Package 'sarks'

November 21, 2024

Title Suffix Array Kernel Smoothing for discovery of correlative sequence motifs and multi-motif domains

Version 1.18.0

Description Suffix Array Kernel Smoothing (see https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797), or SArKS, identifies sequence motifs whose presence correlates with numeric scores (such as differential expression statistics) assigned to the sequences (such as gene promoters). SArKS smooths over sequence similarity, quantified by location within a suffix array based on the full set of input sequences. A second round of smoothing over spatial proximity within sequences reveals multi-motif domains. Discovered motifs can then be merged or extended based on adjacency within MMDs. False positive rates are estimated and controlled by permutation testing.

Depends R (>= 4.0)

Imports rJava, Biostrings, IRanges, utils, stats, cluster, binom

Suggests RUnit, BiocGenerics, ggplot2

biocViews MotifDiscovery, GeneRegulation, GeneExpression, Transcriptomics, RNASeq, DifferentialExpression, FeatureExtraction

SystemRequirements Java (>= 1.8)

License BSD_3_clause + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 6.1.1

BugReports https://github.com/denniscwylie/sarks/issues

URL https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797, https://github.com/denniscwylie/sarks

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

git_branch RELEASE_3_20

git_last_commit 6e332ff

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-20

Author Dennis Wylie [aut, cre] (<https://orcid.org/0000-0003-0380-3549>)

Maintainer Dennis Wylie <denniscwylie@gmail.com>

Contents

`blockInfo` *Get sarks smoothed scores for input sequence*

Description

Returns a data.frame containing the smoothed scores (including spatially smoothed scores, if applicable) as well as other useful sarks parameters for one or more specified input sequences (blocks).

Usage

```
blockInfo(sarks, block, filters, thresholds, kMax = 12L)
```

Arguments

`sarks`	Sarks object from which information will be derived.
`block`	character vector of names of sequence(s) for which results are desired
`filters`	output from sarksFilters function indicating what combinations of filter parameters halfWindow, spatialLength, and minGini were used.
`thresholds`	output from permutationThresholds specifying thresholds used for k-mer peak calling.
`kMax`	integer value indicating the maximum k-mer length to be reported.

Value

data.frame in same format as result of kmerPeaks giving detailed information about every spatial position within specified sequences/blocks.

References

Wylie, D.C., Hofmann, H.A., and Zemelman, B.V. (2019) SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing, Bioinformatics, Vol. 35(20), 3944-3952

https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
filters <- sarksFilters(halfWindow=4, spatialLength=0, minGini=1.1)
permDist <- permutationDistribution(sarks, 250, filters, seed=123)
thresholds <- permutationThresholds(filters, permDist, nSigma=2.0)
bi24 <- blockInfo(sarks, '24', filters, thresholds)
```

---

blockScores *SArKS input scores*

Description

Extracts vector of input scores associated with input sequences from sarks object.

Usage

```
blockScores(sarks)
```

Arguments

sarks	Sarks object from which information will be derived

Value

named numeric vector; names are the sequence names, values are the associated scores. Note: Sarks internally sorts input lexicographically by sequence name.

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
simulatedScores2 <- blockScores(sarks)
## simulatedScores2 will be in different order than simulatedScores,
## but contains same information.
```

`clusterCounts` *Count occurrences of k-mer clusters*

Description

Counts how often any k-mer from a cluster of k-mers (or list of clusters of k-mers) occurs in each element of a character vector.

Usage

```
clusterCounts(kmers, seqs, directional = TRUE, overlap = FALSE)
```

Arguments

`kmers`	character vector or XStringSet of k-mers composing cluster to search for, or a named list of such character vectors or XStringSet objects to count multiple clusters.
`seqs`	character vector or XStringSet of sequences in which to search for and count occurrences of kmers.
`directional`	logical value: if FALSE, counts occurrences of either cluster(s) of k-mers or their reverse-complements. Makes sense only if applying to DNA sequences!
`overlap`	logical value: should overlapping occurrences of k-mers be counted as multiple hits?

