Package 'Rsubread'

April 5, 2014

Type Package

Title Rsubread: high-performance read alignment, quantification and mutation discovery

Version 1.12.6

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 easy-to-use tools for analyzing next-gen sequencing read data. Functions of these tools include quality assessment, read alignment, read summarization, exon-exon junction detection, absolute expression calling and SNP calling. These tools are highly efficient and accurate. They can be used to anlayze data generated from all major sequencing platforms such as Illumina GA/HiSeq/MiSeq, Roche GS-FLX, ABI SOLiD and LifeTech Ion PGM/Proton sequencers. This package can be installed on multiple operating systems incluidng Linux, Mac OS X, FreeBSD and Solaris.

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

License GPL-3

LazyLoad yes

biocViews Bioinformatics, Sequencing, HighThroughputSequencing,SequenceMatching, RNAseq, ChIPseq, GeneExpression,GeneRegulation, Genetics, SNP, GeneticVariability,Preprocessing, QualityControl, SequenceAnnotation, Software

R topics documented:

align 2
atgcContent 6
buildindex 7
createAnnotationFile 8
detectionCall 9
detectionCallAnnotation 10
exactSNP 10
featureCounts 12

getInBuiltAnnotation 17
processExons 18
propmapped 18
qualityScores 19
removeDupReads 20
RsubreadUsersGuide 21
sam2bed 22

Index **23**

`align` *Read mapping for genomic DNA-seq and RNA-seq data via seed-and-vote (Subread and Subjunc)*

Description

Subread and Subjunc perform local and global alignments respectively. The seed-and-vote paradigm enables efficient and accurate alignments to be carried out.

Usage

```
align(index,readfile1,readfile2=NULL,input_format="FASTQ",output_format="SAM",
output_file=paste(readfile1,"subread",output_format,sep="."),nsubreads=10,TH1=3,TH2=1,
nthreads=1,indels=5,phredOffset=33,tieBreakQS=FALSE,tieBreakHamming=TRUE,unique=TRUE,nBestLocations
minFragLength=50,maxFragLength=600,PE_orientation="fr",nTrim5=0,nTrim3=0,readGroupID=NULL,readGroup
DP_GapOpenPenalty=-1,DP_GapExtPenalty=0,DP_MismatchPenalty=0,DP_MatchScore=2,reportFusions=FALSE)

subjunc(index,readfile1,readfile2=NULL,input_format="FASTQ",output_format="SAM",
output_file=paste(readfile1,"subjunc",output_format,sep="."),nsubreads=14,TH1=1,TH2=1,
nthreads=1,indels=5,phredOffset=33,tieBreakQS=FALSE,tieBreakHamming=TRUE,unique=TRUE,
minFragLength=50,maxFragLength=600,PE_orientation="fr",nTrim5=0,nTrim3=0,readGroupID=NULL,readGroup
DNAseq=FALSE,reportAllJunctions=FALSE)
```

Arguments

`index`	character string giving the basename of index file. Index files should be located in the current directory.
`readfile1`	a character vector including names of files that include sequence reads to be aligned. For paired-end reads, this gives the list of files including first reads in each library. File format is FASTQ/FASTA by default. See `input_format` option for more supported formats.
`readfile2`	a character vector giving names of files that include second reads in paired-end read data. Files included in `readfile2` should be in the same order as their mate files included in `readfile1` .`NULL` by default.
`input_format`	character string specifying format of the read input file(s). `FASTQ` by default (this includes FASTA format as well). Acceptable formats include `FASTQ` (including FASTA), `gzFASTQ` (gzipped FASTQ or FASTA), `SAM` and `BAM`. The character string is case insensitive.

output_format — character string specifying format of the output file. SAM by default. Acceptable formats include SAM and BAM.

output_file — a character vector specifying names of output files. By default, names of output files are set as the file names provided in readfile1 added with an suffix string.

nsubreads — numeric value giving the number of subreads extracted from each read.

TH1 — numeric value giving the consensus threshold for reporting a hit. This is the threshold for the first reads if paired-end read data are provided.

