Package 'Rsubread'

October 9, 2013

Type Package

Title Rsubread: an R package for the alignment, summarization and analyses of next-generation sequencing data

Version 1.10.5

Author Wei Shi and Yang Liao with contributions from Jenny Zhiyin Dai and Timothy Triche, Jr.

Maintainer Wei Shi <shi@wehi.edu.au>

Description

This R package provides facilities for processing the read data generated by the next-gen sequencing technologies. These facilities include quality assessment, read alignment, read summarization, exon-exon junction detection, absolute expression calling and SNP discovery. This package can be used to process both short and long reads. It supports major sequencing platforms such as Illumina GA/HiSeq, Roche 454, ABI SOLiD and Ion Torrent.

URL http://bioconductor.org/packages/release/bioc/html/Rsubread.html

License GPL-3

LazyLoad yes

biocViews Sequencing, HighThroughputSequencing

R topics documented:

`align` *The Subread aligner - mapping next-generation sequencing reads via seed-and-vote*

Description

The Subread aligner is a super fast, accurate and scalable read aligner. It is a general purpose read aligner and can be used to map reads generated by both genomic DNA sequencing and RNA-sequencing technologies.

Usage

```
align(index,readfile1,readfile2=NULL,output_file,nsubreads=10,TH1=3,TH2=1,nthreads=1,indels=5,phred
```

Arguments

`index`	character string giving the basename of index file. Index files should be located in the current directory.
`readfile1`	character string giving the name of file which includes sequencing reads. This will be the name of first file of paired-end data are provided. The read file should be in FASTQ or FASTA format.
`readfile2`	character string giving the name of second file when paired-end read data are provided. `NULL` by default.
`output_file`	character string giving the name of output file which includes the mapping results.
`nsubreads`	numeric value giving the number of subreads extracted from each read. 10 by default.
`TH1`	numeric value giving the consensus threshold for reporting a hit. This threshold will be applied to the first read if paired-end read data are provided. 3 by default.
`TH2`	numeric value giving the consensus threhold for the second read in a pair. 1 by default.
`nthreads`	numeric value giving the number of threads used for mapping. 1 by default.
`indels`	numeric value giving the number of insertions/deletions allowed during the mapping. 5 by default.
`phredOffset`	numeric value added to the phred score of each base in each read. `33` by default.

markJunctionReads
: a logical value indicating whether the unmapped bases of discovered junction reads should be marked by "S" operations in their CIGAR strings in the SAM output file. The unmapped bases should originate from an exon which is different from the one the corresponding read was mapped to. 'Subjunc' program can then be called to discover exon junction locations. This options is only applicable to RNAseq data. TRUE by default.

tieBreakQS
: logical. If TRUE, use quality scores to break ties when more than one best locations were found. FALSE by default.

tieBreakHamming
: logical. If TRUE, use Hamming distance to break ties when more than one best locations were found (to use this option, the index must be built with reference sequence included).

unique
: logical. If TRUE, only uniquely mapped reads will be reported (reads mapped to multiple locations in the reference genome will not be reported). This option can be used together with option tieBreakQS or tieBreakHamming.

nBestLocations
: numeric value giving the maximal number of equally-best mapping locations allowed to be reported for the read. 1 by default. The value of nBestLocations has to be within the range of 1 to 16. The number of equally-best locations reported for a read will be less than or equal to this value. For example, if a read has two equally-best mapping locations, but the nBestLocations set to be 5, then only two locations will be reported for this read. Note that the -u takes precedence over nBestLocations, ie. no mapping locations will be reported for multi-mapping reads when unique is set to TRUE.

minFragLength
: numeric value giving the minimum fragment length. 50 by default.

maxFragLength
: numeric value giving the maximum fragment length. 600 by default.

PE_orientation
: character string giving the orientation of the two reads in a pair. "fr" by default, which means the first read is on the forward strand and the second read is on the reverse strand.

DP_GapOpenPenalty
: a numeric value giving the penalty for opening a gap when using the Smith-Waterman dynamic programming algorithm to detect insertions and deletions. The Smith-Waterman algorithm is only applied for those reads which are found to contain insertions or deletions. -2 by default.

