each pwm hits
library(S4Vectors)
elementNROWS(sitesetList)

## Output of SiteSetList
writeGFF3(sitesetList, scoreType="absolute")
as(sitesetList, "DataFrame")
as(sitesetList, "data.frame")
as.data.frame(sitesetList)
as(sitesetList, "GRanges")

## Calculate the p-values
pvalues(sitesetList, type="TFMPvalue")
pvalues(sitesetList, type="sampling")
```

TFFM *The TFFM class*

Description

The TFFM is a virtual class. Two classes are derived from this class: TFFMFirst and TFFMDetail.

TFFMFirst class stands for the first-order TFFMs and TFFMDetail stands for the more detailed and descriptive TFFMs.

Usage

```
## constructors:
  TFFMFirst(ID="Unknown", name="Unknown", matrixClass="Unknown",
            strand="+", bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
            tags=list(), profileMatrix=matrix(),
            type=character(), emission=list(),
            transition=matrix())
  TFFMDetail(ID="Unknown", name="Unknown", matrixClass="Unknown",
             strand="+", bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
             tags=list(), profileMatrix=matrix(),
             type=character(), emission=list(),
             transition=matrix())
```

Arguments

ID, name, matrixClass, strand, bg, tags, profileMatrix	See XMatrix
type	The type of TFFM.
emission	The emission distribution parameters.
transition	The transition probability matrix.

Value

A TFFM object.

Methods

ncol `signature(x = "TFFMFirst")`: Get the length of First-order TFFM.

ncol `signature(x = "TFFMDetail")`: Get the length of detail TFFM.

totalIC `signature(x = "TFFM")`: Get the information content at each position.

Author(s)

Ge Tan

References

Mathelier, A., and Wasserman, W.W. (2013). The next generation of transcription factor binding site prediction. PLoS Comput. Biol. 9, e1003214.

http://cisreg.cmmt.ubc.ca/TFFM/doc/#

Examples

```
xmlFirst <- file.path(system.file("extdata", package="TFBSTools"),
                      "tffm_first_order.xml")
tffmFirst <- readXMLTFFM(xmlFirst, type="First")
tffm <- getPosProb(tffmFirst)
```

toGRangesList-methods *toGRangesList function*

Description

Get the genomic coordinates from SitePairSetList.

Value

A list of two GRanges objects are returned, one for the target sequences and another for query sequences.

In the GRanges, strand is taken from the Axt object. In the meta-data columns, PWM matrix ID, the strand of matrix and match score are also returned.

Methods

signature(x = "SitePairSetList", axt = "Axt") Convert the relative coordinates to absolute coordinates.

Author(s)

Ge Tan

Examples

```
data(MA0003.2)
pwm <- toPWM(MA0003.2)
#library(CNEr)
#axtFilesHg19DanRer7 <- file.path(system.file("extdata", package="TFBSTools"),
#                                 "hg19.danRer7.net.axt")
#axtHg19DanRer7 <- readAxt(axtFilesHg19DanRer7)
#sitePairSet <-  searchAln(pwm, axtHg19DanRer7, min.score="80%",
#                          windowSize=51L, cutoff=0.7, strand="*",
#                          type="any", conservation=NULL, mc.cores=1)
# toGRangesList(sitePairSet, axtHg19DanRer7)
```

toICM *toICM method*

Description

Converts a raw frequency matrix (PFMatrix) to a information content matrix (ICMatrix). It takes the bases background frequencies, pseudocounts and schneider as parameters.

Usage

```
toICM(x, pseudocounts=0.8, schneider=FALSE,
      bg=c(A=0.25, C=0.25, G=0.25, T=0.25))
```

Arguments

x	For toPWM, a PFMatrix, rectangular DNAStringSet object ("rectangular" means that all elements have the same number of characters) with no IUPAC ambiguity letters, a rectangular character vector or a matrix with rownames containing at least A, C, G and T, or a PFMatrixList object
pseudocounts	A default value 0.8 is used.
schneider	This logical parameter controls whether a Schneider correction will be done. See more details below.
bg	bg is a vector of background frequencies of four bases with names containing A, C, G, T. When toPWM is applied to a PFMatrix, if bg is not specified, it will use the bg information contained in PFMatrix.

Details

The information content matrix has a column sum between 0 (no base preference) and 2 (only 1 base used). Usually this information is used to plot sequence log.

The information content at each position is computed

$$D = \log_2(nrow(pfm)) + colSums(postProbs \times \log_2(postProbs)$$

$$icm = posProbs * D$$

where D is the total information contect for each position. For detailed procedure of computation, please refer to the vignette.

If a Schneider correction will be done if requested. Please see the reference below for more comprehensive explanation.

Value

A ICMatrix object which contains the background frequency, pseudocounts and Schneider correction used.

Author(s)

Ge Tan

References

Schneider, T. D., Stormo, G. D., Gold, L., & Ehrenfeucht, A. (1986). Information content of binding sites on nucleotide sequences. Journal of molecular biology, 188(3), 415-431.

See Also

toPWM, XMatrix, seqLogo

Examples