TH2 — numeric value giving the consensus threhold for the second reads in paired-end data.

nthreads — numeric value giving the number of threads used for mapping. 1 by default.

indels — numeric value giving the maximum number of insertions/deletions allowed during the mapping. 5 by default.

phredOffset — numeric value added to the Phred quality scores of read bases by the sequencer. 33 by default.

tieBreakQS — logical indicating if the mapping quality score should be to break the tie when more than one best location was found for a read. FALSE by default. Note that the mapping quality score used for tie breaking was calcuated only from the perfectly matched subreads (16mers) extracted from the read, whereas the mapping quality scores included in the mapping output for mapped reads were calcuated from using all the bases in the read. Also note that there may still be more than one best mapping location found after tie breaking using this option.

tieBreakHamming — logical indicating if the Hamming distance should be used to break the tie when more than one best mapping location was found for a read. TRUE by default. The distance between the mapped read and the target region in the reference is calcuated using all the bases included in the read. Note that there may still be more than one best mapping location found after tie breaking using this option.

unique — logical indicating if uniquely mapped reads should be reported only. TRUE by default. It is recommended that this option is used together with tieBreakHamming or tieBreakQS.

nBestLocations — numeric value giving the maximal number of equally-best mapping locations allowed to be reported for the read. 1 by default. The allowed value is between 1 to 16 (inclusive). Note that the unique argument takes precedence over nBestLocations argument.

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 from the same pair. It has three possible values including fr, ff and rf. Letter f denotes the forward strand and letter r the reverse strand. fr by default (ie. the first read in the pair is on the forward strand and the second read on the reverse strand).

nTrim5 — numeric value giving the number of bases trimmed off from 5' end of each read. 0 by default.

nTrim3 — numeric value giving the number of bases trimmed off from 3' end of each read. 0 by default.

readGroupID a character string giving the read group ID. The specified string is added to the read group header field and also be added to each read in the mapping output. NULL by default.

readGroup a character string in the format of tag:value. This string will be added to the read group (RG) header in the mapping output. NULL by default.

color2base logical. If TRUE, color-space read bases will be converted to base-space bases in the mapping output. Note that the mapping itself will still be performed at color-space. FALSE by default.

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. -1 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.

reportFusions logical indicating if genomic fusion events such as chimeras should be reported. If TRUE, align function will detect and report such events. This option should only be applied for genomic DNA sequencing data. For RNA-seq data, users should use subjunc with the reportAllJunctions option for detection of fusions. Discovered fusions will be saved to a file (*.fusions.txt). Detailed mapping results for fusion reads will also be saved to the SAM/BAM output file. Secondary alignments of fusion reads will be saved to the following optional fields: CC(Chr), CP(Position), CG(CIGAR) and CT(strand). Note that each fusion read occupies only one row in the SAM/BAM output file.

DNAseq logical indicating if the input read data are genomic DNA sequencing data. If TRUE, subjunc function will ignore the splicing signals (donor/receptor sites) when searching for junctions. Junctions that occur within the same chromosome or across different chromosomes (chimerism) will all be reported for DNA seqence data.

reportAllJunctions
logical. This argument should be used for RNA-seq data. If TRUE, subjunc function will report those junctions that do not have the required donor/receptor sites (GT/AG), or cross different chromosomes or are located on different strands within the same chromosome, in addition to the canonical exon-exon junctions.

Details

The align function implements the Subread aligner (Liao et al., 2013) that uses a new mapping paradigm called “seed-and-vote". Subread is general-purpose aligner that can be used to align both genomic DNA-seq reads and RNA-seq reads.

Subjunc is designed for mapping RNA-seq reads. The major difference between Subjunc and Subread is that Subjunc reports discovered exon-exon junctions and it also performs full alignments for every read including exon-spanning reads. Subread uses the largest mappable regions in the reads to find their mapping locations. The seed-and-vote paradigm has been found to be not only more accurate than the conventional seed-and-extend (adopted by most aligners) in read mapping, but it is a lot more efficient (Liao et al., 2013).

Both Subread and Subjunc 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 and Subjunc have adjustable memory usage. See buildindex function for more details.

Mapping performance is largely determined by the number of subreads extracted from each read nsubreads and the consensus threshold TH1 (also TH2 for paired-end read data). Default settings are recommended for most of the read mapping tasks.