DP_GapExtPenalty
: a numeric value giving the penalty for extending the gap, used by the Smith-Waterman algorithm. 0 by default.

DP_MismatchPenalty
: a numeric value giving the penalty for mismatches, used by the Smith-Waterman algorithm. 0 by default.

DP_MatchScore
: a numeric value giving the score for matches used by the Smith-Waterman algorithm. 2 by default.

Details

The Subread aligner is very efficient and accurate in mapping reads generated by the next-gen sequencing technologies. It adopts a mapping paradigm called “seed-and-vote", which extracts a

number of 16mers (called seeds or subreads) for each read and uses these subreads to vote for the mapping location of the read. An in-fill step is then applied to finalize the alignments (Liao et al. 2013).

The Subread aligner is written in C programming language (the C version can be downloaded from http://subread.sourceforge.net/), and this function calls it from R so that R/Bioconductor users can have access to it from their familiar environment.

Subread supports the mappping of both genomic DNA sequencing (gDNA-seq) reads and RNA sequencing (RNA-seq) reads. It can be used to align reads generated from major sequencing platforms including Illumina GA/HiSeq, ABI SOLiD, Roche 454 and Ion Torrent sequencers. Note that to map color-space reads (e.g. SOLiD reads), a color-space index should be built for the reference genome (see buildindex for details).

Subread can map short reads and long reads, as well as reads with variable length (e.g. Roche 454 reads). It only takes $<$20 minutes for Subread to map 10 million 100bp reads to the human genome (one thread). The running time of Subread is determined by the number of subreads extracted from reads (the argument nsubreads), and it only increases slightly with the increase of read length.

The amount of computer memory (RAM) used by this function can be specified via the memory parameter of the buildindex function, which has to called before running this function to perform the read alignment. Subread uses no more than 8GB of memory when mapping reads to the human or mouse genome.

Two key parameters of the Subread aligner are the number of subreads extracted from the read nsubreads and the consensus threshold used for reporting the mapping location of the read TH1 (also TH2 for paired-end read data). Using a higher consensus threshold will reduce the number of incorrect alignments, but it will also reduce the mapping sensitivity. It was found that the default values (10 subreads extracted and a consensus threshold of 3) of these two parameters in this function yielded the best performance (Liao et al. 2013).

The Subread aligner has been demonstrated to have an excellent mapping ccuracy and sensitivity. In particular, it has a superior performance in mapping RNA-seq reads thanks to its capacity in mapping exon-spanning reads (Liao et al. 2013).

Value

No value is produced but a SAM format file is written to the current working directory.

Author(s)

Wei Shi and Yang Liao

References

Yang Liao, Gordon K Smyth and Wei Shi. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. Nucleic Acids Research, 41(10):e108, 2013.

Examples

```
# build an index using a sample reference sequence ('reference.fa') that includes a chromosome called 'chr_dummy' ar
library(Rsubread)
ref <- system.file("extdata","reference.fa",package="Rsubread")
buildindex(basename="./reference_index",reference=ref)
```

```
# align a sample read dataset ('reads.txt') to the sample reference
reads <- system.file("extdata","reads.txt",package="Rsubread")
align(index="./reference_index",readfile1=reads,output_file="./Rsubread_alignment.SAM")
```

`atgcContent` *Calculate percentages of nucletodies A, T, G and C in a sequencing read datafile*

Description

Calculate percentages of nucletodies A, T, G and C

Usage

```
atgcContent(filename, basewise=FALSE)
```

Arguments

`filename`	character string giving the name of input FASTQ/FASTA file
`basewise`	logical. If `TRUE`, nucleotide percentages will be calculated for each base position in the read across all the reads. By default, percentages are calculated for the entire dataset.

Details

Sequencing reads could contain letter "N" besides "A", "T", "G" and "C". Percentage of "N" in the read dataset is calcuated as well.

The `basewise` calculation is useful for examining the GC bias towards the base position in the read. By default, the percentages of nucleotides in the entire dataset will be reported.

Value

A named vector containing percentages for each nucleotide type if `basewise` is `FALSE`. Otherwise, a data matrix containing nucleotide percentages for each base position of the reads.

