Package ‘rbamtools’

May 7, 2018

Type Package
Title Read and Write BAM (Binary Alignment) Files
Version 2.16.11
Date 2018-04-14
Description Provides an R interface to functions of the ‘SAMtools’ C-Library by Heng Li <http://www.htslib.org/>.
License Artistic-2.0
Copyright file inst/COPYRIGHTS
LazyLoad yes
Depends methods,refGenome (>= 1.6.1)
Suggests xtable
Collate rbamtools.R segmentize.R sampleBamFiles.R
Repository CRAN
Date/Publication 2018-05-07 12:40:46 UTC
NeedsCompilation yes
Author Wolfgang Kaisers [aut, cre],
Heng Li [ctb, cph],
Jue Ruan [ctb, cph]
Maintainer Wolfgang Kaisers <wolfgang.kaisers@gmail.com>

R topics documented:

rbamtools-package .................................................. 3
alignDepth-class .................................................... 4
as.data.frame-methods ............................................. 6
bamAlign ............................................................. 6
bamAlign-class ...................................................... 7
bamClose-methods .................................................. 10
bamCount ........................................................... 10
bamCountAll ......................................................... 11
bamGapList-class ................................................... 11
bamHeader-class .................................................. 12
bamHeaderText-class ........................................... 13
bamRange .......................................................... 15
bamRange-class ..................................................... 16
bamReader .......................................................... 19
bamReader-class ................................................... 20
bamSave ............................................................ 23
bamSort ............................................................. 24
bamWriter .......................................................... 25
bamWriter-class ................................................... 26
countNucs .......................................................... 28
countTextLines ..................................................... 29
createIdxBatch ..................................................... 30
exonAlignDepth-class ............................................ 30
exonLoessModel-class ............................................. 32
extractBamRegions ............................................... 34
extractGeneRegions ............................................... 35
extractRanges ....................................................... 36
filename-methods ................................................ 37
gapList-class ....................................................... 37
gapSiteList-class ................................................ 38
geneAlignDepth-class ............................................ 40
GenomePartition-class .......................................... 41
getHeaderText-methods .......................................... 43
getNextAlign-methods .......................................... 44
getQualDf .......................................................... 45
getRefData ........................................................ 46
getVal .............................................................. 47
groupRatio ........................................................ 48
headerLine-class .................................................. 50
headerProgram-class ............................................ 51
headerReadGroup-class ......................................... 52
isOpen-methods ................................................... 53
multSeq ............................................................ 54
nucStats ........................................................... 54
rangeSegCount-class ............................................ 55
rangeToFastq ....................................................... 57
readerToFastq ..................................................... 58
readPooledBamGapDf ............................................ 59
readPooledBamGaps ............................................... 59
refSeqDict-class ................................................. 60
sampleBamFiles-class .......................................... 61
saveAldData ....................................................... 63
segmentize ........................................................ 64
setVal .............................................................. 65

Index ................................................................. 68
rbamtools-package

Reading, writing and manipulating BAM-file format.

Description

BAM is Binary (Sequence) Alignment/Map format. Many sequence alignment products which align second generation sequence reads to a genomic reference (such as the human genome) use BAM-file format as output. Analysis of results of a sequence alignment requires reading and interpreting BAM-files and sometimes manipulating BAM-files. The rbamtools package provides a R-interface to the samtools C-library by Heng Li.

Details

Package: rbamtools
Type: Package
Version: 2.0
Date: 2012-02-03
License: Artistic 2.0 + MIT License (see LICENSE in src/samtools subdirectory)
LazyLoad: yes
Depends: methods

The package is organized in S4 classes. Four classes represent the data and organize the defined functionality:

- **bamReader**: Reading aligns from a BAM-file
- **bamAlign**: Single BAM alignment (including Data accessor functions)
- **bamRange**: (Linked-) List container for BAM alignments
- **bamWriter**: Writing aligns to a BAM-file
- **gapList**: (Linked-) List container for alignment gaps.

Author(s)

Wolfgang Kaisers Maintainer: Wolfgang Kaisers <kaisers@med.uni-duesseldorf.de>

References

The SAM Format Specification (v1.4-r985).
The Sequence alignment/map (SAM) format and SAMtools.
Bioinformatics, 25, 2078-9.
Examples

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
idx <- system.file("extdata", "accepted_hits.bam.bai", package="rbamtools")

#
reader<-bamReader(bam)
isOpen(reader)
#
align<-getNextAlign(reader)
#
loadIndex(reader,idx)
indexInitialized(reader)
#
coords<-as.integer(c(0,0,249250621))
range<-bamRange(reader,coords)
align<-getNextAlign(range)
#
bamClose(reader)
```

alignDepth-class

Class "alignDepth": Alignment depth information from bamRange objects.

Description

The alignDepth class represents data and provides plot functions for alignment depth for genomic regions (represented by bamRange objects).

Details

The params vector contains the following values:

1. seqid 0-based index of reference sequence
2. qrBegin 0-based left boundary of query region (query range begin)
3. qrEnd 0-based right boundary of query region (query range end)
4. complex 0= all aligns included, 1= only aligns with n_cigar > 1 included
5. rSeqLen Length of reference sequence
6. qSeqMinLen Minimum of query sequence length (= read length)
7. qSeqMaxLen Maximum of query sequence length (= read length)

Objects from the Class

Objects can be created by calls of the form alignDepth(object, gap)) on bamRange objects. From the bamRange object, the range is extracted and for each nucleotide position within this range the numbers of align matches are calculated. When alignDepth is called with gap=TRUE, the function counts aligns solely for gap-adjacent match regions (cigar-op’s).
Slots

- **depth**: "integer". Align depth values.
- **pos**: "integer". Corresponding (1-based) positions
- **params**: "numeric". Set of internally used parameters
- **refname**: "character". Name of reference sequence from which bamRange was extracted.

Methods

- **show** signature(object="alignDepth"): Prints a short message with some summarizing data.
- **plotAlignDepth** signature(object="alignDepth", main, xlab, ylab, start, end, transcript, xlim, strand): Plots align depth in a line-plot.
- **getDepth** signature(object="alignDepth"): Returns numeric align depth values.
- **getPos** signature(object="alignDepth"): Returns numeric position values for align depth.
- **getParams** signature(object="alignDepth"): Returns numeric parameter values.

Author(s)

Wolfgang Kaisers

Examples