Multiple input files (libraries) are allowed to be provided to Subread or Subjunc functions.

Subjunc requires donor/receptor sites to be present when detecting exon-exon junctions. It can detect up to four junction locations in a single exon-spanning reads.

Value

No value is produced but SAM or BAM format files are written to the current working directory. For Subjunc, BED files including discovered exon-exon junctions are also 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 and 90
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 and 90
library(Rsubread)
ref <- system.file("extdata","reference.fa",package="Rsubread")
buildindex(basename="./reference_index",reference=ref)
```

---

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)
write.Rsubread(GR)
```

Arguments

GR	The GRanges object to convert to an Rsubread annotation file

Value

A data frame with five columns named GeneID, Chr, Start, End and Strand.

Author(s)

Tim Triche, Jr. and Wei Shi

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)))
  annot_for_featureCounts <- 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

`exactSNP` *exactSNP - an accurate and efficient SNP caller*

Description

Measure background noises and perform Fisher's Exact tests to detect SNPs.

Usage

```
exactSNP(readFile,isBAM=FALSE,refGenomeFile,SNPAnnotationFile=NULL,
outputFile=paste(readFile,".exactSNP.VCF",sep=""),qvalueCutoff=12,minAllelicFraction=0,
minAllelicBases=1,minReads=1,maxReads=3000,minBaseQuality=13,nTrimmedBases=3,nthreads=1)
```

Arguments

readFile	a character string giving the name of a file including read mapping results. This function takes as input a SAM file by default. If a BAM file is provided, the isBAM argument should be set to TRUE.
isBAM	logical indicating if the file provided via readFile is a BAM file. FALSE by default.
refGenomeFile	a character string giving the name of a file that includes reference sequences (FASTA format).
SNPAnnotationFile	a character string giving the name of a VCF-format file that includes annotated SNPs. Such annotation files can be downloaded from public databases such as the dbSNP database. Incorporating known SNPs into SNP calling has been found to be helpful. However note that the annotated SNPs may or may not be called for the sample being analyzed.
outputFile	a character string giving the name of the output file to be generated by this function. The output file includes all the reported SNPs (in VCF format). It includes discovered indels as well.
qvalueCutoff	a numeric value giving the q-value cutoff for SNP calling at sequencing depth of 50X. 12 by default. The q-value is calcuated as -log10(p), where p is the p-value yielded from the Fisher's Exact test. Note that this function automatically adjusts the q-value cutoff for each chromosomal location according to its sequencing depth, based on this cutoff.
minAllelicFraction	a numeric value giving the minimum fraction of allelic bases out of all read bases included at a chromosomal location required for SNP calling. Its value must be within 0 and 1. 0 by default.
minAllelicBases	a numeric value giving the minimum number of allelic (mis-matched) bases a SNP must have at a chromosomal location. 1 by default.
minReads	a numeric value giving the minimum number of mapped reads a SNP-containing location must have (ie. the minimum coverage). 1 by default.
maxReads	Specify the maximum number of mapped reads a SNP-containing location can have. 3000 by default. Any location having more than this threshold number of reads will not be considered for SNP calling. This option is useful for removing PCR artefacts.
minBaseQuality	a numeric value giving the minimum base quality score (Phred score) read bases should satisfy before being used for SNP calling. 13 by default(corresponding to base calling p value of 0.05). Read bases with quality scores less than 13 will be excluded from analysis.
nTrimmedBases	a numeric value giving the number of bases trimmed off from each end of the read. 3 by default.
nthreads	a numeric value giving the number of threads/CPUs used. 1 by default.

Details

This function takes as input a SAM/BAM format file, measures local background noise for each chromosomal location and then performs Fisher's exact tests to find statistically significant SNPs .

This function implements a novel algorithm for discovering SNPs. This algorithm is comparable with or better than existing SNP callers, but it is fast more efficient. It can be used to call SNPs for individual samples (ie. no control samples are required). Detail of the algorithm is described in a manuscript which is currently under preparation.

Value

A VCF format file including detailed information for reported SNPs, such as chromosomal location, reference base, alternative base, read coverage, allele frequency, p value etc.