Author(s)

Zhiyin Dai and Wei Shi

Examples

```
library(Rsubread)
reads <- system.file("extdata","reads.txt",package="Rsubread")
# Fraction of A,T,G and C in the entire dataset
x <- atgcContent(filename=reads,basewise=FALSE)
# Fraction of A,T,G and C at each base location across all the reads
xb <- atgcContent(filename=reads,basewise=TRUE)
```

buildindex *Build index for a reference genome*

Description

An index needs to be built before read mapping can be performed. This function creates a hash table for the reference genome, which can then be used by Subread and Subjunc aligners for read alignment.

Usage

```
buildindex(basename,reference,colorspace=FALSE,memory=3700,TH_subread=24)
```

Arguments

basename	character string giving the basename of created index files.
reference	charater string giving the name of the file containing all the refernece sequences.
colorspace	logical. If TRUE, a color space index will be built. Otherwise, a base space index will be built.
memory	numeric value specifying the amount of memory to be requested in megabytes. 3700 MB by default.
TH_subread	numeric value specifying the threshold for removing highly repetitive subreads (16 mers). 24 by default. Subreads will be excluded from the index if they occur more than threshold number of times in the genome.

Details

This function builds an index for a reference genome and the built index can then be used by Subread (align) and subjunc to map reads. Subread is a general-purpose read aligner that can be used to map both genomic DNA sequencing (gDNA-seq) reads and RNA sequencing (RNA-seq) reads. Subjunc is designed for detecting exon-exon junctions from using RNA-seq data. Subread and Subjunc use a mapping paradigm called 'seed-and-vote', allowing more efficient and accurate read mapping (Liao et al. 2013).

This function generates a hash table for the reference genome, in which keys are subreads (16mers) and values are their locations in the reference genome. Highly repetitive subreads (or uninformative subreads) are excluded from the hash table so as to reduce mapping ambiguity (mapping location of the read is directly determined from the mapping locations of subreads extracted from it). The argument TH_subread specifies the maximal number of times a subread is allowed to occur in the reference genome to be included in the hash table.

The argument memory controls how many index segments will be generated after index building is completed. Because only one index segment is present in the memory at any time when performing read alignments, this argument also controls the amount of computer memory (RAM) being used in read aligning. The default value (3700MB) enables the index (ie. the hash table) built for the human or mouse genome to be partitioned into two segments, each about 4GB in size (including half of the hash table entries and half the actual reference sequences. Note that each reference base in encoded

in 2 bits). The running time of mapping 10 million 100bp reads to the human or mouse genome is typically 30 minutes.

The index for the human or mouse genome can be built into one whole segment by setting `memory` to `8000`. This allows the maximal mapping speed to be achieved. It takes less than 20 minutes to map 10 million reads to the human or mouse genome with this setting.

It takes typically less than one hour and half to build an index for the human or mouse genome. The index only needs to be built once and can be re-used in the subsequent alignments.

Value

No value is produced but index files are written to the current working directory.

Author(s)

Wei Shi and Yang Liao

References

Yang Liao, Gordon K Smyth and Wei Shi. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. Nucleic Acids Research, 41(10):e108, 2013.

Examples

```
# build an index using a sample reference sequence ('reference.fa') that includes a chromosome called 'chr_dummy' ar
library(Rsubread)
ref <- system.file("extdata","reference.fa",package="Rsubread")
buildindex(basename="./reference_index",reference=ref)
```

`callSNPs` *Finding Single Nucleotide Polymorphisms (SNPs) from next-gen sequencing data*

Description

Fisher's exact tests are performed to call SNPs.

Usage

```
callSNPs(SAMfile,refGenomeFile,outputFile,minBaseQuality=13,minReadCoverage=5,minAlleleFraction=0.5
```

Arguments

`SAMfile`	a character string giving the name of a SAM format file.
`refGenomeFile`	a character string giving the name of the reference sequence file (FASTA format).
`outputFile`	a character string giving the name of the output file including called SNPs (BED format).