Value

if cluster is a single character vector or XStringSet (of any length), returns integer vector of counts; if cluster is a list of character vectors, returns matrix of counts: one row per sequence in seqs, one column per character vector/XStringSet in cluster

Examples

```
seqs <- c(
    line1 = "My mind's got a mind of its own",
    line2 = "Takes me out to parties when I'd rather be alone",
    line3 = "Takes me out a-walkin' when I'd rather be at home"
)
clusters <- list(
    antisocial = c('alone', 'at home'),
    mind = 'mind'
)
clCounts <- clusterCounts(clusters, seqs)
```

clusterKmers *Cluster k-mers*

Description

Takes a set of k-mer sequences and returns a list of partitioning the input k-mers into clusters of more similar k-mers. Hierarchical clustering (average linkage) is performed based on Jaccard coefficient distance metric applied treating each k-mer as the set of all tetramers which can be found as substrings within it.

Usage

```
clusterKmers(kmers, k = 4, nClusters = NULL, maxClusters = NULL,
    directional = TRUE)
```

Arguments

kmers	character vector or XStringSet of k-mers to partition into clusters
k	length of sub-k-mers (default k=4 to use tetramers) with which to calculate Jaccard distances for clustering
nClusters	number of clusters to partition kmers into; if set to NULL (default value), selects number of clusters to maximize the average silhouette score (https://en.wikipedia.org/wiki/Silhouette_(clustering)).
maxClusters	if nClusters not specified, can optionally set maximum number of clusters allowed in silhouette score optimization.
directional	logical value: if FALSE, considers each kmer as equivalent to its reverse-complement. Makes sense only if applying to DNA sequences!

Value

list of character vectors (or XStringSet objects as per the class of kmers argument) partitioning kmers into clusters: the character vector at the i-th element of the output list contains the elements from kmers assigned to cluster i.

Examples

```
kmers <- c(
    'CAGCCTGG', 'CCTGGAA', 'CAGCCTG', 'CCTGGAAC', 'CTGGAACT',
    'ACCTGC', 'CACCTGC', 'TGGCCTG', 'CACCTG', 'TCCAGC',
    'CTGGAAC', 'CACCTGG', 'CTGGTCTA', 'GTCCTG', 'CTGGAAG', 'TTCCAGC'
)
clusterKmers(kmers, directional=FALSE)
```

`estimateFalsePositiveRate` *Estimating SArKS false positive rate*

Description

Run second permutation test using the specified number of repetitions, keeping track of maximum observed windowed and spatially-windowed smoothed scores for each combination of filter parameters for each permutation, and comparing these values to thresholds determined by first round of permutation testing.

Usage

```
estimateFalsePositiveRate(sarks, reps, filters, thresholds, seed = NULL,
    conf.level = 0.95)
```

Arguments

`sarks`	Sarks object to test.
`reps`	integer specifying how many repetitions to test.
`filters`	output from sarksFilters function indicating what combinations of filter parameters halfWindow, spatialLength, and minGini to use.
`thresholds`	output from permutationThresholds specifying thresholds for k-mer peak calling.
`seed`	optional seed for random number generator (use in case reproducibility of output is desired). NOTE: do not use the same seed passed to initial permutationDistribution call used to set thresholds.
`conf.level`	level of confidence to be used in the false positive rate confidence interval.

Value

named list with three elements: 'permutation' containing the output from permutationDistribution run.

References

Wylie, D.C., Hofmann, H.A., and Zemelman, B.V. (2019) SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing, Bioinformatics, Vol. 35(20), 3944-3952

https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
filters <- sarksFilters(halfWindow=4, spatialLength=0, minGini=1.1)
permDist <- permutationDistribution(sarks, 250, filters, seed=123)
thresholds <- permutationThresholds(filters, permDist, nSigma=2.0)
fpr <- estimateFalsePositiveRate(
```

```
            sarks, 250, filters, thresholds, seed=123456)
```

`extendKmers` *Extend k-mers in length where possible*

Description

Extend k-mers when adding flanking characters from region in input sequence from which they are derived would result in another reported l-mer string (l > k).

Usage

