Package 'RIPSeeker'

October 16, 2019

Type Package

Title RIPSeeker: a statistical package for identifying protein-associated transcripts from RIP-seq experiments

Version 1.24.0

Date 2013-Apr-13

Author Yue Li

Maintainer Yue Li <yueli@cs.toronto.edu>

Description Infer and discriminate RIP peaks from RIP-seq alignments using two-state HMM with negative binomial emission probability. While RIPSeeker is specifically tailored for RIP-seq data analysis, it also provides a suite of bioinformatics tools integrated within this self-contained software package comprehensively addressing issues ranging from post-alignments processing to visualization and annotation.

Depends R (>= 2.15), methods, S4Vectors (>= 0.9.25), IRanges, GenomicRanges, SummarizedExperiment, Rsamtools, GenomicAlignments, rtracklayer

Suggests biomaRt, ChIPpeakAnno, parallel, GenomicFeatures

License GPL-2

URL http://www.cs.utoronto.ca/~yueli/software.html

Lazyload yes

biocViews Sequencing, RIPSeq

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

git_branch RELEASE_3_9

git_last_commit 50db7d9

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

R topics documented:

`RIPSeeker-package` *RIPSeeker: a statistical package for identifying protein-associated transcripts from RIP-seq experiments*

Description

RIPSeeker infers and discriminates RIP peaks from RIP-seq alignments using two-state HMM with negative binomial emission probability. While RIPSeeker is specifically tailored for RIP-seq data analysis, it also provides a suite of bioinformatics tools integrated within this self-contained software package comprehensively addressing issues ranging from post-alignments processing to visualization and annotation. In addition, a rule-based approach is provided as an additional function named `rulebaseRIPSeek` for user to obtain RPKM/FPKM (and fold-change) for the gene/transcripts expressions in RIP (and control) based on automatically retrieved online Ensembl annotation given single or paired-end alignments.

Details

Package:	RIPSeeker
Type:	Package
Version:	1.4.0
Date:	2012-11-06
License:	GPL-2

The front-end main function `ripSeek` suffices for most applications. The function takes as the only required argument the path to alignment files (BAM/BED/SAM) and outputs predicted RIP regions. Optionally, user may indicate via 'cNAME' which file(s) in the first file argument list is/are control to enable empirical false discover rate (eFDR) computation. If the arguments 'biomaRt_dataset' and/or 'goAnno' are set, ripSeek will return the annotated RIP predictions and the enriched GO terms corresponding to the genomic context of the RIP predictions. User can also specify the thresholds for statistical significance scores via `logOddCutoff`, `pvalCutoff`, `pvalAdjCutoff`, `eFDRCutoff`.

Author(s)

Yue Li <yueli@cs.toronto.edu>

References

Li, Y., Zhao, D. Y., Greenblatt, J. F., & Zhang, Z. (2013). RIPSeeker: a statistical package for identifying protein-associated transcripts from RIP-seq experiments. Nucleic Acids Research. doi:10.1093/nar/gkt142

Zhao, J., Ohsumi, T. K., Kung, J. T., Ogawa, Y., Grau, D. J., Sarma, K., Song, J. J., et al. (2010). Genome-wide Identification of Polycomb-Associated RNAs by RIP-seq. Molecular Cell, 40(6), 939D953. doi:10.1016/j.molcel.2010.12.011

See Also

`ripSeek`,`rulebaseRIPSeek`

Examples

```
library(RIPSeeker)

ls("package:RIPSeeker")
```

`addDummyProb` *Create a dummy GRanges object as a placeholder in case nbh_em fails (Internal function)*

Description

This function is used to generate a place holder in cases the EM fails to converge on a chromosome due to too few number of reads mapped to that chromosome. This is an internal function not expected to be directly called by the user.

Usage

```
addDummyProb(alignGR, K = 2, randomProb = FALSE, runViterbi = FALSE, ...)
```

Arguments

alignGR	GRanges object derived from RIP-seq alignment inputs.
K	Number of hidden states (Default: 2).
randomProb	A binary value to indicate whether to use random probability as a place holder to present posterior probabilities. If set FALSE, (by default), equal probability is used for all states.
runViterbi	A binary value to indicate whether to generate place holder for the Viterbi state sequence (Default: FALSE).
...	Additional arguments expected to contain the bin size used for computing the bin counts in binCount function, and any other extra arguments are ignored.

Details

A priviate function to fall back in case HMM fails to converge mostly due to too many zero counts in the input vector. When that occurs, a GRanges place holder object needs to be returned to keep consistent with the remaining GRanges for each chromosome. Thus, all information slot will be generated as place holder to properly create the GRangesList for the predictions on all chromosomes (each as an GRanges item in the list).

Value

GRanges	A GRanges object containing the read count (in the defined bin size), alpha, beta, TRANS dummy values for the HMM

Author(s)

Yue Li

See Also

mainSeekSingleChrom

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGRList <- GRangesList(as.list(split(alignGR, seqnames(alignGR))))

x <- addDummyProb(alignGRList$chrX, binSize=10000)
```

x

---

addPseudoAlignment *Add a psuedoalignment as a placeholder for the chromosome (Internal function)*

Description

Check whether chromosome has at least one alignment to prevent abnormal behaviour of the subsequent functions. In case no alignment is found on an entire chromosome, add a pseudo-alignment as a placeholder for that chromosome.

Usage

```
addPseudoAlignment(alignGR)
```

Arguments

alignGR	GRanges object containing the alignment information.

Details

In case no alignment is found on an entire chromosome, add an alignment with start 1 and end 20 as a placeholder for the chromosome. This step is necessary to maintian the chromosome information.

Value

alignGR	Original or augmented input GRanges object with pseudoreads, depending on whether there exists empty chromosome(s).

Author(s)

Yue Li

See Also

combineAlignGals, readGAlignments, readGAlignmentPairs, import

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGR
```

```
x <- addPseudoAlignment(alignGR)

x
```

---

annotateRIP *Annotate RIP peaks with genomic information and perform GO enrichement*

Description

Given the genomic coordinates of each predicted RIP regions, query the Ensembl database whether each region is nearby or overlaps any known (noncoding) genes.

Usage

```
annotateRIP(sigGRanges, biomaRt_dataset, featureType = "TSS",
goAnno, strandSpecific = FALSE, exportFormat = "txt",
hasGOdb = !missing(goAnno), goPval = 0.1, outDir, ...)
```

Arguments

sigGRanges	GRanges object indicating the chromosomal coordinates of each RIP peaks.
biomaRt_dataset	Ensembl dataset available from biomaRt (See listDatasets). For instance, the human and mouse annotations are hsapiens_gene_ensembl and mmusculus_gene_ensembl, respectively.
featureType	TSS, miRNA, Exon, 5'UTR, 3'UTR, transcript or Exon plus UTR defined in getAnnotation.
goAnno	Optional argugment that specifies a GO dataset used for GO enrichement analysis performed by getEnrichedGO. For instance, the human and mouse GO datasets are org.Hs.eg.db and org.Mm.eg.db.
strandSpecific	Indicate whether the annotations should be strand-specific (Default: FALSE)
exportFormat	Format to export using exportGRanges (Default: "txt", i.e. tab-delim file).
hasGOdb	A binary flag that indicates whether GO enrichement is performed in order to export the results. hasGOdb can be FALSE either because goAnno is not specifiy or because the GO database does not exist.
goPval	P-value cutoff to determine the significance of enriched GO terms by getEnrichedGO.
outDir	Output directory.
...	Extra arguments passed to useMart to specify the database and to passed getEnrichedGO to specify the GO enrichment procedure.

Details

To access the up-to-date Ensembl database, RIPSeeker employs `useMart` and `getAnnotation` from biomaRt and ChIPpeakAnno Bioconductor packages to dynamically establish internet connection to the database and retrieve the up-to-date annotations. Then, `annotatePeakInBatch` from ChIPpeakAnno is used to efficiently annotate all of the predicted regions based on the Ensembl annotation. A predicted region may overlap multiple genes, all of which will be reported as separate records. Moreover, `getEnrichedGO` from ChIPpeakAnno is applied to the annotated predictions to discover enriched Gene Ontology (GO) terms involving the protein-associated transcriptome.

In order to use old annotation (e.g., mm9 v.s. mm10), user also needs to specify the host and biomart arguments accepted within `useMart`. To access to mouse annotation from Ensembl version 65, for instance, user needs to call annotateRIP(..., dataset="mmusculus_gene_ensembl", biomart="ENSEMBL_MART_ENSEM host="dec2011.archive.ensembl.org", ...), which will run useMart(dataset="mmusculus_gene_ensembl", biomart="ENSEMBL_MART_ENSEMBL", host="dec2011.archive.ensembl.org", ...) to get the mm9 annotation from Ensembl (v65).

Value

`sigGRangesAnnotated`
: `sigGRanges` augmented with genomic information including "ensembl_gene_id", "external_gene_id", and "description"

`enrichedGO`
: Output from getEnrichedGO. All three main GO categories ("Biological Process", "Molecular Function", "Cellular Component") are combined together and returned. The argument is only returned when `hasGOdb` is TRUE.

If `outDir` is specified, then the above `sigGRangesAnnotated` is saved as RIPregions_annotated.txt and RIPregions_annotated.RData, and `enrichedGO` as RIPregions_enrichedGO.txt in the `outDir` directory.

Author(s)

Yue Li

References

Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. Steffen Durinck, Paul T. Spellman, Ewan Birney and Wolfgang Huber, Nature Protocols 4, 1184-1191 (2009).

BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. Steffen Durinck, Yves Moreau, Arek Kasprzyk, Sean Davis, Bart De Moor, Alvis Brazma and Wolfgang Huber, Bioinformatics 21, 3439-3440 (2005).

Lihua Julie Zhu, Herve Pages, Claude Gazin, Nathan Lawson, Jianhong Ou, Simon Lin, David Lapointe and Michael Green (2012). ChIPpeakAnno: Batch annotation of the peaks identified from either ChIP-seq, ChIP-chip experiments or any experiments resulted in large number of chromosome ranges.. R package version 2.4.0.

See Also

`useMart`,`getAnnotation`,`getEnrichedGO`

Examples

```
if(interactive()) { # need internet connection
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

# Parameters setting
binSize <- NULL # automatically determine bin size
minBinSize <- 10000 # min bin size in automatic bin size selection
maxBinSize <- 12000 # max bin size in automatic bin size selection
multicore <- FALSE # use multicore
strandType <- "-" # set strand type to minus strand

biomart <- "ENSEMBL_MART_ENSEMBL" # use archive to get ensembl 65
dataset <- "mmusculus_gene_ensembl" # mouse dataset id name
host <- "dec2011.archive.ensembl.org"  # use ensembl 65 for annotation

goAnno <- "org.Mm.eg.db"


################ run main function for HMM inference on all chromosomes ################
mainSeekOutputRIP <- mainSeek(
    bamFiles=grep(pattern="SRR039214", bamFiles, value=TRUE, invert=TRUE),
binSize=binSize, minBinSize = minBinSize,
maxBinSize = maxBinSize, strandType=strandType,
reverseComplement=TRUE, genomeBuild="mm9",
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = TRUE,
multicore=multicore, silentMain=FALSE, verbose=TRUE)

# use defined binSize from RIP
RIPBinSize <- lapply(mainSeekOutputRIP$nbhGRList, function(x) median(width(x)))


mainSeekOutputCTL <- mainSeek(
    bamFiles=grep(pattern="SRR039214", bamFiles, value=TRUE, invert=FALSE),
binSize=RIPBinSize, strandType=strandType,
reverseComplement=TRUE, genomeBuild="mm9",
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = TRUE,
multicore=multicore, silentMain=FALSE, verbose=TRUE)

################ significance test on Viterbi predicted peaks ################
ripGR <- seekRIP(mainSeekOutputRIP$nbhGRList$chrX, mainSeekOutputCTL$nbhGRList)


################ Annotate peaks ################

annotatedRIPGR <- annotateRIP(sigGRanges = ripGR,
biomaRt_dataset = dataset, goAnno = goAnno,
strandSpecific = !is.null(strandType),
host=host, biomart=biomart)
```

```
head(annotatedRIPGR$sigGRangesAnnotated)
}
```

`binCount` *Count reads in nonoverlapping bins across a chromosome*

Description

Stratify chromosome into nonoverlapping bins of the same size and count the number of reads that fall within each bin.

Usage

```
binCount(alignGR, binSize, returnBinCountOnly = FALSE)
```

Arguments

`alignGR`	GRanges object containing the alignments for a single chromosome.
`binSize`	An integer for the bin size.
`returnBinCountOnly`	Binary indicator. If TRUE, only the integer read count is returned; if FALSE, GRanges of bins with value slot saved as the corresponding read counts is returned.

Details

The funciton is designed to operate to a single chromosome to facilitate parallel computing on multiple chromosomes independently. The function is used in `evalBinSize` to select the optimal bin size based on the read counts and in `mainSeekSingleChrom` to provide the read count as input for the HMM.

Value

If `returnBinCountOnly` is TRUE, then the integer read count is returned; if `returnBinCountOnly` is FALSE, then the GRanges of bins with value slot saved for the corresponding read counts is returned.

Author(s)

Yue Li

References

P. Aboyoun, H. Pages and M. Lawrence. GenomicRanges: Representation and manipulation of genomic intervals. R package version 1.8.9.