Author(s)

Yang Liao and Wei Shi

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

Description

This function assigns mapped sequencing reads to genomic features

Usage

```
featureCounts(files,annot.inbuilt="mm9",annot.ext=NULL,isGTFAnnotationFile=FALSE,GTF.featureType="e
useMetaFeatures=TRUE,allowMultiOverlap=FALSE,nthreads=1,strandSpecific=0,countMultiMappingReads=FAL
isPairedEnd=FALSE,requireBothEndsMapped=FALSE,checkFragLength=FALSE,minFragLength=50,maxFragLength=
countChimericFragments=TRUE,chrAliases=NULL,reportReads=FALSE)
```

Arguments

`files`	a character vector giving names of input files containing read mapping results. The files can be in either SAM format or BAM format. The file format is automatically detected by the function.
`annot.inbuilt`	a character string specifying an in-built annotation used for read summarization. It has three possible values including `mm9`, `mm10` and `hg19`, corresponding to the NCBI RefSeq annotations for genomes 'mm9', 'mm10' and 'hg19', respectively. `mm9` by default. The in-built annotation has a SAF format (see below).
`annot.ext`	A character string giving name of a user-provided annotation file or a data frame including user-provided annotation data. If the annotation is in GTF format, it can only be provided as a file. If it is in SAF format, it can be provided as a file or a data frame. See below for more details about SAF format annotation. `annot.ext` will override `annot.inbuilt` if they are both provided.

isGTFAnnotationFile
: logical indicating whether the annotation provided via the annot.ext argument is in GTF format or not. FALSE by default. This option is only applicable when annot.ext is not NULL.

GTF.featureType
: a character string giving the feature type used to select rows in the GTF annotation which will be used for read summarization. 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.

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 in the SAF-format annotation file 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 satisfy in order 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 (templates or read pairs) will be counted instead of 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.

`chrAliases` a character string giving the name of a file that contains aliases of chromosome names. The file should be a comma delimited text file that includes two columns. The first column gives the chromosome names used in the annotation and the second column gives the chromosome names used by reads. This file should not contain header lines. Names included in this file are case sensitive.

`reportReads` logical indicating if read counting result for each read/fragment is saved to a file. If TRUE, read counting results for reads/fragments will be saved to a tab-delimited file that contains four columns including name of read/fragment, status(assigned or the reason if not assigned), name of target feature/meta-feature and number of hits if the read/fragment is counted multiple times. Name of the file is the same as name of the input read file except a suffix '.featureCounts' is added. Multiple files will be generated if there is more than one input read file.

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 mapped reads to genomic features. Both SAM and BAM format input files are accepted.

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.

The in-built annotations for human and mouse genomes (hg19, mm9 and mm10) provided in this function can be conveniently used for read summarization. These annotations were downloaded from the NCBI ftp server (ftp://ftp.ncbi.nlm.nih.gov/genomes/) and were then postprocessed by removing redundant chromosomal regions within each gene and combining adjacent CDS and UTR sequences. The in-built annotations use the SAF annotation format (see below) and their content can be retrieved using the getInBuiltAnnotation function.

Users may also choose to provide their own annotation for summarization. If users provide a SAF (Simplified Annotation Format) annotation, the annotation should 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 SAF annotation format has five required columns, including GeneID, Chr, Start, End and Strand. These columns can be in any order. More columns can be included in the annotation. Columns are tab-delimited. Column names are case insensitive. GeneID column may contain integers or character strings. Chromosomal names included in the Chr column must match those used inclued in the mapping results, otherwise reads will fail to be assigned. Users may provide a SAF annotation in the form of a data frame or a file using the annot.ext argument.

Users may also provide a GTF/GFF format annotation via annot.ext argument. But GTF/GFF annotation should only be provided as a file, and isGTFAnnotationFile should be set to TRUE when such a annotation is provided. featureCounts function uses the 'gene_id' attribute in a GTF/GFF annotation to group features to form meta-features when performing read summarization at meta-feature level.