```
extendKmers(sarks, sarksTable)
```

Arguments

`sarks`	Sarks object used to obtain k-mer peak call set.
`sarksTable`	data.frame containing called k-mer peaks information (format as output from kmerPeaks function).

Value

modified data.frame containing called k-mer peaks information (format as output from kmerPeaks function).

References

Wylie, D.C., Hofmann, H.A., and Zemelman, B.V. (2019) SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing, Bioinformatics, Vol. 35(20), 3944-3952

https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
filters <- sarksFilters(halfWindow=4, spatialLength=0, minGini=1.1)
permDist <- permutationDistribution(sarks, 250, filters, seed=123)
thresholds <- permutationThresholds(filters, permDist, nSigma=2.0)
peaks <- kmerPeaks(sarks, filters, thresholds)
prunedPeaks <- pruneIntervals(peaks)
extendedPeaks <- extendKmers(sarks, prunedPeaks)
```

kmerCounts *Count occurrences of k-mer*

Description

Counts how often a k-mer (or vector of k-mers) occurs in each element of a character vector.

Usage

```
kmerCounts(kmer, seqs, directional = TRUE, overlap = FALSE)
```

Arguments

kmer	character vector or XStringSet of k-mers to search for.
seqs	character vector or XStringSet of sequences in which to search for and count occurrences of kmer.
directional	logical value: if FALSE, counts occurrences of either kmer or its reverse-complement. Makes sense only if applying to DNA sequences!
overlap	logical value: should overlapping occurrences of kmer be counted as multiple hits?

Value

if length(kmer) is one, returns integer vector of counts; if length(kmer) is more than one, returns matrix of counts: one row per sequence in seqs, one column per expression in regex

Examples

```
data(simulatedSeqs)
motifCounts <- kmerCounts('CATACTGAGA', simulatedSeqs)
otherCounts <- kmerCounts(
    c('AAAAA', 'CG'),
    simulatedSeqs,
    directional = FALSE
)
```

kmerPeaks *Call k-mer peaks*

Description

SArKS identifies sets of short subsequences (k-mers) whose presence as substrings of sequences from the input sequence set tends to be associated with elevated sequence scores. Such k-mers are identified as "peaks" where kernel-smoothed scores exceed specified thresholds (generally set by permutation method).

Usage

```
kmerPeaks(sarks, filters, thresholds, peakify = TRUE, kMax = 12L)
```

Arguments

sarks	Sarks object to use for k-mer peak calling.
filters	output from sarksFilters function indicating what combinations of filter parameters halfWindow, spatialLength, and minGini to use.
thresholds	output from permutationThresholds specifying thresholds for k-mer peak calling.
peakify	logical value specifying whether to restrict output to only spatial positions at which the smoothed score is at least as high as either neighboring position or not.
kMax	integer value indicating the maximum k-mer length to be reported.

Value

data.frame containing called k-mer peak information.

References

Wylie, D.C., Hofmann, H.A., and Zemelman, B.V. (2019) SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing, Bioinformatics, Vol. 35(20), 3944-3952

https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
filters <- sarksFilters(halfWindow=4, spatialLength=0, minGini=1.1)
permDist <- permutationDistribution(sarks, 250, filters, seed=123)
thresholds <- permutationThresholds(filters, permDist, nSigma=2.0)
peaks <- kmerPeaks(sarks, filters, thresholds)
```

---

locateClusters *Locate occurrences of specified clusters of k-mers*

Description

Find locations of matches of list of character vectors of k-mers in each element of a named character vector. Not case sensitive.

Usage

```
locateClusters(clusters, seqs, directional = TRUE, showMatch = FALSE)
```

Arguments

clusters	list of character vectors or XStringSet objects of k-mers to search for
seqs	character vector or XStringSet of sequences in which to locate kmer
directional	logical value: if FALSE, counts occurrences of either k-mers within each cluster or their reverse-complements. Makes sense only if applying to DNA sequences!
showMatch	logical value; if true add additional column to output indicating what the exact regex match for each occurrence (can be slow)

Value

data.frame with three columns: 'seqid' containing the name of the sequence from seqs in which the match was found; 'cluster' indicating the cluster from wich a k-mer was located; and 'location' giving the 1-based position at which the match was found.

Examples

