Package ‘hiReadsProcessor’

March 28, 2017

Title Functions to process LM-PCR reads from 454/Illumina data
Version 1.10.0
Date 2016-05-05
Author Nirav V Malani <malnirav@gmail.com>
Maintainer Nirav V Malani <malnirav@gmail.com>
Description hiReadsProcessor contains set of functions which allow
users to process LM-PCR products sequenced using any platform.
Given an excel/txt file containing parameters for
demultiplexing and sample metadata, the functions automate
trimming of adaptors and identification of the genomic product.
Genomic products are further processed for QC and abundance
quantification.
Depends Biostrings, GenomicAlignments, BiocParallel, hiAnnotator, R
(>= 3.0)
Imports sonicLength, dplyr, BiocGenerics, GenomicRanges, rSFFreader,
readxl, methods
License GPL-3
VignetteBuilder knitr
Suggests knitr, testthat
biocViews Sequencing, Preprocessing
LazyLoad yes
SystemRequirements BLAT, UCSC hg18 in 2bit format for BLAT
RoxygenNote 5.0.1
NeedsCompilation no

R topics documented:

  addFeature ......................................................... 3
  addListNameToReads ............................................ 4
  annotateSites ..................................................... 4
  blatListedSet .................................................... 5
  blatSeqs .......................................................... 6
  chunkize .......................................................... 7
  clusterSites ....................................................... 8
  crossOverCheck .................................................. 10
R topics documented:

dereplicateReads ......................................................... 11
doRCtest ............................................................... 11
extractFeature .......................................................... 12
extractSeqs .............................................................. 13
findAndRemoveVector .................................................... 14
findAndTrimSeq .......................................................... 15
findBarcodes ............................................................. 16
findIntegrations .......................................................... 18
findLinkers ............................................................... 19
findLTRs ................................................................. 21
findPrimers .............................................................. 22
findVector ............................................................... 23
getIntegrationSites ....................................................... 24
getSectorsForSamples .................................................. 25
getSonicAbund ........................................................... 26
hiReadsProcessor ....................................................... 27
isuSites ................................................................. 28
otuSites ................................................................. 29
pairUpAlignments ....................................................... 30
pairwiseAlignSeqs ....................................................... 31
primerIDAignSeqs ....................................................... 33
psl ......................................................................... 34
pslCols ................................................................. 35
pslToRangedObject ..................................................... 35
read.BAMasPSL .......................................................... 36
read.blast8 .............................................................. 37
read.psl ............................................................... 38
read.sampleInfo ........................................................ 39
read.SeqFolder ........................................................ 41
read.seqsFromSector ................................................... 42
removeReadsWithNs ..................................................... 42
replicateReads .......................................................... 43
sampleSummary .......................................................... 44
seqProps ............................................................... 44
splitByBarcode .......................................................... 45
splitSeqsToFleles ....................................................... 46
startgfServer ........................................................... 46
trimSeqs ............................................................... 47
troubleshootLinkers .................................................... 48
vpairwiseAlignSeqs .................................................... 49
write.listedDNAStringSet .......................................... 50
write.psl .............................................................. 52

Index ................................................................. 53
addFeature

Add a specific feature/attribute to the sampleInfo object.

Description

Given a sampleInfo object, the function adds a new feature for the given samples & sectors.

Usage

addFeature(sampleInfo, sector = NULL, samplename = NULL, feature = NULL, value = NULL)

Arguments

sampleInfo sample information SimpleList object, which samples per sector/quadrant information along with other metadata.
sector a vector or a specific sector to add the new feature(s) to. Default is NULL, in which case the sectors are searched from samplename parameter.
samplename a character vector or a specific sample to add the new feature(s) to. Default is NULL.
feature a string of naming the new feature to add for the defined samplename and sector.
value named vector of samplenames & values which is assigned for the defined sector, samplename, and feature. Example: c("Sample1"="ACDTDASD")

Value

modified sampleInfo object with new feature(s) added.

See Also

findPrimers, extractSeqs, trimSeqs, extractFeature, getSectorsForSamples

Examples

load(file.path(system.file("data", package = "hiReadsProcessor"), "FLX_seqProps.RData"))
extractFeature(seqProps, sector="2", samplename="Roth-MLV3p-CD4TMLVWell16-MseI", feature="metadata")
seqProps <- addFeature(seqProps, sector="2", samplename="Roth-MLV3p-CD4TMLVWell16-MseI", feature="foo", value=c("Roth-MLV3p-CD4TMLVWell16-MseI"="woo"))
extractFeature(seqProps, sector="2", samplename="Roth-MLV3p-CD4TMLVWell16-MseI", feature="metadata")
**addListNameToReads**

*Prepend name attribute of a list to DNAStringSet*

**Description**

Given a named listed DNAStringSet object returned from `extractSeqs`, the function prepends the sample name to read names.

**Usage**

```
addListNameToReads(dnaSet, flatten = FALSE)
```

**Arguments**

- `dnaSet`: output from `extractSeqs`
- `flatten`: should the output be unlisted? Default is FALSE.

**Value**

listed DNAStringSet with the names attribute prepended with the name of the list. If flatten is TRUE, then a DNAStringSet object

**See Also**

`extractFeature`, `extractSeqs`, `getSectorsForSamples`, `write.listedDNAStringSet`

**Examples**

```r
load(file.path(system.file("data", package = "hiReadsProcessor"), "FLX_seqProps.RData"))
samples <- c('Roth-MLV3p-CD4TMLVwell6-Tsp509I', 'Roth-MLV3p-CD4TMLVwell6-HseI', 'Roth-MLV3p-CD4TMLVwell5-MuA')
seqs <- extractSeqs(seqProps, sector = '2', samplename = samples, feature="genomic")
addListNameToReads(seqs, TRUE)
```

**annotateSites**

*Find the 5’ primers and add results to SampleInfo object.*

**Description**

Given a sampleInfo object, the function finds 5’ primers for each sample per sector and adds the results back to the object. This is a specialized function which depends on many other functions shown in 'see also section' to perform specialized trimming of 5’ primer/adaptor found in the sampleInfo object. The sequence itself is never trimmed but rather coordinates of primer portion is recorded back to the object and used subsequently by `extractSeqs` function to perform the trimming.
Usage

```r
annotateSites(sampleInfo, annots = NULL, samplenames = NULL, parallel = TRUE, ...)
```

Arguments

- `sampleInfo`: sample information SimpleList object outputted from `findIntegrations`, which holds genomic integration sites.
- `annots`: a named list of GRanges object holding features for annotating integration sites. The name attribute of the list is used as column name.
- `samplenames`: a vector of samplenames to process. Default is NULL, which processes all samples from sampleInfo object.
- `parallel`: use parallel backend to perform calculation. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using `SerialParam`. Parallelization is done at sample level per sector. Use parallel2 for parallelization at sequence level.
- `...`: additional parameters to be passed to `doAnnotation` except for sites.rd, features.rd, and colnam.

Value

a SimpleList object similar to sampleInfo parameter supplied with new data added under each sector and sample. New data attributes include: primed

See Also

`clusterSites`, `isuSites`, `crossOverCheck`, `findIntegrations`, `getIntegrationSites`, `pslToRangedObject`

Examples

```r
load(file.path(system.file("data", package = "hiReadsProcessor"), "FLX_seqProps.RData"))
data(genes)
genes <- makeGRanges(genes)
cpgs <- getUCSTable("cpgIslandExt","CpG Islands")
cpgs <- makeGRanges(cbind(cpgs,strand="*"), chromCol = "chrom")
annotes <- list("RefGenes"=genes,"CpG"=cpgs)
annotateSites(seqProps, annots, annotType="nearest", side="5p")
```

---

**bлатListedSet**

Align a listed DNAStringSet to a reference using gfClient or standalone BLAT.

### Description

Align sequences from a listed DNAStringSet object returned from `extractSeqs` to an indexed reference genome using gfServer/gfClient protocol or using standalone BLAT and return the psl file as a GRanges object. This function heavily relies on defaults of `blatSeqs`.
Usage

blatListedSet(dnaSetList = NULL, ...)

Arguments

dnaSetList DNAStringSet object containing sequences to be aligned against the reference.

... parameters to be passed to blatSeqs.

Value

a list of GRanges object reflecting psl file type per set of sequences.

See Also

pairwiseAlignSeqs, vpairwiseAlignSeqs, startgfServer, stopgfServer, blatSeqs, read.psl, pslToRangedObject, read.blast8

---

**blatSeqs**

*Align sequences using BLAT.*

**Description**

Align batch of sequences using standalone BLAT or gfServer/gfClient protocol against an indexed reference genome. Depending on parameters provided, the function either aligns batch of files to a reference genome using gfClient or takes sequences from query & subject parameters and aligns them using standalone BLAT. If standaloneBlat=FALSE and gfServer is not launched apriori, this function will start one using startgfServer and kill it using stopgfServer upon successful execution.

**Usage**

blatSeqs(query = NULL, subject = NULL, standaloneBlat = TRUE, port = 5560, host = "localhost", parallel = TRUE, numServers = 1L, gzipResults = TRUE, blatParameters = c(minIdentity = 90, minScore = 10, stepSize = 5, tileSize = 10, repMatch = 112312, dots = 50, maxDnaHits = 10, q = "dna", t = "dna", out = "psl"))

**Arguments**

query an object of DNAStringSet, a character vector of filename(s), or a path/ pattern of fasta files to BLAT. Default is NULL.

subject an object of DNAStringSet, a character vector, or a path to an indexed genome (nibs,2bits) to serve as a reference or target to the query. Default is NULL. If the subject is a path to a nib or 2bit file, then standaloneBlat will not work!

standaloneBlat use standalone BLAT as suppose to gfServer/gfClient protocol. Default is TRUE.

port the same number you started the gfServer with. Required if standaloneBlat=FALSE. Default is 5560.

host name of the machine running gfServer. Default is ‘localhost’ and only used when standaloneBlat=FALSE.
**chunkize**  

Breaks an object into chunks of N size.

**Description**

Given a linear/vector like object, the function breaks it into equal sized chunks either by chunkSize. This is a helper function used by functions in 'See Also' section where each chunk is sent to a parallel node for processing.

**Usage**

```r
chunkize(x, chunkSize = NULL)
```
clusterSites

Arguments

x a linear object.
chunkSize number of rows to use per chunk of query. Defaults to length(x)/detectCores() or length(query)/bpworkers() depending on parallel backend registered.

Value

a list of object split into chunks.

See Also

primerIDAlignSeqs, vpairwiseAlignSeqs, pairwiseAlignSeqs

Examples

x <- c("GAGGCTGTCACCGACAAGGTTCTGA", "AATAGCGTGGTGACAGCCCACATGC", "GGTCTTCTAGGGAACCTACGCCACA", "TTTCCGGCGGCAGTCAGAGCCAAAG", "TCTGTCAACTCTGTAGATCCAATCA", "ATGGTACCTACACACACGCCAAT", "GTCAAGGACACCTAATCAACAACCGGA", "AGACCGAGTCTACAGCTGACTAGA", "ATCGTNTCCGCAATCTGCTCAGGTG")
chunkize(DNAStringSet(x), 5)

clusterSites

Cluster/Correct values within a window based on their frequency given discrete factors

Description

Given a group of discrete factors (i.e. position ids) and integer values, the function tries to correct/cluster the integer values based on their frequency in a defined windowsize.