When isPairedEnd is TRUE, fragments (pairs of reads) instead of reads will be counted. featureCounts function checks if reads from the same pair are adjacent to each other (this could happen when reads were for example sorted by their mapping locations), and it automatically reorders those reads that belong to the same pair but are not adjacent to each other in the input read file.

Value

A list with the following components:

counts:	a data matrix containing read counts for each feature or meta-feature for each library.
annotation:	a data frame with six columns including GeneID, Chr, Start, End and Length. When read summarization was
targets:	a character vector giving sample information.
stat:	a data frame giving numbers of unassigned reads and the reasons why they are not assigned (eg. ambiguity, mu

Author(s)

Wei Shi and Yang Liao

References

Yang Liao, Gordon K Smyth and Wei Shi. featureCounts: an efficient general-purpose program for assigning sequence reads to genomic features. Bioinformatics, accepted on Nov 7, 2013, doi: 10.1093/bioinformatics/btt656.

See Also

getInBuiltAnnotation

Examples

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

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

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

# Summarizing single-end reads using 5 threads:
featureCounts(files="mapping_results_SE.sam",nthreads=5)

# Summarizing BAM format single-end read data:
featureCounts(files="mapping_results_SE.bam")

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

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

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

## End(Not run)
```

`getInBuiltAnnotation` *Retrieve in-built annotations provided by featureCounts function*

Description

Retrieve an in-built annotation and save it to a data frame

Usage

```
getInBuiltAnnotation(annotation="mm9")
```

Arguments

`annotation`	a character string specifying the in-built annotation to be retrieved. It has three possible values including `mm9`, `mm10` and `hg19`, corresponding to the NCBI RefSeq annotations for genomes 'mm9', 'mm10' and 'hg19', respectively. `mm9` by default.

Details

The `featureCounts` read summarization function provides in-built annotations for conveniently summarizing reads to genes or exons, and this function allows users to have access to those in-built annotations.

For more information about these annotations, please refer to the help page for `featureCounts` function.

Value

A data frame with five columns including `GeneID`, `Chr`, `Start`, `End` and `Strand`.

Author(s)

Wei Shi

See Also

`featureCounts`

Examples

```
library(Rsubread)
x <- getInBuiltAnnotation("hg19")
x[1:5,]
```

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 data in a sequencing read dataset*

Description

Extract quality strings and convert them to Phred scores

Usage

```
qualityScores(filename, input_format="gzFASTQ", offset=33, nreads=10000)
```

Arguments

filename	character string giving the name of an input file containing sequence reads.
input_format	character string specifying format of the input file. gzFASTQ (gzipped FASTQ) by default. Acceptable formats include gzFASTQ, FASTQ, SAM and BAM. Character string is case insensitive.
offset	numeric value giving the offset added to the Phred scores, 33 by default.
nreads	numeric value giving the number of reads from which quality scores are extracted. 10000 by default.

Details

Quality scores of read bases are represented by ASCII characters in next-gen sequencing data. This function extracts the quality characters from each base in each read and then converts them to Phred scores using the provided offset value (`offset`).

If the total number of reads in a dataset is `n`, then every `n/nreads` read is extracted from the input data.

Value

A data matrix containing Phred scores for read bases. Rows in the matrix are reads and columns are base positions in each read.

Author(s)

Wei Shi, Yang Liao and Zhiyin Dai

Examples

```
library(Rsubread)
reads <- system.file("extdata","reads.txt",package="Rsubread")
x <- qualityScores(filename=reads,input_format="FASTQ",offset=64,nreads=1000)
x[1:10,1:10]
```

`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,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.
`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` number of times.

Value

A SAM file including 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

Index

∗Topic **documentation**
RsubreadUsersGuide, 21

align, 2, *7*, *14*
atgcContent, 6

buildindex, *5*, 7

createAnnotationFile, 8

detectionCall, 9
detectionCallAnnotation, 10

exactSNP, 10

featureCounts, 12, *17*

getInBuiltAnnotation, *15, 16*, 17

processExons, 18
propmapped, 18

qualityScores, 19

removeDupReads, 20
RsubreadUsersGuide, 21

sam2bed, 22
subjunc, *7*
subjunc *(*align*)*, 2

vignette, *21*

write.Rsubread *(*createAnnotationFile*)*, 8