Package 'rtfbs'

August 16, 2016

Maintainer Melissa Hubisz <mjhubisz@cornell.edu>

License BSD_3_clause + file LICENSE

Title Transcription Factor Binding Site Identification Tool

Author Nicholas Peterson, Andre Martins, Melissa Hubisz, and Adam Siepel

Description Identifies and scores possible Transcription Factor Binding Sites and allows for FDR analysis and pruning. It supports splitting of sequences based on size or a specified GFF, grouping by G+C content, and specification of Markov model order. The heavy lifting is done in C while all results are made available via R.

Version 0.3.5

URL http://compgen.cshl.edu/rtfbs

Date 2016-08-15

Imports rphast, methods

Depends stats

Collate 'tfbs.R'

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-08-16 19:49:30

R topics documented:

`rtfbs-package` *Transcription Factor Binding Site Identification Tool*

Description

Identifies and scores possible Transcription Factor Binding Sites and allows for FDR analysis and pruning. It supports splitting of sequences based on size or a specified GFF, grouping by G+C content, and specification of Markov model order. The heavy lifting is done in C while all results are made available via R.

Details

Copyright:	Copyright (c) 2002-2016 University of California, Cornell University, Cold Spring Harbor Laboratory.
Package:	rtfbs
License:	BSD_3_clause + file LICENSE
Version:	0.3.5
URL:	http://compgen.cshl.edu/rtfbs
Date:	2016-08-15
Imports:	

	rphast, methods
Depends:	
	stats
Collate:	
	'tfbs.R'
RoxygenNote:	5.0.1
Built:	R 3.1.2; x86_64-unknown-linux-gnu; 2016-08-15 20:31:08 UTC; unix

Index:

Author(s)

Nicholas Peterson, Andre Martins, Melissa Hubisz, and Adam Siepel

Maintainer: Melissa Hubisz <mjhubisz@cornell.edu>

as.pointer.ms *MS To Pointer*

Description

RTFBS can store MS objects in R's memory, or C's memory for efficency reasons. This function transforms an MS object whose data is stored in memory within R into an MS object whose data is stored in C's memory. The result is a new MS object in R whose data is stored in C memory. Copying an MS object from R into C is performed automatically by RTFBS when it needs to run C code. The user will probably not need to call this function.

Usage

```
as.pointer.ms(src)
```

Arguments

src	An MS object stored by value in R

Value

an MS object containing only a pointer to an object created in C which contains the data.

Author(s)

Nick Peterson

See Also

ms for details on MS storage options.

build.mm *Build Markov Model to represent sequences in an MS object*

Description

Build a Markov Model of user specified order to represent sequences in an MS object.

Usage

```
build.mm(ms, order, pseudoCount = 0, considerReverse = FALSE)
```

Arguments

ms	Sequence used to build Markov Model
order	Order of Markov model to build; ie, the number of preceding bases to consider when calculating the probability of a given base.
pseudoCount	(Optional) Integer added to the number of observed cases of each possible sequence pattern.
considerReverse	(Optional) Logical value. If TRUE, considers reverse complement frequencies in addition to forward strand frequencies when building model.

Value

A list of matrices, each representing a markov model from order 0, 1, ..., order. Each matrix gives the probability of observing a particular base (column) given the preceding bases (row).

See Also

read.ms split.ms groupByGC.ms

Examples

```
require("rtfbs")
exampleArchive <- system.file("extdata", "NRSF.zip", package="rtfbs")
seqFile <- "input.fas"
unzip(exampleArchive, seqFile)
# Read in FASTA file "input.fas" from the examples into an
#   MS (multiple sequences) object
ms <- read.ms(seqFile)
# Build a 3rd order Markov Model to represent the sequences
#   in the MS object "ms".  The Model will be a list of
#   matrices  corrisponding in size to the order of the
#   Markov Model
mm <- build.mm(ms, 3);
# Print the list of 4 Markov Matrices that make up the
#   Markov Model
print(mm)
```

---

calc.fdr *Calculate FDR*

Description

Calculate False Discovery Rate (FDR) of possible binding sites. This function uses two sets of scores, realSeqsScores and simSeqsScores. realSeqsScores are scores for the sequences being scanned for binding sites. simSeqsScores are scores for the simulated sequence. The simulated sequences and simSeqsScores must be made using the same Markov Model as the realSeqsScores.