Usage

clusterSites(posID = NULL, value = NULL, grouping = NULL, psl.rd = NULL, weight = NULL, windowSize = 5L, byQuartile = FALSE, quartile = 0.7, parallel = TRUE, sonicAbund = FALSE)

Arguments

posID a vector of groupings for the value parameter (i.e. Chr,strand). Required if psl.rd parameter is not defined.
value a vector of integer with values that needs to corrected or clustered (i.e. Positions). Required if psl.rd parameter is not defined.
grouping additional vector of grouping of length posID or psl.rd by which to pool the rows (i.e. samplenames). Default is NULL.
psl.rd a GRanges object returned from getIntegrationSites. Default is NULL.
weight a numeric vector of weights to use when calculating frequency of value by posID and grouping if specified. Default is NULL.
windowSize size of window within which values should be corrected or clustered. Default is 5.
clusterSites

byQuartile flag denoting whether quartile based technique should be employed. See notes for details. Default is TRUE.

quartile if byQuartile=TRUE, then the quartile which serves as the threshold. Default is 0.70.

parallel use parallel backend to perform calculation with BiocParallel. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam. Process is split by the grouping the column.

sonicAbund calculate breakpoint abundance using getSonicAbund. Default is FALSE.

Value

da data frame with clusteredValues and frequency shown alongside with the original input. If psl.rd parameter is defined then a GRanges object is returned with three new columns appended at the end: clusteredPosition, clonecount, and clusterTopHit (a representative for a given cluster chosen by best scoring hit!).

Note

The algorithm for clustering when byQuartile=TRUE is as follows: for all values in each grouping, get a distribution and test if their frequency is >= quartile threshold. For values below the quartile threshold, test if any values overlap with the ones that passed the threshold and is within the defined windowSize. If there is a match, then merge with higher value, else leave it as is. This is only useful if the distribution is wide and polynodal. When byQuartile=FALSE, for each group the values within the defined window are merged with the next highest frequently occuring value, if frequencies are tied then lowest value is used to represent the cluster. When psl.rd is passed, then multihits are ignored and only unique sites are clustered. All multihits will be tagged as a good 'clusterTopHit'.

See Also

findIntegrations, getIntegrationSites, otuSites, isuSites, crossOverCheck, pslToRangedObject, getSonicAbund

Examples

clusterSites(posID = c("chr1-", "chr1-", "chr1-", "chr2+", "chr15-", "chr16-","chr1-"), value = c(rep(1000, 2), 5832, 1000, 12324, 65738, 928042), grouping = c("a", "a", "a", "b", "b", "b", "c"), parallel = FALSE)
## Not run:
data(psl)
psl <- psl[sample(nrow(psl), 100), ]
psl.rd <- getIntegrationSites(pslToRangedObject(psl))
psl.rd$grouping <- sub("(.+)-.+", "\1", psl.rd$qName)
clusterSites(grouping = psl.rd$grouping, psl.rd = psl.rd)
## End(Not run)
crossOverCheck

Remove values/positions which are overlapping between discrete groups based on their frequency.

Description

Given a group of discrete factors (i.e. position ids) and integer values, the function tests if they overlap between groups. If overlap is found, then the group having highest frequency of a given position wins, else the position is filtered out from all the groups. The main use of this function is to remove crossover sites from different samples in the data.

Usage

crossOverCheck(posID = NULL, value = NULL, grouping = NULL, weight = NULL, windowSize = 1, psl.rd = NULL)

Arguments

posID  a vector of groupings for the value parameter (i.e. Chr,strand). Required if psl.rd parameter is not defined.
value  a vector of integer locations/positions that needs to be binned, i.e. genomic location. Required if psl.rd parameter is not defined.
grouping  additional vector of grouping of length posID or psl.rd by which to pool the rows (i.e. samplenames). Default is NULL.
weight  a numeric vector of weights to use when calculating frequency of value by posID and grouping if specified. Default is NULL.
windowSize  size of window within which values should be checked. Default is 1.
psl.rd  a GRanges object. Default is NULL.

Value

a data frame of the original input with columns denoting whether a given row was a Candidate and isCrossover. If psl.rd parameter is defined, then a GRanges object with 'isCrossover', 'Candidate', and 'FoundIn' columns appended at the end.

See Also

clusterSites, otuSites, findIntegrations, getIntegrationSites, pslToRangedObject

Examples

crossOverCheck(posID = c('chr1-', 'chr1-', 'chr1-', 'chr1-', 'chr2+', 'chr15-', 'chr16-', 'chr11-'), value = c(rep(1000, 3), 5832, 1000, 12324, 65738, 928042), grouping = c('a', 'a', 'b', 'b', 'b', 'c', 'c'))
**dereglicateReads**

Removes duplicate sequences from DNAStringSet object.

**Description**

Given a DNAStringSet object, the function dereglicates reads and adds counts=X to the definition line to indicate replication.

**Usage**

dereglicateReads(dnaSet)

**Arguments**

dnaSet DNAStringSet object to dereglicate.

**Value**

DNAStringSet object with names describing frequency of repeat.

**See Also**

replicateReads, removeReadsWithNs, findBarcodes, splitByBarcode

**Examples**

```r
dereglicateReads(dnaSet)
```

**doRCtest**

Test if pattern aligns better in +/- orientation.

**Description**

Given a fixed length pattern sequence and variable length subject sequences, the function roughly finds which orientation of pattern yields the most hits. The function doing the heavy lifting is vcountPattern. This is an accessory function used in function listed under See Also section below.

**Usage**

doRCtest(subjectSeqs = NULL, patternSeq = NULL, qualityThreshold = 0.5, parallel = TRUE)
Arguments

subjectSeqs  DNAStringSet object containing sequences to be searched for the pattern.
patternSeq   DNAString object or a sequence containing the query sequence to search.
qualityThreshold percent of patternSeq to match. Default is 0.50, half match. This is supplied to max.mismatch parameter of vcountPattern as round(nchar(patternSeq)*(1-qualityThreshold)).
parallel     use parallel backend to perform calculation with BiocParallel. Defaults to FALSE. If no parallel backend is registered, then a serial version is ran using SerialParam.

Value

patternSeq that aligned the best.

See Also

pairwiseAlignSeqs, vpairwiseAlignSeqs, primerIDAlignSeqs

Examples

subjectSeqs <- c("CCTGAATCTGGCAATGTCATCATC", "ATCCTGGCAATGTCATCATC", "ATCAGTGGTAACGGCTAATACGCG", "ATCAATGGCGATTGCCGCGTCTGCA", "CCGCGTCTGCAATGTGAGGGCCTAA", "GAAGGATGCCAGTTGAAGTTCACAC")
subjectSeqs <- xscat("AAAAAAAAAA", subjectSeqs)
doRCtest(subjectSeqs, "TTTTTTTTT")

extractFeature

Extract a specific feature/attribute of the sampleInfo object.

Description

Given a sampleInfo object, the function extracts a defined feature(s) for given sample or sector.

Usage

extractFeature(sampleInfo, sector = NULL, samplename = NULL, feature = NULL)

Arguments

sampleInfo  sample information SimpleList object, which samples per sector/quadrant information along with other metadata.
sector      a vector or specific sector to extract the feature from. Default is NULL, which extracts all sectors.
samplename  a character vector or a specific sample to extract feature from. Default is NULL, which extracts all samples.
feature     Options include: primed, LTRed, linkeder, decoded, and any of the metadata. Default is NULL. When feature='metadata', then it returns names of all the metadata elements associated with the sample as a comma separated list.
extractSeqs

Value

a list or list of lists depending upon which parameters were supplied.

See Also

addFeature, findPrimers, findLTRs, findLinkers, extractSeqs, trimSeqs, getSectorsForSamples

Examples

load(file.path(system.file("data", package = "hiReadsProcessor"),
"FLX_seqProps.RData"))
samples <- c("Roth-MLV3p-CD4TMLVWell16-Tsp509I",
'Roth-MLV3p-CD4TMLVWell16-MseI', 'Roth-MLV3p-CD4TMLVWell15-MluA')
extactFeature(seqProps, sector=2, samplename=samples, feature="primed")
extactFeature(seqProps, sector=2, samplename=samples, feature="linkered")
extactFeature(seqProps, sector=2, samplename=samples, feature="metadata")

desc:

given a sampleinfo object, the function extracts sequences for a defined feature.

Usage

extractSeqs(sampleInfo, sector = NULL, samplename = NULL,
feature = "genomic", trim = TRUE, minReadLength = 1,
sidReturn = NULL, pairReturn = "both", strict = FALSE)

Arguments

sampleInfo sample information SimpleList object, which samples per sector/quadrant information along with other metadata.
sector specific sector to extract sequences from. Default is NULL, which extracts all sectors.
samplename specific sample to extract sequences from. Default is NULL, which extracts all samples.
feature which part of sequence to extract (case sensitive). Options include: primed, !primed, LTRed, !LTRed, linkered, !linkered, primerIDs, genomic, genomicLinkered, decoded, and unDecoded. If a sample was primerIDed and processed by primerIDAlignSeqs, then all the rejected and unmatched attributes can be prepended to the feature. Example: vectored, Rejectedlinkered, Reject-edprimerIDslinkered, Absentlinkered, or unAnchoredprimerIDslinkered. When feature is genomic, it includes sequences which are primed, LTRed, linkered, and !linkered. The genomicLinkered is same as genomic minus the !linkered. When feature is decoded, it includes everything that demultiplexed. The ‘!’ in front of a feature extracts the inverse. One can only get unDecoded sequences if returnUnmatched was TRUE in findBarcodes. If findVector was run and "vectored" feature was found in the sampleInfo object, then genomic & genomicLinkered output will have vectored reads removed.
findAndRemoveVector

Find and trim vector sequence from reads.

**Description**

This function facilitates finding and trimming of long/short fragments of vector present in LM-PCR products. The algorithm looks for vector sequence present anywhere within the read and trims according longest contiguous match on either end of the read. Alignment is doing using BLAT.

**Usage**

```r
findAndRemoveVector(reads, Vector, minLength = 10, returnCoords = FALSE, parallel = TRUE)
```
findAndTrimSeq

Arguments

reads DNAStringSet object containing sequences to be trimmed for vector.
Vector DNAString object containing vector sequence to be searched in reads.
minLength integer value dictating minimum length of trimmed sequences to return. Default is 10.
returnCoords return the coordinates of vector start-stop for the matching reads. Defaults to FALSE.
parallel use parallel backend to perform calculation with BiocParallel. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam.

Value

DNAStringSet object with Vector sequence removed from the reads. If returnCoords=TRUE, then a list of two named elements "hits" & "reads". The first element, "hits" is a GRanges object with properties of matched region and whether it was considered valid denoted by 'good.row'. The second element, "reads" is a DNAStringSet object with Vector sequence removed from the reads.