`minBaseQuality` a numeric value giving the base quality cutoff for selecting the bases included in SNP calling. 13 by default(corresponding to base calling p value of 0.05). Bases with quality scores less than 13 will be excluded from reads when using them to discover SNPs.

`minReadCoverage` a numeric value giving the minimal number of reads required for considering a chromosomal location to include a SNP. 5 by default.

`minAlleleFraction` a numeric value giving the minimal required fraction of reads supporting a called SNP. 0.5 by default.

Details

This function takes as input a SAM format file, which includes mapping results for a set of reads, and then calls SNPs by performing Fisher's exact tests. Bases of low sequencing quality or low mapping quality were removed so as to use high confidence bases to call SNPs at each chromosomal loci.

Value

A BED format file including detailed information about called SNPs, such as chromosomal location, reference base, alternative base, read coverage, allele frequency, p value etc.

Author(s)

Yang Liao and Wei Shi

---

createAnnotationFile *Create an annotation file from a GRanges object, suitable for featureCounts()*

Description

Any of rtracklayer::import.bed('samplesubjunc.bed', asRangedData=FALSE), unlist(spliceGraph(TxDb)), transcripts(TxDb), exons(TxDb), or features(FDB) will produce a GRanges object containing usable features for read counting.

This function converts a suitably streamlined GRanges object into annotations which can then be used by featureCounts() to quickly count aligned reads.

The GRanges object must contain an elementMetadata column named 'id'.

Usage

```
createAnnotationFile(GR, fname=NULL)
write.Rsubread(GR, fname=NULL)
```

Arguments

GR	The GRanges object to convert to an Rsubread annotation file
fname	An optional filename. If none is provided, one will be autogenerated.

Value

None.

Author(s)

Tim Triche, Jr.

See Also

transcripts, exons, features, import.bed, subjunc

Examples

```
## Not run:
  library(TxDb.Hsapiens.UCSC.hg19.lincRNAsTranscripts)
  hg19LincRNAs <- transcripts(TxDb.Hsapiens.UCSC.hg19.lincRNAsTranscripts)
  names(values(hg19LincRNAs)) <- gsub('tx_id','id',names(values(hg19LincRNAs)))
  createAnnotationFile(hg19LincRNAs)

## End(Not run)
```

---

detectionCall *Determine detection p values for each gene in an RNA-seq dataset*

Description

Use GC content adjusted background read counts to determine the detection p values for each gene

Usage

```
detectionCall(dataset, species="hg", plot=FALSE)
```

Arguments

dataset	a character string giving the filename of a SAM format file, which is the output of read alignment.
species	a character string specifing the species. Options are hg and mm.
plot	logical, indicating whether a density plot of detection p values will be generated.

Value

A data frame which includes detection p values and annotation information for each genes.

Author(s)

Zhiyin Dai and Wei Shi

`detectionCallAnnotation` *Generate annotation data used for calculating detection p values*

Description

This is for internal use only.

Usage

```
detectionCallAnnotation(species="hg", binsize=2000)
```

Arguments

`species`	character string specifying the species to analyase
`binsize`	binsize of integenic region

Value

Two files containing annotation information for exons regions and integenic regions, respectively.

Author(s)

Zhiyin Dai and Wei Shi

`featureCounts` *featureCounts: a general-purpose read summarization function*

Description

This function assigns mapped sequencing reads to genomic features

Usage

```
featureCounts(files,file.type="SAM",annot=NULL,genome="mm9",isGTFAnnotationFile=FALSE,GTF.featureTy
```