```r
# Open (indexed) BAM file
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader <- bamReader(bam, idx=TRUE)

# Extract reads from BAM file
xlim = c(10000, 30000)
coords <- c(0, xlim[1], xlim[2])
range <- bamRange(reader, coords)
bamClose(reader)

# Calculate align depth
ad <- alignDepth(range)
ad
getParams(ad)

# Prepare plotting parameter
gene <- "WASH7P"
ensg_id <- "ENSG00000272732"
enst_id <- "ENST00000538476"

# Get exon positions
start <- c(14411, 15000, 15796, 15904, 16687, 16748, 16858, 17233, 17602, 17915, 18268, 24737, 29534)
end <- c(14502, 15038, 15901, 15947, 16745, 16765, 17055, 17364, 17742, 18061, 18366, 24891, 29806)

# Do plot
plotAlignDepth(ad, lwd = 2, xlim = xlim,
               main = paste("Align depth for gene", gene),
               ylab = "Align depth", start = start,
               ...)```
as.data.frame-methods  Conversion of bamRange or gapList into a data.frame

Description

as.data.frame functions convert objects of type bamRange or gapList into a data.frame. Data from each element (bamAlign or Align-gap) is written into a single line.

Methods

signature(x = "bamRange") Conversion of bamRange into data.frame
signature(x = "gapList") Conversion of gapList into data.frame

bamAlign

Description

The function takes data stored in aling-fields and creates a bamAlign object, which can be stored in a BAM file via bamWriter.

Usage

bamAlign(qname, qseq, qqual, cigar, refid, position, flag=272L, alqual=10L, mrefid=(-1L), mpos=(-1L), insertsize=0L)

Arguments

qname  Query name, e.g. "HWUSI..." for Illumina sequences.
qseq  Query sequence (DNA-sequence)
qqual  Query quality (ASCII coded quality values). Must contain same number of characters as qseq
cigar  CIGAR string. Must be in valid format, e.g. 45M100N56N. Sequence length must match encoded items in CIGAR string.
refid  Integer. 0-based index which must have a counterpart in Reference Sequence Dictionary (otherwise samtools crashed when creating a BAM-index file).
position  integer: Genomic start position of alignment.
flag  integer: Contains information about binary stored flags in align.
**bamAlign-class**

```r
alqual integer: Alignment quality.
mrefid integer: Mate refid. Used for paired end reads.
mpos integer: Mate position. Used for paired end reads.
insertsize integer.
```

**Value**

bamGapList

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
# Create alignment object from scratch
align <- bamAlign("HMUS1-0001", "ATGTACGTCG", "Qual/String", "4M10N6M", refid=0, position=100)

# Print and access data
align
name(align)
alignSeq(align)
alignQual(align)
cigarData(align)
refID(align)
position(align)
```

---

**Class "bamAlign": Representation of single genomic alignments.**

**Description**

The bamAlign class represents the content of a single alignment stored in a BAM-file. An instance of this class can be created by reading an Alignment from an object of class bamReader or bamRange. Accessor functions allow reading and writing of object contents. The alignment can be rewritten to a BAM-file via a bamWriter object or stored in a bamRange object.

**Objects from the Class**

Objects can be created by calls of the form `new("bamAlign", alignment)`.

**Slots**

```r
align: Object of class "externalptr"
```
Methods

alignQual signature(object="bamAlign"): Returns quality scores of read (one for each query-base).

alignSeq signature(object="bamAlign"): Returns base sequence (of read).

cigarData signature(object="bamAlign"): Returns data.frame with listed cigar items.

failedQC signature(object="bamAlign"): Gets flag item.

failedQC< signature(object="bamAlign", value="logical"): Sets flag item.

firstInPair signature(object="bamAlign"): Gets flag item.

firstInPair< signature(object="bamAlign", value="logical"): Sets flag item.

flag signature(object="bamAlign"): Retrieves 16-bit flag values which contains information of all flags in binary form.

flag< signature(object="bamAlign"): Sets 16-bit flag value (all flags at once).

initialize signature(.Object = "bamAlign"): Initializes bamAlign object

insertSize signature(object="bamAlign"): Returns insertSize value.

mapQuality signature(object="bamAlign"): Returns mapQuality value.

matePosition signature(object="bamAlign"): Returns mate Position value.

mateRefID signature(object="bamAlign"): Returns mateRefID value.

mateReverseStrand signature(object="bamAlign"): Gets flag item.

mateReverseStrand< signature(object="bamAlign", value="logical"): Sets mateReverseStrand item.

mateUnmapped signature(object="bamAlign"): Gets mateUnmapped item.

mateUnmapped< signature(object="bamAlign", value="logical"): Sets mateUnmapped item.

name signature(object="bamAlign"): Returns align name (read Identifier).

nCigar signature(object="bamAlign"): Returns number of cigar items in align.

paired signature(object="bamAlign"): Gets flag item.

paired< signature(object="bamAlign", value="logical"): Sets flag item.

pcrORopt_duplicate signature(object="bamAlign"): Gets flag item.

pcrORopt_duplicate< signature(object="bamAlign", value="logical"): Sets flag item.

position signature(object="bamAlign"): Returns align position (on Ref-Sequence).

properPair signature(object="bamAlign"): Gets flag item.

properPair< signature(object="bamAlign", value="logical"): Sets flag item.

refID signature(object="bamAlign"): Returns (0-based) ID of Reference Sequence (As indicated by ID column returned by getRefData).

reverseStrand signature(object="bamAlign"): Gets flag item.
**reverseStrand**<- signature(object="bamAlign", value="logical"): Sets flag item.

**secondaryAlign** signature(object="bamAlign"): Gets flag item.

**secondaryAlign**<- signature(object="bamAlign", value="logical"): Sets flag item.

**secondInPair** signature(object="bamAlign"): Gets flag item.

**secondInPair**<- signature(object="bamAlign", value="logical"): Sets flag item.

**suppAlign** signature(object="bamAlign"): Gets flag item.

**suppAlign**<- signature(object="bamAlign", value="logical"): Sets flag item.

**unmapped** signature(object="bamAlign"): Gets flag item.

**unmapped**<- signature(object="bamAlign", value="logical"): Sets flag item.

**Author(s)**

Wolfgang Kaisers

**References**

The SAM Format Specification (16ede77).


**Examples**