Note

If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following MulticoreParam SnowParam

See Also

pairwiseAlignSeqs, vpairwiseAlignSeqs, pslToRangedObject, blatSeqs, read.blast8, findAndTrimSeq

findAndTrimSeq

Find and trim a short pattern sequence from the subject.

Description

This function facilitates finding and trimming of a short pattern sequence from a collection of subject sequences. The trimming is dictated by side parameter. For more information on the trimming process see the 'side' parameter documentation in trimSeqs. For information regarding the pattern alignment see the documentation for pairwiseAlignSeqs. This function is meant for aligning a short pattern onto large collection of subjects. If you are looking to align a vector sequence to subjects, then please use BLAT.

Usage

findAndTrimSeq(patternSeq, subjectSeqs, side = "left", offBy = 0, alignWay = "slow", ...)
findBarcodes

Arguments

- **patternSeq**
  - DNAString object or a sequence containing the query sequence to search.

- **subjectSeqs**
  - DNAStringSet object containing sequences to be searched for the pattern.

- **side**
  - which side of the sequence to perform the search & trimming: left, right or middle. Default is 'left'.

- **offBy**
  - integer value dictating if the trimming base should be offset by X number of bases. Default is 0.

- **alignWay**
  - method to utilize for detecting the primers. One of following: "slow" (Default), "fast", or "blat". Fast, calls `vpairwiseAlignSeqs` and uses `vmatchPattern` at its core, which is less accurate with indels and mismatches but much faster. Slow, calls `pairwiseAlignSeqs` and uses `pairwiseAlignment` at its core, which is accurate with indels and mismatches but slower. Blat will use `blatSeqs`.

  - ... parameters to be passed to `pairwiseAlignment`, `vpairwiseAlignSeqs` or `blatSeqs` depending on which method is defined in 'alignWay' parameter.

Value

DNAStringSet object with pattern sequence removed from the subject sequences.

Note

If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following `MulticoreParam SnowParam`

See Also

- `pairwiseAlignSeqs`, `vpairwiseAlignSeqs`, `extractFeature`, `extractSeqs`, `primerIDAlignSeqs`, `findPrimers`, `findLinkers`

Examples

```r
findAndTrimSeq(patternSeq="AGACCCTTTT", 
subjectSeqs=DNAStringSet(c("AGACCCTTTTGCAGCAT", "AGACCCTTGCTGACTCA", 
"AGACCCTTTTGAGGAGCTAG")), qualityThreshold=.85, doRC=FALSE, side="left", 
offBy=1, alignWay = "slow")
```

findBarcodes  
**Demultiplex reads by their barcodes**

Description

Given a sample information object, the function reads in the raw fasta/fastq file, demultiplexes reads by their barcodes, and appends it back to the sampleInfo object. Calls `splitByBarcode` to perform the actual splitting of file by barcode sequences. If supplied with a character vector and reads themselves, the function behaves a bit differently. See the examples.

Usage

```r
findBarcodes(sampleInfo, sector = NULL, dnaSet = NULL, showStats = FALSE, 
returnUnmatched = FALSE, dereplicate = FALSE, alreadyDecoded = FALSE)
```
findBarcodes

Arguments

sampleInfo  sample information SimpleList object created using `read.sampleInfo`, which holds barcodes and sample names per sector/quadrant/lane or a character vector of barcodes to sample name associations. Ex: `c("ACATCCAT"="Sample1", "GAATGGAT"="Sample2",...)`

sector  If sampleInfo is a SimpleList object, then a numeric/character value or vector representing sector(s) in sampleInfo. Optionally if on high memory machine sector='all' will decode/demultiplex sequences from all sectors/quadrants. This option is ignored if sampleInfo is a character vector. Default is NULL.

dnaSet  DNAStringSet object containing sequences to be decoded or demultiplexed. Default is NULL. If sampleInfo is a SimpleList object, then reads are automatically extracted using `read.seqsFromSector` and parameters defined in sampleInfo object.

showStats  toggle output of search statistics. Default is FALSE.

returnUnmatched  return unmatched sequences. Returns results as a list where x["unDecodedSeqs"] has culprits. Default is FALSE.

dereplicate  return dereplicated sequences. Calls `dereplicateReads`, which appends counts=X to sequence names/deflines. Default is FALSE. Not applicable for paired end data since it can cause insyncronicity.

alreadyDecoded  if reads have be already decoded and split into respective files per sample and 'seqfilePattern' parameter in `read.SeqFolder` is set to reading sample files and not the sector files, then set this to TRUE. Default is FALSE. Enabling this parameter skips the barcode detection step and loads the sequence file as is into the sampleInfo object.

Value

If sampleInfo is an object of SimpleList then decoded sequences are appended to respective sample slots, else a named list of DNAStringSet object. If returnUnmatched=TRUE, then x["unDecodedSeqs"] has the unmatched sequences.

See Also

`splitByBarcode`, `dereplicateReads`, `replicateReads`

Examples

dnaSet <- DNAStringSet(c("read1" = "ACATCCATAGCTACGACGACATCGACATA",
"read2"="GAATGGATGCGCTAAGAAGAGA", "read4"="ACATCCATTCTACACATCT")
findBarcodes(sampleInfo = c("ACATCCAT" = "Sample1", "GAATGGAT" = "Sample2"),
dnaSet=dnaSet, showStats=TRUE, returnUnmatched=TRUE)
## Not run:
load(file.path(system.file("data", package = "hiReadsProcessor"),
"FLX_seqProps.RData"))
findBarcodes(seqProps, sector = "all", showStats = TRUE)
## End(Not run)
findIntegrations

*Find the integration sites and add results to SampleInfo object.*

**Description**

Given a SampleInfo object, the function finds integration sites for each sample using their respective settings and adds the results back to the object. This is an all-in-one function which aligns, finds best hit per read per sample, cluster sites, and assign ISU IDs. It calls `blatSeqs`, `read.psl`, `getIntegrationSites`, `clusterSites`, `otuSites`. Here must be linker reads within the sampleInfo object in order to use this function using the default parameters. If you are planning on BLATing non-linkered reads, then change the seqType to one of accepted options for the 'feature' parameter of `extractSeqs`, except for ‘!’ based features.

**Usage**

```r
findIntegrations(sampleInfo, seqType = NULL, genomeIndices = NULL, samplenames = NULL, parallel = TRUE, autoOptimize = FALSE, doSonic = FALSE, doISU = FALSE, ...)
```

**Arguments**

- `sampleInfo` sample information SimpleList object outputted from `findLinkers`, which holds decoded, primed, LTRed, and Linkered sequences for samples per sector/quadrant along with metadata.
- `seqType` which type of sequence to align and find integration sites. Default is NULL and determined automatically based on type of restriction enzyme or isolation method used. If restriction enzyme is Fragmentase, MuA, Sonication, or Sheared then this parameter is set to genomicLinkered, else it is genomic. Any one of following options are valid: genomic, genomicLinkered, decoded, primed, LTRed, linker.
- `genomeIndices` an associative character vector of freeze to full or relative path of respective of indexed genomes from BLAT(.nib or .2bit files). For example: `c("hg18"="/usr/local/blatSuite34/hg18.2bit", "mm8"="/usr/local/blatSuite34/mm8.2bit")`. Be sure to supply an index per freeze supplied in the sampleInfo object. Default is NULL.
- `samplenames` a vector of samplenames to process. Default is NULL, which processes all samples from sampleInfo object.
- `parallel` use parallel backend to perform calculation with `BiocParallel`. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using `SerialParam`.
- `autoOptimize` if aligner='BLAT', then should the blatParameters be automatically optimized based on the reads? Default is FALSE. When TRUE, following parameters are adjusted within the supplied blatParameters vector: stepSize, tileSize, minScore, minIdentity. This parameter is useful when aligning reads of various lengths to the genome. Optimization is done using only read lengths. In beta phase!
- `doSonic` calculate integration sites abundance using breakpoints. See `getSonicAbund` for more details. Default is FALSE.
- `doISU` calculate integration site unit for multihits. See `isuSites` for more details. Default is FALSE.
- `...` additional parameters to be passed to `blatSeqs`.
Value

A SimpleList object similar to sampleInfo parameter supplied with new data added under each sector and sample. New data attributes include: psl, and sites. The psl attributes holds the genomic hits per read along with QC information. The sites attribute holds the condensed integration sites where genomic hits have been clustered by the Position column and cherry picked to have each site pass all the QC steps.

Note

If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following MulticoreParam SnowParam

See Also

findPrimers, findLTRs, findLinkers, startgfServer, read.psl, blatSeqs, blatListedSet, pslToRangedObject, clusterSites, isuSites, crossOverCheck, getIntegrationSites, getSonicAbund, annotateSites

Examples

load(file.path(system.file("data", package = "hiReadsProcessor"),
"FLX_seqProps.RData"))
findIntegrations(seqProps,
genomeIndices=c("hg18"="/usr/local/genomeIndexes/hg18.noRandom.2bit"),
numServers=2)

findLinkers

Find the 3’ linkers and add results to SampleInfo object.

Description

Given a sampleInfo object, the function finds 3’ linkers for each sample per sector and adds the results back to the object. This is a specialized function which depends on many other functions shown in 'see also section' to perform specialized trimming of 3’ primer/linker adaptor sequence found in the sampleInfo object. The sequence itself is never trimmed but rather coordinates of linker portion is recorded back to the object and used subsequently by extractSeqs function to perform the trimming. This function heavily relies on either pairwiseAlignSeqs or primerIDAlignSeqs depending upon whether linkers getting aligned have primerID in it or not.

Usage

findLinkers(sampleInfo, showStats = FALSE, doRC = FALSE, parallel = TRUE, samplenames = NULL, bypassChecks = FALSE, parallel2 = FALSE, ...)
findLinkers

Arguments

sampleInfo sample information SimpleList object outputted from findPrimers or findLTRs, which holds decoded sequences for samples per sector/quadrant along with information of sample to primer associations.

showStats toggle output of search statistics. Default is FALSE.

doRC perform reverse complement search of the defined pattern/linker sequence. Default is FALSE.

parallel use parallel backend to perform calculation with BiocParallel. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam. Parallelization is done at sample level per sector.

samplenames a vector of samplenames to process. Default is NULL, which processes all samples from sampleInfo object.

bypassChecks skip checkpoints which detect if something was odd with the data? Default is FALSE.

parallel2 perform parallelization is sequence level. Default is FALSE. Useful in cases where each sector has only one sample with numerous sequences.

... extra parameters to be passed to pairwiseAlignment.

Value

a SimpleList object similar to sampleInfo parameter supplied with new data added under each sector and sample. New data attributes include: linkered. If linkers have primerID then, primerIDs attribute is appended as well.

Note

• For paired end data, qualityThreshold for pair 2 is increased by 0.25 or set to 1 whichever is lower to increase quality & full match to linker sequence.

• If no linker matches are found with default options, then try doRC=TRUE.