Arguments

files	a character vector giving names of SAM format files.
file.type	a character string giving the type of files provided. SAM and BAM formats are supported. SAM by default.
annot	a data frame containing annotation information. When annot is NULL, an in-built annotation file will be used. NULL by default.
genome	a character string specifying the reference genome to which the sequencing reads were mapped. It has the values of mm9, mm10 and hg, corresponding to mouse genome build 'mm9', mouse genome build 'mm10' and human genome build 'hg19', respectively. mm9 by default. This argument is only applicable when annot is NULL.
isGTFAnnotationFile	logical indicating if the annotation file provided via annot is in GTF format. FALSE by default. If isGTFAnnotationFile is TRUE, a GTF format annotation file must be provided via annot argument.
GTF.featureType	a character string giving the feature type. Only rows which have the matched feature type in the provided GTF annotation file will be included for read counting. exon by default. This argument is only applicable when isGTFAnnotationFile is TRUE.
GTF.attrType	a character string giving the attribute type (in the GTF annotation) which will be used to group features (eg. exons) into meta-features (eg. genes). gene_id by default. This argument is only applicable when isGTFAnnotationFile is TRUE. This attribute type is usually the gene identifier.
useMetaFeatures	logical indicating whether the read summarization should be performed at the feature level (eg. exons) or meta-feature level (eg genes). If TRUE, features in the annotation (each row is a feature) will be grouped into meta-features using their values in the "GeneID" column or using the "gene_id" attribute in the GTF-format annotation file, and reads will assiged to the meta-features instead of the features. See below for more details.
allowMultiOverlap	logical indicating if a read is allowed to be assigned to more than one feature (or meta-feature) if it is found to overlap with more than one feature (or meta-feature). FALSE by default.
nthreads	integer giving the number of threads used for running this function. 1 by default.
strandSpecific	integer indicating if strand-specific read counting should be performed. It has three possible values: 0 (unstranded), 1 (stranded) and 2 (reversely stranded). 0 by default.
countMultiMappingReads	logical indicating if multi-mapping reads/fragments should be counted, FALSE by default. If TRUE, a multi-mapping read will be counted up to N times if it has N reported mapping locations. This function uses the 'NH' tag to find multi-mapping reads.

`minMQS` integer giving the minimum mapping quality score a read must have so as to be counted. For paired-end reads, at least one end should satisfy this criteria. 0 by default.

`isPairedEnd` logical indicating if paired-end reads are used. If TRUE, fragments (two reads are generated from each fragment in paired-end data) will be counted rather than individual reads. FALSE by default.

`requireBothEndsMapped` logical indicating if both ends from the same fragment are required to be successfully aligned before the fragment can be assigned to a feature or meta-feature. This parameter is only appliable when `isPairedEnd` is TRUE.

`checkFragLength` logical indicating if the two ends from the same fragment are required to satisify the fragment length criteria before the fragment can be assigned to a feature or meta-feature. This parameter is only appliable when `isPairedEnd` is TRUE. The fragment length criteria are specified via `minFragLength` and `maxFragLength`.

`minFragLength` integer giving the minimum fragment length for paired-end reads. 50 by default.

`maxFragLength` integer giving the maximum fragment length for paired-end reads. 600 by default. `minFragLength` and `maxFragLength` are only applicable when `isPairedEnd` is TRUE. Note that when a fragment spans two or more exons, the observed fragment length might be much bigger than the nominal fragment length.

`countChimericFragments` logical indicating whether a chimeric fragment, which has its two reads mapped to different chromosomes, should be counted or not. If this fragment overlaps with only one feature (or one meta-feature), typically by one of its two read, this fragment will be assigned to that feature (or meta-feature). If it is found to overlap more than one feature (or meta-feature), for example one of its two reads overlaps meta-feature A and the other overlaps meta-feature B, and `allowMultiOverlap` is FALSE, then this fragment will not be counted. This parameter is only appliable when `isPairedEnd` is TRUE.

Details

`featureCounts` is a general-purpose read summarization function, which assigns to the genomic features (or meta-features) the mapped reads that were generated from genomic DNA and RNA sequencing.

This function takes as input a set of files containing read mapping results output from a read aligner (e.g. `align`), and then assigns the mapped reads to the genomic features. The acceptable formats of the input files are SAM and BAM.

The argument `useMetaFeatures` specifies the read summarization should be performed at the feature level or at the meta-feature level. Each entry in the annotation data is a feature, which for example could be an exon. When `useMetaFeatures` is TRUE, the `featureCounts` function creates meta-features by grouping features using the gene identifiers included in the "GeneID" column in the annotation data (or in the "gene_id" attribute in the GTF format annotation file) and then assigns reads to meta-features instead of features. The `useMetaFeatures` is particularly useful for gene-level expression analysis, because it instructs this function to count reads for genes (meta-features) instead of exons (features). Note that when meta-features are used in the read summarization, if

a read is found to overlap two or more features belong to the same meta-feature it will be only counted once for that meta-feature.