```
seqs <- c(
    line1 = "My mind's got a mind of its own",
    line2 = "Takes me out to parties when I'd rather be alone",
    line3 = "Takes me out a-walkin' when I'd rather be at home"
)
clusters <- list(
    antisocial = c('alone', 'at home'),
    mind = 'mind'
)
clusterLoci <- locateClusters(clusters, seqs)
```

---

locateKmers *Locate occurrences of specified k-mers*

Description

Find locations of matches of vector of k-mers in each element of a named character vector. Not case sensitive.

Usage

```
locateKmers(kmers, seqs, directional = TRUE, showMatch = FALSE)
```

Arguments

kmers	character vector or XStringSet of k-mers to search for
seqs	named character vector or XStringSet of sequences in which to locate kmer
directional	logical value: if FALSE, counts occurrences of either kmers or their reverse-complements. Makes sense only if applying to DNA sequences!
showMatch	logical value; if true add additional column to output indicating what the exact regex match for each occurrence (can be slow)

Value

data.frame with three columns: 'seqid' containing the name of the sequence from seqs in which the k-mer was found; 'kmer' indicating the k-mer located; and 'location' giving the 1-based position at which the match was found.

Examples

```
data(simulatedSeqs)
kmerLoci <- locateKmers(c('AAAAA', 'CATACTGAGA'), simulatedSeqs)
```

`mergedKmerSubPeaks` *Identify and merge k-mer sub-peaks within multi-motif domains*

Description

When spatials smoothing is employed, SArKS identifies spatial windows containing elevated spatially-averaged sequence-smoothed scores (multi-motif domains, or MMDs). This function finds k-mers within these MMDs whose sequence-smoothed scores are above the threshold used for MMD calling and merges such k-mers when their spatial positions overlap.

Usage

```
mergedKmerSubPeaks(sarks, filters, thresholds, peakify = TRUE,
    kMax = 12L)
```

Arguments

`sarks`	Sarks object to use for k-mer peak calling.
`filters`	output from sarksFilters function indicating what combinations of filter parameters halfWindow, spatialLength, and minGini to use.
`thresholds`	output from permutationThresholds specifying thresholds for k-mer peak calling.
`peakify`	logical value specifying whether to restrict initial k-mer peak calling to only spatial positions at which the smoothed score is at least as high as either neighboring position (or not).
`kMax`	integer value indicating the maximum k-mer length for initial k-mer peak calling.

Value

data.frame containing called k-mer peak information.

References

Wylie, D.C., Hofmann, H.A., and Zemelman, B.V. (2019) SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing, Bioinformatics, Vol. 35(20), 3944-3952

https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 3, 1)
filters <- sarksFilters(halfWindow=4, spatialLength=3, minGini=1.1)
permDist <- permutationDistribution(sarks, 250, filters, seed=123)
thresholds <- permutationThresholds(filters, permDist, nSigma=4.0)
mergedSubPeaks <- mergedKmerSubPeaks(sarks, filters, thresholds)
```

`permutationDistribution` *Estimating distribution of maximum smoothed sequence scores under permutation*

Description

Run permutation test using the specified number of repetitions, keeping track of maximum observed windowed and spatially-windowed smoothed scores for each combination of filter parameters for each permutation.

Usage

```
permutationDistribution(sarks, reps, filters, seed = NULL)
```

Arguments

`sarks`	Sarks object to test.
`reps`	integer specifying how many repetitions to test.
`filters`	output from sarksFilters function indicating what combinations of filter parameters halfWindow, spatialLength, and minGini to use.
`seed`	optional seed for random number generator (use in case reproducibility of output is desired).

Value

named list with three elements: 'windowed' containing a data.frame with the maximum smoothed scores for each permutation at each combination of filter parameter values, 'spatial' containing a data.frame with the maximum spatially-smoothed scores for each permutation and each filter parameter specification, and '.java' containing the R representation of the java object encoding this information.