• If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following MulticoreParam SnowParam

See Also

pairwiseAlignSeqs, vpairwiseAlignSeqs, primerIDAlignSeqs, findLTRs, findPrimers, extractFeature, extractSeqs, FindAndTrimSeq, findIntegrations

Examples

load(file.path(system.file("data", package = "hiReadsProcessor"), "FLX_seqProps.RData"))
findLinkers(seqProps, showStats=TRUE, doRC=TRUE)
findLTRs

Find the 5' LTRs and add results to SampleInfo object.

Description

Given a sampleInfo object, the function finds 5' LTR following the primer for each sample per sector and adds the results back to the object. This is a specialized function which depends on many other functions shown in 'see also section' to perform specialized trimming of 5' viral LTRs found in the sampleInfo object. The sequence itself is never trimmed but rather coordinates of LTR portion is added to primer coordinates and recorded back to the object and used subsequently by extractSeqs function to perform the trimming. This function heavily relies on pairwiseAlignSeqs.

Usage

findLTRs(sampleInfo, showStats = FALSE, doRC = FALSE, parallel = TRUE, samplenames = NULL, bypassChecks = FALSE, parallel2 = FALSE, ...)

Arguments

sampleInfo sample information SimpleList object outputted from findPrimers, which holds decoded and primed sequences for samples per sector/quadrant along with information of sample to LTR associations.
showStats toggle output of search statistics. Default is FALSE. For paired end data, stats for "pair2" is relative to decoded and/or primed reads.
doRC perform reverse complement search of the defined pattern/LTR sequence. Default is FALSE.
parallel use parallel backend to perform calculation with BiocParallel. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam. Parallelization is done at sample level per sector.
samplenames a vector of samplenames to process. Default is NULL, which processes all samples from sampleInfo object.
bypassChecks skip checkpoints which detect if something was odd with the data? Default is FALSE.
parallel2 perform parallelization is sequence level. Default is FALSE. Useful in cases where each sector has only one sample with numerous sequences.
... extra parameters to be passed to pairwiseAlignment.

Value

a SimpleList object similar to sampleInfo parameter supplied with new data added under each sector and sample. New data attributes include: LTRed

Note

• For paired end data, qualityThreshold for pair 2 is decreased by 0.05 to increase chances of matching LTR sequence.
• If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following MulticoreParam SnowParam
findPrimers

See Also

pairwiseAlignSeqs, vpairwiseAlignSeqs, extractFeature, extractSeqs, primerIDA OPalignSeqs, findPrimers, findLinkers, findAndTrimSeq

Examples

load(file.path(system.file("data", package = "hiReadsProcessor"), "FLX_seqProps.RData"))
findLTRs(seqProps, showStats=TRUE)

findPrimers Find the 5' primers and add results to SampleInfo object.

Description

Given a sampleInfo object, the function finds 5’ primers for each sample per sector and adds the results back to the object. This is a specialized function which depends on many other functions shown in ‘see also section’ to perform specialized trimming of 5’ primer/adaptor found in the sampleInfo object. The sequence itself is never trimmed but rather coordinates of primer portion is recorded back to the object and used subsequently by extractSeqs function to perform the trimming.

Usage

findPrimers(sampleInfo, alignWay = "slow", showStats = FALSE, doRC = FALSE, parallel = TRUE, samplenames = NULL, bypassChecks = FALSE, parallel2 = FALSE, ...)

Arguments

sampleInfo sample information SimpleList object outputted from findBarcodes, which holds decoded sequences for samples per sector/quadrant along with information of sample to primer associations.

alignWay method to utilize for detecting the primers. One of following: "slow" (Default), or "fast". Fast, calls vpairwiseAlignSeqs and uses vmatchPattern at its core, which is less accurate with indels and mismatches but much faster. Slow, calls pairwiseAlignSeqs and uses pairwiseAlignment at its core, which is accurate with indels and mismatches but slower.

showStats toggle output of search statistics. Default is FALSE.

doRC perform reverse complement search of the defined pattern/primer. Default is FALSE.

parallel use parallel backend to perform calculation . Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam. Parallelization is done at sample level per sector. Use parallel2 for parallelization at sequence level.

samplenames a vector of samplenames to process. Default is NULL, which processes all samples from sampleInfo object.
**findVector**

Find vector DNA in reads and add results to SampleInfo object.

### Description

Given a sampleInfo object, the function finds vector fragments following the LTR piece for each sample per sector and adds the results back to the object. This is a specialized function which depends on many other functions shown in ‘see also section’ to perform specialized trimming of 5’ viral LTRs found in the sampleInfo object. The sequence itself is never trimmed but rather coordinates of vector portion is added to LTR coordinates and recorded back to the object and used subsequently by `extractSeqs` function to perform the trimming. This function heavily relies on `blatSeqs`. In order for this function to work, it needs vector sequence which is read in using `vectorFile` metadata supplied in the sample information file in `read.sampleInfo`

### Usage

```r
findVector(sampleInfo, showStats = FALSE, parallel = TRUE, samplenames = NULL)
```

---

**bypassChecks**

skip checkpoints which detect if something was odd with the data? Default is FALSE.

**parallel2**

perform parallelization is sequence level. Default is FALSE. Useful in cases where each sector has only one sample with numerous sequences.

... extra parameters to be passed to either `vmatchPattern` or `pairwiseAlignment` depending on `alignWay` parameter.

### Value

a SimpleList object similar to sampleInfo parameter supplied with new data added under each sector and sample. New data attributes include: primed

### Note

- For paired end data, qualityThreshold for pair 2 is decreased by 0.10 to increase chances of matching primer sequence.
- If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following MulticoreParam SnowParam

### See Also

`pairwiseAlignSeqs`, `vpairwiseAlignSeqs`, `extractFeature`, `extractSeqs`, `primerIDAlignSeqs`, `findLTRs`, `findLinkers`, `findAndTrimSeq`

### Examples

```r
load(file.path(system.file("data", package = "hiReadsProcessor"),
"FLX_seqProps.RData"))
findPrimers(seqProps, showStats=TRUE)
```
getIntegrationSites

Obtain integration sites from BLAT output

Description

Given a GRanges object from `read.psl`, the function uses specified filtering parameters to obtain integration sites and maintain sequence attrition. The function will remove any non-best scoring alignments from the object if not already filtered apriori.

Usage

```r
ggetIntegrationSites(psl.rd = NULL, startWithin = 3,
                   alignRatioThreshold = 0.7, genomicPercentIdentity = 0.98,
                   correctByqStart = TRUE, oneBased = FALSE)
```

Arguments

- **sampleInfo**: sample information SimpleList object outputted from `findLTRs`, which holds decoded, primed, and LTRed sequences for samples per sector/quadrant.
- **showStats**: toggle output of search statistics. Default is FALSE.
- **parallel**: use parallel backend to perform calculation with `BiocParallel`. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using `SerialParam`. Parallelization is done at sample level per sector. Use parallel2 for parallelization at sequence level.
- **samplenames**: a vector of samplenames to process. Default is NULL, which processes all samples from sampleInfo object.

Value

A SimpleList object similar to sampleInfo parameter supplied with new data added under each sector and sample. New data attributes include: vectored

Note

- If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following `MulticoreParam` `SnowParam`

See Also

- `pairwiseAlignSeqs`, `blatSeqs`, `extractFeature`, `extractSeqs`, `findPrimers`, `findLTRs`, `findLinkers`, `findAndTrimSeq`, `findAndRemoveVector`

Examples

```r
load(file.path(system.file("data", package = "hiReadsProcessor"),
"FLX_seqProps.RData"))
findVector(seqProps, showStats=TRUE)
```
getSectorsForSamples

Arguments

psl.rd  a GRanges object reflecting psl format where tName is the seqnames.
startWithin upper bound limit of where the alignment should start within the query. Default is 3.
alignRatioThreshold cutoff for (alignment span/read length). Default is 0.7.
genomicPercentIdentity cutoff for (1-(misMatches/matches)). Default is 0.98.
correctByqStart use qStart to correct genomic position. This would account for sequencing/trimming errors. Position=ifelse(strand=="+",tStart-qStart,tEnd+qStart). Default is TRUE.
oneBased the coordinates in psl files are "zero based half open". The first base in a sequence is numbered zero rather than one. Enabling this would add +1 to the start and leave the end as is. Default is FALSE.

Value

a GRanges object with integration sites which passed all filtering criteria. Each filtering parameter creates a new column to flag if a sequence/read passed that filter which follows the scheme: 'pass.FilterName'. Integration Site is marked by new column named 'Position'.

See Also

startgfServer, read.psl, blatSeqs, blatListedSet, findIntegrations, pslToRangedObject, clusterSites, isuSites, crossOverCheck, read.blast8

Examples

data(psl)
psl.rd <- pslToRangedObject(psl)
getIntegrationSites(psl.rd)

getSectorsForSamples Get sectors for samples defined in the sampleInfo object.

Description

Given a sampleInfo object, the function gets the sectors for each samplename. This is an accessory function utilized by other functions of this package to aid sector retrieval.

Usage

getsectorsForSamples(sampleInfo, sector = NULL, samplename = NULL, returnDf = FALSE)
**Arguments**

- **sampleInfo**  
  sample information SimpleList object, which samples per sector/quadrant information along with other metadata.

- **sector**  
  a specific sector or vector of sectors if known ahead of time. Default is NULL, which extracts all sectors.

- **samplename**  
  a specific sample or vector of samplenames to get sectors for. Default is NULL, which extracts all samples.

- **returnDf**  
  return results in a dataframe. Default is FALSE.

**Value**

If `returnDf=TRUE`, then a dataframe of sector associated with each samplename, else a named list of length two: x["sectors"] and x["samplenames"]

**See Also**

- `extractSeqs`
- `extractFeature`
- `addFeature`

**Examples**

```r
load(file.path(system.file("data", package = "hiReadsProcessor"), "FLX_seqProps.RData"))
samples <- c('Roth-MLV3p-CD4TMLVWell16-Tsp509I', 'Roth-MLV3p-CD4TMLVWell16-MseI', 'Roth-MLV3p-CD4TMLVWell15-MuA')
getSectorsForSamples(seqProps, samplename=samples)
getSectorsForSamples(seqProps, samplename=samples, returnDf=TRUE)
```

---

**getSonicAbund**  
Calculate breakpoint/sonic abundance of integration sites in a population

**Description**

Given distinct fragment lengths per integration, the function calculates sonic abundance as described in `sonicLength`. This function is called by `clusterSites` and needs all individual fragments lengths per position to properly estimate the clonal abundance of an integration sites in a given population.

**Usage**

```r
gsonicAbund(posID = NULL, fragLen = NULL, grouping = NULL, replicateNum = NULL, psl.rd = NULL, parallel = TRUE)
```

**Arguments**

- **posID**  
  a vector of discrete positions, i.e. Chr,strand,Position. Required if psl.rd parameter is not defined.

- **fragLen**  
  a vector of fragment length per posID. Required if psl.rd parameter is not defined.