```r
# Retrieve align from file
bam<-system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader<-bamReader(bam)
align<-getNextAlign(reader)
bamClose(reader)

# Use accessors
name(align)
refID(align)
position(align)
NCigar(align)
cigarData(align)
alignSeq(align)
alignQual(align)
alignQualVal(align)
reverseStrand(align)
reverseStrand(align) <- TRUE
reverseStrand(align)
suppAlign(align)
suppAlign(align) <- TRUE
```
bamClose-methods

*bamClose(bamReader), bamClose(bamWriter): Closing of file connections.*

Description

Closing open file connections for bamReader an bamWriter.

Methods

signature(object = "bamReader") Closes connection of bamReader to BAM-file.
signature(object = "bamWriter") Closes connection of bamWriter to BAM-file.

bamCount

*bamCount: Counting of CIGAR-OP items*

Description

The bamCount function takes a bamReader object, a set of reference coordinates and the 'complex' argument and returns an integer vector.

Usage

bamCount(object, coords)

Arguments

  object       An instance of bamReader. Must be opened and contain initialized index.
  coords       Integer vector of length 3: coords=c(refid, start, stop)

Details

The method returns integer vector of length 10. Entries 1 to 9 contain the number of CIGAR-OP items found in the given range. The 10th entry is the total number of aligns in the range. The returned vector is named. The names of the vector are "M", "I","N","S","H","P","=","X" and nAligns. The first 9 names are the abbreviations for CIGAR-OP items which are defined in the SAM file format reference.

Examples

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader<-bamReader(bam, idx=TRUE)
coords<-getRefCoords(reader,"chr1")
count<-bamCount(reader,coords)
count
bamClose(reader)
```
**Description**

The `bamCountAll` function takes a `bamReader` object and returns a data.frame. The data.frame contains the counts for all contained reference sequences. Because align numbers are counted for all reference sequences separately, the function needs to load a BAM index file.

**Usage**

```
bamCountAll(object, verbose=FALSE)
```

**Arguments**

- `object`: An instance of `bamReader`. Must be opened and contain initialized index.
- `verbose`: Logical. Determines the amount of textual output during runtime.

**Details**

The method returns a data.frame. Each row contains count data for one reference sequence. Each column contains the counts for one CIGAR-OP type ("M", "I", "N", "S", "H", "P", ".", ":", ":", ":"). Columns with the total number of aligns, the refid (ID) and the length of the reference sequence (LN), as retrieved by `getRefData` are added.

**Examples**

```r
library(rgamtools)
bam <- system.file("extdata", "accepted_hits.bam", package="rgamtools")
reader <- bamReader(bam, idx=TRUE)
count <- bamCountAll(reader, verbose=TRUE)
count
bamClose(reader)
```

---

**bamGapList-class**

Class "bamGapList"

**Description**

The `bamGapList` class represents a list of Alignment gap (N-items in Cigar-data) sites. For each gap-site, left and right start and end positions as well as the gap-length are reported. Numbers of aligns supporting this site, number of left-sided start positions (\(\leq 8\)) and the sum of overlapping nucleotides on the left side are given.

**Objects from the Class**

Objects can be created by calls of the form `siteList(reader, coords)`.
**Slots**

- list: "externalptr". Point to double linked list struct.
- refdata: "data.frame". Contains bamHeader like data for stored aligns.

**Methods**

- `size` signature(x = "bamGapList") returns number of site-items in list.
- `coerce` signature(from = "bamGapList", to = "data.frame"): Coercion of bamGapList to data.frame.
- `coerce` signature("bamGapList","data.frame"): Coercion of bamGapList to data.frame.
- `show` signature(object = "bamGapList"): Prints a short message with some summarizing data.
- `nAligns` signature(object = "bamGapList"): Returns number of aligns in specified Range.
- `nAlignGaps` signature(object = "bamGapList"): Returns number of align gaps in specified Range.

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
library(rbamtools)
bam<-system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader<-bamReader(bam, idx=TRUE)
bsl<-bamGapList(reader)
bsl
size(bsl)
nAligns(bsl)
nAlignGaps(bsl)
summary(bsl)
dfr<-.as.data.frame(bsl)
head(dfr)
bamClose(reader)
```

**bamHeader-class**

Class "bamHeader": Representation of data contained in the header section of BAM files.

**Description**

bamHeader represents data contained in the header Section of BAM-files.

**Objects from the Class**

Objects can be created by calls of the form `header<-new("bamHeader")`. 
Slots

header: Object of class "externalptr". Points to samtools bam_header_t struct.

Methods

getHeaderText signature(x = "bamHeader"): Returns textual representation of data stored in this class as described in SAM Format Specification.

Author(s)

Wolfgang Kaisers

References

The SAM Format Specification (v1.4-r985).

Examples

bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
  # Open bam file
reader<-bamReader(bam)
  # Extract binary header structure
header<-getHeader(reader)
  # Extract textual representation
htxt<-getHeaderText(header)
  # Extract header line segment
hl<-headerLine(htxt)
  # Get header program
hp<-headerProgram(htxt)
  # Re-create binary header structure
header2<-bamHeader(htxt)
  # Use created structure for new BAM-file
writer<-bamWriter(header2,"test.bam")
bamClose(reader)
bamClose(writer)

bamHeaderText-class  Class "bamHeader": Textual representation of header section of BAM files.

Description

bamHeader manages textual representation of data contained in the header section of BAM-files. The binary representation (from which new BAM-files can be created) is encapsulated in class 'bamHeader'. Both types can be converted into each other.
**Objects from the Class**

Objects can be created by calls of the form `h1 <- new("bamHeader").`

**Slots**

- `com`: character
- `head`: `headerLine`
- `dict`: `refSeqDict`
- `group`: `headerReadGroup`
- `prog`: `headerProgram`

**Methods**

- `headerLine` signature(x = "bamHeader"): Gets `headerLine` object.
- `headerLine<-` signature(x = "bamHeader"): Sets `headerLine` object.
- `refSeqDict` signature(object = "bamHeader"): Gets `refSeqDict` object.
- `refSeqDict<-` signature(object = "bamHeader"): Sets `refSeqDict` object.
- `headerReadGroup` signature(object = "bamHeader"): Gets `headerReadGroup` object.
- `headerReadGroup<-` signature(object = "bamHeader"): Sets `headerReadGroup` object.
- `headerProgram` signature(.Object = "bamHeader"): Gets `headerProgram` object.
- `headerProgram<-` signature(.Object = "bamHeader"): Sets `headerProgram` object.
- `getHeaderText` signature(.Object = "bamHeader"): Returns whole information encoded in a character string as described in SAM Format Specification.

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader <- bamReader(bam)
header <- getHeader(reader)
bamClose(reader)
htxt <- getHeaderText(header)
headLine <- headerLine(htxt)
headerLine(htxt) <- headLine
readGroup <- headerReadGroup(htxt)
headerReadGroup(htxt) <- readGroup
headerProg <- headerProgram(htxt)
headerProgram(htxt) <- headerProg
headTxt <- getHeaderText(htxt)
```
bamRange