```
## Constructor a PFMatrix
pfm <- PFMatrix(ID="MA0004.1", name="Arnt", matrixClass="Zipper-Type",
                strand="+",
                bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
                tags=list(family="Helix-Loop-Helix",
                          species="10090",
```

```
                          tax_group="vertebrates",
                          medline="7592839", type="SELEX", ACC="P53762",
                          pazar_tf_id="TF0000003",
                          TFBSshape_ID="11", TFencyclopedia_ID="580"),
                 profileMatrix=matrix(c(4L,  19L, 0L,  0L,  0L,  0L,
                                        16L, 0L,  20L, 0L,  0L,  0L,
                                        0L,  1L,  0L,  20L, 0L,  20L,
                                        0L,  0L,  0L,  0L,  20L, 0L),
                                      byrow=TRUE, nrow=4,
                                      dimnames=list(c("A", "C", "G", "T")))
               )
## Convert it into a PWMatrix
icm <- toICM(pfm, pseudocounts=0.8, schneider=TRUE)

## Conversion on PWMatrixList
data(MA0003.2)
data(MA0004.1)
pfmList <- PFMatrixList(pfm1=MA0003.2, pfm2=MA0004.1, use.names=TRUE)
icmList <- toICM(pfmList, pseudocounts=0.8, schneider=TRUE)
```

`toPWM` *toPWM method*

Description

Converts a raw frequency matrix (PFMatrix) to a position weight matrix (PWMatrix). It takes the type, bases background frequencies, pseudocounts as parameters.

Usage

```
toPWM(x, type=c("log2probratio", "prob"), pseudocounts=0.8,
      bg=c(A=0.25, C=0.25, G=0.25, T=0.25))
```

Arguments

`x`	For `toPWM`, a PFMatrix, rectangular DNAStringSet object ("rectangular" means that all elements have the same number of characters) with no IUPAC ambiguity letters, a rectangular character vector or a matrix with rownames containing at least A, C, G and T, or a PFMatrixList object.
`type`	The type of PWM generated, should be one of "log2probratio" or "prob". "log2probratio" will generate the PWM matrix in log-scale, while "prob" will give the PWM matrix in probability scale of 0 to 1.
`pseudocounts`	pseudocounts is a numeric non-negative vector, which means you can specify different pseudocounts for each site. The values will be recycled if shorter than the length of sites. 0.8 is recommended. See the reference below for more details. In the TFBS perl module, the squared root of the column sum of the matrix, i.e., the number of motifs used to construct the PFM, is used.
`bg`	bg is a vector of background frequencies of four bases with names containing A, C, G, T. When toPWM is applied to a `PFMatrix`, if bg is not specified, it will use the bg information contained in `PFMatrix`.

Details

The raw position frequency matrix (PFM) is usually converted into a position weight matrix (PWM), also known as position specific scoring matrix (PSSM). The PWM provides the probability of each base at certain position and used for scanning the genomic sequences. The implementation here is slightly different from `PWM` in `Biostrings` package by choosing the pseudocounts. Pseudocounts is necessary for correcting the small number of counts or eliminating the zero values before log transformation.

$$postProbs = \frac{PFM + bg * pseudocounts}{colSums(PFM) + sum(bg) * pseudocounts}$$

$$priorProbs = bg/sum(bg)$$

$$PWM_{log2probratio} = \log_2 \frac{postProbs}{priorProbs}$$

$$PWM_{prob} = postProbs$$

Value

A `PWMatrix` object that contains the background frequency and pseudocounts used.

Author(s)

Ge Tan

References

Wasserman, W. W., & Sandelin, A. (2004). Applied bioinformatics for the identification of regulatory elements. Nature Publishing Group, 5(4), 276-287. doi:10.1038/nrg1315

Nishida, K., Frith, M. C., & Nakai, K. (2009). Pseudocounts for transcription factor binding sites. Nucleic acids research, 37(3), 939-944. doi:10.1093/nar/gkn1019

See Also

`toICM`, `XMatrix`

Examples