- **grouping**  
  additional vector of grouping of length posID or psl.rd by which to pool the rows (i.e. samplenames). Default is NULL.
replicateNum  an optional vector of the replicate number per grouping and posID. Default is NULL.
psl.rd  a GRanges object returned from getIntegrationSites Default is NULL.
parallel  use parallel backend to perform calculation with BiocParallel. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam. Process is split by the grouping the column.

Value
a data frame with estimated sonic abundance shown alongside with the original input. If psl.rd parameter is defined then a GRanges object is returned with a new column 'estAbund'.

Note
For samples isolated using traditional restriction digest method, the abundance will be inaccurate as it is designed for sonicated or sheared sample preparation method.

See Also
clusterSites, otuSites, findIntegrations, getIntegrationSites, pslToRangedObject

Examples

data("A1", package='sonicLength')
bore <- with(A1, getSonicAbund(locations, lengths, "A", replicates))
head(bore)

hiReadsProcessor  Functions to process LM-PCR reads from 454/Illumina data

Description
hiReadsProcessor contains set of functions which allow users to process LM-PCR products sequenced using any platform. Given an excel/txt file containing parameters for demultiplexing and sample metadata, the functions automate trimming of adaptors and identification of the genomic product. Genomic products are further processed for QC and abundance quantification.

Author(s)
Nirav V Malani
isuSites

Bin values or make ISUs by assigning a unique ID to them within discrete factors.

Description

Given a group of values or genomic positions per read/clone, the function tries to yield a unique ISU (Integration Site Unit) ID for the collection based on overlap of locations to other reads/clones by grouping. This is mainly useful when each read has many locations which needs to be considered as one single group of sites.

Usage

isuSites(posID = NULL, value = NULL, readID = NULL, grouping = NULL, psl_rd = NULL, maxgap = 5, parallel = TRUE)

Arguments

- posID: a vector of groupings for the value parameter (i.e. Chr,strand). Required if psl.rd parameter is not defined.
- value: a vector of integer locations/positions that needs to be binned, i.e. genomic location. Required if psl.rd parameter is not defined.
- readID: a vector of read/clone names which is unique to each row, i.e. deflines.
- grouping: additional vector of grouping of length posID or psl.rd by which to pool the rows (i.e. samplenames). Default is NULL.
- psl_rd: a GRanges object returned from clusterSites. Default is NULL.
- maxgap: max distance allowed between two non-overlapping position to trigger the merging. Default is 5.
- parallel: use parallel backend to perform calculation with BiocParallel. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam. Process is split by the grouping the column.

Value

a data frame with binned values and isuID shown alongside the original input. If psl.rd parameter is defined, then a GRanges object where object is first filtered by clusterTopHit column and the isuID column appended at the end.

Note

The algorithm for making isus of sites is as follows: for each readID check how many positions are there. Separate readIDs with only position from the rest. Check if any readIDs with >1 position match to any readIDs with only one position. If there is a match, then assign both readIDs with the same ISU ID. Check if any positions from readIDs with >1 position match any other readIDs with >1 position. If yes, then assign same ISU ID to all readIDs sharing 1 or more positions.

See Also

clusterSites, isuSites, crossOverCheck, findIntegrations, getIntegrationSites, pslToRangedObject
**otuSites**

**Examples**

```r
isuSites(posID = c('chr1-', 'chr1-', 'chr1-', 'chr2+', 'chr15-', 'chr16-', 'chr11-'),
  value = c(rep(1000, 2), 5832, 1000, 12324, 65738, 928042),
  readID = paste('read', sample(letters, 7), sep = '-'),
  grouping = c('a', 'a', 'a', 'b', 'b', 'b', 'c'), parallel = FALSE)
```

---

**Description**

Given a group of values or genomic positions per read/clone, the function tries to yield a unique OTU (operation taxonomic unit) ID for the collection based on overlap of locations to other reads/clones by grouping. This is mainly useful when each read has many locations which needs to be considered as one single group of sites.

**Usage**

```r
otuSites(posID = NULL, value = NULL, readID = NULL, grouping = NULL,
  psl.rd = NULL, maxgap = 5, parallel = TRUE)
```

**Arguments**

- `posID`: a vector of groupings for the value parameter (i.e. Chr,strand). Required if psl.rd parameter is not defined.
- `value`: a vector of integer locations/positions that needs to be binned, i.e. genomic location. Required if psl.rd parameter is not defined.
- `readID`: a vector of read/clone names which is unique to each row, i.e. deflines.
- `grouping`: additional vector of grouping of length posID or psl.rd by which to pool the rows (i.e. sample names). Default is NULL.
- `psl.rd`: a GRanges object returned from `clusterSites`. Default is NULL.
- `maxgap`: max distance allowed between two non-overlapping position to trigger the merging. Default is 5.
- `parallel`: use parallel backend to perform calculation with `BiocParallel`. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using `SerialParam`. Process is split by the grouping the column.

**Value**

A data frame with binned values and otuID shown alongside the original input. If psl.rd parameter is defined, then a GRanges object.

**Note**

The algorithm for making OTUs of sites is as follows:

- for each grouping & posID, fix values off by maxgap parameter
- create bins of fixed values per readID
pairUpAlignments

Pair up alignments in a GRanges object

Description

Given a GRanges object, the function uses specified gaplength parameter to pair up reads where the qName column ends with "atpersand pairname atpersand" which is outputted by extractSeqs.

Usage

pairUpAlignments(psl.rd = NULL, maxGapLength = 2500, sameStrand = TRUE, parallel = TRUE)

Arguments

psl.rd a GRanges object with qNames ending in "atpersand pairname atpersand".
maxGapLength maximum gap allowed between end of pair1 and start of pair2. Default is 2500 bp.
sameStrand should pairs be aligned to the same strand or in same orientation in the reference genome? Default is TRUE. This is 'TRUE' because pair2 reads are reverse complemented when reading in data in findBarcodes
parallel use parallel backend to perform calculation with BiocParallel. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam.
Value

a GRanges object with reads paired up denoted by "paired" column. Improper pairs or unpaired reads are returned with "paired" column as FALSE.

See Also

pairwiseAlignSeqs, blatSeqs, read.blast8, read.psl, getIntegrationSites, read.BAMasPSL

Examples

```r
psl.rd <- read.BAMasPSL(bamFile=c("sample1hits.bam","sample2hits.bam"))
pairUpAlignments(psl.rd)
```

pairwiseAlignSeqs

Align a short pattern to variable length target sequences.

Description

Align a fixed length short pattern sequence (i.e. primers or adaptors) to subject sequences using pairwiseAlignment. This function uses default of type="overlap", gapOpenings=-1, and gapExtension=-1 to align the patternSeq against subjectSeqs. One can adjust these parameters if preferred, but not recommended. This function is meant for aligning a short pattern onto large collection of subjects. If you are looking to align a vector sequence to subjects, then please use BLAT or see one of following blatSeqs, findAndRemoveVector

Usage

```r
pairwiseAlignSeqs(subjectSeqs = NULL, patternSeq = NULL, side = "left",
                    qualityThreshold = 1, showStats = FALSE, bufferBases = 5, doRC = TRUE,
                    returnUnmatched = FALSE, returnLowScored = FALSE, parallel = FALSE, ...)
```

Arguments

- **subjectSeqs**: DNAStringSet object containing sequences to be searched for the pattern. This is generally bigger than patternSeq, and cases where subjectSeqs is smaller than patternSeq will be ignored in the alignment.
- **patternSeq**: DNAString object or a sequence containing the query sequence to search. This is generally smaller than subjectSeqs.
- **side**: which side of the sequence to perform the search: left, right or middle. Default is 'left'.
- **qualityThreshold**: percent of patternSeq to match. Default is 1, full match.
- **showStats**: toggle output of search statistics. Default is FALSE.
- **bufferBases**: use x number of bases in addition to patternSeq length to perform the search. Beneficial in cases where the pattern has homopolymers or indels compared to the subject. Default is 5. Doesn’t apply when side='middle'.
- **doRC**: perform reverse complement search of the defined pattern. Default is TRUE.
pairwiseAlignSeqs

returnUnmatched
return sequences which had no or less than 5% match to the patternSeq. Default is FALSE.

returnLowScored
return sequences which had quality score less than the defined qualityThreshold. Default is FALSE.

parallel
use parallel backend to perform calculation with BiocParallel. Defaults to FALSE. If no parallel backend is registered, then a serial version is ran using SerialParam.

extra parameters for pairwiseAlignment

Value

• IRanges object with starts, stops, and names of the aligned sequences.

• If returnLowScored or returnUnmatched = T, then a CompressedIRangesList where x["hits"] has the good scoring hits, x["Rejected"] has the failed to match qualityThreshold hits, and x["Absent"] has the hits where the aligned bit is <=10% match to the patternSeq.

Note

• For qualityThreshold, the alignment score is calculated by (matches*2)-(mismatches+gaps) which programatically translates to round(nchar(patternSeq)*qualityThreshold)*2.

• Gaps and mismatches are weighed equally with value of -1 which can be overridden by defining extra parameters ‘gapOpening’ & ‘gapExtension’.

• If qualityThreshold is 1, then it is a full match, if 0, then any match is accepted which is useful in searching linker sequences at 3’ end. Beware, this function only searches for the pattern sequence in one orientation. If you are expecting to find the pattern in both orientation, you might be better off using BLAST/BLAT!

• If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following MulticoreParam SnowParam

See Also

primerIDAlignSeqs, vpairwiseAlignSeqs, doRCtest, findAndTrimSeq, blatSeqs, findAndRemoveVector

Examples

subjectSeqs <- c("CCTGAATCCTGGCAATGTCATCATC", "ATCCTGGCAATGTCATCATCAATGG", "ATCAGTTGTCAACGGCTAATACGCG", "ATCAATGGCGATTGCCGCGTCTGCA", "CCGCGTCTGCAATGTGAGGGCCTAA", "GAAGGATGCCAGTTGAAGTTCACAC")
subjectSeqs <- DNAStringSet(xscat("AAAAAAAAAA", subjectSeqs))
pairwiseAlignSeqs(subjectSeqs, "AAAAAAAAAA", showStats=TRUE)
pairwiseAlignSeqs(subjectSeqs, "AAAAAAAATA", showStats=TRUE, qualityThreshold=0.5)
primerIDAlignSeqs

**Align a short pattern with PrimerID to variable length target sequences.**

**Description**

Align a fixed length short pattern sequence containing primerID to variable length subject sequences using `pairwiseAlignment`. This function uses default of `type="overlap"`, `gapOpening=-1`, and `gapExtension=-1` to align the patternSeq against subjectSeqs. The search is broken up into as many pieces +1 as there are primerID and then compared against subjectSeqs. For example, `patternSeq="AGCATCAGCANNNNNNACGATCTACGCC"` will launch two search jobs one per either side of Ns. For each search, `qualityThreshold` is used to filter out candidate alignments and the area in between is chosen to be the primerID. This strategy is beneficial because of Indels introduced through homopolymer errors. Most likely the length of primerID(s) will not the same as you expected!