Usage

```
calc.fdr(realSeqs, realSeqsScores, simSeqs, simSeqsScores, interval = 0.01)
```

Arguments

realSeqs	MS object containing non-simulated sequences
realSeqsScores	Feat object obtained from scoring realSeqs
simSeqs	MS object containing simulated sequences
simSeqsScores	Feat object obtained from scoring simSeqs
interval	Float specifying distance between steps at which the FDR will be calculated (lower is better). If NULL, calculate FDR for each unique score.

Value

Data.Frame with two columns 'score' and 'FDR' mapping a single score to a single FDR. Data frame is sorted by score if any exist.

Note

realSeqsScores and simSeqsScores are both objects returned by score.ms; the same arguments (threshold, conservative, strand) should be used in both calls to score.ms or FDR will not be valid.

If calc.fdr returns an fdr of zero for all scores, then you can probably increase the number of significant results by re-running score.ms with a lower threshold for both simulated and real sequences.

See Also

score.ms

Examples

```
require("rtfbs")
exampleArchive <- system.file("extdata", "NRSF.zip", package="rtfbs")
seqFile <- "input.fas"
unzip(exampleArchive, seqFile)
# Read in FASTA file "input.fas" from the examples into an
#   MS (multiple sequences) object
ms <- read.ms(seqFile);
pwmFile <- "pwm.meme"
unzip(exampleArchive, pwmFile)
# Read in Position Weight Matrix (PWM) from MEME file from
#  the examples into a Matrix object
pwm <- read.pwm(pwmFile)
# Build a 3rd order Markov Model to represent the sequences
#   in the MS object "ms".  The Model will be a list of
#   matrices  corrisponding in size to the order of the
#   Markov Model
mm <- build.mm(ms, 3);
# Match the PWM against the sequences provided to find
#   possible transcription factor binding sites.  A
```

```
#   Features object is returned, containing the location
#   of each possible binding site and an associated score.
#   Sites with a negative score are not returned unless
#   we set threshold=-Inf as a parameter.
cs <- score.ms(ms, pwm, mm, threshold=-2)
# Generate a sequence 1000 bases long using the supplied
#   Markov Model and random numbers
v <- simulate.ms(mm, 100000)
# Match the PWM against the sequences provided to find
#   possible transcription factor binding sites.  A
#   Features object is returned, containing the location
#   of each possible binding site and an associated score.
#   Sites with a negative score are not returned unless
#   we set threshold=-Inf as a parameter. Any identified
#   binding sites from simulated data are false positives
#   and used to calculate False Discovery Rate
xs <- score.ms(v, pwm, mm, threshold=-2)
# Calculate the False Discovery Rate for each possible
#   binding site in the Features object CS.  Return
#   a mapping between each binding site score and the
#   associated FDR.
fdr <- calc.fdr(ms, cs, v, xs)
# Print the Data.Frame containing the FDR/Score mapping
fdr
```

`concat.ms` *Concat MS*

Description

Concatinate two or more MS objects

Usage

```
concat.ms(...)
```

Arguments

`...`	MS objects to concatenate

Value

MS object containing all input objects'

from.pointer.ms *MS From Pointer*

Description

RTFBS can store MS objects in R's memory, or C's memory for efficency reasons. This function transforms an MS object whose data is stored in memory on the C side to an MS object whose data is stored on the R side. The result is a new MS object containing data stored in memory on the R side copied from memory on the C side Copying an MS object from C into R enables modification of the MS object using generic R functions. To do the reverse, (copying an MS object from R to C), use the as.pointer.ms function.

Usage

```
from.pointer.ms(src)
```

Arguments

src	An MS object stored by reference (pointer.only=TRUE)

Value

an MS object stored in R. If src is already stored in R, returns the original object.

Author(s)

Nick Peterson

See Also

ms for details on MS storage options.

gcContent.ms *Get GC content of each sequence in an MS object*

Description

Get GC content of each sequence in an MS object

Usage

```
gcContent.ms(ms)
```

Arguments

ms	MS object

Value

numeric vector containing GC content of each sequence in ms. GC content is computed as (# of GC bases)/(# of ACGT bases).