See Also

`selectBinSize`,`evalBinSize`

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGRList <- GRangesList(as.list(split(alignGR, seqnames(alignGR))))

binSize <- 1000

binGR <- binCount(alignGRList$chrX, binSize)
```

`combineAlignGals` *Combine alignment files into a single GAlignments object*

Description

Import and process individual BAM/SAM/BED alignment files using `getAlignGal` and combine them into a single GAlignments.

Usage

```
combineAlignGals(bamFiles, ...)
```

Arguments

`bamFiles`	A list of paths to the alignment files.
`...`	Arguments passed to `getAlignGal`.

Details

If there is only one BAM file, then simply return the output from `getAlignGal`; otherwise, all processed alignments are pooled to form a single GAlignments object.

Value

`combinedGal`	GAlignments object containning the (combined) processed alignments with the `values` slot saved for the "uniqueHits" binary flag defined in `getAlignGal` and `metadata` saved as a list containing argument setting for `reverseComplement,returnDuplicate,fl` defined in `getAlignGal`

.

Note

User are recommanded to pool technical replicates but keep biological replicate separate for confirmation.

Author(s)

Yue Li

See Also

getAlignGal,readGAlignments,readGAlignmentPairs,import

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

# combine the alignments for technical replicates
alignGal <-
combineAlignGals(bamFiles=grep(pattern="SRR039214",
                 bamFiles, value=TRUE, invert=TRUE),
                 reverseComplement=TRUE, genomeBuild="mm9")
```

---

combineRIP *Combined predictions from (presumably) biological replicates.*

Description

A simple helper function that combines multiple prediction lists from biological replicates into a single list.

Usage

```
combineRIP(ripPath, pattern="gff3$",
combineOption="intersect",
pvalCutoff=1, pvalAdjCutoff=1, eFDRCutoff=1,
logOddCutoff=-Inf, maxgap=1e3, minIntersect, genomeBuild)
```

Arguments

ripPath	Path to predictions list in a select format as indicated by the file extension.
pattern	Pattern for the names of the prediction files to combine. The file names are expected to have a common extension such as "bed", "gff3", "gtf", but this is not enforced. Default: "gff3$" (i.e. the default output RIPregions.gff3 from ripSeek).

combineOption	Options on *how* to combine the peaks including: "intersect": is selected (default), only peaks in each biological replicate list that overlap with or are adjacent within maxgap nucleotides to at least minIntersect other replicates will be kept. If minIntersect is unspecified, then only the peaks that consistently predicted in all replicates are kept. "merge": All overlapping peaks from the replicates will be merged into one peak. "union": All overlapping peaks from the replicates will be merged into one peak.
pvalCutoff	Threshold for the p-value cutoff. Only peaks with p-value *less* than the logOddCutoff will be reported. Default: 1 (i.e. no cutoff).
pvalAdjCutoff	Threshold for the adjusted p-value cutoff. Only peaks with adjusted p-value *less* than the logOddCutoff will be reported. Default: 1 (i.e. no cutoff).
eFDRCutoff	Threshold for the empirical false discovery rate (eFDR). Only peaks with eFDR *less* than the eFDRCutoff will be reported. Default: 1 (i.e. no cutoff).
logOddCutoff	Threshold for the log odd ratio of posterior for the RIP over the background states (See seekRIP). Only peaks with logOdd score *greater* than the logOddCutoff will be reported. Default: -Inf (i.e. no cutoff).
maxgap	Maximum gap allowed to determine two peaks agree with each other.
minIntersect	Mininum number of replicates required to have peaks either intersect or are adjacent to the peak in other replciate.
genomeBuild	Genome build used to obtain the chromosome information from online UCSC database to assign chromosome length to the GRanges object created as the combined peak list.

Value

gr	GRanges object containning chromosome locations of the combined peaks.

Note

Please run ripSeek first on all biological replicates and renamed each "RIPregions.gff3" output to correspond to different biological replicates and place all of the files into a single folder. The path of this folder can then be used as the input argument for ripPath.

Author(s)

Yue Li

References

P. Aboyoun, H. Pages and M. Lawrence. GenomicRanges: Representation and manipulation of genomic intervals. R package version 1.8.9.

Michael Lawrence, Vince Carey and Robert Gentleman. rtracklayer: R interface to genome browsers and their annotation tracks. R package version 1.16.3.

See Also

combineAlignGals,ripSeek,import,import,reduce,countOverlaps

Examples

```
# Retrieve system files
ripPath <- system.file("extdata/RIPregions", package="RIPSeeker")

gr1 <- combineRIP(ripPath, combineOption="intersect", genomeBuild="mm9")

gr2 <- combineRIP(ripPath, combineOption="merge", genomeBuild="mm9")

gr3 <- combineRIP(ripPath, combineOption="union", genomeBuild="mm9")

length(gr1)

length(gr2)

length(gr3)
```

---

computeLogOdd *Compute the log odd ratio of RIP over background.*

Description

The RIPScore is computed as the log odd ratio of the posterior for the RIP state ($z_i = 2$) over the posterior for the background state ($z_i = 1$)

Usage

```
computeLogOdd(nbhGR)
```

Arguments

nbhGR	GRanges of bins with the value slot saved for the posterior probabilities for the background and RIP state.

Details

To assess the statistical significance of the RIP predictions, we assign each bin a RIPScore defined as the log odd ratio of the posterior for the RIP state ($z_i = 2$) over the posterior for the background state ($z_i = 1$). When control is available, the RIPScore is updated as the difference between the RIPScores evaluated separately for RIP and control libraries. The scoring system captures the model confidence for the RIP state of each bin in the RIP library penalized by the false confidence for the RIP state of the same bin in the control library. In addition, RIPScore obviates scaling of read counts. Since sequencing depth usually differs between RIP and control libraries, scaling is necessary if the statistical score were derived from the read count differences. On the other hand, simplistic linear scaling may distort the data.

Value

A vector of log odd scores for each bin in nbhGR.

Author(s)

Yue Li

See Also

seekRIP,scoreMergedBins,logScoreWithoutControl,logScoreWithControl

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGRList <- GRangesList(as.list(split(alignGR, seqnames(alignGR))))

################ run main function for HMM inference on a single chromosome ################
nbhGR <- mainSeekSingleChrom(alignGR=alignGRList$chrX, K = 2, binSize=1e5)

ripscore <- computeLogOdd(nbhGR)
```

---

computeRPKM *Compute RPKM based on gene annotations*

Description

Given a list of single-end or paired-end read alignment files in BAM/SAM/BED format, compute the read counts and normalized read counts as expression of annotated transcript in the unit of "reads per kilobase of exon per million mapped reads" (RPKM).

Usage

```
computeRPKM(bamFiles, RIPSeekerRead = TRUE, paired = FALSE,
countMode = "IntersectionNotEmpty", featureGRanges,
idType = "ensembl_transcript_id", featureType = "exon",
ignore.strand = FALSE, txDbName = "biomart",
moreGeneInfo = FALSE, saveData, justRPKM = TRUE, ...)
```

Arguments

bamFiles	A list of one or more BAM/SAM/BED alignment files.
RIPSeekerRead	Binary flag. If TRUE, then import and process the alignment files using the built-in function combineAlignGals from RIPSeeker package; if FALSE, then import the files by directly calling the required functions. The flag makes using the function outside of RIPSeeker package become possible.
paired	Binary to indicate whether the alignments files are paired-end. The alignments file must be either paired-end or single-end but not both.
countMode	An argument used to set the mode argument in the underlying function summarizeOverlaps employed to compute the read counts for each feature. The possible mode includes "Union", "IntersectionStrict", and "IntersectionNotEmpty". All three modes avoid double counting the reads by either discarding reads that completely fall into multiple features or counting the read only once for the feature that uniquely and completely includes it. Please refer to summarizeOverlaps for details.
featureGRanges	GRanges of features as an optional argument for function to compute RPKM/FPKM just for those features without retrieving online annotations.
idType	A character string that specifies the type of the annotations, which can "ensembl_transcript_id", "ensembl_gene_id", "ucsc", etc. Refer to listFilters for more information.
featureType	Features that will be groupped by genes/transcripts in a GRangesList. The available options are "exon" (Default), "intron", "fiveUTR", "threeUTR", and "CDS" corresponding to the functions exonsBy, cdsBy, intronsByTranscript, fiveUTRsByTranscript, threeUTRsByTranscript, and cdsBy, respectively.
ignore.strand	Whether to ignore strand when counting the reads (Default: FALSE).
txDbName	Name of the transcript database to use to retreive the annotation. The available options are "biomart" (Default) or "UCSC" corresponding to the functions makeTxDbFromBiomart and makeTxDbFromUCSC, respectively.
moreGeneInfo	Binary indicator to indicate whether to download more information for each genes/transcripts rather than having only the gene/transcript IDs (Default: FALSE).
saveData	Path of output file.
justRPKM	Binary for whether to return only the RangedSummarizedExperiment.
...	Extra arguments passed to functions makeTxDbFromBiomart, makeTxDbFromUCSC, useMart, combineAlignGals.

Details

The function is a wrapper function making use of several external functions from several well maintained and freely available Bioconductor packages including GenomicFeatures, GenomicRanges, biomaRt and Rsamtools packages. The paired-end alignments are converted into single-end using function galp2gal and then subject to read count computation by summarizeOverlaps, which does not yet directly support paired-end alignments.

Value

rpkmSEobject	A RangedSummarizedExperiment object with assays slot saved for counts, rowRanges holds the features, metadata for RPKM/FPKM (normalized) gene expression.

rpkmDF	Data frame with or without the detailed gene information columns depending on whether moreGeneInfo is TRUE or FALSE. rpkmDF is only returned within in a list when justRPKM is FALSE.
featureGRanges	The features in GRanges object that are used to compute the gene expression. featureGRanges is only returned within in a list when justRPKM is FALSE.

Note

Also works for RNA-seq alignments.

Author(s)

Yue Li

References

M. Carlson, H. Pages, P. Aboyoun, S. Falcon, M. Morgan, D. Sarkar and M. Lawrence. GenomicFeatures: Tools for making and manipulating transcript centric annotations. R package version 1.8.2.

P. Aboyoun, H. Pages and M. Lawrence (). GenomicRanges: Representation and manipulation of genomic intervals. R package version 1.8.9.

Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. Steffen Durinck, Paul T. Spellman, Ewan Birney and Wolfgang Huber, Nature Protocols 4, 1184-1191 (2009).

BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. Steffen Durinck, Yves Moreau, Arek Kasprzyk, Sean Davis, Bart De Moor, Alvis Brazma and Wolfgang Huber, Bioinformatics 21, 3439-3440 (2005).

Martin Morgan and Herv\'e Pag\'es (). Rsamtools: Binary alignment (BAM), variant call (BCF), or tabix file import. R package version 1.8.5. http://bioconductor.org/packages/release/bioc/html/Rsamtools.html

See Also

makeTxDbFromBiomart,makeTxDbFromUCSC,useMart,exonsBy,cdsBy,intronsByTranscript,fiveUTRsByTranscr

Examples

```
if(interactive()) { # need internet connection

# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

# use biomart
txDbName <- "biomart"
biomart <- "ENSEMBL_MART_ENSEMBL" # use archive to get ensembl 65
dataset <- "mmusculus_gene_ensembl"
host <- "dec2011.archive.ensembl.org"  # use ensembl 65 for annotation

resultlist <- computeRPKM(bamFiles=grep(pattern="SRR039214",
        bamFiles, value=TRUE, invert=TRUE), #featureGRanges=featureGRanges,
```

```
dataset=dataset, moreGeneInfo=TRUE, justRPKM=FALSE,
idType="ensembl_transcript_id", txDbName=txDbName,
biomart=biomart, host=host, by="tx")

}
```

`disambiguateMultihits` *Assign each multihit to a unique region based on the posterior for the read-enriched hidden state*

Description

Among multiple alignments of the same read (i.e. multihit), select the alignment corresponding to the bin with the maximum posterior for the enriched state.

Usage

```
disambiguateMultihits(alignGal, nbhGRList, postprobCutoff = 0)
```

Arguments

`alignGal`	GAlignments object with an additional column in the `values` slot that indicates whether the read corresponding to the current alignment is a unique hit (i.e., read mapped uniquely to a single loci) or multihit (i.e., read mapped to multiple loci).
`nbhGRList`	GRangesList each item containig the HMM training results on a single chromosome. Importantly, the posterior probabilities for the background and enriched states need to be present the `metadata` slot and used to disambiguate multihits, which is done by `mainSeekSingleChrom`.
`postprobCutoff`	Posterior cutoff for returning only the reads with maximum posterior that is greater than the threshold (Default: 0; i.e., no cutoff).

Details

Each multihit (i.e., read aligned to multiple loci) flagged in the `getAlignGal` function are assigned to a unique locus corresponding to the j^th bin with the highest posterior or responsibility from the RIP state. Intuitively, the RIP state corresponds to the read-enriched loci. Disambiguating multihits in this way will potentially improve the power of detecting more RIP regions but may also introduce certain bias towards the idea of "rich gets richer". After this step, RIPSeeker will rerun the functions from `selectBinSize` to `nbh` to improve the HMM model estimation with augmented read count data. Optionally, user can choose not to reiterate the training process to go straight to the next step to detect RIP regions (See `seekRIP`).

Value

GAlignments with each read mapped uniquely to a single locus.

Author(s)

Yue Li

See Also

getAlignGal,ripSeek,mainSeek,mainSeekSingleChrom

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

# Parameters setting
binSize <- 1e5   # use a large fixed bin size for demo only
minBinSize <- NULL # turn off min bin size in automatic bin size selection
maxBinSize <- NULL # turn off max bin size in automatic bin size selection
multicore <- FALSE # use multicore
strandType <- "-"   # set strand type to minus strand

# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- combineAlignGals(bamFiles=grep(pattern="SRR039214",
          bamFiles, value=TRUE, invert=TRUE), reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGR <- addPseudoAlignment(alignGR)

alignGRList <- GRangesList(as.list(split(alignGR, seqnames(alignGR))))

################ run mainSeekSingleChrom function for HMM inference on a single chromosome ################
nbhGRList <- lapply(alignGRList, mainSeekSingleChrom, K = 2, binSize=binSize,

minBinSize = minBinSize, maxBinSize = maxBinSize, runViterbi=FALSE)

nbhGRList <- GRangesList(nbhGRList)

alignGalFiltered <- disambiguateMultihits(alignGal, nbhGRList)
```

---

empiricalFDR *Compute empirical false discovery rate*

Description

At a p-value, find the number of regions in RIP library (denoted as "trueCount") and the number of regions in control library (denoted as "falseCount"). The empirical false discovery rate (eFDR) is estimated as the ratio of the falseCount over the trueCount.