**Usage**

```r
primerIDAlignSeqs(subjectSeqs = NULL, patternSeq = NULL,
                    qualityThreshold1 = 0.75, qualityThreshold2 = 0.5,
                    doAnchored = FALSE, doRC = TRUE, returnUnmatched = FALSE,
                    returnRejected = FALSE, showStats = FALSE, ...)
```

**Arguments**

- `subjectSeqs` DNAStringSet object containing sequences to be searched for the pattern.
- `patternSeq` DNAString object or a sequence containing the query sequence to search with the primerID.
- `qualityThreshold1` percent of first part of patternSeq to match. Default is 0.75.
- `qualityThreshold2` percent of second part of patternSeq to match. Default is 0.50.
- `doAnchored` for primerID based patternSeq, use the base before and after primer ID in patternSeq as anchors?. Default is FALSE.
- `doRC` perform reverse complement search of the defined pattern. Default is TRUE.
- `returnUnmatched` return sequences if it had no or less than 5% match to the first part of patternSeq before the primerID. Default is FALSE.
- `returnRejected` return sequences if it only has a match to one side of patternSeq or primerID length does not match # of Ns +/-2 in the pattern. Default is FALSE.
- `showStats` toggle output of search statistics. Default is FALSE.
- `...` extra parameters for `pairwiseAlignment`

**Value**

- A CompressedIRangesList of length two, where x["hits"] is hits covering the entire patternSeq, and x["primerIDs"] is the potential primerID region.
- If `returnUnmatched = T`, then x["Absent"] is appended which includes reads not matching the first part of patternSeq.
• If returnRejected=TRUE, then x["Rejected"] includes reads that only matched one part of patternSeq or places where no primerID was found in between two part of patternSeq, and x["RejectedprimerIDs"] includes primerIDs that didn’t match the correct length.

• If doAnchored=TRUE, then x["unAnchoredprimerIDs"] includes reads that didn’t match the base before and after primer ID on patternSeq.

Note

• For qualityThreshold1 & qualityThreshold2, the alignment score is calculated by \((\text{matches} \times 2) - (\text{mismatches} + \text{gaps})\) which programatically translates to \(\text{round}(\text{nchar(patternSeq)}/\text{qualityThreshold}) \times 2\)

• Gaps and mismatches are weighed equally with value of -1 which can be overridden by defining extra parameters ‘gapOpening’ & ‘gapExtension’.

• If qualityThreshold is 1, then it is a full match, if 0, then any match is accepted which is useful in searching linker sequences at 3’ end. Beware, this function only searches for the pattern sequence in one orientation. If you are expecting to find the pattern in both orientation, you might be better off using BLAST/BLAT!

See Also

vpairwiseAlignSeqs, pairwiseAlignSeqs, doRCtest, blatSeqs, findAndRemoveVector

Examples

subjectSeqs <- c("CCTGAATCCTGGCAATGTCATCATC", "ATCCTGGCAATGTCATCATCAATGG", "ATCAGTTGTCAACGGCTAATACGCG", "ATCAATGGCGATTGCCGCGTCTGCA", "CCGCGTCTGCAATGTGAGGGCCTAA", "GAAGGATGCCAGTTGAAGTTCACAC")
ids <- c("GGTTCTACGT", "AGGAGTATGA", "TGTCGGTATA", "GTTATAAAC", "AGGCTATATC", "ATGGTTTGTT")
subjectSeqs <- xscat(subjectSeqs, xscat("AAGCGGAGCCC", ids, "TTTTTTTTTTT"))
patternSeq <- "AAGCGGAGCCCNNNNNNNNNNTTTTTTTTTTT"
primerIDAlignSeqs(DNAStringSet(subjectSeqs), patternSeq, doAnchored = TRUE)

psl

PSL file output

Description

Sample BLAT PSL file output from samples included Integration Sites Sequencing Data seqProps

Format

a data frame of 1000 rows and 21 columns
pslCols

Return PSL file columns with classes

Description
Print out required fields & classes of PSL file format

Usage
pslCols(withClass = TRUE)

Arguments
withClass return classes for each column.

Value
vector of PSL column names

See Also
pairwiseAlignSeqs, vpairwiseAlignSeqs, startgfServer, blatSeqs, read.blast8, read.BAMasPSL, pslToRangedObject

Examples
pslCols()

pslToRangedObject
Convert psl dataframe to GRanges

Description
Convert psl dataframe to GRanges object using either the query or target as the reference data column.

Usage
pslToRangedObject(x, useTargetAsRef = TRUE, isblast8 = FALSE)

Arguments
x dataframe reflecting psl format
useTargetAsRef use target(tName) or query(qName) as the chromosome or the reference data. Default is TRUE.
isblast8 the input dataframe blast8 format output from BLAT. Default is FALSE.

Value
a GRanges object reflecting psl file type.
See Also

read.psl, read.blast8, blatListedSet

Examples

data(psl)
psl <- head(psl)
pslToRangedObject(psl)
pslToRangedObject(psl, useTargetAsRef = FALSE)

---

read.BAMasPSL

Reads a BAM/SAM file and converts it into a PSL like format.

Description

Given filename(s), the function reads the BAM/SAM file, converts into a PSL like format. Any other file format will yield errors or erroneous results. This is intended to be used independently with other short read aligners.

Usage

read.BAMasPSL(bamFile = NULL, removeFile = TRUE, asGRanges = TRUE)

Arguments

bamFile

BAM/SAM filename, or vector of filenames, or a pattern of files to import.

removeFile

remove the file(s) supplied in bamFile parameter after importing. Default is FALSE.

asGRanges

return a GRanges object. Default is TRUE

Value

a GRanges or GAlignments object reflecting psl file type.

See Also

pairwiseAlignSeqs, blatSeqs, read.blast8, read.psl, pslToRangedObject, pairUpAlignments

Examples

read.BAMasPSL(bamFile = "processed.*.bam")
read.BAMasPSL(bamfile = c("sample1hits.bam", "sample2hits.bam"))
Description

Given filename(s), the function reads the blast8 file format from BLAT as a data frame and performs basic score filtering if indicated. Any other file format will yield errors or erroneous results.

Usage

read.blast8(files = NULL, asGRanges = FALSE, removeFile = TRUE, parallel = FALSE)

Arguments

files  
blast8 filename, or vector of filenames, or a pattern of files to import.

asGRanges  
return a GRanges object instead of a dataframe. Default is TRUE Saves memory!

removeFile  
remove the blast8 file(s) after importing. Default is FALSE.

parallel  
use parallel backend to perform calculation with BiocParallel. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam.

Value

a dataframe or GRanges object reflecting blast8 file type.

Note

If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following MulticoreParam SnowParam

See Also

pairwiseAlignSeqs, vpairwiseAlignSeqs, startgfServer, blatSeqs, read.psl

Examples

# this function works similar to read.psl #
#read.blast8(files="processed.*.blast8")
#read.blast8(files=c("sample1hits.blast8","sample2hits.blast8"))
Description

Given filename(s), the function reads the PSL file format from BLAT as a data frame and performs basic score filtering if indicated. Any other file format will yield errors or erroneous results. Make sure there is no header row! See required columns in pslCols.

Usage

read.psl(pslFile = NULL, bestScoring = TRUE, asGRanges = FALSE, removeFile = TRUE, parallel = FALSE)

Arguments

pslFile PSL filename, or vector of filenames, or a pattern of files to import.
bestScoring report only best scoring hits instead of all hits. Default is TRUE. Score is calculated by matches-misMatches-qBaseInsert-tBaseInsert.
asGRanges return a GRanges object instead of a dataframe. Default is FALSE
removeFile remove the PSL file(s) after importing. Default is FALSE.
parallel use parallel backend to perform calculation with BiocParallel. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam.

Value

da dataframe reflecting psl file type. If asGRanges=TRUE, then a GRanges object.

Note

If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following MulticoreParam SnowParam

See Also

pairwiseAlignSeqs, vp pairwiseAlignSeqs, startgfServer, blatSeqs, read.blast8, read.BAMasPSL, pslToRangedObject, write.psl

Examples

## Not run:
data(psl)
pslFile <- tempfile()
write.psl(psl, filename = pslFile)
head(read.psl(pslFile = pslFile))
# read many PSL files matching the regex#
psl <- read.psl(pslFile = "processed.*.psl")

## End(Not run)
**read.sampleInfo**

Read a sample information file and format appropriate metadata.

**Description**

Given a sample information file, the function checks if it includes required information to process samples present on each sector/quadrant/region/lane. The function also adds other columns required for processing with default values if not already defined ahead of time.

**Usage**

```r
read.sampleInfo(sampleInfoPath = NULL, splitBySector = TRUE)
```

**Arguments**

- **sampleInfoPath** full or relative path to the sample information file, which holds samples to quadrant/lane associations along with other metadata required to trim sequences or process it.
- **splitBySector** split the data frame into a list by sector column. Default is TRUE.

**Details**

- **Required Column Description**:
  - **sector** => region/quadrant/lane of the sequencing plate the sample comes from. If files have been split by samples apriori, then the filename associated per sample without the extension. If this is a filename, then be sure to enable ‘alreadyDecoded’ parameter in `findBarcodes`, since contents of this column is pasted together with ‘seqfilePattern’ parameter in `read.SeqFolder` to find the appropriate file needed. For paired end data, this is basename of the FASTA/Q file holding the sample data from the LTR side. For example, files such as Lib3_L001_R2_001.fastq.gz or Lib3_L001_R2_001.fastq would be Lib3_L001_R2_001, and consequently Lib3_L001_R1_001 would be used as the second pair!
  - **barcode** => unique 4-12bp DNA sequence which identifies the sample. If providing filename as sector, then leave this blank since it is assumed that the data is already demultiplexed.
  - **primerltrsequence** => DNA sequence of the viral LTR primer with/without the viral LTR sequence following the primer landing site. If already trimmed, then mark this as SKIP.
  - **sampleName** => Name of the sample associated with the barcode
  - **sampleDescription** => Detailed description of the sample
  - **gender** => sex of the sample: male or female or NA
  - **species** => species of the sample: homo sapien, mus musculus, etc.
  - **freeze** => UCSC freeze to which the sample should be aligned to.
  - **linkerSequence** => DNA sequence of the linker adaptor following the genomic sequence. If already trimmed, then mark this as SKIP.
  - **restrictionEnzyme** => Restriction enzyme used for digestion and sample recovery. Can also be one of: Fragmentase or Sonication!

- **Metadata Parameter Column Description**:
  - **ltrBitSequence** => DNA sequence of the viral LTR following the primer landing site. Default is last 7bps of the primerltrsequence.