Examples

```
require("rtfbs")
seqs <- ms(seqs=c("AAAA", "ACACACAC", "CGCCG", "ACGTACGTACGT", "CGGGGGGGGGG"),
           paste("fake", 1:5, sep=""))
gcContent.ms(seqs)
```

---

groupByGC.ms *Group sequences by GC*

Description

Group sequences in an MS object by their GC content.

Usage

```
groupByGC.ms(ms, ngroups)
```

Arguments

ms	MS object, containing at least ngroups sequences.
ngroups	Number of quantiles to group sequences into.

Value

List of MS objects, where element i represents the i'th quantile according to GC content.

See Also

read.ms

Examples

```
require("rtfbs")
exampleArchive <- system.file("extdata", "NRSF.zip", package="rtfbs")
seqFile <- "input.fas"
unzip(exampleArchive, seqFile)
# Read in FASTA file "input.fas" from the examples into an
#   MS (multiple sequences) object
seqs <- read.ms(seqFile)
# Group sequences from the "seqs" MS object based on each
#   sequences's GC content into 4 new MS objects, one for
#   each GC content range
groups <- groupByGC.ms(seqs, 4)
sapply(groups, length)
sapply(groups, function(x) {range(gcContent.ms(x))})
```

is.pointer.ms *Data in R or C*

Description

Determine if data in MS object is stored in R (by value) or C (by reference). If data is stored in C, it can be copied to R using the function `from.pointer.ms`. Conversely, if the data is stored in R, it can be copied to C using the function `as.pointer.ms`.

Usage

```
is.pointer.ms(x)
```

Arguments

x	MS object containing at least one sequence

Value

TRUE if data in MS is stored in C, otherwise FALSE indicating it is stored in R

label.matrix *Name PWM & MM rows and columns*

Description

Given a list of PWM or MM matrices, name the rows and columns (if applicable) This function is used for specifying the names of rows and columns for Markov Models and Position Weight Matrices(PWMs) only. In the case of naming PWMs, set columnsOnly to true, as it doesn't make sense to name the rows of a PWM. The columns are named 'A,C,G,T'. The rows are uniquely named, staring with A going to T If there are more than 4 rows, two characters are used AA, AC..., if more than 16 rows 3 characters are used, and so on.

Usage

```
label.matrix(mat, columnsOnly = FALSE)
```

Arguments

mat	Matrix of which have the column and row names populated
columnsOnly	Only name the columns, not the rows

Value

Matrix with columns and (rows optional) named

length.ms *Length of MS object*

Description

Return number of sequences contained in MS object

Usage

```
## S3 method for class 'ms'
length(x)
```

Arguments

x	MS object

Value

Integer containing the number of sequences the MS object contains

lengths.ms *MS sequence lengths*

Description

Returns the length of each sequence in an MS object.

Usage

```
lengths.ms(x)
```

Arguments

x	an MS object

Value

integer vector giving the length of each sequence in the MS object

Author(s)

Nick Peterson

makeFdrPlot *Plot FDR*

Description

Plot one or more false discovery rate data.frame(s). False discovery rate data.frame(s) are created using calc.fdr. This plot enables the user to get an idea of an appropreate FDR threshold to use for determining likely binding sites.

Usage

```
makeFdrPlot(x, xlim = NULL, ylim = NULL, col = "black", lty = 1, ...)
```

Arguments

x	data.frame created by calc.fdr, or list of such data frames
xlim	If given, specifies the x coordinate boundaries. Otherwise these are taken by the observed range of scores
ylim	If given, specifies the y coordinate boundaries. Otherwise these are taken by the observed range of FDR values.
col	The color to plot (see par() for description). Will be recycled to the number of data.frames in x.
lty	The line type (see par() for description). Will be recycled to the number of data.frames in x.
...	Additional arguments to be passed to plot()