Usage

```
empiricalFDR(pval, pvalRIP, pvalCTL)
```

Arguments

pval	A scalar p-value.
pvalRIP	A column vector of p-values for the peaks identifed from RIP v.s. control comparison.
pvalCTL	A column vector of p-values for the peaks identifed from control v.s. RIP comparison.

Details

Only when the control is available, is an empirical false discovery rate (eFDR) estimated based on the idea of "sample swap" inspired by MACS (a ChIP-seq algorithm from Zhange *el al.* (2008). At each p-value, RIPSeeker finds the number of significnat RIP-regions over control (CTL) based on pvalRIP and the number of significant control regions over RIP based on pvalCTL. The eFDR is defined as the ratio of the number of "RIP" (false positive) regions identified from CTL-RIP comparison over the number of RIP regions from the RIP-CTL comparison. The maximum value for eFDR is 1 and minimum value for eFDR is max(p-value, 0). The former takes care of the case where the numerator is bigger than the denominator, and the latter for zero numerator.

Value

A scalar probabibility value that represents the eFDR.

Note

This is an internal funciton used in seekRIP.

Author(s)

Yue Li

References

Yong Zhang, Tao Liu, Clifford A Meyer, J\'er\^ome Eeckhoute, David S Johnson, Bradley E Bernstein, Chad Nusbaum, Richard M Myers, Myles Brown, Wei Li, and X Shirley Liu. Model-based analysis of ChIP-Seq (MACS). Genome Biology, 9(9):R137, 2008.

See Also

logScoreWithControl, seekRIP, computeLogOdd, scoreMergedBins

Examples

```
pvalRIP <- runif(100)

pvalCTL <- runif(100)

eFDR <- empiricalFDR(pvalRIP[1], pvalRIP, pvalCTL)

pvalRIP[1]
```

```
eFDR

# more significant pval
pvalRIP[1] <- 1e-4

eFDR <- empiricalFDR(pvalRIP[1], pvalRIP, pvalCTL)

pvalRIP[1]

eFDR
```

`evalBinSize` *Evaluate bin size using Shimazaki cost function*

Description

Given a bin size and a GRanges alignment object, the funciton computes the bin count and returns the cost of the bin size based on Shimazaki cost function.

Usage

```
evalBinSize(binSize, alignGR)
```

Arguments

`binSize`	An integer that indicates the bin size applied to the binning of the chromosome.
`alignGR`	GRanges object of alignments to a single chromosome.

Details

The function implements the algorithm developed by Shimazaki and Shinomoto (2007), which is based on the goodness of the fit of the time histogram to estimate the rate of neural response of an animal to certain stimuli in a spike-in experiment. The algorithm involves four simple steps:

1. Divide chromosome sequence into N bins of width b.
2. Count number of read counts x_i that enter the i'th bin.
3. Compute: $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $v = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$.
4. Compute: $C(b) = \frac{2\bar{x}-v}{b^2}$

Value

`cost`	A scalar value for the cost of the bin size.

Author(s)

Yue Li

References

Hideaki Shimazaki and Shigeru Shinomoto. A method for selecting the bin size of a time histogram. Neural computation, 19(6):1503-1527, June 2007.

See Also

selectBinSize,binCount

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGRList <- GRangesList(as.list(split(alignGR, seqnames(alignGR))))

binSize <- 1000

costs <- evalBinSize(binSize, alignGRList$chrX)
```

---

exportGRanges *Export GRanges object in a specified format*

Description

A wrapper function of export with additional support for exporting tab-delimted format with no re-arrangement of the original GRanges output.

Usage

```
exportGRanges(gRanges, outfile, exportFormat)
```

Arguments

gRanges	GRanges object to export.
outfile	File path for output.
exportFormat	Desirable format including "txt" and other formats specified in export.

Value

Output the text to the file stream defined in outfile.

Note

The function is used in ripSeek to export desired format and can be used as general purpose function.

Author(s)

Yue Li

References

Michael Lawrence, Vince Carey and Robert Gentleman. rtracklayer: R interface to genome browsers and their annotation tracks. R package version 1.16.3.

See Also

export

Examples

```
if(interactive()) { # need permission to write to the current dir
gr <-
    GRanges(seqnames =
            Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
            ranges =
            IRanges(1:10, width = 10:1, names = head(letters,10)),
            strand =
            Rle(strand(c("-", "+", "*", "+", "-")),
                c(1, 2, 2, 3, 2)),
            score = 1:10,
            GC = seq(1, 0, length=10))

outfile <- paste(getwd(), "/gr.txt", sep="/")

exportGRanges(gr, outfile=outfile, exportFormat="txt")
}
```

---

galp2gal *Convert GAlignmentPairs to GAlignments*

Description

Convert GAlignmentPairs to GAlignments using CIGAR to mark flanked portion of the pairs as 'N'.

Usage

```
galp2gal(galp)
```

Arguments

galp

Details

Each proper read pairs is combined into a single alignment record making use of the CIGAR flag 'N' to indicate the number of bases between the mate pairs (i.e., the difference between the start of the right mate pair and the end of the left mate pair). In other words, the paired-end alignments are treated as gapped alignments of long fragments. The function is used within getAlignGal but can be used as a stand-alone function as well.

Value

gal — GAlignments object containing for each paired alignments a single alignment record.

Author(s)

Yue Li

References

P. Aboyoun, H. Pages and M. Lawrence. GenomicRanges: Representation and manipulation of genomic intervals. R package version 1.8.9.

See Also

getAlignGal,combineAlignGals,readGAlignments,readGAlignmentPairs,import

Examples

```
library(Rsamtools)

extdata.dir <- system.file("extdata", package="RIPSeeker")

ex1_file <- list.files(extdata.dir, "ex1.bam", recursive=TRUE, full.names=TRUE)

galp <- readGAlignmentPairs(ex1_file, use.names=TRUE)

galp

gal <- galp2gal(galp)

gal
```

---

getAlignGal *Import and processs in BAM/SAM/BED format*

Description

Import and process single-end or paired-end alignments in a BAM/SAM/BED file to retain valid alignments defined by the arguments below. Multihits (same read mapped to multiple loci) are flagged for the subsequent disambiguation with function disambiguateMultihits). The final output is a GAlignments object.

Usage

```
getAlignGal(alignFilePath, format, genomeBuild,
deleteGeneratedBAM = FALSE, reverseComplement = FALSE,
returnDuplicate = FALSE, flagMultiHits = TRUE,
returnOnlyUniqueHits = FALSE, paired = FALSE, ...)
```

Arguments

`alignFilePath`	Path to the alignment file.
`format`	The alignmnet format can be determiend automatically from the file extension or specified by the user. The supported formats are BAM, SAM, and BED.
`genomeBuild`	Genome build used to obtain the chromosome information from online UCSC database in order to construct GAlignments object. Since the BAM/SAM header provides the chromosome information, the argument needs to be set only in the absence of the header information for some BAM/SAM files or when BED file is used. Examples for the common `genomeBuild` are "mm9" for mouse or "hg19" for human reference genomes. Note that an appropriate genome build that has been used in the alignment is important for desirable outcome. For instance, user should use "mm10" if the alignments are based on "mm10" rather than "mm9" genome build.
`deleteGeneratedBAM`	Binary indicator to indicate whether the converted BAM from the original SAM input file needs to be deleted from the local disk (Default: FALSE).
`reverseComplement`	Binary indicator to indicate whether the reads were sequenced from the opposite strand of the original RNA molecule. `reverseComplement` only applies to strand-specific sequencing in which case only the strand generated during *second strand synthesis* is sequenced. Thus, if `reverseComplement=TRUE`, the strand signs of the alignments are switched (i.e. + to -, - to +, and * unchanged); otherwise (`reverseComplement=FALSE`) retian the original the strand signs.
`returnDuplicate`	Indicator (TRUE, FALSE, NA) to instruct whether the duplicate alignmnets need to be returned (Default: FALSE). Duplicate reads are a set of reads that align to exactly the same genomic coordinate. Because transcripts are usually hundreds or thousands of base pairs long and thus much longer than the read (25-100 nt), the chance that the same 25-100 nt portion of the transcript being sequenced multiple times is very small and may very likely be due to PCR artifact. This argument is acutally passed to 'isDuplicate' in `scanBamFlag`.
`flagMultiHits`	Binary indicator for whether to add additional binary column named "uniqueHits" to indicate whether the corresponding aligned reads are unique hit (uniqueHits==TRUE) or multihit (uniqueHits==FALSE). Multihits represent multiple alignments of the same read due to gene duplications or repetitive elements of the genome. The multhits typically constitute a substantial proportion of the total mapped reads. Rather than being removed, these multihits are flagged (`flagMultiHits=TRUE` by default) and in the later step assigned to a unique region by (`disambiguateMultihits`).
`returnOnlyUniqueHits`	Binary indicator to return only the unique hits and discard all of the multihits (Default: FALSE).
`paired`	Binary indicator to indicate whether the alignments are paired-end (Default: FALSE). For paired-end alignments, properly paired reads are combined into a single alignment record making use of the CIGAR flag 'N' to indicate the number of bases between the mate pairs (i.e., the length of the insert fragment). In other words, the paired-end alignments are treated as gapped alignments of long fragments (See `galp2gal`).
`...`	Extra arguments are ignored.

Details

The BAM file is imported using readGAlignments for single-end or readGAlignmentPairs for paired-end alignments. The SAM file is converted to BAM first and then imported as above. The BED file is first imported by import as GRanges object and subsequently converted to GAlignments via the constructor function GAlignments.

Value

alignGal	GAlignments object containning the processed alignments with the values slot saved for the "uniqueHits" binary flag (See flagMultiHits above) and metadata saved as a list containing argument setting for reverseComplement,returnDuplicate,flagMultiH

Author(s)

Yue Li

References

P. Aboyoun, H. Pages and M. Lawrence. GenomicRanges: Representation and manipulation of genomic intervals. R package version 1.8.9.

Michael Lawrence, Vince Carey and Robert Gentleman. rtracklayer: R interface to genome browsers and their annotation tracks. R package version 1.16.3.

See Also

combineAlignGals,readGAlignments,readGAlignmentPairs,import

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")
```

---

logScoreWithControl *Compute RIPScore based on RIP and control posteriors and test for significance*

Description

Compute the RIPScore using both RIP and control posteriors for each bins, merge and summarize the scores for the merged bins, and finally compute the p-value and adjusted p-value for the summary RIPScore.

Usage

```
logScoreWithControl(nbhGRRIP, nbhGRCTL, padjMethod = "BH", getControlStats = TRUE)
```

Arguments

nbhGRRIP	GRanges object for the RIP library created from mainSeek containing the posterior probabilities of the hidden states for each observed read count.
nbhGRCTL	An optional arugment as a GRanges object for the control library created from mainSeek containing the posterior probabilities of the hidden states for each observed read count.
padjMethod	Method used to adjust multiple testing performed in p.adjust (Default: "BH").
getControlStats	Binary indicator to whether return statistics including computed for the control library alone. If TRUE, then all control specific score columns will be reported with a prefix "CTL".

Details

The RIPScore is compupted in computeLogOdd as the log odd ratio of the posterior for the RIP state ($z_i = 2$) over the posterior for the background state ($z_i = 1$) in RIP library subtracted by the log odd ratio computed from the control library. The adjacent bins with hidden states predicted by nbh_vit as the enriched state (corresponding to the NB with larger mean) are merged. The RIPSscores are averaged over the merged bins. To assess the statistical significance of the RIPScore for each region, we assume that the RIPScore follows a *Gaussian* (Normal) distribution with mean and standard deviation estimated using the RIPScores over all of the bins. The rationale is based on the assumption that most of the RIPScores correspond to the background state and together contribute to a stable estimate of the test statistics (TS) and p-value computed using the R built-in function pnorm. The p-value is adjusted by p.adjust with BH method by default. The same procedure is applied optionally to the control library.

Value

GRanges of merged bins with values slot saved for RIPScore (lodOdd), p-value (pval), adjusted p-value (pvalAdj) for RIP and optionally for control.

Note

Internal function used by seekRIP.

Author(s)

Yue Li

See Also

logScoreWithoutControl, seekRIP, computeLogOdd, scoreMergedBins

Examples

```
if(interactive()) { # check the example in seekRIP
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)
```

```
# Parameters setting
binSize <- 1e5 # use a large fixed bin size for demo only
multicore <- FALSE # use multicore
strandType <- "-" # set strand type to minus strand

################ run main function for HMM inference on all chromosomes ################
mainSeekOutputRIP <-
    mainSeek(bamFiles=grep(pattern="SRR039214",
    bamFiles, value=TRUE, invert=TRUE),
binSize=binSize, strandType=strandType,
reverseComplement=TRUE, genomeBuild="mm9",
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = FALSE,
multicore=multicore, silentMain=FALSE, verbose=TRUE)

mainSeekOutputCTL <- mainSeek(bamFiles=grep(pattern="SRR039214",
    bamFiles, value=TRUE, invert=FALSE),
binSize=binSize, strandType=strandType,
reverseComplement=TRUE, genomeBuild="mm9",
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = FALSE,
multicore=multicore, silentMain=FALSE, verbose=TRUE)


################ Compute log score and test for significance WITH control ################
ripGR.wicontrol <- logScoreWithControl(mainSeekOutputRIP$nbhGRList$chrX, mainSeekOutputCTL$nbhGRList$chrX)

ripGR.wicontrol
}
```

`logScoreWithoutControl` *Compute RIPScore based on RIP posteriors alone and test for significance*

Description

Compute the RIPScore using only the RIP (typically when control is unavailable) posteriors for each bins, merge and summarize the scores for the merged bins, and finally compute the p-value and adjusted p-value for the summary RIPScore.