The argument allowMultiOverlap specifies how those reads, which are found to overlap with more than one feature (or meta-feature), should be assigned. When allowMultiOverlap is FALSE, a read overlapping multiple features (or meta-features) will not be assigned to any of them (not counted). Otherwise, it will be assigned to all of them. A read overlaps a meta-feature if it overlaps at least one of the features belonging to this meta-feature.

gene and exon are typically used when summarizing RNA-seq read data, which will yield read counts for genes and exons, respectively.

In-built annotation files for a few model organisms are included in the Rsubread package and these annotations can be easily used with this function to summarize reads for genes and exons. These annotations were downloaded from the NCBI ftp server (ftp://ftp.ncbi.nlm.nih.gov/genomes/) and then processed to remove redundant chromosomal regions within each gene. The resulting annotations contain non-overlapping exon regions for each gene. Users can use these in-built annotation files simply by setting the parameter genome to a proper value (mm9, mm10 or hg). The format of in-built annotation files is the same as that of an example format shown below (a data frame provided by users).

This function supports the use of GTF format annotation file. Users should be provide the name of the annotation file via the annot argument and set the argument isGTFAnnotationFile to be TRUE. This function uses the 'gene_id' attribute to group features to form meta-features if meta-feature level read summarization is needed.

Alternatively, users may create a data frame including the annotation information and then provide that data frame to this function via the annot parameter for read summarization. The data frame must have the following format:

```
GeneID Chr Start End Strand
497097 chr1 3204563 3207049 -
497097 chr1 3411783 3411982 -
497097 chr1 3660633 3661579 -
100503874 chr1 3637390 3640590 -
100503874 chr1 3648928 3648985 -
100038431 chr1 3670236 3671869 -
...
```

The GeneID column may include numbers or character strings. Note that the chromosomal names included in the Chr column must match the chromosomal names used in the sequences of the reference genome to which the reads were mapped, otherwise reads cannot be assigned to the features.

When paired-end read data are provided and the argument isPairedEnd is set to TRUE, fragments will be counted for each feature/meta-feature instead of reads. For RNA-seq data, the observed lengths of some fragments could be much greater than their nominal fragment length when they span two or more exons. Such fragments can be counted when a large value of the argument maxFragLength is used. Note that if the two reads from the same fragment do not meet the fragment length criteria, the Subread (align) and Subjunc (subjunc) aligners will treat them as single-end reads and map them individually.

Value

A list with the following components:

counts:	a data matrix containing read counts for each feature in each library
annotation:	a data frame including columns named GeneID, Chr, Start and End. The GeneID column includes Entrez gene
targets:	a character vector giving sample information.

Author(s)

Wei Shi and Yang Liao

Examples

```
## Not run:
library(Rsubread)

# Summarizing single-end reads using built-in RefSeq annotation for mouse genome mm9:
featureCounts(files="mapping_results_SE.sam",genome="mm9")

# Summarizing single-end reads using a user-provided GTF annotation file:
featureCounts(files="mapping_results_SE.sam",annot="annotation.gtf",isGTFAnnotationFile=TRUE,GTF.featureType="

# Summarizing single-end reads using 5 threads:
featureCounts(files="mapping_results_SE.sam",nthreads=5,annot="annotation.gtf",isGTFAnnotationFile=TRUE,GTF.fe

# Summarizing BAM format single-end read data:
featureCounts(files="mapping_results_SE.bam",file.type="BAM",annot="annotation.gtf",isGTFAnnotationFile=TRUE,G

# Performing strand-specific read counting (strandSpecific=2 if reversely stranded):
featureCounts(files="mapping_results_SE.sam",strandSpecific=1,annot="annotation.gtf",isGTFAnnotationFile=TRUE,

# Summarizing paired-end reads and counting fragments (instead of reads):
featureCounts(files="mapping_results_PE.sam",isPairedEnd=TRUE,annot="annotation.gtf",isGTFAnnotationFile=TRUE,

# Counting fragments satisfying the fragment length criteria, eg. [50bp, 600bp]:
featureCounts(files="mapping_results_PE.sam",isPairedEnd=TRUE,checkFragLength=TRUE,minFragLength=50,maxFragLen

# Counting fragments that have both ends successfully aligned without checking the fragment length:
featureCounts(files="mapping_results_PE.sam",isPairedEnd=TRUE,requireBothEndsMapped=TRUE,annot="annotation.gtf

# Excluding chimeric fragments from the fragment counting:
featureCounts(files="mapping_results_PE.sam",isPairedEnd=TRUE,countChimericFragments=FALSE,annot="annotation.g

## End(Not run)
```