Examples

```
require("rtfbs")
exampleArchive <- system.file("extdata", "NRSF.zip", package="rtfbs")
seqFile <- "input.fas"
unzip(exampleArchive, seqFile)
# Read in FASTA file "input.fas" from the examples into an
#   MS (multiple sequences) object
ms <- read.ms(seqFile);
pwmFile <- "pwm.meme"
unzip(exampleArchive, pwmFile)
# Read in Position Weight Matrix (PWM) from MEME file from
#  the examples into a Matrix object
pwm <- read.pwm(pwmFile)
# Build a 3rd order Markov Model to represent the sequences
#   in the MS object "ms".  The Model will be a list of
#   matrices  corrisponding in size to the order of the
#   Markov Model
mm <- build.mm(ms, 3);
# Match the PWM against the sequences provided to find
#   possible transcription factor binding sites.  A
```

```
#   Features object is returned, containing the location
#   of each possible binding site and an associated score.
#   Sites with a negative score are not returned unless
#   we set threshold=-Inf as a parameter.
cs <- score.ms(ms, pwm, mm)
# Generate a sequence 1000 bases long using the supplied
#   Markov Model and random numbers
v <- simulate.ms(mm, 10000)
# Match the PWM against the sequences provided to find
#   possible transcription factor binding sites.  A
#   Features object is returned, containing the location
#   of each possible binding site and an associated score.
#   Sites with a negative score are not returned unless
#   we set threshold=-Inf as a parameter. Any identified
#   binding sites from simulated data are false positives
#   and used to calculate False Discovery Rate
xs <- score.ms(v, pwm, mm)
# Calculate the False Discovery Rate for each possible
#   binding site in the Features object CS.  Return
#   a mapping between each binding site score and the
#   associated FDR.
fdr <- calc.fdr(ms, cs, v, xs)
# Plot the False Discovery Rate v.s. score for one or
#   more groups.  To plot multiple FDR/Score mapping
#   data frames on a single plot, simply supply a list
#   of FDR/Score data.frames
makeFdrPlot(fdr)
```

`ms` *Multiple Sequence (MS) Objects*

Description

Creates a new Multiple Sequences (MS) object to hold given sequences.

Usage

```
ms(seqs, names = NULL, offsets = NULL, pointer.only = FALSE)
```

Arguments

`seqs`	A character vector containing sequences, one per sample
`names`	A character vector identifying the sample name for each sequence. If `NULL`, use "seq1", "seq2", ...
`offsets`	List of integers giving the offset for each sequences from the start of its unsplit sequence. If `NULL`, offsets of zero are assumed for each sequence.
`pointer.only`	a boolean indicating whether returned alignment object should be stored by reference (see Details)

Details

Make a new multiple sequence (MS) object given a vector of character strings. They can be optionally annotated with sample names.

The number of elements in names (if provided) must match the number of elements in seqs.

An alphabet (valid non-missing characters) of "ACGT" is automatically assumed for all sequences that RTFBS operates on.

About storing objects as pointers: If `pointer.only==FALSE`, the MS object will be stored in R and can be viewed and modified by base R code as well as RTFBS functions. Setting `pointer.only=TRUE` will cause the object to be stored by reference, as an external pointer to an object created by C code. This may be necessary to improve performance, but the object can then only be viewed/manipulated via RTFBS functions. Furthermore, if an object is stored as a pointer, then its value is liable to be changed when passed as an argument to a function.

Value

An ms object. These are stored in an array-like format, so that they can be subsetted with the [] operator.

Author(s)

Nick Peterson

See Also

Functions for accessing/viewing ms objects: `sequences.ms`, `offsets.ms`, `length.ms`,`lengths.ms`, `names.ms`, `print.m`

`names.ms` *MS Sequence Names*

Description

Returns a list of sequence names contained in an an MS object.

Usage

```
## S3 method for class 'ms'
names(x)
```

Arguments

`x`	an MS object

Value

a character vector giving the names of the sequences, or NULL if they are not defined

Author(s)

Nick Peterson

`offsets.ms` *Get index offsets*

Description

Return list of index offsets for each sequence in an MS object

Usage

```
offsets.ms(ms)
```

Arguments

`ms`	An MS object

Value

List of integers, each indicating the offset at which this sequence starts compared to the unsplit sequence it came from

---

`output.sites` *Threshold possible binding sites by Score or FDR*

Description

Threshold the possible binding sites based on score, or False Discovery Rate (FDR). To threshold on FDR, you must have computed an FDR/Score map using calc.fdr, and chosen an FDR threshold, for which makeFdrPlot() is helpful.