Usage

```
logScoreWithoutControl(nbhGRRIP, padjMethod = "BH")
```

Arguments

`nbhGRRIP`	GRanges object for the RIP library created from `mainSeek` containing the posteriors probabilities of the hidden states for each observed read count.
`padjMethod`	Method used to adjust multiple testing performed in `p.adjust` (Default: "BH").

Details

The RIPScore is compupted in computeLogOdd as the log odd ratio of the posterior for the RIP state ($z_i = 2$) over the posterior for the background state ($z_i = 1$) in RIP library alone (typically when control is unavailable). The adjacent bins with hidden states predicted by nbh_vit as the enriched state (corresponding to the NB with larger mean) are merged. The RIPSscores are averaged over the merged bins. To assess the statistical significance of the RIPScore for each region, we assume that the RIPScore follows a *Gaussian* (Normal) distribution with mean and standard deviation estimated using the RIPScores over all of the bins. The rationale is based on the assumption that most of the RIPScores correspond to the background state and together contribute to a stable estimate of the test statistics (TS) and p-value computed using the R built-in function pnorm. The p-value is adjusted by p.adjust with BH method by default.

Value

GRanges of merged bins with values slot saved for RIPScore (lodOdd), p-value (pval), adjusted p-value (pvalAdj) for RIP

Note

Internal function used by seekRIP.

Author(s)

Yue Li

See Also

logScoreWithControl,seekRIP,computeLogOdd,scoreMergedBins

Examples

```
if(interactive()) { # check the example in seekRIP
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

# Parameters setting
binSize <- 1e5 # use a large fixed bin size for demo only
multicore <- FALSE # use multicore
strandType <- "-" # set strand type to minus strand

################ run main function for HMM inference on all chromosomes ################
mainSeekOutputRIP <- mainSeek(bamFiles=grep(pattern="SRR039214",
    bamFiles, value=TRUE, invert=TRUE),
binSize=binSize, strandType=strandType,
reverseComplement=TRUE, genomeBuild="mm9",
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = FALSE,
multicore=multicore, silentMain=FALSE, verbose=TRUE)


################ Compute log score and test for significance WITHOUT control ################
```

```
ripGR.wocontrol <- logScoreWithoutControl(mainSeekOutputRIP$nbhGRList$chrX)
}
```

`mainSeek` *Train HMM paramters on each chromosome independently from the alignments.*

Description

A back-end function used by the front-end function `ripSeek` to train HMM paramters on all of the chromosomes indepdently. This function in turn calls another function `mainSeekSingleChrom` to compute HMM paramters on each chromosome separately or in parallel (if multicore is TRUE).

Usage

```
mainSeek(bamFiles, reverseComplement = FALSE,
genomeBuild = "mm9", uniqueHit = TRUE,
assignMultihits = TRUE, strandType = NULL,
  paired=FALSE, rerunWithDisambiguatedMultihits = TRUE,
  silentMain = FALSE, multicore = TRUE,
  returnAllResults = TRUE, ...)
```

Arguments

`bamFiles`	A list of paths to individual BAM files. BED and SAM files are also accepted.
`reverseComplement`	Whether the reads came from the original or the opposite strand of the RNA being sequenced. If former, then `reverseComplement` should be FALSE; otherwise TRUE, in which case the strand signs will be switched from + to -, - to +, and * is unchanged.
`genomeBuild`	When the input alignment format is BED, genomeBuild is only required in `getAlignGal` to determine the chromosome lengths for the GAlignments obejct using function `SeqinfoForUCSCGenome`. BAM and SAM header have chromosome information, and thus genomeBuild is not needed.
`uniqueHit`	Binary indicator. If uniqueHit=TRUE, only reads mapped to single unique loci are used to train the HMM. Otherwise, all of the reads including multihits will be used for the HMM. A multihit is a read mapped to more than one loci. The flags for uniqueHits and multihits are the metadata values of GAlignments object constructed in `getAlignGal`.
`assignMultihits`	Binary indicator used by ripSeek to tell the function whether disambiguate multihits by assigning them to unique loci with the maximum posterior probability obtained from running HMM (See `nbh_em`)
`strandType`	A character variable indicate which strand the RIPSeeker needs to operate on. The options are NULL, '+', '-', '*'. If NULL or '*', then all of the reads will be used (preferable for non-strand specific sequencing). If '+' or '-', only reads from '+' or '-' strand will be used, respectively. Note that the sign is assumed to be THE SAME AS the strand sign of the processed alignment object and will be the opposite sign if reverseComplement is TRUE (See `reverseComplement` above).

paired — Binary to indicate whether the library is paired-end (TRUE) or single-end (FALSE by default) (see getAlignGal).

rerunWithDisambiguatedMultihits — After multihits have been asigned to unique loci, rerunWithDisambiguatedMultihits (Default: TRUE) indicates whether to re-run the HMM on the augmented read alignmnet data. If FALSE, the HMM step will not be re-run, and the workflow will proceed to RIP detection (See seekRIP) using the nondisambiguated alignments, which can either be the alignments containing only the uniqueHits (if uniqueHit=TRUE) or the alignments containing both the uniqueHits and multiHits (if uniqueHit=FALSE).

silentMain — Binary indicator to indicate whether to disable the verbose from the mainSeekSingleChrom function. If FALSE (by default), the EM training process will be output to the console for user to keep track of the training progress.

multicore — Binary indicator to indicate whether to use mclapplyfunction to compute HMM on chromosomes in parallel. The multicore function will speed up the computation by a factor proportional to the total number of CPU cores on the machine but may impose larger memory overhead than the singe-threading approach.

returnAllResults — Binary indicator to indicate whether to return all (HMM trained parameters, original, and disambiguated GAlignments) or just the HMM results.

... — Arguments passed to mainSeekSingleChrom.

Value

A list containing:

nbhGRList — GRangesList each item containig the HMM training results on a single chromosome.

alignGal — Original alignment data in GAlignments object

alignGalFiltered — Disambiguated alignmnet data with multihits assigned to unique loci.

Author(s)

Yue Li

See Also

ripSeek,mainSeekSingleChrom,mclapply

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

# Parameters setting
binSize <- 1e5   # use a large fixed bin size for demo only
minBinSize <- NULL # min bin size in automatic bin size selection
maxBinSize <- NULL # max bin size in automatic bin size selection
```

```
multicore <- FALSE # use multicore
strandType <- "-" # set strand type to minus strand

################ run mainSeekSingleChrom function for HMM inference on all chromosomes ################
mainSeekOut <- mainSeek(bamFiles=grep(pattern="SRR039214",
    bamFiles, value=TRUE, invert=TRUE),
binSize=binSize, minBinSize = minBinSize,
maxBinSize = maxBinSize, strandType=strandType,
reverseComplement=TRUE, genomeBuild="mm9",
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = FALSE,
multicore=multicore, silentMain=FALSE, verbose=TRUE)
```

`mainSeekSingleChrom` *Automatic bin size selection, bin count, and HMM parameters optimization on read count vector from a single chromosome (Internal function)*

Description

This an internal function used by `mainSeek` to accomplish three major tasks *on a single chromosome*: automatically select bin size, compute read counts within the bins, and obtain optimal HMM paramters.

Usage

```
mainSeekSingleChrom(alignGR, K = 2, binSize = NULL, minReadCount = 10,
backupNumBins = 10, minBinSize = 200, maxBinSize = 1200,
increment = 5, pathToSavePlotsOfBinSizesVersusCosts,
verbose = TRUE, allowSecondAttempt = TRUE, ...)
```

Arguments

`alignGR`	GRanges containing the alignments on a single chromosome .
`K`	Number of hidden states (Default: 2). By default, state 1 specifies the background and state 2 the RIP regions. The two states are recognized by the means for the two distributions (See `nbh_em`).
`binSize`	Size to use for binning the read counts across each chromosome. If NULL, optimal bin size within a range (default: minBinSize=200, maxBinSize=1200) will be automatically selected (See `selectBinSize`).
`minReadCount`	Minimum aligned read counts needed for HMM to converge (Default: 10). Note that HMM may not converge some times when majority of the read counts are zero even if some read count > 10. When that happens, a back-up function `addDummyProb` comes in to create a placeholder for the corresponding chromosome in GRangeList to maintain the data structure to preserve all information (successfully) obtained from other chromosomes.
`backupNumBins`	If read count is less than `minReadCount`, then use `backupNumBins` (Default: 10) to bin the chromosome.

`minBinSize` Minimum bin size to start with the bin selection (See `selectBinSize`). Default to 200, common minimum band size selected in RIP or RNA-seq library construction.

`maxBinSize` Maximum bin size to stop with the bin selection (See `selectBinSize`). Default: 1200.

`increment` Step-wise increment in bin size selection (See `selectBinSize`). Default: 5.

`pathToSavePlotsOfBinSizesVersusCosts`
Directory used to save the diagnostic plots for bin size selection.

`verbose` Binary indicator for disable (FALSE) or enable (TRUE) HMM training message from function `nbh` to output to the console.

`allowSecondAttempt`
In case HMM fails to converge due to malformed paramters in EM iteraction, re-iterating the HMM process each time with a different suboptimal bin size in attempt to succeed in some trial. If all yeild nothing, fall back up to `addDummyProb` to return the place holder for the chromosome.

`...` Argumnets passed to `nbh`.

Value

`nbhGR` GRanges object containing the optimized HMM parameters (and the Viterbi hidden state sequence) accompanied with the read count vector following the (automatic) binning scheme.

Note

Unless a highly customized workflow is needed, `ripSeek` is the high-level front-end main function that should be used in most cases.

Author(s)

Yue Li

See Also

`ripSeek`,`mainSeek`,`nbh_em`

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

# Parameters setting
binSize <- 1e5   # use a large fixed bin size for demo only
minBinSize <- NULL # min bin size in automatic bin size selection
maxBinSize <- NULL # max bin size in automatic bin size selection
multicore <- FALSE # use multicore
strandType <- "-" # set strand type to minus strand

# Retrieve system files
```

```
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGRList <- GRangesList(as.list(split(alignGR, seqnames(alignGR))))

################ run main function for HMM inference on a single chromosome ################
nbhGR <- mainSeekSingleChrom(alignGR=alignGRList$chrX, K = 2, binSize=binSize,

minBinSize = minBinSize, maxBinSize = maxBinSize)

nbhGR
```

---

nbh *Generic function of negative binomial HMM*

Description

Generic function for nbh.GRanges and nbh.integer

Usage

```
nbh(x, ...)
```

Arguments

x	Object of class Integer or GRanges.
...	Extra arguments passed to either nbh.GRanges or nbh.integer.

Author(s)

Yue Li

References

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition (Vol. 77, pp. 257-286). Presented at the Proceedings of the IEEE. doi:10.1109/5.18626

Bishop, Christopher. Pattern recognition and machine learning. Number 605-631 in Information Science and Statisitcs. Springer Science, 2006.

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

mainSeekSingleChrom,nbh.integer,nbh.GRanges

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGRList <- GRangesList(as.list(split(alignGR, seqnames(alignGR))))

binSize <- 1e5  # use a large fixed bin size for demo only

binGR <- binCount(alignGRList$chrX, binSize)

# test on GRanges object
nbhGR <- nbh(binGR, 2, runViterbi=TRUE)

# test on integer object
nbhList <- nbh(values(binGR)$count, 2, runViterbi=TRUE)
```

`nbh.GRanges` *Optimize HMM parameters based on the read counts on a chromosome*

Description

Inheritance function from `nbh` that receives an object of GRanges class with additional column of read counts (for each strand) and call `nbh.integer` to derive the most probable sequence of hidden states

Usage

```
## S3 method for class 'GRanges'
nbh(x, K, ...)
```

Arguments

`x`	GRanges with 'values' slot used for bin counts in 1D vector of integers.
`K`	Number of hidden states.
`...`	Extra arguments passed to nbh.integer for the actual HMM computation.

Value

binGR	GRanges of bin counts with metadata slot saved for the optimized HMM parameters including alpha, beta for the K negative binomial mixture components and TRANS (the transition probabilities)

Author(s)

Yue Li

References

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition (Vol. 77, pp. 257-286). Presented at the Proceedings of the IEEE. doi:10.1109/5.18626

Bishop, Christopher. Pattern recognition and machine learning. Number 605-631 in Information Science and Statisitcs. Springer Science, 2006.

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

mainSeekSingleChrom,nbh.integer

Examples

```
if(interactive()) ?nbh # see nbh for example of nbh running on GRanges object
```

---

nbh.integer *HMM posterior decoding and NB parameter optimization*

Description

Inherithance function from nbh that receives a vector of integers and compute optimal HMM parameters via EM algorithm.

Usage