References

Wylie, D.C., Hofmann, H.A., and Zemelman, B.V. (2019) SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing, Bioinformatics, Vol. 35(20), 3944-3952

https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
filters <- sarksFilters(halfWindow=4, spatialLength=0, minGini=1.1)
permDist <- permutationDistribution(sarks, 250, filters, seed=123)
```

`permutationThresholds` *Set smoothed score thresholds based on permutation distribution*

Description

Calculate thresholds for SArKS k-mer calling from permutation distribution.

Usage

```
permutationThresholds(filters, permDist, nSigma = 4)
```

Arguments

`filters`	output from sarksFilters function indicating what combinations of filter parameters halfWindow, spatialLength, and minGini to use.
`permDist`	output from permutationDistribution function.
`nSigma`	number of standard deviations above mean of permutation distribution at which to set threshold for either windowed or spatially-windowed score.

Value

named list with two elements: 'theta' containing a data.frame with the threshold information and '.java' containing an R representation of the java object with this information.

References

Wylie, D.C., Hofmann, H.A., and Zemelman, B.V. (2019) SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing, Bioinformatics, Vol. 35(20), 3944-3952

https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
filters <- sarksFilters(halfWindow=4, spatialLength=0, minGini=1.1)
permDist <- permutationDistribution(sarks, 250, filters, seed=123)
thresholds <- permutationThresholds(filters, permDist, nSigma=2.0)
```

`pruneIntervals` *Prune nested k-mer intervals from called k-mer peak set.*

Description

Every k-mer identified by SArKS is derived as a substring defined by the interval running position i to position i+k-1 of the concatenation of all input sequences. In some cases a j-mer (with j < k) may be separately identified as a peak by SArKS for which the j-mer interval is entirely contained within [i, i+k-1]; this function removes such nested intervals from the reported collection of peaks.

Usage

```
pruneIntervals(intervals, start = "s", end = NULL)
```

Arguments

`intervals`	data.frame containing called k-mer peaks information (format as output from kmerPeaks function).
`start`	name of column in intervals data.frame containing interval start coordinates
`end`	name of column in interval data.frame containing interval end coordinates; if no such column present, default NULL value indicates that end coordinates should be obtained by adding nchar(intervals$kmer) to the start coordinates to obtain end coordinates.

Value

modified data.frame containing called k-mer peaks information (format as output from kmerPeaks function).

References

Wylie, D.C., Hofmann, H.A., and Zemelman, B.V. (2019) SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing, Bioinformatics, Vol. 35(20), 3944-3952

https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
filters <- sarksFilters(halfWindow=4, spatialLength=0, minGini=1.1)
permDist <- permutationDistribution(sarks, 250, filters, seed=123)
thresholds <- permutationThresholds(filters, permDist, nSigma=2.0)
peaks <- kmerPeaks(sarks, filters, thresholds)
prunedPeaks <- pruneIntervals(peaks)
```

regexCounts *Count occurrences of regular expression*

Description

Counts how often a regular expression (or vector of regular expressions) occurs in each element of a character vector.

Usage

```
regexCounts(regex, seqs, overlap = FALSE)
```

Arguments

regex	character vector of regular expressions to search for
seqs	character vector or XStringSet of sequences in which to search for and count occurrences of regex
overlap	logical value: should overlapping occurrences of regex be counted as multiple hits?

Value

if length(regex) is one, returns integer vector of counts; if length(regex) is more than one, returns matrix of counts: one row per sequence in seqs, one column per expression in regex

Examples

```
data(simulatedSeqs)
reCounts1 <- regexCounts('AAAAA|TTTTT', simulatedSeqs)
reCounts2 <- regexCounts(c('AAAAA|TTTTT', 'CG'), simulatedSeqs)
```

regexLocate *Locate occurrences of regular expression*

Description

Find locations of matches of a regular expression (or vector of regular expressions) in each element of a named character vector. Not case sensitive.

Usage

```
regexLocate(regex, seqs, showMatch = FALSE)
```

Arguments

regex	character vector or XStringSet of regular expressions to search for
seqs	named character vector or XStringSet of sequences in which to locate regex
showMatch	logical value; if true add additional column to output indicating what the exact regex match for each occurrence (can be slow)

Value

If only a single regex is searched for: data.frame with two columns: 'seqid' containing the name of the sequence from seqs in which the regex was found and 'location' giving the 1-based position at which the regex was found. If length(regex) greater than one, adds additional column 'regex' indicating the name of the regex located.

Examples