bamRange(object, coordinates, complex=FALSE): Function for reading of alignments in genomic regions.

Description

The bamRange function takes a bamReader object, a set of reference coordinates and the 'complex' argument and returns an instance of class 'bamRange'.

Usage

bamRange(object=NULL, coords=NULL, complex=FALSE)

Arguments

object An instance of bamReader. Must be opened and contain initialized index
coords Integer vector of length 3: coords=c(refid, start, stop)
complex A logical value (of length 1). Default value: FALSE

Details

The method returns a list of bamAlign's from which overlap with the specified region. When complex is TRUE, the function only retrieves Aligns where nCigar > 1 ('complex' aligns, e.g. 45M329N56M). When reader is NULL, an empty range-list is constructed (can be filled with push_back). When complex is FALSE, the function retrieves all alignments which fall into the given range.

Value

An instance of class bamRange which can be accessed sequentially, modified or written to a BAM-file.

Examples

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
idx<-paste(bam,"bai",sep=".")

# Open BAM file
reader<-bamReader(bam)

# Create empty range and fill with push_back
range<-bamRange()
for(i in 1:10)
{ 
   align<-getNextAlign(reader)
   push_back(range,align)
}
size(range)
```
bamRange-class

### Description

bamRange represents a double linked list of bamAlign objects which overlap with a defined region in a BAM-file. The bamRange-function retrieves all alignments in the depicted Range from
BAM-File into a bamRange object. A bamRange object maintains a double-linked list of aligns. The list keeps a pointer to a current align structure for iteration purposes. Additionally there are some summarizing values stored (which are displayed by show) which describe the range inside the reference from which the bamRange object was read (seqid, qrBegin, qrEnd, complex) and some statistics (size, qSeqMinLen, qSeqMaxLen).

Details

bamRange objects internally keep the following values:

1. seqid 0-based index of reference sequence
2. qrBegin 0-based left boundary of query region (query range begin)
3. qrEnd 0-based right boundary of query region (query range end)
4. complex 0= all aligns included, 1= only aligns with n_cigar > 1 included
5. rSeqLen Length of reference sequence
6. qSeqMinLen Minimum of query sequence length (= read length)
7. qSeqMaxLen Maximum of query sequence length (= read length)

For the bamRange class exists a rudimentary subsetting ('[ ]') operator. '[ ]' allows only for indexes > 0 and <= size(x). Index values are sorted in ascending order before values are extracted.

Objects from the Class

Objects can be created by calls of the form range<-bamRange(reader, coords).

Slots

range: External pointer. Points to double linked list of bamAligns.

Methods

as.data.frame signature(x="bamRange"): Returns data.frame representation of aligns.

coerce signature(from="bamRange", to="data.frame"): Coercion of bamRange to data.frame.

bamSave signature(object="bamRange"): Saves aligns stored in this list to BAM-file via a bamWriter object.

getAlignRange signature(object="bamRange"): Iterates through the list and returns 0-based position of the leftmost and rightmost matching nucleotide in range.

getNextAlign signature(object="bamRange"): Returns next align from current position and shifts current position to next one.

getParams signature(object="bamRange"): Returns named vector of stored parameters

getPrevAlign signature(object="bamRange"): Returns previous align from current position and shifts current position to previous one.

getRefName signature(.Object="bamRange"): Returns the reference sequence name from which the range was retrieved.

getQualDf signature(object="bamRange", prob=logical"): Returns position dependent counts of phred quality values.
getQualQuantiles signature(object="bamRange", quantiles="numeric"): Returns position dependent quantile values for phred scores.

plotQualQuant signature(object="bamRange"): Plots phred quality quantiles for sequence positions.

initialize signature(.Object="bamRange"): Initializes bamRange object.

insertPastCurrent signature(object="bamRange"): Inserts align past current position into list.

insertPreCurrent signature(object="bamRange"): Insert align before current position into list.

pop_back signature(object="bamRange"): Removes last align from list.

pop_front signature(object="bamRange"): Removes first align from list.

push_back signature(object="bamRange"): Adds align at the end of the list.

push_front signature(object="bamRange"): Adds align at the front of the list.

rewind signature(object="bamRange"): Shifts current align to position before first align.

size signature(object="bamRange"): Returns number of aligns in list.

writeCurrentAlign signature(object="bamRange"): Overwrites current align with given align.

Author(s)

Wolfgang Kaisers

Examples

```r
## + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
## A) Open reader
## + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
idx<-paste(bam,"bai",sep=".")
# Open BAM file
reader<-bamReader(bam)
## Not run: create.index(reader,idx)
# Load BAM index file
loadIndex(reader,idx)
indexInitialized(reader)  # Should return 'TRUE'

## + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
## B) Read range
## + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
# Find appropriate refid (=ID)
# Returns a data.frame with three columns:
# ID=refid, SN=Sequence Name, LN=Sequence length
rdf<-getRefData(reader)
head(rdf)

## + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
# The sequence length (LN) also determines valid range for start and stop coordinates
# Invalid refid-, start- or stop-coordinates will release an error.
```
The `bamReader` function constructs and returns an S4 object of class `bamReader`. The `bamReader` object represents a connection to an opened BAM-file. When an indexname is given or `idx=TRUE`,
the function tries to load an existing BAM-index.

Usage

bamReader(filename, indexname, idx=FALSE, verbose=0)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Filename of BAM-file to be opened for reading.</td>
</tr>
<tr>
<td>indexname</td>
<td>Optional: Name of BAM-index file</td>
</tr>
<tr>
<td>idx</td>
<td>Logical</td>
</tr>
<tr>
<td>verbose</td>
<td>Numeric: Quantifies the extent of textual feedback (levels: 0,1,2).</td>
</tr>
</tbody>
</table>

Author(s)

Wolfgang Kaisers

Examples

```r
bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
reader<-bamReader(bam)
#
align<-getNextAlign(reader)
name(align)
position(align)
bamClose(reader)
#
# Open reader and initialize BAM-index
reader<-bamReader(bam, idx=TRUE)
indexInitialized(reader)
```

Description

A bamReader object encapsulates functionality for reading of bamAligns from a BAM-file. It optionally contains a pointer to a bam-index structure which allows fast access to aligns that overlap a specified region (random access). The index is loaded via the loadIndex function.

Objects from the Class

Objects can be created by calls of the form reader<-bamReader(filename).
Slots

filename: Character denoting name of BAM-file.
reader: External pointer to opened BAM-file.
startpos: Numeric value returned by bam_tell immediately after opening BAM-file. It is used as target position for rewinding by bam_seek.
index: External pointer to BAM index (used for retrieving bamRange objects from BAM-file).

Methods

bamClose signature(object = "bamReader"): Closes connection to BAM-file.
bamCopy signature(object = "bamReader", writer="bamWriter", refids, verbose): Copies all aligns for given refids from reader to writer. When no refids are given, aligns for all refids are copied. refids refer to Reference-ID's returned by 'getRefData' (ID column). Missing matches (from refids into Reference-ID's) will terminate the function with an error.
bamSave signature(object = "bamReader", writer="bamWriter"): Copies all aligns from reader to writer.
bamSort signature(object = "bamReader", prefix="character", byName=FALSE, maxmem=1e+9, path): Sorting an existing BAM-file.
createIndex signature(object = "bamReader", idx_filename="character"): Creates an index file for opened BAM-file. Therefore the BAM-file must be position-sorted. idx_filename will be the name of the new BAM-index file. idx_filename is an optional argument. The standard value is "bam_filename.bam".bai.
loadIndex signature(object="bamReader", filename="character"): Loads index from given index file. A loaded index is a prerequisite for random access to a BAM file.
indexInitialized signature(object="bamReader"): Returns TRUE when a loaded BAM index is present in bamReader.
create.index Deprecated. See "createIndex".
load.index Deprecated. See "loadIndex".
index.initialized Deprecated. See "indexInitialized".
filename signature(object = "bamReader"): Returns filename of opened BAM-file.
getHeader signature(object = "bamReader"): Returns object of class bamHeader which contains binary representation of bam-header data.
getHeaderText signature(object = "bamReader"): Returns object of class bamHeaderText which contains textual representation of bam-header data.
getNextAlign signature(object = "bamReader"): Returns object of class bamAlign which contains data of next Align from file. When EOF is reached the function returns NULL.
getRefCoords signature(object="bamReader", sn="character"): Helper function takes a sequence name and returns coordinates of entire reference sequence for usage with bamRange, gapList or siteList function. The function returns a vector of length 3. The vector elements are named "refid", "start", "stop".
getRefCount signature(object = "bamReader"): Returns number of reference sequences.
**getRefData** signature(object = "bamReader"): Returns data frame which contains three columns: For each reference sequence, the corresponding row contains the Reference-ID (1st column, refID), the Reference name (2nd column, refName) and the length of the Reference sequence (3rd column, refLength).

**isOpen** signature(object = "bamReader"): Returns TRUE when file connection is open.

**rangeSegCount** signature(object = "bamReader", coords=numeric, segments=numeric, complex=logical): Counts alignments for specified genomic segment regions (genes, exons, ...)

**rewind** signature(object = "bamReader"): Resets current file position. The subsequent call to getNextAlign will return the first align in the BAM-file.

**initialize** signature(object = "bamReader"): Initializes object and opens BAM-file for reading.

**Author(s)**

Wolfgang Kaisers

**References**

The SAM Format Specification (v1.4-r985).

**Examples**

```r
bam<- system.file("extdata", "accepted_hits.bam", package="rbamtools")
idx<- system.file("extdata", "accepted_hits.bam.bai", package="rbamtools")

# # Open BAM-file for reading
reader<-bamReader(bam)
getNextAlign(reader)
rewind(reader)
getNextAlign(reader)
isOpen(reader)

# # Create and load new index
## Not run:
new_idx<="index.bam.bai"
createIndex(reader,new_idx)
loadIndex(reader,new_idx)
indexInitialized(reader)