---

processExons *Obtain chromosomal coordiates of each exon using NCBI annotation*

Description

This is for internal use.

Usage

```
processExons(filename="human_seq_gene.md", species="hg")
```

Arguments

filename	a character string giving the name of input .md file (NCBI annotation file)
species	a character string specifying the species

Details

The NCBI annotation file gives the chromosomal coordinates of UTR (Untranslated region) and CDS (Coding sequence). This function uses these information to derive the chromosomal coordinates of exons. The first and last exons of genes usually contain both UTR sequence and CDS sequence.

Value

A text file containing chromosomal coordinates of each exon.

Author(s)

Zhiyin Dai and Wei Shi

---

propmapped *Obtain the proportion of mapped reads*

Description

Use mapping information stored in a SAM format file to count the number of mapped reads

Usage

```
propmapped(samfiles)
```

Arguments

samfiles	a character vector giving the names of SAM format files.

Details

This function counts of number of mapped reads using the mapping informtion stored in SAM format files.

Value

A data frame containing the total number of reads, number of mapped reads and proportion of mapped reads for each library.

Author(s)

Wei Shi

Examples

```
# build an index using the sample reference sequence provided in the package
# and save it to the current directory
library(Rsubread)
ref <- system.file("extdata","reference.fa",package="Rsubread")
buildindex(basename="./reference_index",reference=ref)

# align the sample read data provided in this packge to the sample reference
# and save the mapping results to the current directory
reads <- system.file("extdata","reads.txt",package="Rsubread")
align(index="./reference_index",readfile1=reads,output_file="./Rsubread_alignment.SAM")

# get the percentage of successfully mapped reads
propmapped("./Rsubread_alignment.SAM")
```

---

qualityScores *Extract quality score infromation from a sequencing read dataset*

Description

Extract quality scores and convert them to ASCII code

Usage

```
qualityScores(filename, offset=64, nreads=10000)
```

Arguments

filename	character string giving the name of input FASTQ file.
offset	numeric value giving the offset added to the original quality score, 64 by default.
nreads	numeric value giving the number of reads from which quality scores are extracted

Details

Quality scores are given in the form of characters in datasets which contain sequencing reads. This function extracts the quality scores and then convert them to the ASCII codes which encode these characters. These ASCII codes are then subtracted by the offset to obtain the original quality scores.

If the total number of reads is n, then every n/nreads read will be used for quality score retrieval.

Value

A data matrix containing the quality scores with rows being reads and columns being base positions in the read.

Author(s)

Zhiyin Dai and Wei Shi

Examples

```
library(Rsubread)
reads <- system.file("extdata","reads.txt",package="Rsubread")
x <- qualityScores(filename=reads,nreads=1000)
boxplot(x)
```

---

removeDupReads *Remove sequencing reads which are mapped to identical locations*

Description

Remove reads which are mapped to identical locations, using mapping location of the first base of each read.

Usage

```
removeDupReads(SAMfile,threshold=50,nthreads=1,outputFile)
```

Arguments

SAMfile	a character string giving the name of a SAM format input file.
threshold	a numeric value giving the threshold for removing duplicated reads, 50 by default. Reads will be removed if they are found to be duplicated equal to or more than threshold times.
nthreads	a numeric value giving the number of CPUs used.
outputFile	a character string giving the base name of output files.

Details

This function uses the mapping location of first base of each read to find duplicated reads. Reads are removed if they are duplicated more than threshold times.\

Value

A SAM file which includes the remaining reads after duplicate removal.