```
## Constructe a PFMatrix
pfm <- PFMatrix(ID="MA0004.1", name="Arnt", matrixClass="Zipper-Type",
                strand="+", bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
                tags=list(family="Helix-Loop-Helix", species="10090",
                          tax_group="vertebrates",
                          medline="7592839", type="SELEX", ACC="P53762",
                          pazar_tf_id="TF0000003",
                          TFBSshape_ID="11", TFencyclopedia_ID="580"),
                profileMatrix=matrix(c(4L,  19L, 0L,  0L,  0L,  0L,
                                       16L, 0L,  20L, 0L,  0L,  0L,
                                       0L,  1L,  0L,  20L, 0L,  20L,
                                       0L,  0L,  0L,  0L,  20L, 0L),
                                     byrow=TRUE, nrow=4,
                                     dimnames=list(c("A", "C", "G", "T")))
                )
## Convert it into a PWMatrix
pwm <- toPWM(pfm, type="log2probratio", pseudocounts=0.8)
```

```
## Conversion on PWMatrixList
data(MA0003.2)
data(MA0004.1)
pfmList <- PFMatrixList(pfm1=MA0003.2, pfm2=MA0004.1, use.names=TRUE)
pwmList <- toPWM(pfmList, pseudocounts=0.8)
```

---

writeGFF3-methods writeGFF3, writeGFF2 *functions*

Description

write the SiteSet, SitePairSet, SiteSetList, SitePairSetList into the GFF3 or GFF2 format.

Usage

```
writeGFF3(x, scoreType=c("absolute", "relative"))
writeGFF2(x, scoreType=c("absolute", "relative"))
```

Arguments

x	A SiteSet, SitePairSet, SiteSetList, or SitePairSetList object.
scoreType	The score column can have absolute value or relative value.

Value

It returns nothing.

Author(s)

Ge Tan

XMatrix "XMatrix" *objects*

Description

XMatrix is a virtual class. No objects can be created from it directly. Three classes are derived from this class: PFMatrix, PWMatrix and ICMatrix.

PFMatrix is a class whose instances are objects representing raw position frequency matrices (PFMs).

PWMatrix is a class whose instances are objects representing position weight matrices (PWMs). Compared with PFMatrix, it has extra slot pseudocounts.

ICMatrix is a class whose instances are objects representing information content matrices (ICMs). Compared with PWMatrix, it has extra slot schneider.

Usage

```
## Constructors:
PFMatrix(ID="Unknown", name="Unknown", matrixClass="Unknown",
         strand="+", bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
         tags=list(), profileMatrix=matrix())
PWMatrix(ID="Unknown", name="Unknown", matrixClass="Unknown",
         strand="+", bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
         tags=list(), profileMatrix=matrix(), pseudocounts=numeric())
ICMatrix(ID="Unknown", name="Unknown", matrixClass="Unknown",
         strand="+", bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
         tags=list(), profileMatrix=matrix(), pseudocounts=numeric(),
         schneider=logical())
## Accessor-like methods:
## S4 method for signature 'XMatrix'
ID(x)
## S4 method for signature 'XMatrix'
bg(x)

## ... and more (see Methods)
```

Arguments

ID	Object of class "character": A unique identifier for each matrix.
name	Object of class "character": The name of the transcription factor. In JASPAR, as far as possible, the name is based on the standardized Entrez gene symbols. In the case the model describes a transcription factor hetero-dimer, two names are concatenated, such as RXR-VDR. In a few cases, different splice forms of the same gene have different binding specificity: in this case the splice form information is added to the name, based on the relevant literature.
matrixClass	Object of class "character": Structural class of the transcription factor, based on the TFCaT system
strand	Object of class "character": Which strand is the binding sites sequences from.
bg	Object of class "numeric": Background frequencies of the four bases. By default, it is equally 0.25.
tags	Object of class "list": Some tags information about this model. Tags include: (1) "family": Structural sub-class of the transcription factor, based on the TFCaT system. (2) "species": The species source for the sequences, in NCBI tax IDs. (3) "tax_group": Group of species, currently consisting of 4 larger groups: vertebrate, insect, plant, chordate. (4) "medline": a ID to the relevant publication reporting the sites used in the mode building. (5) "type": Methodology used for matrix construction. (6) "ACC": A representative protein accession number in Genbank for the transcription factor. Human takes precedence if several exists. (6) "pazar_tf_id": a ID to PAZAR database. (7) "TFBSshape_ID": a ID to TFBSshape database. (8) "TFencyclopedia_ID": a ID to the Transcription Factor Encyclopedia. (9) "comment": For some matrices, a curator comment is added.

profileMatrix	Object of class "matrix": This is the matrix information.
pseudocounts	Object of class "numeric": This is the pseudocounts used when computing ICM or PWM from PFM. By default, a threshold of 0.8 is used based on the previous research (doi:10.1093/nar/gkn1019).
schneider	Obejct of class "logical": this logical value indicates whether the schneider correction is used during the conversion from PFM to ICM.
x	Object of class XMatrix.

Value

A XMatrix object.

Methods

bg signature(x = "XMatrix"): Gets the background base frequencies.

bg<- signature(x = "XMatrix"): Sets the background base frequencies.

ID signature(x = "XMatrix"): Gets the ID information.

ID<- signature(x = "XMatrix"): Sets the ID information.

length signature(x = "XMatrix"): Gets the pattern length in nucleotides (i.e. number of columns in the matrix).

reverseComplement signature(x = "PWMatrix"): Generates the reverse complement matrix object. Note than the strand is XMatrix will also be changed to the opossite strand.

as.matrix signature(x = "XMatrix"): Returns the matrix in the XMatrix class.

totalIC signature(x = "ICMatrix"): Returns the information content vector.

Matrix signature(x = "XMatrix"): Gets the matrix stored in XMatrix object.

Matrix<- signature(x = "XMatrix"): Sets the matrix stored in XMatrix object.

matrixClass signature(x = "XMatrix"): Gets the matrix type of a XMatrix object.

matrixClass<- signature(x = "XMatrix"): Sets the matrix type of a XMatrix object.

name signature(x = "XMatrix"): Gets the name information.

name<- signature(x = "XMatrix"): Sets the name information.

strand signature(x = "XMatrix"): Gets the strand information of a XMatrix object.

tags signature(x = "XMatrix"): Gets a list object of tags information.

Author(s)

Ge Tan

See Also

toPWM, toICM

Examples