## End(Not run)
#
# # Load package provided index
loadIndex(reader,idx)
indexInitialized(reader)
#
# # Read align
align<-getNextAlign(reader)
coords<-as.integer(c(0,0,249250621))
ranges<-bamRange(reader,coords)
align<-getNextAlign(ranges)
```
# bamsave

# Copy all aligns
writer<-bamWriter(getHeader(reader),"newFile1.bam")
bamsave(reader,writer)
bamclose(writer)

# reader2<-bamReader("newFile1.bam")
## Not run:
bamSort(reader,"newFile1s")
# Copy align for Reference-ID '0'
writer<-bamWriter(getHeader(reader),"newFile2.bam")
bamCopy(reader,writer,refid=0)
bamclose(writer)

## End(Not run)
#
# Closing reader
bamClose(reader)

---

bamsave

**bamsave**: Creation of a sorted copy of a BAM file.

---

**Description**

The methods for bamsave write BAM alignment data (either single alignments as bamAlign objects or lists of alignments as bamRange objects) out to a bamWriter.

**Usage**

`bamsave(object, value, refid, ...)`

**Arguments**

| object | bamWriter. The writer must be opened. |
| value  | Object of type bamAlign or bamRange. |
| refid  | (Optional) Defines sequence association (ID) in Reference sequence dictionary of output BAM file. May be given as numeric, or character. When missing, ID is determined from reference name (only for bamRange). |
| ...    | Currently unused. |

**Details**

The different methods specialize on different methods for determining the Reference sequence ID in the output file.

**Author(s)**

Wolfgang Kaisers
bamSort

*bamSort: Creation of a sorted copy of a BAM file.*

**Description**

The function takes an opened instance of *bamReader*, a prefix for the output file and produces a sorted BAM file which is (apart from being sorted) identical to the input file. BAM-files must be sorted before an index can be created. During this routine, some content of the source file is copied into the working memory, sorted and written into temporary files. Finally, the sorted files are merged into a single target. The name of the sorted BAM-file will be `[prefix].bam`. Number and size of temporary files depend on source file size and used working memory (`maxmem`). Small `maxmem` values result in large numbers of temporary files. The minimum `maxmem` value is `100000000`. Smaller `maxmem` values cause an error.

**Usage**

```r
bamSort(object, prefix="sorted",
        byName=FALSE, maxmem=1e+9, path=dirname(filename(object)))
```

**Arguments**

- **object**: `bamReader` (must be opened) or `sampleBamFiles`
- **prefix**: The prefix of the output file. When 'sorted' is given as prefix, the routine produces a file named 'sorted.bam'.
- **byName** *(Optional)*. Logical. Must have length 1. When TRUE the file will be sorted by align name. When FALSE the file will be sorted by coordinate. Sorting by coordinate is the prerequisite for creation of an index file.
- **maxmem** *(Optional)*. Numeric. Must have length 1. Minimum value is `100000000`. Smaller `maxmem` values cause an error. Determines how many aligns are sorted inside the working memory before they are written into a temporary file.
- **path**. Character. Directory where sorted output file is written to. Default value is the directory where the input file is located.
Details

The function does not take a complete name for the output file but only a prefix. The prefix is internally completed with a `.bam` suffix. This is because the `samtools` function `bam_sort_core_ext` only takes a prefix. `samtools` in turn produces intermediate files which also use the prefix and which are removed again when the `bam_sort_core_ext` cleans up.

Author(s)

Wolfgang Kaisers

Examples

```r
bam<system.file("extdata","accepted_hits.bam",package="rbamtools")
reader< bamReader(bam)
bamSort(reader)
## ----- ----- ----- ----- ----- ----- ----- ##
## sampleBamFiles
## ----- ----- ----- ----- ----- ----- ----- ##
bs <- sampleBamFiles(bam)
```
Examples

# +++++++++++++++++++++++++++++++++++++++++++++++++++++++
# In this example, we copy some complex (i.e. interesting) aligns
# into a new BAM file
bam<-system.file("extdata", "accepted_hits.bam", package="rbamtools")
idx<-paste(bam,"bai",sep=".")

# +++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Open BAM file and read complex aligns from chr1
reader<-bamReader(bam)
loadIndex(reader,idx)
range<-bamRange(reader,coords,complex=TRUE)
bamClose(reader)

# +++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Create bamHeader from scratch
bh<-new("bamHeaderText")
headl<-new("headerLine")
setVal(headl,"SO","coordinate")
dict<-new("refSeqDict")
addSeq(dict,SN="chr1",LN=249250621)
dict
prog<-new("headerProgram")
setVal(prog,"ID","1")
setVal(prog,"PN","tophat")
setVal(prog,"CL","tophat-p8 --library-type fr-unstranded hs_ucsc rna033.fastq")
setVal(prog,"VN","2.0.0")
bh<-bamHeaderText(head=headl,dict=dict,prog=prog)
## Not run: getHeaderText(bh)
header<-bamHeader(bh)

# +++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Copy aligns in range into new BAM file
## Not run:
writer<-bamWriter(header,"chr1_complex.bam")
bamSave(writer,range,refid=0)
bamClose(writer)

## End(Not run)
# +++++++++++++++++++++++++++++++++++++++++++++++++++++++

bamWriter-class

Class "bamWriter": Representation of a file connection to a BAM file for writing access.
Description

The bamWriter class represents the writing connection to a BAM-file. Usually, this class is used to create an empty BAM-file and to copy aligns from another BAM-file into the new file.

Objects from the Class

Objects can be created by calls of the form writer<-bamWriter(reader, filename).

Slots

- filename: Object of class "character". Points to samtools struct samtile_t.
- writer: Object of class "externalptr". Points to samtools struct samtile_t.

Methods

- bamClose signature(object = "bamWriter"): Closes BAM file.
- bamSave signature(object = "bamWriter", value = "bamAlign" or "bamRange", refid = "numeric"): Saves bamAlign or bamRange object to BAM-file. refid will be overwritten in all written aligns.
- filename signature(object = "bamWriter"): Returns filename of opened BAM-file.
- initialize signature(.Object = "bamWriter"): Opens BAM file for writing.
- isOpen signature(object = "bamWriter"): Returns TRUE when file connection is open.

Author(s)

Wolfgang Kaisers

References

The SAM Format Specification (v1.4-r985).

Examples

```r
# ++++                                  
# In this example, we copy some complex (i.e. interesting) aligns
# into a new BAM file
bam<system.file("extdata", "accepted_hits.bam", package="rbamtools")
idx<paste(bam,"bai",sep=".")