```
## S3 method for class 'integer'
nbh(x, K, NBM_NIT_MAX = 250,
NBM_TOL = 0.01, NBH_NIT_MAX = 250,
NBH_TOL = 0.001, runViterbi = FALSE, ...)
```

Arguments

x	A vector of integers, conceptaully representing the read counts within bins of chromosome.
K	Number of hidden states.
NBM_NIT_MAX	Maximum number of EM iterations (Default: 250) for the negative binomial mixture model (NBM) intialization step (See nbm_em).

NBM_TOL	Threshold as fraction of increase in likelihood (given the current NBM parameters) comparing with the likelihood from the last iteration. EM for the NBM stops when the improvement is below the threshold (Default: 0.01).
NBH_NIT_MAX	Maximum number of EM iterations (Default: 250) for the negative binomial hidden Markov model (NBH).
NBH_TOL	Threshold as fraction of increase in likelihood (given the current NBH parameters) comparing with the likelihood from the last iteration. EM for the NBH stops when the improvement is below the threshold (Default: 0.001).
runViterbi	Binary indicator. If TRUE, Viterbi algorithm will be applied to derive the maximum likelihood hidden state sequence using the optimized HMM paramters obtained from the EM (See nbh_em).
...	Extra arguments are ignored.

Details

The function consists of three major steps: (1) negarive binomail mixture model used to initialized HMM parameters; (2) optimization of HMM paramters using EM algorithm; (3) Viterbi maximum-liklihood estimation of hidden state sequence. Step (1) involves optimization of NBM parameters assuming the data points are independently sampled from a mixture of K NB distributions (See nbh_init). Given the optimized paramters for K-NBM, step (2) drops the independence assumption by introducing the transition probibility between hidden variables, which is initlaized as the mixing proportions of NBM (See nbh_init). Given the optimized HMM paramters, step (3) derives the maximum liklihood hidden state sequence using Viterbi algorithm. Step (3) is run only when runViterbi is TRUE.

Value

A list containing:

initAlpha	Initialized alpha of NBM from nbh_init.
initBeta	Initialized beta of NBM from nbh_init.
initTRANS	Initialized mixing proportion of NBM from nbh_init.
postprob	Posteriors of the K hidden states for each observed count derived from nbh_em (e.g., posteriors of background and enriched state in a two-state HMM).
alpha	Optimized alpha of the NB mixture components in the HMM using nbh_em.
TRANS	Optimized transition probability of the HMM using nbh_em.
viterbi_state	Sequence of discrete values representing the hidden states derived from the maxmium likelihood estimation using Viterbi algorithm (See nbh_vit).

Author(s)

Yue Li

References

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition (Vol. 77, pp. 257-286). Presented at the Proceedings of the IEEE. doi:10.1109/5.18626

Bishop, Christopher. Pattern recognition and machine learning. Number 605-631 in Information Science and Statisitcs. Springer Science, 2006.

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

mainSeekSingleChrom,nbh,nbh.GRanges

Examples

```
if(interactive()) ?nbh # see nbh for example of nbh running on integer object
```

---

nbh_chk *Check the parameters of the negative binomial HMM*

Description

The function verifies the numerical range and dimension of the NBH paramters alpha, beta, and TRANS and returns the number of hidden states. It is used in nbh_em before running EM.

Usage

```
nbh_chk(TRANS, alpha, beta)
```

Arguments

TRANS	Expected a squared matrix of probabilities ($0 \leq p \leq 1$) with row and column length equal to that of alpha and beta and row sum and column sum both equal to 1 (within some numerical deviation of 1e-6).
alpha	Expected a vector of positive values with length equal to that of beta and the row/column of TRANS.
beta	Expected a vector of positive values with length equal to that of alpha and the row/column of TRANS.

Value

N	Number of components or equivalently the length of alpha, beta, or wght.

Author(s)

Yue Li

References

Bishop, Christopher. Pattern recognition and machine learning. Number 605-631 in Information Science and Statisitcs. Springer Science, 2006.

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

nbh_em,nbm_chk

Examples

```
# two hidden states
TRANS <- matrix(c(0.9, 0.1, 0.3, 0.7), nrow=2, byrow=TRUE)

alpha <- c(2, 4)

beta  <- c(1, 0.25)

nbh_chk(TRANS, alpha, beta)
```

---

nbh_em *Expectation conditional maximization of negative binomial HMM parameters using forward-backward algorithm*

Description

Given an input read count vector of integers, the function optimizes the parameters for the negative binomial HMM of K hidden states using expectation conditional maximization with forward-backward algorithm to acheive the exact inference.

Usage

```
nbh_em(count, TRANS, alpha, beta, NBH_NIT_MAX = 250,
NBH_TOL = 1e-05, MAXALPHA = 1e+07, MAXBETA = 1e+07)
```

Arguments

count	A vector of integers, conceptaully representative of the read counts within bins of chromosome.
TRANS	Transition probability matrix, a squared matrix of probabilities ($0 \leq p \leq 1$) with row and column length equal to that of alpha and beta and row sum and column sum both equal to 1 (within some numerical deviation of 1e-6).
alpha	Shape parameter of the NB as a vector of positive values with length equal to that of beta and the row/column of TRANS.
beta	Inverse scale parameter of the NB as a vector of positive values with length equal to that of beta and the row/column of TRANS.
NBH_NIT_MAX	Maximum number of EM iterations (Default: 250) for the negative binomial hidden Markov model (NBH).
NBH_TOL	Threshold as fraction of increase in likelihood (given the current NBH parameters) comparing with the likelihood from the last iteration. EM for the NBH stops when the improvement is below the threshold (Default: 0.001).
MAXALPHA	The maximum value of alpha in case the update goes beyond the numerical upper limit of the system. Once alpha becomes larger than MAXALPHA, the EM itaration is prematurely terminated to prevent malfunction.
MAXBETA	The maximum value of beta in case the update goes beyond the numerical upper limit of the system. Once beta becomes larger than MAXBETA, the EM itaration is prematurely terminated to prevent malfunction.

Details

Given a K-state HMM with NB emission (NBH), the goal is to maximize the likelihood function with respect to the parameters comprising of α_k and β_k for the K NB components and the transition probabilities A_jk between any state j and k, which are the priors $p(z = k)$. Because there is no analytical solution for the maximum likelihood (ML) estimators of the above quantities, a modified EM procedures called Expectation Conditional Maximization is employed (Meng and Rubin, 1994).

In E-step, the posterior probability is evaluated by forward-backward algorithm using NB density functions with initialized alpha, beta, and TRANS. In the CM step, A_jk is evaluated first followed by Newton updates of α_k and β_k. EM iteration terminates when the percetnage of increase of log likelihood drop below `NBH_TOL`, which is deterministic since EM is guaranteed to converge. For more details, please see the manuscript of RIPSeeker.

Value

A list containing:

`alpha`	optimized alpha_k for NB at state K
`beta`	optimized beta_k for NB at state K
`TRANS`	optimized transition probability matrix
`logl`	Log likelihood in each EM iteration.
`postprob`	Posterior probabilities for each observed data point at the last EM iteration.
`dens`	the negative binomial probabilities computed at the last EM iteration

Author(s)

Yue Li

References

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition (Vol. 77, pp. 257-286). Presented at the Proceedings of the IEEE. doi:10.1109/5.18626

Christopher Bishop. Pattern recognition and machine learning. Number 605-631 in Information Science and Statisitcs. Springer Science, 2006.

X. L. Meng, D. B. Rubin, Maximum likelihood estimation via the ECM algorithm: A general framework, Biometrika, 80(2):267-278 (1993).

J. A. Fessler, A. O. Hero, Space-alternating generalized expectation-maximization algorithm, IEEE Tr. on Signal Processing, 42(10):2664 -2677 (1994).

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

`nbh_init`,`nbh`,`nbh.GRanges`,`nbh_vit`,`nbm_em`

Examples

```
# Simulate data
TRANS_s <- matrix(c(0.9, 0.1, 0.3, 0.7), nrow=2, byrow=TRUE)
alpha_s <- c(2, 4)
beta_s  <- c(1, 0.25)
Total <- 100
```

```
x <- nbh_gen(TRANS_s, alpha_s, beta_s, Total);

count <- x$count
label <- x$label

Total <- length(count)

# dummy initialization
TRANS0 <- matrix(rep(0.5,4), 2)

alpha0 <- c(1, 20)

beta0 <- c(1, 1)

NIT_MAX <- 50
TOL <- 1e-100
nbh <- nbh_em(count, TRANS0, alpha0, beta0, NIT_MAX, TOL)

map.accuracy <- length(which(max.col(nbh$postprob) == label))/Total

vit <- nbh_vit(count, nbh$TRANS, nbh$alpha, nbh$beta)

vit.accuracy <- length(which(vit$class == label))/Total

# Plots
par(mfrow=c(2,2), cex.lab=1.2, cex.main=1.2)

plot(count, col="blue", type="l", main=sprintf("A. Simulated Data (Total = %i)",Total))

plot(as.numeric(nbh$logl), xlab="EM Iteration", ylab="Log-Likelihood",
main="B. Log-Likelihood via EM");grid()


# Marginal postprob
plot(nbh$postprob[,2], col="blue", type="l", ylim = c(0,1),
ylab="Marginal Posteriror or True State")
points(label-1, col="red")
title(main = sprintf("C. MAP Prediciton Accuracy = %.2f%s", 100 * map.accuracy, "%"))


# Viterbi states
plot(vit$class - 1, col="dark green", type="l", ylim = c(0,1),
ylab="Viterbi or True State")
points(label-1, col="red")
title(main = sprintf("D. Viterbi Prediciton Accuracy = %.2f%s", 100 * vit.accuracy, "%"))
```

---

nbh_gen *Simulate data from a negative binomial HMM.*

Description

Generate count data and the hidden states based on the user-supplied HMM paramters.

Usage

```
nbh_gen(TRANS, alpha, beta, Total)
```

Arguments

TRANS	Expected a squared matrix of probabilities ($0 \leq p \leq 1$) with row and column length equal to that of alpha and beta and row sum and column sum both equal to 1 (within some numerical deviation of 1e-6).
alpha	Expected a vector of positive values with length equal to that of beta and the row/column of TRANS.
beta	Expected a vector of positive values with length equal to that of alpha and the row/column of TRANS.
Total	Total number of data points to generate.

Value

A list containing:

count	Simulation count data.
label	Hidden states associated with the simulated data.

Author(s)

Yue Li

References

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

nbh_em, nbm_chk, randindx

Examples

```
# Simulate data using user-supplied transition prob, alpha and beta for the NB HMM parameters
TRANS_s <- matrix(c(0.9, 0.1, 0.3, 0.7), nrow=2, byrow=TRUE)
alpha_s <- c(2, 4)
beta_s  <- c(1, 0.25)
Total <- 100
x <- nbh_gen(TRANS_s, alpha_s, beta_s, Total)
```

nbh_init *Initialize negative binomial HMM parameters using negative binomial mixture model*

Description

The function finds a sensible set of initial NB HMM parameters by fitting a NB mixture model of K components using the read count data.

Usage

```
nbh_init(count, K, NBM_NIT_MAX = 250, NBM_TOL = 0.001)
```

Arguments

count	A vector of integers, conceptaully representing the read counts within bins of chromosome.
K	Number of hidden states.
NBM_NIT_MAX	Maximum number of EM iterations (Default: 250) for the negative binomial mixture model (NBM) intialization step (See nbm_em).
NBM_TOL	Threshold as fraction of increase in likelihood (given the current NBM parameters) comparing with the likelihood from the last iteration. EM for the NBM stops when the improvement is below the threshold (Default: 0.01).

Details

Because the EM algorithm in HMM tends to fall into local optimal with poor initialization, NB mixture model with K mixture components (K-NBM) is first applied to the data to obtain a reasonable estimate for the HMM parameters. Given the read count vector, the function applied the lower level function nbm_em (NB mixture model) to find alpha, beta, and mixing proportion of the K NB mixture components. Alpha and beta are the parameters of the NB mixture components initialized as the last K quantiles of the nonzero read counts and 1, respectively. The mixing proportions or component weights (wght) of the NB distributions are first initialized as uniform and after EM optimization are used to form a symmetrical transition probability matrix such that probability of state 1 transitioning to state 2 is equal to the probability of state 2 transitioning to state 1. Such matrix is used as the initial transition probability for the HMM model tranining (See nbh_em).

Value

A list containing:

alpha	Alpha paramter of the K NB components optimized using nbm_em
beta	Beta paramter of the K NB components optimized using nbm_em
TRANS	Transition probability intialized as a symmetrical matrix of mixing proportion of the K NB components optimized using nbm_em.

Author(s)

Yue Li

References

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition (Vol. 77, pp. 257-286). Presented at the Proceedings of the IEEE. doi:10.1109/5.18626

Christopher Bishop. Pattern recognition and machine learning. Number 605-631 in Information Science and Statisitcs. Springer Science, 2006.

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

nbm_em, nbh, nbh.GRanges, nbh_em

Examples

```
# Simulate data
Total_train <- 1000

Total_test <- 200

TRANS_s <- matrix(c(0.9, 0.1, 0.5, 0.5), nrow=2, byrow=TRUE)
alpha_s <- c(2, 2)
beta_s  <- c(1, 0.25)

train <- nbh_gen(TRANS_s, alpha_s, beta_s, Total_train)

test <- nbh_gen(TRANS_s, alpha_s, beta_s, Total_test)

nbhInit <- nbh_init(train$count, ncol(TRANS_s))

count <- train$count
label <- train$label

# NBH initialization
nbhInit <- nbh_init(count, ncol(TRANS_s))

TRANS0 <- nbhInit$TRANS
alpha0 <- nbhInit$alpha
beta0 <- nbhInit$beta

# NBH EM
nbh <- nbh_em(count, TRANS0, alpha0, beta0)

map.accuracy <- length(which(max.col(nbh$postprob) == label))/Total_train

vit <- nbh_vit(count, nbh$TRANS, nbh$alpha, nbh$beta)

vit.accuracy <- length(which(vit$class == label))/Total_train

vit_test <- nbh_vit(test$count, nbh$TRANS, nbh$alpha, nbh$beta)

vit_test.accuracy <- length(which(vit_test$class == test$label))/Total_test

nbh_wt_KMLinit <- list(mapAccuracy_train=map.accuracy, vitAccuracy_train=vit.accuracy,
vitLogl_train=vit$logl, vitAccuracy_test=vit_test.accuracy,
vitLogl_test=vit_test$logl)
```

nbh_vit *Derive maximum likelihood hidden state sequence using Viterbi algorithm*

Description

Given read counts and HMM parameters (optimized by nbh_em), derive the sequence of hidden states that maximizes the joint likelihood of observed and latent data.

Usage