```
## Constructorpf PFMatrix
## Note that there is no XMatrix() constructor,
## but an XMatrix family of constructors: PFMatrix(), PWMatrix(), ICMatrix()
pfm <- PFMatrix(ID="MA0004.1", name="Arnt", matrixClass="Zipper-Type",
                strand="+", bg=c(A=0.25, C=0.25, G=0.25, T=0.25),
                tags=list(family="Helix-Loop-Helix", species="10090",
                tax_group="vertebrates", medline="7592839", type="SELEX",
                ACC="P53762", pazar_tf_id="TF0000003",
                TFBSshape_ID="11", TFencyclopedia_ID="580"),
                profileMatrix=matrix(c(4,  19, 0,  0,  0,  0,
                                       16, 0,  20, 0,  0,  0,
                                       0,  1,  0,  20, 0,  20,
                                       0,  0,  0,  0,  20, 0),
                                       byrow=TRUE, nrow=4,
                                       dimnames=list(c("A", "C", "G", "T")))
               )

## Construction from a set of binding sites sequences
sitesSeqs <- c("Human Gli1"= "GACCACCCA", "hIGFBP-6"= "GACCCCCCA",
               "HNF-3beta"="GAACACCCA", "hPlakoglobin"= "GACCACCAA",
               "rIGFBP-6"= "GTCCACCCA", "Sox-9"= "GGCCACCCA")
countMatrix <- consensusMatrix(sitesSeqs)
pfm <- PFMatrix(ID="Gli-1", name="Gli-1", profileMatrix=countMatrix)

## Coersion
as.matrix(pfm)
as(pfm, "matrix")

## Methods
pwm <- toPWM(pfm)
reverseComplement(pwm)
length(pfm)
```

XMatrixList *Class* "XMatrixList"

Description

The XMatrixList virtual class is a container for storing a collection of XMatrix objects. No object can be constructed directly from this virtual and it has three subclasses: PFMatrixList, PWMatrixList and ICMatrixList. Basically it is a SimpleList and is designed for manipulating the set of XMatrix objects as a whole.

Usage

```
## Constructors:
PFMatrixList(..., use.names=TRUE)
PWMatrixList(..., use.names=TRUE)
ICMatrixList(..., use.names=TRUE)

## Accessor-like methods:
## S4 method for signature 'XMatrixList'
```

```
ID(x)
## S4 method for signature 'XMatrixList'
name(x)
## S4 method for signature 'XMatrixList'
bg(x)
## S4 method for signature 'XMatrixList'
tags(x)
## S4 method for signature 'XMatrixList'
name(x)
## S4 method for signature 'XMatrixList'
strand(x)
```

Arguments

...	The XMatrix objects are supplied in ....
use.names	A logical value. When TRUE, the names of the XMatrix will be kept.
x	A XMatrixList object.

Value

A XMatrixList object.

Author(s)

Ge Tan

See Also

XMatrix,

Examples

```
data(MA0003.2)
data(MA0004.1)