---

```r
read.sampleInfo(sampleInfoPath = NULL, splitBySector = TRUE)
```
– `ltrBitIdentity` => percent of LTR bit sequence to match during the alignment. Default is 1.
– `primerLTRidentity` => percent of primer to match during the alignment. Default is .85.
– `linkerIdentity` => percent of linker sequence to match during the alignment. Default is 0.55. Only applies to non-primerID/random tag based linker search.
– `primerIdInLinker` => whether the linker adaptor used has primerID/random tag in it? Default is FALSE.
– `primerIdInLinkerIdentity1` => percent of sequence to match before the random tag. Default is 0.75. Only applies to primerID/random tag based linker search and when primeridinlinker is TRUE.
– `primerIdInLinkerIdentity2` => percent of sequence to match after the random tag. Default is 0.50. Only applies to primerID/random tag based linker search and when primeridinlinker is TRUE.
– `celltype` => celltype information associated with the sample
– `user` => name of the user who prepared or processed the sample
– `pairedEnd` => is the data paired end? Default is FALSE.
– `vectorFile` => fasta file containing the vector sequence

• Processing Parameter Column Description:
  – `startWithin` => upper bound limit of where the alignment should start within the query. Default is 3.
  – `alignRatioThreshold` => cutoff for (alignment span/read length). Default is 0.7.
  – `genomicPercentIdentity` => cutoff for (1-(misMatches/matches)). Default is 0.98.
  – `clusterSitesWithin` => cluster integration sites within a defined window size based on frequency which corrects for any sequencing errors. Default is 5.
  – `keepMultiHits` => whether to keep sequences reads that return multiple best hits, aka ambiguous locations.
  – `processingDate` => the date of processing

**Value**

If `splitBySector=TRUE`, then an object of SimpleList named by quadrant/lane information defined in sampleInfo file, else a dataframe.

**See Also**

`read.SeqFolder, findBarcodes, splitByBarcode`

**Examples**

```r
runData <- system.file(file.path("extdata", "FLX_sample_run"),
package = "hiReadsProcessor")
read.sampleInfo(file.path(runData, "sampleInfo.xlsx"))
```
read.SeqFolder  

Read contents of a sequencing folder and make a SimpleList object

Description

Given a sequencing folder path, sample information file path, and sequence file extension pattern, the function returns a list of variables required to process the data. The function also calls `read.sampleInfo` which reads in sample processing metadata and formats it if needed.

Usage

```r
read.SeqFolder(sequencingFolderPath = NULL, sampleInfoFilePath = NULL, seqfilePattern = NULL)
```

Arguments

- **sequencingFolderPath**
  full or relative path to the sequencing folder

- **sampleInfoFilePath**
  full or relative path to the sample information file, which holds samples to quadrant/lane associations along with other metadata required to trim sequences or process it. Default to NULL, where the function tries to find xls/xlsx or tab delimited txt file in the sequencing folder which sounds similar to 'sampleinfo' and present you with choices of file to select from.

- **seqfilePattern** regex/string to describe sequence file endings. See examples. Default is NULL.

Value

a SimpleList list which is used by other functions to process and decode the data.

Note

- One must make sure that each sequencing file has sector name/number prefixed at the beginning, else `findBarcodes` will fail trying to find the filename.

- For paired end Illumina runs, make sure the filenames include R1, R2, and I1 somewhere in the name denoting pair1, pair2, and index/barcode reads, respectively.

See Also

`read.sampleInfo`, `findBarcodes`, `splitByBarcode`

Examples

```r
runData <- system.file("extdata/FLX_sample_run/", package = "hiReadsProcessor")
read.SeqFolder(runData, seqfilePattern=".fna.gz$")
```

```r
## Not run:
read.SeqFolder(., seqfilePattern = "\TCA.454Reads.fna$")
read.SeqFolder(., seqfilePattern = ".fastq$")
read.SeqFolder(., seqfilePattern = ".sff$")
```

## End(Not run)
removeReadsWithNs

Remove sequences with ambiguous nucleotides.

Description
Given a DNAStringSet object, the function removes any reads that has either repeating or total Ns which is greater than to maxNs threshold

Usage
removeReadsWithNs(dnaSet, maxNs = 5, consecutive = TRUE)
replicateReads

Arguments

dnaSet DNAStringSet object to evaluate.
maxNs integer value denoting the threshold of maximum allowed Ns. Default is 5.
consecutive boolean flag denoting whether Ns to filter is consecutive or total. Default is TRUE.

Value

DNAStringSet object.

See Also

dereplicateReads, replicateReads, findBarcodes, splitByBarcode

Examples

dnaSet <- c("CCTGAATCTNNCAATGTCATCATC", "ATCCTGGCNATGTCATCATCAATGG", "ATCAATTGCGATGCGCTTGCAGTGTA", "CGCATTGCATGTAATGGCTCAGG", "GAAGNNNNGTGTGAATTCACAC")
removeReadsWithNs(dnaSet)
removeReadsWithNs(dnaSet, maxNs = 4, consecutive = FALSE)

replicateReads  Replicate sequences from DNAStringSet object using counts identifier or vector

Description

Given a DNAStringSet object, the function replicates reads using counts=X marker at the end of definition line.

Usage

replicateReads(dnaSet, counts = NULL)

Arguments

dnaSet DNAStringSet object to replicate.
counts an integer or a numeric vector of length length(dnaSet) indicating how many times to repeat each sequence. Default is NULL, in which it uses counts=X notation from the definition line to replicate reads.

Value

DNAStringSet object.

See Also

dereplicateReads, removeReadsWithNs, findBarcodes, splitByBarcode
seqProps

Example

```r
dnaSet <- dereplicateReads(dnaSet)
replicateReads(dnaSet)
```

Description

Simple summary of a sampleInfo object.

Usage

```r
sampleSummary(object, ...)
```

Arguments

- `object` sample information SimpleList object, which samples per sector/quadrant information along with other metadata.
- `...` ignored for now.

Value

a dataframe summarizing counts of major attributes per sample and sector.

Examples

```r
data(FLX_seqProps)
sampleSummary(seqProps)
```

seqProps

Sample Integration Sites Sequencing Data

Description

This is a processed data object containing raw sequences and respective alignments to UCSC freeze hg18 from 112 integration site samples. The object is of SimpleList class and follows a certain structural hierarchy explained by the Introductory vignette.

Format

A SimpleList object
splitByBarcode

Split DNAStringSet object using first X number of bases defined by a vector.

Description

Given a character vector of barcodes/MID to sample association and a DNAStringSet object, the function splits/demultiplexes the DNAStringSet object by first few bases dictated by length of barcodes/MID supplied. This is an accessory function used by findBarcodes

Usage

splitByBarcode(barcodesSample, dnaSet, trimFrom = NULL, showStats = FALSE, returnUnmatched = FALSE)

Arguments

barcodesSample a character vector of barcodes to sample name associations. Ex: c("ACATCCAT"="Sample1", "GAATGGAT"="Sample2",...)
dnaSet DNAStringSet object to evaluate.
trimFrom integer value serving as start point to trim the sequences from. This is calculated internally length barcode+1. Default is NULL.
showStats boolean flag denoting whether to show decoding statistics per sample & barcode. Default is FALSE.
returnUnmatched boolean flag denoting whether to return unmatched reads. Default is FALSE.

Value

DNAStringSet object split by sample name found in barcodesSample.

See Also

findBarcodes, dereplicateReads, replicateReads

Examples

dnaSet <- DNAStringSet(c("read1" = "ACATCCATAGCTACGACGACATCGACATA", "read2"="GAATGGATGACGACTACAGCACGACGAGCAGCTACT", "read3"="GAATGGATGCGCTAAGAAGAGA", "read4"="ACATCCATCTACACATCT"))
splitByBarcode(c("ACATCCAT" = "Sample1", "GAATGGAT" = "Sample2"), dnaSet, showStats=TRUE)
splitSeqsToFiles  Split DNA sequences into smaller files.

Description
Given a vector of sequences or DNAStringSet or a FASTA filename, the function splits it into smaller pieces as denoted by totalFiles parameter.

Usage
splitSeqsToFiles(x, totalFiles = 4, suffix = "tempy", filename = "queryFile.fa", outDir = getwd())

Arguments
x a DNAStringSet object, or a FASTA filename.
totalFiles an integer indicating how many files to create. Default is 4.
suffix a word to add to each file created. Default is "tempy".
filename name of the file if x is a DNAStringSet object. Default is "queryFile.fa".
outDir directory to write the output file. Default is current directory.

Value
a vector of filename names created.

See Also
blatSeqs

Examples
seqs <- DNAStringSet(sapply(sample(c(100:1000), 500),
function(size) paste(sample(DNA_BASES, size, replace = TRUE), collapse = ""))
splitSeqsToFiles(seqs, 5, "tempyQ", "myDNASEqs.fa", tempdir())

startgfServer  Start/Stop a gfServer instance

Description
Start or Stop a gfServer with indexed reference genome to align batch of sequences using BLAT gfServer/gfClient protocol.

Usage
startgfServer(seqDir = NULL, host = "localhost", port = 5560, gfServerOpts = c(repMatch = 112312, stepSize = 5, tileSize = 10, maxDnaHits = 10))
stopgfServer(host = "localhost", port = NULL)
trimSeqs

Arguments

seqDir absolute or relative path to the genome index (nib/2bit files).

host name of the machine to run gfServer on. Default: localhost

port a port number to host the gfServer with. Default is 5560.

gfServerOpts a character vector of options to be passed to gfServer command on top of server
defaults. Default: c(repMatch=112312, stepSize=5, tileSize=10, maxDnaHits=10).
Set this to NULL to start gfServer with defaults.

Value

system command status for executing gfServer command.

See Also

stopgfServer, read.psl, blatSeqs, read.blast8

Examples

#startgfServer(seqDir="/usr/local/blatSuite34/hg18.2bit",port=5560)
#stopgfServer(port=5560)

trimSeqs

Trim sequences from a specific side.

Description

This function trims a DNAStringSet object using the ranges from left, right, or middle of the se-
quency. This is a helper function utilized in primerIDAlignSeqs and extractSeqs. If dnaSet
and coords are not the same length, then they are required to have a names attribute to perform the
matched trimming.

Usage

trimSeqs(dnaSet, coords, side = "middle", offBy = 0)

Arguments

dnaSet DNAStringSet object containing sequences to be trimmed.

coords IRanges object containing boundaries.

side either ’left’,’right’,or the Default ’middle’.

offBy integer value dictating if the supplied coordinates should be offset by X number
of bases. Default is 0.

Value

a DNAStringSet object with trimmed sequences.
troubleshootLinkers

Note

If side is left, then any sequence following end of coords+offBy is returned. If side is right, then sequence preceding start of coords-offBy is returned. If side is middle, then sequence contained in coords is returned where offBy is added to start and subtracted from end in coords.

See Also

extractSeqs, primerIDAlignSeqs

Examples

dnaSet <- DNAStringSet(c("AAAAACCTGAATGTCATCATC", "AAAAACCTGCAATGTCATCATC", "AAAAAATCAATGGCGATGTCATCAT", "AAAAAATCAATGGCGATGTCATCATC", "AAAAAATCAATGGCGATGTCATCATC", "AAAAAAAAGGATGTCGGAATGTCACAC"))
coords <- IRanges(start=1, width=rep(10,6))
trimSeqs(dnaSet, coords, side="left", offBy=1)
trimSeqs(dnaSet, coords, side="middle")

troubleshootLinkers

Compare LTRed/Primed sequences to all linkers.