Usage

```
output.sites(seqsScores, scoreThreshold = NULL, fdrScoreMap = NULL,
  fdrThreshold = NULL)
```

Arguments

`seqsScores`	score.ms output representing scores for candidate binding sites
`scoreThreshold`	A numeric value giving the lower score boundary significance threshold. Sequences with scores higher than this boundary will be selected. (Not required if thresholding by FDR.)
`fdrScoreMap`	calc.fdr output giving mapping between score/FDR (only required if thresholding by FDR).
`fdrThreshold`	A numeric value between 0 and 1 giving upper FDR boundary- any site with a lower FDR score will be output. (only required if thresholding by FDR)

Value

Features object containing thresholded Transcription Factor Binding Sites, their locations, scores, strand, etc. If thresholding by score, this is equivalent to `seqsScores[seqsScores$score > scoreThreshold,]`.

Examples

```
require("rtfbs")
exampleArchive <- system.file("extdata", "NRSF.zip", package="rtfbs")
seqFile <- "input.fas"
unzip(exampleArchive, seqFile)
# Read in FASTA file "input.fas" from the examples into an
#   MS (multiple sequences) object
ms <- read.ms(seqFile);
pwmFile <- "pwm.meme"
unzip(exampleArchive, pwmFile)
# Read in Position Weight Matrix (PWM) from MEME file from
#  the examples into a Matrix object
pwm <- read.pwm(pwmFile)
# Build a 3rd order Markov Model to represent the sequences
#   in the MS object "ms".  The Model will be a list of
#   matrices  corrisponding in size to the order of the
#   Markov Model
mm <- build.mm(ms, 3);
# Match the PWM against the sequences provided to find
#   possible transcription factor binding sites.  A
#   Features object is returned, containing the location
#   of each possible binding site and an associated score.
#   Sites with a negative score are not returned unless
#   we set threshold=-Inf as a parameter.
cs <- score.ms(ms, pwm, mm)
# Generate a sequence 1000 bases long using the supplied
#   Markov Model and random numbers
v <- simulate.ms(mm, 10000)
# Match the PWM against the sequences provided to find
#   possible transcription factor binding sites.  A
#   Features object is returned, containing the location
#   of each possible binding site and an associated score.
#   Sites with a negative score are not returned unless
#   we set threshold=-Inf as a parameter. Any identified
#   binding sites from simulated data are false positives
#   and used to calculate False Discovery Rate
xs <- score.ms(v, pwm, mm)
# Calculate the False Discovery Rate for each possible
#   binding site in the Features object CS.  Return
#   a mapping between each binding site score and the
#   associated FDR.
fdr <- calc.fdr(ms, cs, v, xs)
# Output identified transcription factor binding sites
#   below a 0.5 FDR threshold
output.sites(cs, NULL, fdr, 0.5)
# OR
# Output identified transcription factor binding sites
```

```
#   above a 5.2 score threshold
output.sites(cs, 5.2)
```

`print.ms` *Prints an MS (multiple sequence) object. #' @title Printing MS objects*

Description

Prints an MS (multiple sequence) object. #' @title Printing MS objects

Usage

```
## S3 method for class 'ms'
print(x, ..., print.all = ((length(x) < 15) &&
  (sum(lengths.ms(x)) < 500)))
```

Arguments

`x`	an object of class ms
`...`	additional arguments sent to `print`
`print.all`	whether to suppress printing of the bases, names, offsets

Author(s)

Nick Peterson

`read.mm` *Read Markov Model from file*

Description

Read in a Markov model from file created by MEME's fasta-get-markov tool

Usage

```
read.mm(filename)
```

Arguments

`filename`	Full path to output file from fasta-get-markov

Value

Markov Model (list of Markov Matrices, one for each order)

`read.ms` *Reading in sequences from file*

Description

Read Multiple Sequences from a FASTA file.

Usage

```
read.ms(filename, pointer.only = FALSE)
```

Arguments

`filename`	The name of the input file containing the sequences. File should be in FASTA format.
`pointer.only`	If `TRUE`, sequences within the MS will be stored by reference as external pointers to objects created by C code, rather than directly in R memory. This improves performance and may be necessary for large files, but reduces functionality.

Value

MS object containing sequences read from file

See Also

`ms` for further description of MS objects and pointer.only option.