## Construction of PFMatrixList
pfmList <- PFMatrixList(pfm1=MA0003.2, pfm2=MA0004.1, use.names=TRUE)

## Construction of PFM<atrixList from list of PFMatrix
pfmList <- do.call(PFMatrixList, list(pfm1=MA0003.2, pfm2=MA0004.1))
```

Index

∗ **classes**
MotifSet, 13
SitePairSet, 31
SitePairSetList-class, 32
SiteSet, 33
SiteSetList, 34
XMatrix, 41
XMatrixList, 44
∗ **datasets**
MA0004.1, 12
∗ **methods**
calConservation-methods, 3
deleteMatrixHavingID, 4
dmmEM-methods, 5
PFMSimilarity-methods, 16
runMEME, 21
searchAln, 23
searchPairBSgenome-methods, 26
searchSeq, 27
toICM, 37
toPWM, 39
writeGFF3-methods, 41
∗ **package**
TFBSTools-package, 3
[,MotifSet,ANY,ANY,ANY-method *(MotifSet)*, 13
[,MotifSet,ANY,ANY-method *(MotifSet)*, 13
[,MotifSet-method *(MotifSet)*, 13
[,SiteSet,ANY,ANY,ANY-method *(SiteSet)*, 33
[,SiteSet,ANY,ANY-method *(SiteSet)*, 33
[,SiteSet-method *(SiteSet)*, 33
[-methods *(XMatrix)*, 41

as.data.frame,SitePairSet-method *(SitePairSet)*, 31
as.data.frame,SitePairSetList-method *(SitePairSetList-class)*, 32
as.data.frame,SiteSet-method *(SiteSet)*, 33
as.data.frame,SiteSetList-method *(SiteSetList)*, 34
as.matrix,XMatrix-method *(XMatrix)*, 41

bg *(XMatrix)*, 41
bg,XMatrix-method *(XMatrix)*, 41
bg,XMatrixList-method *(XMatrixList)*, 44
bg<-,XMatrix-method *(XMatrix)*, 41

c,SiteSet-method *(SiteSet)*, 33
c-methods *(SiteSet)*, 33
calConservation *(calConservation-methods)*, 3
calConservation,character,character-method *(calConservation-methods)*, 3
calConservation,character,missing-method *(calConservation-methods)*, 3
calConservation,DNAString,DNAString-method *(calConservation-methods)*, 3
calConservation,DNAStringSet,missing-method *(calConservation-methods)*, 3
calConservation-methods, 3
character, *38*, *39*
colSums,XMatrix-method *(XMatrix)*, 41
consensusMatrix,MotifSet-method *(MotifSet)*, 13

deleteMatrixHavingID, 4
deleteMatrixHavingID,character-method *(deleteMatrixHavingID)*, 4
deleteMatrixHavingID,JASPAR2014-method *(deleteMatrixHavingID)*, 4
deleteMatrixHavingID,JASPAR2016-method *(deleteMatrixHavingID)*, 4
deleteMatrixHavingID,JASPAR2018-method *(deleteMatrixHavingID)*, 4
deleteMatrixHavingID,JASPAR2020-method *(deleteMatrixHavingID)*, 4
deleteMatrixHavingID,JASPAR2022-method *(deleteMatrixHavingID)*, 4
deleteMatrixHavingID,SQLiteConnection-method *(deleteMatrixHavingID)*, 4
dim,XMatrix-method *(XMatrix)*, 41
dmmEM, *21*
dmmEM *(dmmEM-methods)*, 5
dmmEM,ANY-method *(dmmEM-methods)*, 5
dmmEM,matrix-method *(dmmEM-methods)*, 5

dmmEM,PFMatrixList-method
(dmmEM-methods), 5
dmmEM-methods, 5
DNAStringSet, *38*, *39*

end,SiteSet-method (SiteSet), 33

getEmissionProb, 6, *10*
getEmissionProb,TFFMDetail-method
(getEmissionProb), 6
getEmissionProb,TFFMFirst-method
(getEmissionProb), 6
getMatrixByID, 7, *10*
getMatrixByID,character-method
(getMatrixByID), 7
getMatrixByID,JASPAR2014-method
(getMatrixByID), 7
getMatrixByID,JASPAR2016-method
(getMatrixByID), 7
getMatrixByID,JASPAR2018-method
(getMatrixByID), 7
getMatrixByID,JASPAR2020-method
(getMatrixByID), 7
getMatrixByID,JASPAR2022-method
(getMatrixByID), 7
getMatrixByID,SQLiteConnection-method
(getMatrixByID), 7
getMatrixByName, *10*
getMatrixByName (getMatrixByID), 7
getMatrixByName,character-method
(getMatrixByID), 7
getMatrixByName,JASPAR2014-method
(getMatrixByID), 7
getMatrixByName,JASPAR2016-method