```
nbh_vit(count, TRANS, alpha, beta)
```

Arguments

count	A vector of integers, conceptaully representative of the read counts within bins of chromosome.
TRANS	Optimized transition probability matrix, a squared matrix of probabilities ($0 \leq p \leq 1$) with row and column length equal to that of alpha and beta and row sum and column sum both equal to 1 (within some numerical deviation of 1e-6).
alpha	Optimized shape parameter of the NB as a vector of positive values with length equal to that of beta and the row/column of TRANS.
beta	Optimized inverse scale parameter of the NB as a vector of positive values with length equal to that of beta and the row/column of TRANS.

Details

Given a K-state HMM with NB emission (NBH), the goal is to find the latent states corresponding to the observed data that maximize the joint likelihood $lnp(X, Z) = lnp(x_1, \ldots, x_N, z_1, \ldots, z_N)$. The optimal solution is obtained via Viterbi algorithm, which essentially belongs to the more general framework of Dynamic Programming.

Briefly, starting from the second node of the Markov chain, we select state of the first node that maximizes $lnp(x_1, x_2, z_2|z_1)$ for *every* state of z_2. Then, we move on to the next node and the next until reaching to the last node. In the end, we make choice for the state of the last node that together leads to the maximum $lnp(X, Z)$. Finally, we backtrack to find the choices of states in all of the intermeidate nodes to form the final solution.

Value

A list containing:

class	ML sequence of latent states
logl	Log-likelihood corresponding to the latents states class

Note

The function is expected to run after learning the model parameters of HMM using nbh_em and (optionally) disambiguating the multihits using nbh_vit. However, nothing prevents user from running it with a random set of HMM parameters. Also, note that Viterbi algorithm finds the most probable *sequence of states*, which is not the same as maximizing the posterior probabilities for all the individual latent variables. For instance, a observed data point may be classified as from state 2 in the most probable chain in spite its marginal posterior probability for state 2 is zero.

Author(s)

Yue Li

References

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition (Vol. 77, pp. 257-286). Presented at the Proceedings of the IEEE. doi:10.1109/5.18626

Christopher Bishop. Pattern recognition and machine learning. Number 605-631 in Information Science and Statisitcs. Springer Science, 2006.

X. L. Meng, D. B. Rubin, Maximum likelihood estimation via the ECM algorithm: A general framework, Biometrika, 80(2):267-278 (1993).

J. A. Fessler, A. O. Hero, Space-alternating generalized expectation-maximization algorithm, IEEE Tr. on Signal Processing, 42(10):2664 -2677 (1994).

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

nbh_init,nbh,nbh.GRanges,nbh_em,nbm_em

Examples

```
# Simulate data
TRANS_s <- matrix(c(0.9, 0.1, 0.3, 0.7), nrow=2, byrow=TRUE)
alpha_s <- c(2, 4)
beta_s  <- c(1, 0.25)
Total <- 100

x <- nbh_gen(TRANS_s, alpha_s, beta_s, Total);

count <- x$count
label <- x$label

Total <- length(count)

# dummy initialization
TRANS0 <- matrix(rep(0.5,4), 2)

alpha0 <- c(1, 20)

beta0 <- c(1, 1)

NIT_MAX <- 50
TOL <- 1e-100
nbh <- nbh_em(count, TRANS0, alpha0, beta0, NIT_MAX, TOL)

map.accuracy <- length(which(max.col(nbh$postprob) == label))/Total

vit <- nbh_vit(count, nbh$TRANS, nbh$alpha, nbh$beta)

vit.accuracy <- length(which(vit$class == label))/Total

# Plots
par(mfrow=c(2,2), cex.lab=1.2, cex.main=1.2)
```

```
plot(count, col="blue", type="l", main=sprintf("A. Simulated Data (Total = %i)",Total))

plot(as.numeric(nbh$logl), xlab="EM Iteration", ylab="Log-Likelihood",
main="B. Log-Likelihood via EM");grid()


# Marginal postprob
plot(nbh$postprob[,2], col="blue", type="l", ylim = c(0,1),
ylab="Marginal Posteriror or True State")
points(label-1, col="red")
title(main = sprintf("C. MAP Prediciton Accuracy = %.2f%s", 100 * map.accuracy, "%"))


# Viterbi states
plot(vit$class - 1, col="dark green", type="l", ylim = c(0,1),
ylab="Viterbi or True State")
points(label-1, col="red")
title(main = sprintf("D. Viterbi Prediciton Accuracy = %.2f%s", 100 * vit.accuracy, "%"))
```

---

nbm_chk *Check the parameters of the negative binomial mixture model*

Description

The function verifies the numerical range and dimension of the NBM paramters alpha, beta, and wght and returns the number of components. It is used in nbm_em before running EM.

Usage

```
nbm_chk(alpha, beta, wght)
```

Arguments

alpha	Expected a vector of positive values with length equal to that of beta and wght.
beta	Expected a vector of positive values with length equal to that of alpha and wght.
wght	Expected a vector of probabilities ($0 \leq p \leq 1$) with length equal to that of alpha and beta and summed to 1 (within some numerical deviation of 1e-6).

Value

N	Number of components or equivalently the length of alpha, beta, or wght.

Author(s)

Yue Li

References

Christopher Bishop. Pattern recognition and machine learning. Number 605-631 in Information Science and Statisitcs. Springer Science, 2006.

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

nbm_em, nbh_chk

Examples

```
# two mixing components
wght <- c(0.5,0.5)

alpha <- c(1, 20)

beta <- c(1, 1)

nbm_chk(alpha, beta, wght)
```

---

nbm_em *Expectation conditional maximization of likelihood for negative binomial mixture model*

Description

Given an input read count vector of integers, the function optimzes the parameters for the negative binomial mixture model of K components using expectation conditional maximization.

Usage

```
nbm_em(count, alpha, beta, wght, NBM_NIT_MAX = 250, NBM_TOL = 0.01)
```

Arguments

count	A vector of integers, conceptaully representing the read counts within bins of chromosome.
alpha	Initial values for α_k for all K NB.
beta	Initial values for β_k for all K NB.
wght	Initial values for π_k for all K NB.
NBM_NIT_MAX	Maximum number of EM iterations (Default: 250).
NBM_TOL	Threshold as fraction of increase in likelihood (given the current NBM parameters) comparing with the likelihood from the last iteration. EM for the NBM stops when the improvement is below the threshold (Default: 0.01).

Details

Given a K-NBM, the goal is to maximize the likelihood function with respect to the parameters comprising of α_k and β_k for the K NB components and the mixing coefficients π_k, which are the priors $p(z = k)$. Because there is no analytical solution for the maximum likelihood (ML) estimators of the above quantities, a modified EM procedures called Expectation Conditional Maximization is employed (Meng and Rubin, 1994).

In E-step, the posterior probability is evaluated using NB density functions with initialized α_k, β_k, and π_k. In the CM step, π_k is evaluated first followed by Newton updates of α_k and β_k. EM iteration terminates when the percetnage of increase of log likelihood drop below NBM_TOL, which is deterministic since EM is guaranteed to converge. For more details, please see the manuscript of RIPSeeker.

Value

A list containing:

alpha	alpha_k for all K components of NB.
beta	beta_k for all K components of NB.
wght	pi_k for all K components of NB.
logl	Log likelihood in each EM iteration.
postprob	Posterior probabilities for each observed data point in the last EM iteration.

Author(s)

Yue Li

References

Bishop, Christopher. Pattern recognition and machine learning. Number 605-631 in Information Science and Statisitcs. Springer Science, 2006.

X. L. Meng, D. B. Rubin, Maximum likelihood estimation via the ECM algorithm: A general framework, Biometrika, 80(2):267-278 (1993).

J. A. Fessler, A. O. Hero, Space-alternating generalized expectation-maximization algorithm, IEEE Tr. on Signal Processing, 42(10):2664 -2677 (1994).

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

nbh_init,nbh,nbh.GRanges,nbh_em

Examples

```
# Simulate data
TRANS_s <- matrix(c(0.9, 0.1, 0.3, 0.7), nrow=2, byrow=TRUE)
alpha_s <- c(2, 4)
beta_s  <- c(1, 0.25)
Total <- 1000
x <- nbh_gen(TRANS_s, alpha_s, beta_s, Total);

N <- 2

cnt <- x$count
label <- x$label

Total <- length(cnt)

# dummy initialization
wght0 <- c(0.5,0.5)

alpha0 <- c(1, 20)

beta0 <- c(1, 1)

NIT_MAX <- 50
TOL <- 1e-100
```

```
# initialize param with nbm

nbm <- nbm_em(cnt, alpha0, beta0, wght0, NIT_MAX, TOL)

map.accuracy <- length(which(max.col(nbm$postprob) == label))/Total

print(map.accuracy)
```

---

plotCoverage *Plot read coverage for a GRanges object*

Description

An internal function used by plotStrandedCoverage to plot read counts within each fixed bin across the entire chromosome.

Usage

```
plotCoverage(x, plotLegend = FALSE, legend.cex = 1, ...)
```

Arguments

x	GRanges object with values slot saved for read counts within the corresponding ranges.
plotLegend	Binary indcator. If TRUE, legend will be plotted on the top left the plot. Legend is expected to be the chromsome name and length, which must be available in the GRange object argument.
legend.cex	Font size of the legend.
...	Extra arguments passed to either the plot or the legend.

Details

The read counts is plotted in blue bars as positive integer across the x-axis as the sorted positions across the chromosome. The plot can be used to examine the overall alignment properties for each chromosome.

Note

Users are not recommanded run this function directly but rather via a much more user friendly function plotStrandedCoverage.

Author(s)

Yue Li

References

P. Aboyoun, H. Pages and M. Lawrence (). GenomicRanges: Representation and manipulation of genomic intervals. R package version 1.8.9.

See Also

plotStrandedCoverage,plot,legend

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGRList <- GRangesList(as.list(split(alignGR, seqnames(alignGR))))

binSize <- 1000

binGR <- binCount(alignGRList$chrX, binSize)

plotCoverage(binGR, plotLegend=TRUE)
```

---

plotStrandedCoverage *Plot strand-specific read coverage for a GRanges object*

Description

Plot read counts within fixed bin across the entire chromosome.

Usage

```
plotStrandedCoverage(gr, binSize = 1000, plotLegend = FALSE, ylim, ...)
```

Arguments

gr	GRanges object containing the alignments.
binSize	Integer indicate the size of the bin used to compute and plot the read counts.
plotLegend	Binary indcator. If TRUE, legend will be plotted on the top left the plot. Legend is expected to be the chromsome name and length, which must be available in the GRange object argument.
ylim	A two element scale on the y-axis, indicating the maximum read counts on the + and - strand to be plotted (e.g., ylim=c(-200, 200)).
...	Extra arguments passed to plotCoverage.

Details

Read count on + and - strand are displayed as red and blue bars on the positive and negative y-axis, respectively. The x-axis indicates the positions across the chromosmoe. The plot can be used to examine for each chromosome the overall alignment properties such as strand specificity (expected in non-strand-specific sequencing) and aggregation of reads.

Author(s)

Yue Li

References

P. Aboyoun, H. Pages and M. Lawrence (). GenomicRanges: Representation and manipulation of genomic intervals. R package version 1.8.9.

See Also

plotCoverage,plot,legend

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGRList <- GRangesList(as.list(split(alignGR, seqnames(alignGR))))

binSize <- 1000

plotStrandedCoverage(gr=alignGRList$chrX, binSize=binSize,

xlab="", ylab="", plotLegend=TRUE, box.lty=0, legend.cex=2 )
```

---

randindx *Generates random indexes with a specified probability distribution*

Description

Returns an array of T indexes distributed as specified by p (which should be a normalized probability vector).

Usage

```
randindx(p, Total, NO_CHK)
```

Arguments

p	A row vector of normalized probabilities that dictate the transition probability from the current state to the next state. For example, $p = [0.2, 0.8]$ indicates that the current state transitoins to state 1 at 0.2 and 2 at 0.8. The current state itself can either be the state 1 or 2.
Total	Total number of states needed to be generated using the input transition vector.
NO_CHK	Check whether the first argument is a valid row vector of normalized probabilities.

Details

The function is used by nbh_gen to generate random data point based on the user-supplied transition probability matrix.

Value

I	Index/Indices or state(s) sampled following the transition.

probability.

Author(s)

Yue Li

References

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

nbh_gen

Examples

```
# Total contains the length of data to simulate
Total <- 100

# number of states
N <- 2

# transition probabilities between states
TRANS <- matrix(c(0.9, 0.1, 0.3, 0.7), nrow=2, byrow=TRUE)

label <- matrix(0, Total, 1)

# Simulate initial state
label[1] <- randindx(matrix(1,ncol=N)/N, 1, 1)

# Use Markov property for the following time index
for(t in 2:Total) {

label[t] <- randindx(TRANS[label[t-1],], 1, 1)
}
```

```
plot(label)
```

`ripSeek` *HMM-based de novo RIP predictions using alignment data*

Description

This function is the main interface to most essential functions of RIPSeeker package.

Usage