Examples

```
require("rtfbs")
exampleArchive <- system.file("extdata", "NRSF.zip", package="rtfbs")
seqFile <- "input.fas"
unzip(exampleArchive, seqFile)
# Read in FASTA file "input.fas" from the examples into an
#   MS (multiple sequences) object
seqs <- read.ms(seqFile)
# Print the number of sequences in the MS object and whether
#   stored in C or R
print(seqs)
```

read.pwm *Read PWM object*

Description

Read and log transform Position Weight Matrices (PWMs) from file. Each position weight matrix represents a Transcription Factor motif that is being searched for. Files may contain one or more motifs. Only MEME Text formated files are supported. Look in the examples below to find an example PWM file.

Usage

```
read.pwm(filename)
```

Arguments

filename	Full Path to file of MEME Text format

Value

Matrix object containing the Position Weight Matrix read from the file. If the file contains more than one PWM, a list of Matrix objects are returned, one for each PWM. The returned PWMs are log transformed, so that entry [i,j] of the matrix represents the log probability of observing the base from column j in the i'th position of a transcription factor binding site.

Examples

```
require("rtfbs")
exampleArchive <- system.file("extdata", "NRSF.zip", package="rtfbs")
pwmFile <- "pwm.meme"
unzip(exampleArchive, pwmFile)
# Read in Position Weight Matrix (PWM) from MEME file from
#  the examples into a Matrix object
pwm <- read.pwm(pwmFile)
# Print PWM as an R matrix
print(pwm)
```

score.ms *Score sequences against a PWM*

Description

Score all potential binding sites in an MS object. If a PWM has N rows, then score every observed N-mer in the MS object. The score is given by the log likelihood of the N-mer given the PWM, minus the log likelihood of the N-mer under the Markov model specified by mm. By default, only potential binding sites with scores > 0 are returned, but this can be modified with the threshold argument.

Usage

```
score.ms(ms, pwm, mm, conservative = TRUE, threshold = 0, strand = "best",
  return_posteriors = FALSE)
```

Arguments

ms	MS object containing at least one sequence
pwm	Position Weight matrix representing transcription factor motif
mm	Markov Model associated with given sequences, which represents the null model
conservative	(Logical value) If TRUE, sequences containing N's are given a log likelihood of negative infinity under the PWM model. If FALSE, any 'N' encountered does not contributes to the score.
threshold	(Numeric value) Only sites with scores above this threshold are returned (default = 0)
strand	One of "best", "both", "+", or "-" specifying which strand(s) to return results for. If "both" search for binding sites in both directions, return all results found. If "best" search for binding sites in both directions, but for each N-mer, return the maximum score over either strand. If "+" look only on the forward strand, and if "-" look only on the reverse strand.
return_posteriors	If TRUE, will return a list structure. Scores represent the motif 'match score', or the product of the probability of observing each base under the motif or background models. Scores are returned under the motif model for all positions in the sequence, on both forward and reverse strands, and under the background model. Note that strand and threshold options are both ignored. If FALSE, returns scores and locations for possible binding sites as a feature object.

Value

Scores and locations for possible binding sites returned as a feature object. Optionally, if return_posteriors is TRUE, will return a list structure (see above).

Note

If a PWM file contains multiple PWMs, then read.pwm will return a list of PWMs. This function takes a single PWM.

See Also

read.ms split.ms groupByGC.ms build.mm read.pwm

Examples

```
require("rtfbs")
exampleArchive <- system.file("extdata", "NRSF.zip", package="rtfbs")
seqFile <- "input.fas"
unzip(exampleArchive, seqFile)
# Read in FASTA file "input.fas" from the examples into an
```

```
#   MS (multiple sequences) object
ms <- read.ms(seqFile);
pwmFile <- "pwm.meme"
unzip(exampleArchive, pwmFile)
# Read in Position Weight Matrix (PWM) from MEME file from
#  the examples into a Matrix object
pwm <- read.pwm(pwmFile)
# Build a 3rd order Markov Model to represent the sequences
#   in the MS object "ms".  The Model will be a list of
#   matrices  corrisponding in size to the order of the
#   Markov Model
mm <- build.mm(ms, 3);
# Match the PWM against the sequences provided to find
#   possible transcription factor binding sites.  A
#   Features object is returned, containing the location
#   of each possible binding site and an associated score.
#   Sites with a negative score are not returned unless
#   we set threshold=-Inf as a parameter.
score.ms(ms, pwm, mm)
```

`sequences.ms` *Get sequences*

Description

Returns list of sequences in an MS object

Usage

```
sequences.ms(ms)
```

Arguments

`ms`	An MS object

Value

character vector containing the sequences in the MS object

simulate.ms *Generate sequence from Markov Model*

Description

Simulate a single sequence based from a Markov Model. These are referred to as simulated sequences and used compute the background rates and False Discovery Rates.