(getMatrixByID), 7
getMatrixByName,JASPAR2018-method
(getMatrixByID), 7
getMatrixByName,JASPAR2020-method
(getMatrixByID), 7
getMatrixByName,JASPAR2022-method
(getMatrixByID), 7
getMatrixByName,SQLiteConnection-method
(getMatrixByID), 7
getMatrixSet, *8*, 8
getMatrixSet,character-method
(getMatrixSet), 8
getMatrixSet,JASPAR2014-method
(getMatrixSet), 8
getMatrixSet,JASPAR2016-method
(getMatrixSet), 8
getMatrixSet,JASPAR2018-method
(getMatrixSet), 8
getMatrixSet,JASPAR2020-method
(getMatrixSet), 8
getMatrixSet,JASPAR2022-method
(getMatrixSet), 8
getMatrixSet,SQLiteConnection-method
(getMatrixSet), 8
getPosProb, *6*, 10
getPosProb,TFFMDetail-method
(getPosProb), 10
getPosProb,TFFMFirst-method
(getPosProb), 10

ICMatrix, *29*, *30*
ICMatrix (XMatrix), 41
ICMatrix-class (XMatrix), 41
ICMatrixList (XMatrixList), 44
ICMatrixList-class (XMatrixList), 44
ID (XMatrix), 41
ID,XMatrix-method (XMatrix), 41
ID,XMatrixList-method (XMatrixList), 44
ID<-,XMatrix-method (XMatrix), 41
initializeJASPARDB
(deleteMatrixHavingID), 4
initializeJASPARDB,character-method
(deleteMatrixHavingID), 4
initializeJASPARDB,JASPAR2014-method
(deleteMatrixHavingID), 4
initializeJASPARDB,JASPAR2016-method
(deleteMatrixHavingID), 4
initializeJASPARDB,JASPAR2018-method
(deleteMatrixHavingID), 4
initializeJASPARDB,JASPAR2020-method
(deleteMatrixHavingID), 4
initializeJASPARDB,JASPAR2022-method
(deleteMatrixHavingID), 4
initializeJASPARDB,SQLiteConnection-method
(deleteMatrixHavingID), 4
IUPAC2Matrix, 11

length,MotifSet-method (MotifSet), 13
length,SitePairSet-method
(SitePairSet), 31
length,SiteSet-method (SiteSet), 33
length,XMatrix-method (XMatrix), 41
length-methods (XMatrix), 41

MA0003.2 (MA0004.1), 12
MA0004.1, 12
MA0043 (MA0004.1), 12
MA0048 (MA0004.1), 12
makeFlatFileDir, 12, *18*
matchPWM, *28*
Matrix (XMatrix), 41

matrix, *38, 39*
Matrix,XMatrix-method *(XMatrix)*, 41
Matrix,XMatrixList-method *(XMatrixList)*, 44
Matrix<- *(XMatrix)*, 41
Matrix<-,XMatrix-method *(XMatrix)*, 41
matrixClass *(XMatrix)*, 41
matrixClass,XMatrix-method *(XMatrix)*, 41
matrixClass,XMatrixList-method *(XMatrixList)*, 44
matrixClass<-,XMatrix-method *(XMatrix)*, 41
matrixType *(XMatrix)*, 41
matrixType,ICMatrix-method *(XMatrix)*, 41
matrixType,PFMatrix-method *(XMatrix)*, 41
matrixType,PWMatrix-method *(XMatrix)*, 41
matrixType,XMatrixList-method *(XMatrixList)*, 44
matrixType-methods *(XMatrix)*, 41
MotifSet, 13, *22*
MotifSet-class *(MotifSet)*, 13

name *(XMatrix)*, 41
name,XMatrix-method *(XMatrix)*, 41
name,XMatrixList-method *(XMatrixList)*, 44
name<-,XMatrix-method *(XMatrix)*, 41
ncol,TFFMDetail-method *(TFFM)*, 35
ncol,TFFMFirst-method *(TFFM)*, 35
ncol,XMatrix-method *(XMatrix)*, 41

parseMEMEOutput, 14
pattern,SiteSet-method *(SiteSet)*, 33
permuteMatrix *(permuteMatrix-methods)*, 15
permuteMatrix,matrix-method *(permuteMatrix-methods)*, 15
permuteMatrix,PFMatrix-method *(permuteMatrix-methods)*, 15
permuteMatrix,PFMatrixList-method *(permuteMatrix-methods)*, 15
permuteMatrix-methods, 15
PFMatrix, *12*, *38, 39*
PFMatrix *(XMatrix)*, 41
PFMatrix-class *(XMatrix)*, 41
PFMatrixList, *9*, *38, 39*
PFMatrixList *(XMatrixList)*, 44
PFMatrixList-class *(XMatrixList)*, 44
PFMSimilarity, *17*
PFMSimilarity *(PFMSimilarity-methods)*, 16
PFMSimilarity,matrix,character-method *(PFMSimilarity-methods)*, 16