```
ripSeek(bamPath, cNAME, binSize = NULL, strandType = NULL,
paired=FALSE, biomaRt_dataset, goAnno, exportFormat = "gff3",
annotateFormat = "txt", annotateType = "TSS", outDir,
padjMethod = "BH", logOddCutoff = 0, pvalCutoff = 1,
pvalAdjCutoff = 1, eFDRCutoff = 1, ...)
```

Arguments

`bamPath`	Either a path to all of the bam files or a list of paths to individual BAM files. BED and SAM files are also accepted.
`cNAME`	An identifer pattern found in the control alignment files. Once specified, these files will be used as control and the remaining files as RIP for discriminative analysis (see `seekRIP`).
`binSize`	Size to use for binning the read counts across each chromosome. If NULL, optimal bin size within a range (default: minBinSize=200, maxBinSize=1200) will be automatically selected (see `selectBinSize`).
`strandType`	Type of strand can be +, -, or * as in GAlignments, GAlignmentPairs, or GRanges (see `GenomicRanges`).
`paired`	Binary to indicate whether the library is paired-end (TRUE) or single-end (FALSE by default) (see `getAlignGal`).
`biomaRt_dataset`	The dataset name used in biomaRt for retrieving genomic information for a given species name (see `annotateRIP`).
`goAnno`	GO dataset name used for GO enrichment analysis (See `annotateRIP`).
`exportFormat`	Format to export the RIP predictions. The commonly used ones are GFF and BED, which can be directly imported as a track to a genomic viewer such as Integrative Genomic Viewer, SAVANT or USCSC browser.
`annotateFormat`	Format to export the annotated RIP predictions. The default "txt" is a tab-delimited format, recommanded for viewing in Excel.
`annotateType`	Type of genomic information in association with the RIP predictions that can be retrieved from Ensembl database (Default: TSS; See `annotateRIP`).
`outDir`	Output directory to save the results. The output data include ...
`padjMethod`	Method to adjust multiple testing (Benjamini-Hocherge method by default).
`logOddCutoff`	Threshold for the log odd ratio of posterior for the RIP over the background states (See `seekRIP`). Only peaks with logOdd score *greater* than the `logOddCutoff` will be reported. Default: 1.

pvalCutoff	Threshold for the p-value for the logOdd score. Only peaks with p-value *less* than the pvalCutoff will be reported. Default: 1 (i.e. no cutoff).
pvalAdjCutoff	Threshold for the adjusted p-value for the logOdd score. Only peaks with adjusted p-value *less* than the pvalAdjCutoff will be reported. Default: 1 (i.e. no cutoff).
eFDRCutoff	Threshold for the empirical false discovery rate (eFDR). Only peaks with eFDR *less* than the eFDRCutoff will be reported. Default: 1 (i.e. no cutoff).
...	Arguments passed to mainSeek.

Details

This is the main front-end function of RIPSeeker and in many cases the only function that users need to get RIP predictions and all relevant information.

Value

A list is returned with the following items:

mainSeekOutputRIP
: A (inner) list comprising three items:

  nbhGRList: GRangesList of the HMM trained parameters for each chromosome on RIP.

  alignGal,alignGalFiltered: GAlignments objects of the RIP alignment outputs from combineAlignGals and disambiguateMultihits, respectively. The former may contain multiple alignments due to the same reads whereas the latter contains a one-to-one mapping from read to alignment after disambiguating the multihits.

mainSeekOutputCTL
: Same as mainSeekOutputRIP but for the control library (if available).

RIPGRList
: The results as GRangesList generated from the RIP peak detection. Each list item represents the RIP peaks on a chromosome accompanied with statistical scores including (read) count, logOddScore, pval, pvalAdj, eFDR for the RIP and control (if available). Please refer to seekRIP for more details.

annotatedRIPGR
: If annotatedRIPGR is TRUE, the additional genomic information will be retreived according to the genomic coordinates of the peaks in RIPGRList. The results are saved in this separate GRanges object as the final results that user will find the most useful.

Note

You may only want to know the expression/abundance of known transcripts/genes or the foldchange between two conditions. In that case, use rulebaseRIPSeek and computeRPKM, respectively. Both singl-end and paired-end alignments are supported in these functions.

Author(s)

Yue Li

References

Zhao, J., Ohsumi, T. K., Kung, J. T., Ogawa, Y., Grau, D. J., Sarma, K., Song, J. J., et al. (2010). Genome-wide Identification of Polycomb-Associated RNAs by RIP-seq. Molecular Cell, 40(6), 939D953. doi:10.1016/j.molcel.2010.12.011

The RIPSeeker manuscript has been submitted to NAR for review.

See Also

rulebaseRIPSeek

Examples

```
if(interactive()) { # need internet connection

# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

cNAME <- "SRR039214"  # specify control name

# output file directory
outDir <- paste(getwd(), "ripSeek_example", sep="/")

# Parameters setting
binSize <- NULL # automatically determine bin size
minBinSize <- 10000 # min bin size in automatic bin size selection
maxBinSize <- 12000 # max bin size in automatic bin size selection
multicore <- TRUE # use multicore
strandType <- "-" # set strand type to minus strand

biomart <- "ENSEMBL_MART_ENSEMBL" # use archive to get ensembl 65
dataset <- "mmusculus_gene_ensembl" # mouse dataset id name
host <- "dec2011.archive.ensembl.org"  # use ensembl 65 for annotation

goAnno <- "org.Mm.eg.db"


################ run main function ripSeek to predict RIP ################
seekOut <- ripSeek(bamPath=bamFiles, cNAME=cNAME,
binSize=binSize, minBinSize = minBinSize,
maxBinSize = maxBinSize, strandType=strandType,
outDir=outDir, silentMain=FALSE,
verbose=TRUE, reverseComplement=TRUE, genomeBuild="mm9",
biomart=biomart, host=host,
biomaRt_dataset = dataset,
goAnno = goAnno,
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = TRUE, multicore=multicore)


################ visualization ################

viewRIP(seekOut$RIPGRList$chrX, seekOut$mainSeekOutputRIP$alignGalFiltered,
```

```
seekOut$mainSeekOutputCTL$alignGalFiltered, scoreType="eFDR")

}
```

`rulebaseRIPSeek` *Compute the RPKM and foldchange between two conditions for the annotated genes*

Description

The function takes alignments in two conditions (with replicates) as input and computes the gene expression in the two conditions in the unit of RPKM (reads per kilobase of exon per million mapped reads) or FPKM for paired-end alignments (where "F" stands for the fragment the mate-pairs are derived from), and then the foldchange ratio of the RPKM of each gene in RIP or treatment condition in general over control condition. The control files (i.e. the denominator in the foldchange ratio) is specified by user in the "cNAME" argument.

Usage

```
rulebaseRIPSeek(bamFiles, cNAME, featureGRanges, rpkmCutoff = 0.4,
fcCutoff = 3, moreRIPGeneInfo = TRUE, idType = "ensembl_transcript_id",
myMin = .Machine$double.xmin, saveRData, ...)
```

Arguments

`bamFiles`	A list of one or more BAM/SAM/BED alignment files.
`cNAME`	An identifer pattern found in the control alignment files. Once specified, these files will be used as control as the denomenator of the foldchange ratio and the remaining files as RIP, the numarator of the foldchange ratio.
`featureGRanges`	GRanges of features as an optional argument for function to compute RPKM/FPKM just for those features without retrieving online annotations.
`rpkmCutoff`	Cutoff for RPKM in RIP above which the genes will be reported if the `fcCutoff` is also satisfied (Default: 0.4).
`fcCutoff`	Cutoff for foldchange in RIP relative to the control above which the genes will be reported if the `rpkmCutoff` is also satisfied (Default: 3).
`moreRIPGeneInfo`	Binary indicator to indicate whether to download more information for each genes/transcripts rather than having only the gene/transcript IDs (Default: TRUE).
`idType`	A character string that specifies the type of the annotations, which can "ensembl_transcript_id" (Default), "ensembl_gene_id", "ucsc", etc. Refer to `listFilters` for more information.
`myMin`	Add a small value to both the numerator and denomenator as "pseudocount" to prevent the case where the denomenator is zero and the ratio becomes infinity regardless the value of the numerator (Default: `.Machine$double.xmin`).
`saveRData`	Path of output RData and tab-delim results.
`...`	Extra arguments passed to `computeRPKM` and/or`useMart`.

Details

The function uses computeRPKM to download annotation and compute RPKM/FPKM of the annotated genes in the list of files. The alignments file are separated into control as identified by the "cNAME" and the RIP (or any treatment) that do not have the cNAME in their file names. The alignments in either group are pooled together. If moreRIPGeneInfo is specified, the function witll query the Ensembl database. The chromosome ID in the numerical format used in Ensembl is prefixed with "chr" and the strand 1 and -1 converted to + and - for convenience.

Value

A list containing the following items:

nRPKM	RPKM of genes in RIP or treatment condition ('n' stands for numerator in the foldchange ratio).
dRPKM	RPKM of genes in control condition ('d' stands for denomenator in the foldchange ratio)
rpkmDF	Data frame containing read count, RPKM for the RIP (or treatment) and control, foldchange, and optional gene information including "chromosome_name", "start_position", "end_position", "strand", "external_gene_id", "ensembl_transcript_id", "ensembl_gene_id", "ucsc", "description"
rpkmCutoff	Cutoff used for RPKM as book keeping value.
fcCutoff	Cutoff used for foldchange as book keeping value.
featureGRanges	GRanges object of the features for which the RPKM and foldchange are computed.

Note

Also works for RNA-seq alignments.

Author(s)

Yue Li

References

Zhao, J., Ohsumi, T. K., Kung, J. T., Ogawa, Y., Grau, D. J., Sarma, K., Song, J. J., et al. (2010). Genome-wide Identification of Polycomb-Associated RNAs by RIP-seq. Molecular Cell, 40(6), 939D953. doi:10.1016/j.molcel.2010.12.011

M. Carlson, H. Pages, P. Aboyoun, S. Falcon, M. Morgan, D. Sarkar and M. Lawrence. GenomicFeatures: Tools for making and manipulating transcript centric annotations. R package version 1.8.2.

P. Aboyoun, H. Pages and M. Lawrence (). GenomicRanges: Representation and manipulation of genomic intervals. R package version 1.8.9.

Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. Steffen Durinck, Paul T. Spellman, Ewan Birney and Wolfgang Huber, Nature Protocols 4, 1184-1191 (2009).

BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. Steffen Durinck, Yves Moreau, Arek Kasprzyk, Sean Davis, Bart De Moor, Alvis Brazma and Wolfgang Huber, Bioinformatics 21, 3439-3440 (2005).

Martin Morgan and Herv\'e Pag\'es (). Rsamtools: Binary alignment (BAM), variant call (BCF), or tabix file import. R package version 1.8.5. http://bioconductor.org/packages/release/bioc/html/Rsamtools.html

See Also

makeTxDbFromBiomart,makeTxDbFromUCSC,useMart,exonsBy,cdsBy,intronsByTranscript,fiveUTRsByTranscr

Examples

```
if(interactive()) {

# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

cNAME <- "SRR039214"  # specify control name

# output file directory
outDir <- paste(getwd(), "ripSeek_example")

# use biomart
txDbName <- "biomart"
biomart <- "ENSEMBL_MART_ENSEMBL" # use archive to get ensembl 65
dataset <- "mmusculus_gene_ensembl"
host <- "dec2011.archive.ensembl.org"  # use ensembl 65 for annotation

resultlist <- rulebaseRIPSeek(bamFiles, "SRR039214", dataset=dataset,
txDbName=txDbName, biomart=biomart, host=host, by="tx")

}
```

---

scoreMergedBins *Average log odd scores over bins being merged into a single region*

Description

Sum, normalize the read counts, and average the logOdd score over the bins being merged into a single enriced region.

Usage

```
scoreMergedBins(findOverlapsHits, unmergedGRAll, mergedGRAll)
```

Arguments

findOverlapsHits	Output from findOverlaps as two columns indices with the first column containing the indices for unmerged GRanges and the second column the indices of the matched merged GRanges.
unmergedGRAll	GRanges before merging.
mergedGRAll	GRanges after merging.

Details

The consecutive RIP-bins predicted by the Viterbi function (See nbh_vit) are merged into a single RIP region. An aggregate RIPScore as the averaged RIPScores over the associated merged bins is assigned to each merged RIP region. In the RIPSeeker workflow, the averaged RIPScore then becomes the representative score for the region and subject to significance test carried out in seekRIP.

Value

A merged GRanges each with scores including summed read count, averaged log odd scores, and FPK (fragment per kilobase of region length). The latter score represent a normalized read count.

Note

This function is expected to be called only from logScoreWithoutControl and logScoreWithControl.

Author(s)

Yue Li

See Also

seekRIP,computeLogOdd logScoreWithControl,logScoreWithoutControl

Examples

```
if(interactive()) { # see example in seekRIP
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

# Parameters setting
binSize <- 1e5 # use a large fixed bin size for demo only
multicore <- FALSE # use multicore
strandType <- "-" # set strand type to minus strand

################ run main function for HMM inference on all chromosomes ################
mainSeekOutputRIP <- mainSeek(bamFiles=
    grep(pattern="SRR039214", bamFiles, value=TRUE, invert=TRUE),
binSize=binSize, strandType=strandType,
reverseComplement=TRUE, genomeBuild="mm9",
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = TRUE,
multicore=multicore, silentMain=FALSE, verbose=TRUE)


nbhGRRIP <- mainSeekOutputRIP$nbhGRList$chrX

logOddScore <- computeLogOdd(nbhGRRIP)

values(nbhGRRIP) <- cbind(as.data.frame(values(nbhGRRIP)), logOddScore)
```

```
enrichIdx <- which(values(nbhGRRIP)$viterbi_state == 2)

unmergedRIP <- nbhGRRIP[enrichIdx]

mergedRIP <- reduce(unmergedRIP, min.gapwidth = median(width(unmergedRIP) ))

overlapIdx <- findOverlaps(mergedRIP, unmergedRIP)

# a list with query hits as names and subject hits as items
findOverlapsHits <- split(overlapIdx, queryHits(overlapIdx))

# get the score for the first merged region
x <- scoreMergedBins(findOverlapsHits[[1]], unmergedRIP, mergedRIP)

# get scores for all of the merged regions
mergedRIPList <- lapply(split(overlapIdx, queryHits(overlapIdx)),

scoreMergedBins, unmergedRIP, mergedRIP)

names(mergedRIPList) <- NULL

mergedRIP <- do.call(c, mergedRIPList)

# logOddScore is the averaged logOddScore across merged bins
mergedRIP
}
```

`seekRIP` *Identify significant peaks*

Description

Based on the posteriors derived from HMM by `mainSeek`, find the significant RIP regions derived from merging the adjacent RIP bins. The significance test makes use of the log odd ratio of the posteriors for RIP over the background states.

Usage

