

Package ‘Rsubread’

March 26, 2013

Type Package

Title Rsubread: a super fast, sensitive and accurate read aligner for mapping next-generation sequencing reads

Version 1.8.2

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

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

Description This package performs read mapping, exon junction detection and other tasks for next-generation sequencing data. The read mapping function implements a novel mapping paradigm based on seed-and-vote. It can be used to map both short reads and long reads (>200bp) and reads of variable lengths. It also provides functions to summarize read counts to genes or exons and gives detection p values for each gene in the RNA-seq data.

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

License GPL-3

LazyLoad yes

biocViews Sequencing, HighThroughputSequencing

R topics documented:

align	2
atgcContent	4
buildindex	5
callSNPs	6
createAnnotationFile	7
detectionCall	8
detectionCallAnnotation	8
featureCounts	9
processExons	10
propmapped	11
qualityScores	12
removeDupReads	13
sam2bed	13
subjunc	14
Index	16

align

*Align next-gen sequencing reads to reference genome***Description**

This function performs read mapping using the seed-and-vote method. It can be used for mapping both gDNA and RNA sequencing data. It supports indel detection.

Usage

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

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 threshold 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.
min_distance	numeric value giving the minimal distance between the pair of reads. 50 by default.
max_distance	numeric value giving the maximal distance between the pair of reads. 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

This function implements a novel mapping strategy which uses a set of 16bp substrings (called subreads) extracted from each read to map them to the reference genome. Different from "seed-and-extend" mapping strategy, this new strategy does not have an extension step therefore it is a lot faster than the competing aligners. Due to the short length of the selected subreads (16bp long), this strategy has a much high sensitivity than other aligners (seed length is usually around 30bp), i.e. it can align a lot more reads than competing aligners. Our evaluation results (using both simulation dataset and real dataset) showed that the accuracy of the new strategy is comparable to or slightly better than other aligners.

Two key parameters used by this new strategy are the number of subreads selected `nsubreads` and the consensus threshold for determining mapping locations `TH1` (also `TH2` for paired-end read data). We recommend using the default setting of these parameters to map reads of around 100bp long. However, users can choose to use more subreads when mapping longer reads. We recommend to set the value of consensus threshold to be 30 percent of the number of subreads used.

The C implementation of this strategy can be found at <http://subread.sourceforge.net/>. This R function calls the corresponding C function to perform the alignment. Therefore, it has the mapping speed as the C program.

`buildindex` function should be called if the index has not been built for the reference genome. The index can be re-used once it has been built.

If paired-end read data is provided, file `readfile1` will assumed to contain the first read from the read pair and `readfile2` the second read.

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*, In Press, Accepted on 8 March 2013.

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 package 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")
```

atgcContent

Calculate percentages of nucleotides A, T, G and C in a sequencing read datafile

Description

Calculate percentages of nucleotides 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 calculated 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 for read mapping***Description**

This function builds an index for the reference genome using a hash table. Memory usage can be adjusted when loading the index into the computer memory.

Usage

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

Arguments

basename	character string giving the basename of created index files.
reference	character string giving the name of the file containing all the reference 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.

Details

A hash table will be built for the reference genome. Keys in the hash table are the 16bp sequences and hash values are their chromosomal locations. A 16bp sequence could have one or more than one chromosomal locations. They are all recorded in the hash table. Non-informative 16bp sequences, which are highly repetitive in the reference genome (occurring 24 times or more in the genome), are excluded from the hash table.

After the index is built, reads can then be mapped to the reference genome by using [align](#) function. It takes around 1 hour to build an index for human genome.

Value

Index files with basename provided in basename. These files are saved in the current directory.

Author(s)

Wei Shi and Yang Liao

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)
```

callSNPs	<i>Finding Single Nucleotide Polymorphisms (SNPs) from next-gen sequencing data</i>
----------	---

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	<i>Create an annotation file from a GRanges object, suitable for featureCounts()</i>
----------------------	--

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 specifying 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 analyse
binsize	binsize of intergenic region

Value

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

Author(s)

Zhiyin Dai and Wei Shi

featureCounts	<i>Count the number of mapped reads for each feature</i>
---------------	--

Description

Summarize read counts to features including genes and exons

Usage

```
featureCounts(SAMfiles,type="gene",species="mm9",annot=NULL,isPairedEnd=FALSE,min_distance=10,r
```

Arguments

SAMfiles	a character vector giving names of SAM format files.
type	a character string giving the feature type. Its value could be gene or exon.
species	a character string specifying the species. The allowed value is mm9, mm10 or hg. Values of this argument determines which in-built annotation file will be used, if annot is NULL.
annot	a character string giving the name of an annotation file provided by the user, or a data frame containing annotation information. If NULL (by default), the in-built annotation file will be used. See below for more details.
isPairedEnd	logical indicating if paired-end reads are used. FALSE by default.
min_distance	integer giving the minimal allowable fragment/template length. 10 by default.
max_distance	integer giving the maximal allowable fragment/template length. 2000 by default.