# ++++                                  
# Open BAM file and read complex aligns from chr1
reader<-bamReader(bam)
loadIndex(reader,idx)
coords<-as.integer(c(0,0,249250621))
range<-bamRange(reader,coords,complex=TRUE)
bamClose(reader)
# ++++
```
# Create bamHeader from scratch
bh<-new("bamHeaderText")
headl<-new("headerLine")
setVal(headl,"SO","coordinate")
dict<-new("refSeqDict")
addSeq(dict,SN="chr1",LN=249250621)
addSeq(dict,SN="chr10",LN=135534747)
dict
prog<-new("headerProgram")
setVal(prog,"ID","1")
setVal(prog,"PN","tophat")
setVal(prog,"CL","tophat -p8 --library-type fr-unstranded hs_ucsc test.fq")
setVal(prog,"VN","2.0.0")
bh<-bamHeaderText(headl=headl,dict=dict,prog=prog)
# getHeaderText(bh)
header<- bamHeader(bh)

# Copy aligns in range into new BAM file
## Not run:
writer<-bamWriter(header, "chr1_complex.bam")
bamSave(writer, range, refid=0)
bamClose(writer)

## End(Not run)

---

**countNucs**  
*Counting nucleotides in bamAlign and bamRange*

**Description**

The function counts occurrence of the Nucleotides A,C,G,T in bamAlign and bamRange objects. Any other values will be combined counted in the last value (N). The function returns an integer vector of length 5. The names indicate which position contains the count value each nucleotide.

**Usage**

```r
countNucs(object)
```

**Arguments**

- **object**  
  bamAlign or bamRange object.

**Value**

Integer (of length 4).
**countTextLines**

**Author(s)**

Wolfgang Kaisers

**Examples**

```
# A) For bamAlign
align<-bamAlign("HMUSI-0001","ACCGGTTTT","Qual/Strng","4M10NGM",refid=0,position=100)
countNucs(align)

# B) For bamRange
bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
reader<-bamReader(bam, idx=TRUE)
coords<-c(0,0,14730)
range<-bamRange(reader,coords)
countNucs(range)
```

---

countTextLines: Counting lines in text files.

**Description**

The `countTextLines` function takes a filename vector and returns the number of lines contained in each file.

**Usage**

```
countTextLines(filenames)
```

**Arguments**

- `filenames`: Character. Vector of file-names to be opened.

**Details**

Fastq files usually (but not guaranteed) contain four lines of text for each read. In many cases, this function can be used to count the number of reads in a fastq file.

**Examples**

```
filename <- system.file("extdata", "test.fastq", package="rbamtools")
countTextLines(filename)
```
**createIdxBatch**

*createIdxBatch: Creation of index files for multiple BAM files.*

**Description**

The function takes a vector of BAM-file names (plus optionally accompanying names of BAM-index-files) and checks for existing BAM-index files. When index files do not exist, the function creates the missing index files.

**Usage**

```r
createIdxBatch(bam, idx=paste(bam, ".bai", sep="" ), rebuild=FALSE)
```

**Arguments**

- **bam** Filenames of BAM-files to be opened.
- **idx** Optional: Name of BAM-index files
- **rebuild** Optional: Logical value. When `TRUE` the function rebuilds existing BAM-index files.

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
## Not run: bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
## Not run: createIdxBatch(bam)
```

---

**exonAlignDepth-class**

*Class “exonAlignDepth”: Align-depth data for single genes on multiple samples (BAM files). Intronic regions are cut out of align depth matrix.*

**Description**

Objects of this class combine data from geneAlignDepth. During construction, the align depth matrix is segmentized, so that only (unified) exonic regions are represented in align depth matrix.

**Objects from the Class**

Objects can be created by calls of the form `exonAlignDepth(sal)`.
Slots

- **ald** "matrix": Matrix containing alignment depth data.
- **aldRatio** "data.frame": Contains mean alignment depth and alignment-depth-ratio data. `nr` is the ratio between the in-place and next alignment depth. `pr` is the ratio between in-place and previous alignment depth.
- **junctions** "data.frame": Contains alignment-gap-sites which had been generated when reading `geneAlignDepth` data.
- **gene_id**: "character": Gene identifier (e.g. Ensembl or UCSC).
- **gene_name**: "character": Gene name.
- **seq_name**: "character": Reference sequence (chromosome) name.
- **strand**: "character": Strand orientation of gene on reference sequence (+, - or *)
- **naligns**: "numeric": Total number of alignments in each BAM file.
- **group**: "factor": Group assignment
- **label** "character": Short textual identifier for each sample.

Methods

- **show** signature(object="exonAlignDepth"): Prints a short message with some summarizing data.
- **exonAlignDepth** signature(object=c("sampleBamFiles", "geneModel"): Constructs `exonAlignDepth` object and reads align depth data from BAM files.
- **plot** signature(object="exonAlignDepth"): Plots align depth data.
- **aldRatio** signature(object="exonAlignDepth"): Returns data.frame containing alignment-depth-ratio data used for identification of exon junctions.
- **junctionSites** signature(object="exonAlignDepth"): Returns data.frame containing junction positions.
- **getNormFactor** signature(object="exonAlignDepth"): Returns numeric value which will be uses for plotting and `groupAldMatrix` and `groupAldTable`.
- **groupAldMatrix** signature(object="exonAlignDepth"): Returns matrix containing mean alignment depth values. Data for each sample is stored in one column. Data for each genomic position is stored in one row. A summarizing function `f` may be given (Default is mean).
- **groupAldTable** signature(object="exonAlignDepth"): Returns data.frame containing three columns. The first column contains genomic positions, the second position contains group assignment and the third position contains (normalized) alignment dept values. A summarizing function `f` may be given (Default is mean).

Author(s)

Wolfgang Kaisers
Examples

```r
## - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Construct sampleBamFiles object
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
bs <- sampleBamFiles(1)
bamFiles(bs) <- bam
sampleLabels(bs) <- "s1"
sampleGroups(bs) <- "g1"
checkBamFiles(bs)
nAligns(bs) <- bamCountAll(bs)
bs
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Construct geneModel object
library(refGenome)
ucfile <- system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc <- loadGenome(umyfile)
gt <- getGeneTable(uuc)
gene_id <- as.character(gt$gene_id[1])
gm <- geneModel(uuc, gene_id)
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Construct geneAlignDepth object
gad <- geneAlignDepth(bs, gm)
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Extract exonAlignDepth object
ead <- exonAlignDepth(gad, ratioLim=5, infVal=1000)
ald <- aldratio(ead)
jc <- junctionSites(ead)
getNormFactor(ead)
cead <- cutFlatAlignDepth(ead, ratio=0.1)
```

`exonLoessModel-class`  
Class "exonLoessModel": Align-depth data for single genes on multiple samples (BAM files). Intrinsic regions are cut out of align depth matrix.