Author(s)

Yang Liao and Wei Shi

RsubreadUsersGuide *View Rsubread Users Guide*

Description

Users Guide for Rsubread and Subread

Usage

```
RsubreadUsersGuide()
```

Details

The Subread/Rsubread Users Guide provides detailed description to the functions and programs included in the Subread and Rsubread software packages. It also includes case studies for analyzing next-gen sequencing data.

The Subread package is written in C and it can be downloaded from http://subread.sourceforge.net. The Rsubread package provides R wrappers functions for many of the programs included in Subread package.

Value

Character string giving the file location.

Author(s)

Wei Shi

See Also

vignette

sam2bed *Convert a SAM format file to a BED format file*

Description

SAM to BED conversion

Usage

```
sam2bed(samfile,bedfile,readlen)
```

Arguments

samfile	character string giving the name of input file. Input format should be in SAM format.
bedfile	character string giving the name of output file. Output file is in BED format.
readlen	numeric value giving the length of reads included in the input file.

Details

This function converts a SAM format file to a BED format file, which can then be displayed in a genome browser like UCSC genome browser, IGB, IGV.

Value

There is no return value, but a BED format file is created in the current working directory. This file contains six columns including chromosomal name, start position, end position, name('.'), mapping quality score and strandness.

Author(s)

Wei Shi

---

subjunc *Discovering exon-exon junctions from using RNA-seq data*

Description

Use the seed-and-vote paradigm to accurately detect exon-exon junctions from RNA-seq data

Usage

```
subjunc(index,samfile,output_file,nsubreads=14,paired_end=FALSE,nthreads=1,indels=5,minFragLength=5
```

Arguments

index	character string giving the basename of index file. Index files should be located in the current directory.
samfile	character string giving the name of SAM file generated from read alignment (eg. output from `align` function).
output_file	character string giving the name of output file which includes the mapping results.
nsubreads	numeric value giving the number of subreads extracted from each read. `14` by default.
paired_end	logical, indicating whether the data is paired ended or single ended. `FALSE` by default.
nthreads	numeric value giving the number of threads used for mapping. 1 by default.

indels	numeric value giving the number of insertions/deletions allowed during the mapping. 5 by default.
minFragLength	numeric value giving the minimum fragment length. 50 by default.
maxFragLength	numeric value giving the maximum fragment length. 600 by default.
PE_orientation	character string giving the orientation of the two reads in a pair. "fr" by default, which means the first read is on the forward strand and the second read is on the reverse strand.

Details

The RNA-seq technology provides a unique opportunity to identify the alternative splicing events that occur during the gene transcription process. This function makes use of a powerful read mapping paradigm called "seed-and-vote" to accurately detect the exon-exon junctions (Liao et al. 2013).

This function takes as input the mapping results (in SAM format) from a read aligner, preferably the Subread aligner included in this package (`align`). It extracts a number of subreads (16mers) from each read, maps them to the reference genome and then identifies the two best mapping locations for each read. A verification step is then applied to determine whether each read should be mapped as an exon-spanning read or not and to report the discovered exon-exon junctions. The donor and receptor sites, which are splicing signals, are required to be present when calling exon-exon junctions. The indels identified in the mapped reads included in the provided SAM file will be kept in the output of this function. This function also reports the indels discovered in the exon-spanning reads.

It has the capacity to detect junction locations that are present at any position of the reads. The pair of exons that are spliced together are allowed to be up to 500kb apart in the genome.

This function can run on both single-ended and paired-end data.

It calls the *Subjunc* program (written in C programming language) included in the Subread package (http://subread.sourceforge.net/) for this analysis.

Value

This function outputs two files: one is a SAM format file which includes the mapping results for all the reads (including both junction reads and exonic reads), and the other is a BED format file which inlcudes pairs of splicing points (exon-exon junctions) found from each junction read, number of supporting junction reads for each exon-exon junction and so on.

Author(s)

Wei Shi and Yang Liao

References

Yang Liao, Gordon K Smyth and Wei Shi. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. Nucleic Acids Research, 41(10):e108, 2013.

Index