Description

Given a SampleInfo object, the function compares LTRed sequences from each sample per sector to all the linker sequences present in the run. The output is a summary table of counts of good matches to all the linkers per sample.

Usage

troubleshootLinkers(sampleInfo, qualityThreshold = 0.55,
  qualityThreshold1 = 0.75, qualityThreshold2 = 0.5, doRC = TRUE,
  parallel = TRUE, samplenames = NULL, ...)

Arguments

sampleInfo sample information SimpleList object outputted from findPrimers or findLTRs, which holds decoded sequences for samples per sector/quadrant along with information of sample to primer associations.

qualityThreshold percent of linker length to match, round(nchar(linker)*qualityThreshold). Default is 0.55. Only applies to non-primerID based linkers

qualityThreshold1 percent of first part of patternSeq to match. Default is 0.75. Only applies to primerID based linker search.

qualityThreshold2 percent of second part of patternSeq to match. Default is 0.50. Only applies to primerID based linker search.

doRC perform reverse complement search of the linker sequence. Default is TRUE. Highly recommended!
Align a short pattern to variable length target sequences.

Align a fixed length short pattern sequence to subject sequences using \texttt{vmatchPattern}. This function is meant for aligning a short pattern onto large collection of subjects. If you are looking to align a vector sequence to subjects, then please use BLAT.

\texttt{vpairwiseAlignSeqs(subjectSeqs = NULL, patternSeq = NULL, side = "left", qualityThreshold = 1, showStats = FALSE, bufferBases = 5, doRC = TRUE, parallel = FALSE, \ldots)}

**Arguments**

- \texttt{subjectSeqs} DNAStringSet object containing sequences to be searched for the pattern. This is generally bigger than patternSeq, and cases where subjectSeqs is smaller than patternSeq will be ignored in the alignment.
- \texttt{patternSeq} DNAString object or a sequence containing the query sequence to search. This is generally smaller than subjectSeqs.
- \texttt{side} which side of the sequence to perform the search: left, right, or middle. Default is ‘left’.
- \texttt{qualityThreshold} percent of patternSeq to match. Default is 1, full match. This is supplied to max.mismatch parameter of \texttt{vmatchPattern} as round(nchar(patternSeq)*(1-qualityThreshold)).
- \texttt{showStats} toggle output of search statistics. Default is FALSE.
bufferBases: use x number of bases in addition to patternSeq length to perform the search. Beneficial in cases where the pattern has homopolymers or indels compared to the subject. Default is 5. Doesn’t apply when side='middle'.

doRC: perform reverse complement search of the defined pattern. Default is TRUE.

parallel: use parallel backend to perform calculation with BiocParallel. Defaults to FALSE. If no parallel backend is registered, then a serial version is ran using SerialParam.

... extra parameters for vmatchPattern except for 'max.mismatch' since it’s calculated internally using the 'qualityThreshold' parameter.

Value

IRanges object with starts, stops, and names of the aligned sequences.

Note

- For qualityThreshold, the alignment score is calculated by \((\text{matches} \times 2) - (\text{mismatches} + \text{gaps})\) which programatically translates to \(\text{round}(2 \times \text{nchar(patternSeq)} \times \text{qualityThreshold})\).
- No indels are allowed in the function, if expecting indels then use pairwiseAlignSeqs.
- If qualityThreshold is 1, then it is a full match, if 0, then any match is accepted which is useful in searching linker sequences at 3’ end. Beware, this function only searches for the pattern sequence in one orientation. If you are expecting to find the pattern in both orientation, you might be better off using BLAST/BLAT!
- If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following MulticoreParam SnowParam

See Also

pairwiseAlignSeqs, primerIDAlignSeqs, doRCtest, findAndTrimSeq, blatSeqs, findAndRemoveVector

Examples

subjectSeqs <- c("CCTGAATCTGCAATGTCATCATC", "ATCCTGGCAATGTCATCATCAATGG", "ATCAGTTGTCAACGGCTAATACGCG", "ATCAATGGCGATTGCCGCGTCTGCA", "CCCGCTCTGCAATGTGAGGGCCTAA", "GAAGGATGCCAGTTGAAGTTCACAC")
subjectSeqs <- DNAStringSet(xscat("AAAAAAAAAA", subjectSeqs))
vpairwiseAlignSeqs(subjectSeqs, "AAAAAAAAAA", showStats=TRUE)
vpairwiseAlignSeqs(subjectSeqs, "AAAAAAAAAA", showStats=TRUE, qualityThreshold=0.5)

write.listedDNAStringSet

Write a fasta file per sample in parallel

Description

Given a listed DNAStringSet object return from extractSeqs, the function writes a fasta file for each sample as defined in filePath parameter.
write.listedDNAStringSet

Usage

write.listedDNAStringSet(dnaSet, filePath = ".", filePrefix = "processed", prependSampleNames = TRUE, format = "fasta", parallel = FALSE)

Arguments

dnaSet listed DNAStringSet object containing sequences to be written.
filePath a path write the fasta files per sample. Default is current working directory.
filePrefix prefix the filenames with a string. Default is 'processed' followed by sample-name.
prependSampleNames Prepend definition lines with sample names. Default is TRUE. Make sure the dnaSet parameter is a named list where names are used as sample names.
format either fasta (the default) or fastq.
parallel use parallel backend to perform calculation with BiocParallel. Defaults to TRUE. If no parallel backend is registered, then a serial version is ran using SerialParam.

Note

• Writing of the files is done using writeXStringSet with parameter append=TRUE. This is to aggregate reads from a sample which might be present in more than one sector.

• If data is paired end, then each pair will be written separately with designations in the filename as well as in the definition line as (at)pairX(at) appended at the end.

• If parallel=TRUE, then be sure to have a parallel backend registered before running the function. One can use any of the following MulticoreParam SnowParam

See Also

findBarcodes, read.SeqFolder, extractSeqs, addListNameToReads

Examples

load(file.path(system.file("data", package = "hiReadsProcessor"), "FLX_seqProps.RData"))
samples <- c("Roth-MLV3p-CD4TMLVWell16-Tsp509I", "Roth-MLV3p-CD4TMLVWell16-MseI", "Roth-MLV3p-CD4TMLVWell15-MuA")
seqs <- extractSeqs(seqProps, sector=2, sampleName=samples, feature="primed")
write.listedDNAStringSet(seqs)
write.psl

Write PSL file from data frame or GRanges

Description
Given a data frame or GRanges object, the function write a tab delimited PSL file

Usage
write.psl(x, filename = "out.psl", header = FALSE,
includeOtherCols = FALSE)

Arguments
x data frame or GRanges object with required columns for psl file format.
filename name for the output PSL file. Default is "out.psl"
header include PSL header line. Default is FALSE.
includeOtherCols include other non PSL specific columns from x in the output. Default is FALSE.

Value
name of the output PSL file

See Also
read.psl, blatSeqs, read.blast8, read.BAMasPSL, pslToRangedObject

Examples
data(psl)
pslFile <- tempfile()
write.psl(psl, filename = pslFile)
Index

*Topic datasets
  psl, 34
  seqProps, 44

addFeature, 3, 13, 26
addListNameToReads, 4, 51
annotateSites, 4, 19

BiocParallel, 7, 9, 12, 15, 18, 20, 21, 24, 27–30, 32, 37, 38, 49–51
blatListedSet, 5, 19, 25, 36
blatSeqs, 5, 6, 13, 16, 18, 19, 23–25, 31, 32, 34–38, 46, 47, 50, 52

chunkize, 7
crossOverCheck, 5, 9, 10, 19, 25, 28, 30
decodeByBarcode (findBarcodes), 16
dereplicateReads, 11, 17, 43, 45
doAnnotation, 5
doRCTest, 11, 32, 34, 50
dosRCtest, 3, 4, 12, 14, 16, 20, 22–24, 26
doextractSeqs, 3–5, 13, 13, 16, 18–24, 26, 30, 42, 47, 48, 50, 51

findAndRemoveVector, 14, 24, 31, 32, 34, 50
findAndTrimSeq, 15, 15, 20, 22–24, 32, 49, 50
findBarcodes, 11, 13, 16, 22, 30, 39–43, 45, 51
findIntegrations, 5, 9, 10, 18, 20, 25, 27, 28, 30
findLinkers, 13, 14, 16, 18, 19, 19, 22–24
findLTRs, 13, 14, 19, 20, 21, 23, 24, 48, 49
findPrimers, 3, 13, 14, 16, 19–22, 22, 24, 48, 49
findVector, 13, 23

getIntegrationSites, 5, 8–10, 18, 19, 24, 27, 28, 30, 31
getSectorsForSamples, 3, 4, 13, 14, 25
getSonicAbund, 9, 18, 19, 26

hiReadsProcessor, 27
hiReadsProcessor-package
  (hiReadsProcessor), 27

isuSites, 5, 9, 18, 19, 25, 28, 28, 30

MulticoreParam, 15, 16, 19–21, 23, 24, 32, 37, 38, 49–51

otuSites, 9, 10, 18, 27, 29

pairUpAlignments, 30, 36
pairwiseAlignment, 16, 20–23, 31–33, 49
pairwiseAlignSeqs, 6–8, 12, 15, 16, 19–24, 31, 34–38, 49, 50
primerIDAlignSeqs, 8, 12, 13, 16, 19, 20, 22, 23, 32, 33, 47–50

psl, 34
pslCols, 35, 38
pslToRangedObject, 5, 6, 9, 10, 15, 19, 25, 27, 28, 30, 35, 36, 38, 52

read.BAMasPSL, 31, 35, 36, 38, 52
read.blast8, 6, 7, 15, 25, 31, 35, 36, 37, 38, 47, 52
read.psl, 6, 7, 18, 19, 24, 25, 31, 36, 37, 38, 47, 52
read.sampleInfo, 17, 23, 39, 41
read.SeqFolder, 17, 39, 40, 41, 42, 51
read.seqsFromSector, 17, 42
removeReadsWithNs, 11, 42, 43
replicateReads, 11, 17, 43, 43, 45

sampleSummary, 44
seqProps, 34, 44
SerialParam, 5, 7, 9, 12, 15, 18, 20–22, 24, 27–30, 32, 37, 38, 49–51
SnowParam, 15, 16, 19–21, 23, 24, 32, 37, 38, 49–51

sonicLength, 26
splitByBarcode, 11, 16, 17, 40, 41, 43, 45
splitSeqsToFiles, 7, 46
startgfServer, 6, 7, 19, 25, 35, 37, 38, 46
stopgfServer, 6, 7, 47
stopgfServer (startgfServer), 46
trimSeqs, 3, 13–15, 47
troubleshootLinkers, 48

vcountPattern, 11, 12
vmatchPattern, 16, 22, 23, 49, 50
vpairwiseAlignSeqs, 6–8, 12, 15, 16, 20, 22, 23, 32, 34, 35, 37, 38, 49, 49

write.listedDNAStringSet, 4, 50
write.psl, 38, 52
writeXStringSet, 51