Details

This function takes as input a set of SAM format files and assigns reads to the features. Currently, only feature types including gene and exon are supported. gene is the aggregation of all the exons for each gene.

There are three in-built annotation files which are used by this function to summarize reads for genes or exons for mm9, mm10 and hg19, respectively. Gene annotation were downloaded from NCBI (<ftp://ftp.ncbi.nlm.nih.gov/genomes/>) and then processed to create annotation files used by this function. The species argument specifies which annotation file should be used. These in-built annotation will be used when annot is NULL.

Users can also provide their own annotation for read summarization, via the annot argument. In this case, user provided annotation will override the in-built annotation. Users can either provide the name of their own annotation file, or they can provide a data frame which contain annotation information. If a data frame is provided, at least four columns, named entrezid, chromosome, chr_start and chr_stop, are required. If users provide an annotation file, then it should be a tab delimited file and its first four columns should be named as entrezid, chromosome, chr_start and chr_stop, respectively. The file has to be sorted by the chromosome column (either increasing or decreasing order is OK), and chromosome names should not contain more than 40 characters. Below is an example for the annotation file provided by the user (sorted by the chromosome column):

```
entrezid chromosome chr_start chr_stop
497097 chr1 3204563 3207049
497097 chr1 3411783 3411982
```

```
497097 chr1 3660633 3661579
100503874 chr1 3637390 3640590
100503874 chr1 3648928 3648985
100038431 chr1 3670236 3671869
...
```

Although this function is designed for summarizing reads from RNA-seq experiments, it can be used to summarize reads from other next-gen sequencing experiments as well, for example ChIP-seq or other DNA sequencing experiments. Simply by setting type to exon and providing an annotation, this function will yield numbers of mapped reads for each feature.

Value

A list with the following components:

counts:	a data matrix containing read counts for each gene or exon.
annotation:	a data frame containing Entrez gene identifiers, gene/exon length etc.
targets:	a character vector giving sample information.

Author(s)

Wei Shi

processExons

Obtain chromosomal coordinates 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	<i>Obtain the proportion of mapped reads</i>
------------	--

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 information 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 package 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 information from a sequencing read dataset*

Description

Extract quality scores and convert them to ASCII code

Usage

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

Arguments

<code>filename</code>	character string giving the name of input FASTQ file.
<code>offset</code>	numeric value giving the offset added to the original quality score, 64 by default.
<code>nreads</code>	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	<i>Remove sequencing reads which are mapped to identical locations</i>
----------------	--

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

sam2bed	<i>Convert SAM format file to BED format</i>
---------	--

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

SAM format is the de facto standard format of output from read aligner. This format not only includes the mapping coordinates of the reads but also includes other using information such as mapping quality, CIGAR information and so on. 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 etc.

Value

A BED format file.

Author(s)

Wei Shi

 subjunc

Discovering exon-exon junctions from RNA-seq data

Description

Discover exon-exon junction locations from RNA-seq data using seed-and-vote paradigm

Usage

```
subjunc(index,samfile,output_file,nsubreads=14,paired_end=FALSE,nthreads=1,indels=5,min_distance=50)
```

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.

min_distance	numeric value giving the minimal distance between the pair of reads. 50 by default.
max_distance	numeric value giving the maximal distance between the pair of reads. 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

This function takes as input the SAM format file generated from read alignment (eg. output from align function), and then tries to discover exon-exon junctions. Donor and receptor sites are being examined when detecting junctions. This function can detect junction locations located at any position of the read. It can detect spliced exons which are as far as 500kb. It supports INDEL detection during the junction detection as well. align function should be run first before calling this function.

This function calls the underlying C program to perform the exon junction discovery. The C program can be run on its own as well, which is included in the subread software package (<http://subread.sourceforge.net/>)

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 includes 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

Index

align, [2](#), [5](#)
atgcContent, [4](#)

buildindex, [3](#), [5](#)

callSNPs, [6](#)
createAnnotationFile, [7](#)

detectionCall, [8](#)
detectionCallAnnotation, [8](#)

featureCounts, [9](#)

processExons, [10](#)
propmapped, [11](#)

qualityScores, [12](#)

removeDupReads, [13](#)

sam2bed, [13](#)
subjunc, [14](#)

write.Rsubread (createAnnotationFile), [7](#)