```
data(simulatedSeqs)
reLoci <- regexLocate('AAAAA|TTTTT', simulatedSeqs)
```

Sarks *Suffix Array Kernel Smoothing*

Description

Sarks class implements suffix array kernel smoothing for de novo correlative motif discovery.

Usage

```
Sarks(fasta, scores, halfWindow, spatialLength = 0L, nThreads = 1L)
```

Arguments

fasta	specification of fasta file containing sequences to be analyzed; may also be a *named* character vector or XStringSet whose elements are sequences to be analyzed.
scores	specification of scores associated with sequences in fasta argument; can be provided as two column tab-delimited file (should have header, first column should provide sequence names identical to those in fasta argument, second column should have numeric scores) or may be a named numeric vector.
halfWindow	half-width of smoothing window (integer).
spatialLength	full length of spatial smoothing window (integer); use 0 to disable spatial smoothing.
nThreads	number of threads to use for computing permutation distributions.

Value

R representation of java Sarks object.

References

Wylie, D.C., Hofmann, H.A., and Zemelman, B.V. (2019) SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing, Bioinformatics, Vol. 35(20), 3944-3952

https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
```

`sarksFilters` *Smoothing window and Gini impurity filter settings*

Description

Sarks methodology involves testing a range of different filter parameter values; sarksFilters builds set of filters with all combinations of desired halfWindow, spatialLength, and minGini values.

Usage

```
sarksFilters(halfWindow, spatialLength, minGini = 1.1)
```

Arguments

`halfWindow`	integer vector of halfWindow values to test.
`spatialLength`	integer vector of spatialLength values to test; use a single 0 value to disable spatial smoothing.
`minGini`	numeric vector giving minimum Gini impurity value(s) for suffix position to be analyzed; use a value above 1 to calculate minimum Gini impurity based on median of observed Gini impurities so as to constrain variance under permutation testing to less than minGini multiples of median variance.

Value

R representation of java object containing specified combinations of filter parameters for running permutation tests.

References

Wylie, D.C., Hofmann, H.A., and Zemelman, B.V. (2019) SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing, Bioinformatics, Vol. 35(20), 3944-3952

https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
filters <- sarksFilters(
        halfWindow=c(4, 8), spatialLength=c(0, 5), minGini=1.1)
```

simulatedScores *Scores associated with simulated sequences from SArKS paper.*

Description

Scores associated with simulated DNA sequences used to illustrate suffix array kernel smoothing method in https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797 (first 20 sequences assigned score of 0.0, last 10 assigned score of 1.0).

Usage

```
simulatedScores
```

Format

Named numeric vector.

Source

https://github.com/denniscwylie/sarks/tree/master/examples/simulated_scores.tsv

simulatedSeqs *Simulated sequences from SArKS paper.*

Description

Simulated DNA sequences used to illustrate suffix array kernel smoothing method in https://academic.oup.com/bioinformatics/article-abstract/35/20/3944/5418797.

Usage

```
simulatedSeqs
```

Format

Named character vector.

Source

https://github.com/denniscwylie/sarks/tree/master/examples/simulated_seqs.fa

`sourceBlock` *Identify associated input sequence for given position(s) in suffix array*

Description

Any positiion in a suffix array for SArKS concatenated sequence can be identified either via its position i in lexicographically sorted list of suffixes or by its spatial position s in the concatenated sequence. This function indicates which input sequence contributed the block of the concatenated sequence within which the specified position(s) can be found.

Usage

```
sourceBlock(sarks, s = NULL, i = NULL)
```

Arguments

`sarks`	Sarks object from which information will be derived
`s`	the spatial position(s) to query; use NULL (default value) if you instead want to specify sorted suffix position i
`i`	the position(s) in the sorted suffix list to query

Value

character vector containing name(s) of corresponding input sequence(s)

Examples

```
data(simulatedSeqs, simulatedScores)
sarks <- Sarks(simulatedSeqs, simulatedScores, 4, 0, 1)
blocks <- sarks$sourceBlock(i=2253:2261)
```

Index