```
seekRIP(nbhGRRIP, nbhGRCTL = NULL, padjMethod = "BH",
logOddCutoff = -Inf, pvalCutoff = 1, pvalAdjCutoff = 1,
eFDRCutoff = 1)
```

Arguments

`nbhGRRIP`	`GRanges` object for the RIP library created from `mainSeek` containing the posterior probabilities of the hidden states for each observed read count.
`nbhGRCTL`	An optional argument as a `GRanges` object for the control library created from `mainSeek` containing the posterior probabilities of the hidden states for each observed read count.
`padjMethod`	Method used to adjust multiple testing performed in `p.adjust` (Default: "BH").
`logOddCutoff`	Threshold for the log odd ratio of posterior for the RIP over the background states (See `seekRIP`). Only peaks with logOdd score *greater* than the `logOddCutoff` will be reported. Default: -Inf (i.e. no cutoff).

pvalCutoff	Threshold for the adjusted p-value for the logOdd score. Only peaks with adjusted p-value *less* than the pvalAdjCutoff will be reported. Default: 1 (i.e. no cutoff).
pvalAdjCutoff	Threshold for the adjusted p-value for the logOdd score. Only peaks with adjusted p-value *less* than the pvalAdjCutoff will be reported. Default: 1 (i.e. no cutoff).
eFDRCutoff	Threshold for the empirical false discovery rate (eFDR). Only peaks with eFDR *less* than the pvalAdjCutoff will be reported. Default: 1 (i.e. no cutoff).

Details

The RIPScore is compupted in computeLogOdd as the log odd ratio of the posterior for the RIP state ($z_i = 2$) over the posterior for the background state ($z_i = 1$) in RIP library. When control is available, the RIPScore is updated by the difference between the RIPScores between RIP and control. The adjacent bins with hidden states predicted by nbh_vit as the enriched state (corresponding to the NB with larger mean) are merged. The RIPSscores are averaged over the merged bins. To assess the statistical significance of the RIPScore for each region, we assume that the RIPScore follows a *Gaussian* (Normal) distribution with mean and standard deviation estimated using the RIPScores over all of the bins. The rationale is based on the assumption that most of the RIPScores correspond to the background state and together contribute to a stable estimate of the test statistics (TS) and p-value computed using the R built-in function pnorm.

The p-value is adjusted by p.adjust with BH method by default. The same procedure is applied optionally to the control library. Only when the control is available, is an empirical false discovery rate (eFDR) estimated based on the idea of "sample swap" inspired by MACS (a ChIP-seq algorithm from Zhange *el al.* (2008). At each p-value, RIPSeeker finds the number of significnat RIP-regions over control (CTL) based on pvalRIP and the number of significant control regions over RIP based on pvalCTL. The eFDR is defined as the ratio of the number of "RIP" (false positive) regions identified from CTL-RIP comparison over the number of RIP regions from the RIP-CTL comparison. The maximum value for eFDR is 1 and minimum value for eFDR is max(p-value, 0). The former takes care of the case where the numerator is bigger than the denominator, and the latter for zero numerator.

Value

GRanges object containing the merged bins with values slot saved for RIPScore (lodOdd), p-value (pval), adjusted p-value (pvalAdj) for RIP and optionally for control.

Note

Internal function used by ripSeek.

Author(s)

Yue Li

References

Yong Zhang, Tao Liu, Clifford A Meyer, J\'er\^ome Eeckhoute, David S Johnson, Bradley E Bernstein, Chad Nusbaum, Richard M Myers, Myles Brown, Wei Li, and X Shirley Liu. Model-based analysis of ChIP-Seq (MACS). Genome Biology, 9(9):R137, 2008.

See Also

logScoreWithControl,logScoreWithoutControl,empiricalFDR,computeLogOdd,scoreMergedBins,ripSeek

Examples

```
if(interactive()) {
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

# Parameters setting
binSize <- 1e5        # use a large fixed bin size for demo only
multicore <- FALSE    # use multicore
strandType <- "-" # set strand type to minus strand

################ run main function for HMM inference on all chromosomes ################
mainSeekOutputRIP <- mainSeek(bamFiles=grep(pattern="SRR039214",
    bamFiles, value=TRUE, invert=TRUE),
binSize=binSize, strandType=strandType,
reverseComplement=TRUE, genomeBuild="mm9",
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = FALSE,
multicore=multicore, silentMain=FALSE, verbose=TRUE)

mainSeekOutputCTL <- mainSeek(bamFiles=grep(pattern="SRR039214",
    bamFiles, value=TRUE, invert=FALSE),
binSize=binSize, strandType=strandType,
reverseComplement=TRUE, genomeBuild="mm9",
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = FALSE,
multicore=multicore, silentMain=FALSE, verbose=TRUE)

# with control
ripGR.wicontrol <- seekRIP(mainSeekOutputRIP$nbhGRList$chrX, mainSeekOutputCTL$nbhGRList)

# without control
ripGR.wocontrol <- seekRIP(mainSeekOutputRIP$nbhGRList$chrX)
}
```

---

selectBinSize *Select optimal bin size based on Shimazaki formula*

Description

The function iteratively estimates the cost of increasing bin size within a defined range and finally selects the bin size with minimum cost.

Usage

```
selectBinSize(alignGR, minBinSize, maxBinSize = 1000,
increment = 5, getFullResults = FALSE)
```

Arguments

alignGR	GRanges object of alignments on a single chromosome
minBinSize	Minimum bin size to start with (Default: 5 * read length)
maxBinSize	Maximum bin size to end with (Default: 1000).
increment	Number of bases to increment the bin size in searching for the optimal bin size within the defined range (Default: 5).
getFullResults	Binary indicator. If TRUE, the optimal bin size (with the minimum cost), minimum cost, and all of the bin sizes considered and their costs are returned. If FALSE, only the optimal bin size is returned.

Details

Based on the preprocessed alignments for a chromosome, RIPSeeker divides the chromosome into bins of equal size b and compute the number of reads that b needs to be determined either empirically (e.g., based on the gel-selected length of the RNA fragment) or computationally. If the bin size is too small, the read counts fluctuates greatly, making it difficult to discern the underlying read count distribution. Additionally, input size to HMM increases as bin size decreases. A very small bin size results in a very long Markov chain of read counts to model, making the computation inefficient. On the other hand, if a bin size is too large, resolution becomes poor. Consequently, one cannot detect the local RIP region with subtle but intrinsic difference from the background, and the RIP regions tend to be too wide for designing specific primer for validation.

Intuitively, selecting an appropriate bin size for each chromosome is metaphorically equivalent to choosing an optimal intervals for building a histogram (Song, 2011). Here we implement the algorithm developed by Shimazaki and Shinomoto (2007), which is based on the goodness of the fit of the time histogram to estimate the rate of neural response of an animal to certain stimuli in a spike-in experiment. This approach has been successfully applied in a recently developed ChIP-seq program (Song and Smith, 2011). Algorithm 1 describes the pseudocode adapted from Shimazaki and Shinomoto (2007) that iteratively estimates the cost C of increasing bin size b within a defined range is outlined as follows.

For b = minBinSize to maxBinSize; do

1. Divide chromosome sequence into N bins of width b.

2. Count number of read counts x_i that enter the i'th bin.

3. Compute: $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $v = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$.

4. Compute: $C(b) = \frac{2\bar{x}-v}{b^2}$

End For

Choose b that minimize $C(b)$

Value

When getFullResults is TRUE, return a list containing:

bestBinSize	the optimal bin size (with the minimum cost)
minCosts	cost of the optimal bin size
binSizes	all of the bin sizes considered
costs	all of the costs

When getFullResults is FALSE, only the optimal bin size (bestBinSize) is returned.

Author(s)

Yue Li

References

Hideaki Shimazaki and Shigeru Shinomoto. A method for selecting the bin size of a time histogram. Neural computation, 19(6):1503-1527, June 2007.

Qiang Song and Andrew D. Smith. Identifying dispersed epigenomic domains from ChIP-Seq data. Bioinformatics (Oxford, England), 27(6):870-871, March 2011.

See Also

evalBinSize,binCount

Examples

```
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

alignGal <- getAlignGal(bamFiles[1], reverseComplement=TRUE, genomeBuild="mm9")

alignGR <- as(alignGal, "GRanges")

alignGRList <- GRangesList(as.list(split(alignGR, seqnames(alignGR))))

minBinSize <- 200

maxBinSize <- 1200

gr <- alignGRList$chrX

b <- selectBinSize(gr, minBinSize, maxBinSize, increment=100, getFullResults=TRUE)

plot(b$binSizes, b$costs)

chrname <- as.character(runValue(seqnames(gr)))

chrlen <- seqlengths(gr)[chrname]

legend("topright", box.lty=0,

sprintf("%s: 1-%d;\nTotal mapped reads: %d;\nOptimal bin size = %d bp",

chrname, chrlen, length(gr), b$bestBinSize))
```

statdis *Returns the stationary distribution of a Markov chain.*

Description

Given a transition matrix A, returns the stationary distribution of a Markov chain by computing the eigen vectors of A.

Usage

```
statdis(A)
```

Arguments

A	Transition probability matrix, a squared matrix of probabilities ($0 \leq p \leq 1$) with row and column length equal to that of alpha and beta and row sum and column sum both equal to 1 (within some numerical deviation of 1e-6).

Value

w	Stationary weights for the distributions of K components based on the transition probability matrix.

Author(s)

Yue Li

References

Capp\'e, O. (2001). H2M : A set of MATLAB/OCTAVE functions for the EM estimation of mixtures and hidden Markov models. (http://perso.telecom-paristech.fr/cappe/h2m/)

See Also

nbh_em

Examples

```
# Simulate data
TRANS_s <- matrix(c(0.9, 0.1, 0.3, 0.7), nrow=2, byrow=TRUE)
alpha_s <- c(2, 4)
beta_s  <- c(1, 0.25)
Total <- 100

x <- nbh_gen(TRANS_s, alpha_s, beta_s, Total);

count <- x$count
label <- x$label

Total <- length(count)

# dummy initialization
```

```
TRANS0 <- matrix(rep(0.5,4), 2)

alpha0 <- c(1, 20)

beta0 <- c(1, 1)

NIT_MAX <- 50
TOL <- 1e-100
nbh <- nbh_em(count, TRANS0, alpha0, beta0, NIT_MAX, TOL)

map.accuracy <- length(which(max.col(nbh$postprob) == label))/Total

vit <- nbh_vit(count, nbh$TRANS, nbh$alpha, nbh$beta)

vit.accuracy <- length(which(vit$class == label))/Total

# Plot the marginal distribution (in the stationnary regime)
# Compute negative binomial distributions for all model states
t <- 0:max(count)

tmp <- nbh_em(t, nbh$TRANS, nbh$alpha, nbh$beta, 1)

dens <- tmp$dens

w <- statdis(nbh$TRANS)

# Plot estimate of marginal probabilities
marprob <- apply(t(dens) * (t(w) %*% matrix(1, ncol=length(t))), 2, sum)

plot(t, marprob, pch=8, col="blue", main="Estimated marginal distribution")

# Plot empirical estimated probabilities
dhist <- matrix(0, ncol=length(t))

for(i in t){
dhist[1+i] <- sum(count == i)/Total
}

points(t, dhist, pch=3, col="red")
```

`viewRIP` *Visualize peaks from UCSC genome browser.*

Description

Upload alignments, peaks, statistical scores to UCSC genome browser for comparative visualization of the results and data available in the UCSC database.

Usage

```
viewRIP(seekedRIP, alignGR, alignGRCTL,
binGR = seekedRIP, scoreType = "eFDR",
cutoffLine = 0.001, displayALLChr = FALSE, ...)
```

Arguments

seekedRIP	GRangesList obtained from ripSeek. Each list item represents the RIP peaks on a chromosome accompanied with statistical scores including (read) count, logOddScore, pval, pvalAdj, eFDR for the RIP and control (if available). Please refer to seekRIP for more details.
alignGR	GRanges of read alignments for the RIP.
alignGRCTL	GRanges of read alignments for the control.
binGR	GRanges containing read count column corresponding to the peaks. By default, alignGR is used as binGR to display the read count in RIP condition.
scoreType	Type of statistical score to display as another track in the browser (Default: eFDR). eFDR/pval/pvalAdj is displayed at -log10 scale.
cutoffLine	Draw a cutoffline in the browser to indicate the significance level above which the peaks are considered significant.
displayALLChr	Binary indicator when TRUE upload and display the information for only one chromosome rather than upload all chromosomes (Default: TRUE).
...	Extra arguments are ignored.

Details

The function is a wrapper function of browserSession, track, and browserView.

Note

If input contain multiple chromosomes, then multiple browser window will be open to display each chromosome. A more user-friendly way is to upload all of the information to UCSC and open a single browser for visualization, which may become one of the new features in future release.

Author(s)

Yue Li

References

Michael Lawrence, Vince Carey and Robert Gentleman (). rtracklayer: R interface to genome browsers and their annotation tracks. R package version 1.16.3.

See Also

ripSeek,browserSession,track,browserView

Examples

```
if(interactive()) { # need internet connection
# Retrieve system files
extdata.dir <- system.file("extdata", package="RIPSeeker")

bamFiles <- list.files(extdata.dir, ".bam$", recursive=TRUE, full.names=TRUE)

bamFiles <- grep("PRC2", bamFiles, value=TRUE)

cNAME <- "SRR039214"  # specify control name
```

```
# Parameters setting
binSize <- NULL # automatically determine bin size
minBinSize <- 10000 # min bin size in automatic bin size selection
maxBinSize <- 12000 # max bin size in automatic bin size selection
multicore <- TRUE # use multicore
strandType <- "-" # set strand type to minus strand


################ run main function ripSeek to predict RIP ################
seekOut <- ripSeek(bamPath=bamFiles, cNAME=cNAME,
binSize=binSize, minBinSize = minBinSize,
maxBinSize = maxBinSize, strandType=strandType,
silentMain=TRUE, verbose=FALSE,
reverseComplement=TRUE, genomeBuild="mm9",
uniqueHit = TRUE, assignMultihits = TRUE,
rerunWithDisambiguatedMultihits = TRUE, multicore=multicore)


################ visualization ################

viewRIP(seekOut$RIPGRList$chrX, seekOut$mainSeekOutputRIP$alignGalFiltered,
seekOut$mainSeekOutputCTL$alignGalFiltered, scoreType="eFDR")
}
```

Index