PFMSimilarity,matrix,matrix-method *(PFMSimilarity-methods)*, 16
PFMSimilarity,matrix,PFMatrix-method *(PFMSimilarity-methods)*, 16
PFMSimilarity,PFMatrix,character-method *(PFMSimilarity-methods)*, 16
PFMSimilarity,PFMatrix,matrix-method *(PFMSimilarity-methods)*, 16
PFMSimilarity,PFMatrix,PFMatrix-method *(PFMSimilarity-methods)*, 16
PFMSimilarity,PFMatrixList,character-method *(PFMSimilarity-methods)*, 16
PFMSimilarity,PFMatrixList,matrix-method *(PFMSimilarity-methods)*, 16
PFMSimilarity,PFMatrixList,PFMatrix-method *(PFMSimilarity-methods)*, 16
PFMSimilarity-methods, 16
primary,SiteSet-method *(SiteSet)*, 33
pvalues *(SiteSet)*, 33
pvalues,SiteSet-method *(SiteSet)*, 33
pvalues,SiteSetList-method *(SiteSetList)*, 34
PWMatrix, *23*, *26, 27*, *34*
PWMatrix *(XMatrix)*, 41
PWMatrix-class *(XMatrix)*, 41
PWMatrixList, *23*, *26–28*
PWMatrixList *(XMatrixList)*, 44
PWMatrixList-class *(XMatrixList)*, 44
PWMSimilarity *(PWMSimilarity-methods)*, 17
PWMSimilarity,matrix,matrix-method *(PWMSimilarity-methods)*, 17
PWMSimilarity,matrix,PWMatrix-method *(PWMSimilarity-methods)*, 17
PWMSimilarity,PWMatrix,matrix-method *(PWMSimilarity-methods)*, 17
PWMSimilarity,PWMatrix,PWMatrix-method *(PWMSimilarity-methods)*, 17
PWMSimilarity,PWMatrixList,matrix-method *(PWMSimilarity-methods)*, 17
PWMSimilarity,PWMatrixList,PWMatrix-method *(PWMSimilarity-methods)*, 17
PWMSimilarity,PWMatrixList,PWMatrixList-method *(PWMSimilarity-methods)*, 17
PWMSimilarity-methods, 17

readJASPARMatrix, 18
readXMLTFFM, 19
relScore *(SiteSet)*, 33
relScore,SiteSet-method *(SiteSet)*, 33
relScore,SiteSetList-method *(SiteSetList)*, 34

reverseComplement,XMatrix-method (XMatrix), 41
rowSums,XMatrix-method (XMatrix), 41
rPWMDmm, *6*
rPWMDmm (rPWMDmm-methods), 19
rPWMDmm,matrix-method (rPWMDmm-methods), 19
rPWMDmm,PFMatrix-method (rPWMDmm-methods), 19
rPWMDmm,PFMatrixList-method (rPWMDmm-methods), 19
rPWMDmm-methods, 19
runMEME, *14, 15*, 21
runMEME,character-method (runMEME), 21
runMEME,DNAStringSet-method (runMEME), 21

sampleRanges, 22
score,SiteSet-method (SiteSet), 33
searchAln, *3*, 23, *27, 28*, *31, 32*, *34, 35*
searchAln,PWMatrix,Axt,missing-method (searchAln), 23
searchAln,PWMatrix,character,character-method (searchAln), 23
searchAln,PWMatrix,character,missing-method (searchAln), 23
searchAln,PWMatrix,DNAString,DNAString-method (searchAln), 23
searchAln,PWMatrix,DNAStringSet,missing-method (searchAln), 23
searchAln,PWMatrixList,character,character-method (searchAln), 23
searchAln,PWMatrixList,character,missing-method (searchAln), 23
searchAln,PWMatrixList,DNAString,DNAString-method (searchAln), 23
searchAln,PWMatrixList,DNAStringSet,missing-method (searchAln), 23
searchPairBSgenome (searchPairBSgenome-methods), 26
searchPairBSgenome,PWMatrix-method (searchPairBSgenome-methods), 26
searchPairBSgenome,PWMatrixList-method (searchPairBSgenome-methods), 26
searchPairBSgenome-methods, 26
searchSeq, *24*, 27, *34, 35*
searchSeq,PWMatrix-method (searchSeq), 27
searchSeq,PWMatrixList-method (searchSeq), 27
searchSeq-methods (searchSeq), 27
seqLogo, 29, *38*
seqLogo,ICMatrix-method (seqLogo), 29
seqLogo,TFFM-method (seqLogo), 29
seqname,SiteSet-method (SiteSet), 33
shannon.entropy, 30
show,SiteSet-method (SiteSet), 33
showAsCell,XMatrixList-method (XMatrixList), 44
site1,SitePairSet-method (SitePairSet), 31