Usage

```
## S3 method for class 'ms'
simulate(object, nsim, seed = NULL, pointer.only = FALSE, ...)
```

Arguments

object	Markov Model build.mm
nsim	Length of the sequence to simulate. Can be a vector, in which case multiple sequences of the specified length will be simulated.
seed	A random number seed. Either NULL (the default; do not re-seed random number generator), or an integer to be sent to set.seed.
pointer.only	If TRUE, keep sequence data stored in a C structure, otherwise it is automatically copied into an R object.
...	Not used; for S3 compatibility

Value

MS object containing a single sequence with nsim bases.

See Also

build.mm for details on Markov models, ms for details on MS objects

Examples

```
require("rtfbs")
exampleArchive <- system.file("extdata", "NRSF.zip", package="rtfbs")
seqFile <- "input.fas"
unzip(exampleArchive, seqFile)
# Read in FASTA file "input.fas" from the examples into an
#   MS (multiple sequences) object
ms <- read.ms(seqFile);
# Build a 3rd order Markov Model to represent the sequences
#   in the MS object "ms".  The Model will be a list of
#   matrices  corrisponding in size to the order of the
#   Markov Model
mm <- build.mm(ms, 3);
# Generate a sequence 1000 bases long using the supplied
#   Markov Model and random numbers
v <- simulate.ms(mm, 1000);
```

split.ms *Split sequences*

Description

Split sequences in MS object at given locations, or based on a numeric window size. If splitting based on a numeric window size, (i.e. every 500 bases) specify the maximum number of bases per sequence in the `f` parameter. If splitting based on specific locations, use read.feat(FullPath) to read in a BED file specifying the locations, and use the resulting object in the `f` parameter.

Usage

```
## S3 method for class 'ms'
split(x, f, drop = FALSE, ...)
```

Arguments

`x`	Multiple sequences object
`f`	Numeric window size, or Features object used to determine where to split sequences
`drop`	Currently not used (for S3 compatibility)
`...`	Currently not used (for S3 compatibility)

Value

MS object, containing the split sequences

See Also

`read.ms`, or read.feat from package rphast for more about Features object

summary.ms *MS Summary*

Description

Prints a short description of an Multiple Sequence (MS) object. Omits names, offsets, and bases by default, but these can be printed using the print.all argument.

Usage

```
## S3 method for class 'ms'
summary(object, ..., print.all = (length.ms(object) < 15 &&
  sum(lengths.ms(object)) < 500))
```

Arguments

object	MS object
...	Not used (exists for S3 compatibility)
print.all	whether to suppress printing of the bases, offsets, and names. If TRUE prints sequences, offsets, and names of all sequences.

Author(s)

Nick Peterson

See Also

print.ms

---

write.mm *Write Markov Model to file*

Description

Write markov model into the format used by MEME's fasta-get-markov tool

Usage

```
write.mm(x, file = NULL, ...)
```

Arguments

x	Markov Model (list of Markov Matrices, one for each order)
file	Full path of where to save file, if NULL print to screen
...	Not Used (for S3 compatability)

---

write.ms *Writing MS Object to FASTA file*

Description

Writes a multiple sequence (MS) object to a FASTA file.

Usage

```
write.ms(x, file=NULL)
```

Arguments

x	an object of class ms
file	File to write (will be overwritten). If NULL, output goes to terminal.

Author(s)

Nick Peterson

[.ms *Extract, replace, reorder MS*

Description

Treat multiple sequences as a array where each row corresponds to a sequence for one species.

Usage

```
## S3 method for class 'ms'
x[rows]
```

Arguments

x	An object of type ms
rows	A numeric vector of sequence indices, character vector (containing sequence name), or logical vector (containing sequences to keep). If logical vector it will be recycled as necessary to the same length as nrow.ms(x).

Details

The bracket notation can return a set of sequences, or re-order rows.

Value

An MS object sampled from x as indicated by rows.

Note

This function will not alter the value of x even if it is stored as a pointer to a C structure.

Author(s)

Nick Peterson

Index