site2,SitePairSet-method (SitePairSet), 31
SitePairSet, *31*, 31, *32*, *34*
SitePairSet-class (SitePairSet), 31
SitePairSetList, *32*
SitePairSetList (SitePairSetList-class), 32
SitePairSetList-class, 32
SiteSet, *28*, 33, *35*
SiteSet-class (SiteSet), 33
SiteSetList, *28*, *34*, 34
SiteSetList-class (SiteSetList), 34
sitesource,SiteSet-method (SiteSet), 33
sitesSeq (MotifSet), 13
sitesSeq,MotifSet-method (MotifSet), 13
start,SiteSet-method (SiteSet), 33
storeMatrix (deleteMatrixHavingID), 4
storeMatrix,character,PFMatrix-method (deleteMatrixHavingID), 4
storeMatrix,character,PFMatrixList-method (deleteMatrixHavingID), 4
storeMatrix,JASPAR2014,PFMatrix-method (deleteMatrixHavingID), 4
storeMatrix,JASPAR2014,PFMatrixList-method (deleteMatrixHavingID), 4
storeMatrix,JASPAR2016,PFMatrix-method (deleteMatrixHavingID), 4
storeMatrix,JASPAR2016,PFMatrixList-method (deleteMatrixHavingID), 4
storeMatrix,JASPAR2018,PFMatrix-method (deleteMatrixHavingID), 4
storeMatrix,JASPAR2018,PFMatrixList-method (deleteMatrixHavingID), 4
storeMatrix,JASPAR2020,PFMatrix-method (deleteMatrixHavingID), 4
storeMatrix,JASPAR2020,PFMatrixList-method (deleteMatrixHavingID), 4
storeMatrix,JASPAR2022,PFMatrix-method (deleteMatrixHavingID), 4
storeMatrix,JASPAR2022,PFMatrixList-method (deleteMatrixHavingID), 4

storeMatrix,SQLiteConnection,PFMatrix-method (deleteMatrixHavingID), 4
storeMatrix,SQLiteConnection,PFMatrixList-method (deleteMatrixHavingID), 4
storeMatrix-methods (deleteMatrixHavingID), 4
strand,SiteSet-method (SiteSet), 33
strand,XMatrix-method (XMatrix), 41
strand,XMatrixList-method (XMatrixList), 44
strand<-,XMatrix-method (XMatrix), 41

tags (XMatrix), 41
tags,XMatrix-method (XMatrix), 41
tags,XMatrixList-method (XMatrixList), 44
TFBSTools (TFBSTools-package), 3
TFBSTools-package, 3
TFFM, *19*, *29*, 35
TFFMDetail, *6*, *10*, *19*
TFFMDetail (TFFM), 35
TFFMFirst, *6*, *10*, *19*
TFFMFirst (TFFM), 35
toGRangesList (toGRangesList-methods), 37
toGRangesList,SitePairSetList,Axt-method (toGRangesList-methods), 37
toGRangesList-methods, 37
toICM, *30*, 37, *40*, *43*
toICM,character-method (toICM), 37
toICM,DNAStringSet-method (toICM), 37
toICM,matrix-method (toICM), 37
toICM,PFMatrix-method (toICM), 37
toICM,PFMatrixList-method (toICM), 37
toPWM, *38*, 39, *43*
toPWM,character-method (toPWM), 39
toPWM,DNAStringSet-method (toPWM), 39
toPWM,matrix-method (toPWM), 39
toPWM,PFMatrix-method (toPWM), 39
toPWM,PFMatrixList-method (toPWM), 39
totalIC (XMatrix), 41
totalIC,ICMatrix-method (XMatrix), 41
totalIC,TFFM-method (TFFM), 35
totalIC-methods (XMatrix), 41

views (SiteSet), 33
views,SiteSet-method (SiteSet), 33

writeGFF2 (writeGFF3-methods), 41
writeGFF2,SitePairSet-method (writeGFF3-methods), 41
writeGFF2,SitePairSetList-method (writeGFF3-methods), 41
writeGFF2,SiteSet-method (writeGFF3-methods), 41
writeGFF2,SiteSetList-method (writeGFF3-methods), 41
writeGFF2-methods (writeGFF3-methods), 41
writeGFF3 (writeGFF3-methods), 41
writeGFF3,SitePairSet-method (writeGFF3-methods), 41
writeGFF3,SitePairSetList-method (writeGFF3-methods), 41
writeGFF3,SiteSet-method (writeGFF3-methods), 41
writeGFF3,SiteSetList-method (writeGFF3-methods), 41
writeGFF3-methods, 41

XMatrix, *36*, *38*, *40*, 41, *45*
XMatrix-class (XMatrix), 41
XMatrixList, 44
XMatrixList,list-method (XMatrixList), 44