Description

Objects of this class combine data from `geneAlignDepth`. During construction, the align depth matrix is segmentized, so that only (unified) exonic regions are represented in align depth matrix.

Objects from the Class

Objects can be created by calls of the form `exonLoessModel(sal)`.

Slots

- `gene_id`: "character": Gene identifier (e.g. Ensembl or UCSC).
- `gene_name`: "character": Gene name.
- `seq_name`: "character": Reference sequence (chromosome) name.
strand: "character": Strand orientation of gene on reference sequence (+, - or *)
nAligns: "numeric": Total number of alignments in each BAM file.
group: "character": Group assignment
loessPred: "matrix": Loess estimated align depth values for each group. Each column contains estimates for one group.

Methods

show signature(object="exonLoessModel"): Prints a short message with some summarizing data.
exonLoessModel signature(object=c("sampleBamFiles", "geneModel"): Constructs exonLoessModel object and reads align depth data from BAM files.
plot signature(object="exonLoessModel"): Plots align depth data.
groupRatio signature(object="exonLoessModel", lim="numeric"): Returns single numeric value. From align depth estimates in loessPred matrix, the ratio between a column and the precedent column is calculated. For each position, the minimum ratio is extracted. The function returns the relative number of positions, where the minimum ratio exceeds given lim value.

Author(s)

Wolfgang Kaisers

Examples

```r
## - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ##
# Construct sampleBamFiles object
bam<-system.file("extdata", "accepted_hits.bam", package="rbamtools")
bs <-sampleBamFiles(!)
bamFiles(bs) <- bam
sampleLabels(bs) <- "s1"
sampleGroups(bs) <- "g1"
checkBamFiles(bs)
nAligns(bs) <- bamCountAll(bs)
bs
## - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ##
# Construct geneModel object
library(refGenome)
ucfile<-system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc<-loadGenome(ucline)
gt <- getGeneTable(ucline)
gene_id <- as.character(gt$gene_id[1])
gm <- geneModel(ucline, gene_id)
## - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ##
# Construct geneAlignDepth object
gad <- geneAlignDepth(bs, gm)
## - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ##
# Extract exonLoessModel object
ead <- exonAlignDepth(gad, ratioLim=5, infVal=1000)
```
extractBamRegions

extractBamRegions: Extraction of alignments from given (genetic) regions and BAM files into a set of output BAM files.

Description

The function extractBamRegions writes aligns or some part of the stored aligns in a BAM file as reads into a BAM output file.

Usage

extractBamRegions(bamFiles, ranges, idxFiles, outFiles)

Arguments

bamFiles character: Vector of BAM file names.
ranges data.frame: Must contain three columns seqid, start and end. The seqid
idxFiles character: Vector of BAM index file names.
outFiles character: Vector of output file names.

Details

bamFiles, idxFiles and outFiles are checked for equal length. bamFiles and idxFiles are checked for existance. There are sensible default values given for idxFiles and outFiles. The default location for writing output files is the current working directory.

Value

No return value given

Author(s)

Wolfgang Kaisers

Examples

bam <- system.file("extdata","accepted_hits.bam", package="rbamtools")
gene_pos <- data.frame(seqid="chr1", start=15e3, end=20e3)
## Not run: extractBamRegions(bam, gene_pos)
extractGeneRegions

**Description**

The function `extractGeneRegions` writes aligns or some part of the stored aligns in a BAM file as reads into a BAM output file.

**Usage**

```r
extractGeneRegions(src, trg, gl)
```

**Arguments**

- `src` : sampleBamFiles: List of source BAM files where alignments are read from. BAM index files must exist and be present in src object.
- `trg` : sampleBamFiles: List of target BAM files where alignments are written to.
- `gl` : geneList: Objects from which genetic regions are taken.

**Details**

Reference sequence names in geneList object must all match sequence names in source BAM files.

**Value**

Numeric. Number of written alignments for each file.

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
# + + + + + + + + + + + + + + + + #
# A) Create sampleBamFiles objects
# + + + + + + + + + + + + + + + + #
.bam <- system.file("extdata","accepted_hits.bam",package="rbamtools")
.outdir <- tempdir()
# Input sampleBamFiles object
.ibs <- sampleBamFiles(bam)
# Output sampleBamFiles object
.obs <- sampleBamFiles(file.path(outdir, "outu.bam"))
# + + + + + + + + + + + + + + + + #
# B) Create geneList Object
# + + + + + + + + + + + + + + + + #
.rfile <- system.file("extdata",
    "hs.ucsc.small.RData",
```
package="refGenome"

ucr <- loadGenome(rfile)
gt <- getGeneTable(ucr)
gl <- geneListTable(ucr, gt$gene_id)

extractGeneRegions(ibs, obs, gl)

bamFiles(obs) <- file.path(outdir, bamSort(obs, "out"))
bamIdxFiles(obs) <- paste(bamFiles(obs), "bai", sep=".")
createIndex(bam, obs)

## End(Not run)

---

**extractRanges**

**extractRanges**: Extraction of alignments

### Description

The function `extractRanges` takes an opened and indexed bamReader object, a data.frame containing range data, and an output filename. From the opened BAM file, aligns in the depicted ranges are transferred into a temporary BAM file. The new file will be sorted. The sorted file will be indexed.

### Usage

```r
extractRanges(object, ranges, filename, complex=FALSE, header, idxname)
```

### Arguments

- **object**: bamReader. Must be opened. Index must be initialized.
- **ranges**: data.frame. Must contain columns 'seqid','start','end'. Each line represents a range on the reference genome for which aligns are extracted.
- **filename**: Character. Name of output BAM-file. The filename will be modified. The modified filename will have suffix '.bam'.
- **complex**: (default=FALSE) Logical. When TRUE, only aligns with nCigar > 1 are copied. When FALSE, all aligns in the depicted ranges are copied.
- **header**: (Optional) bamHeader. It's possible to provide a bamHeader for the new BAM file (in order to control the header entries in the new file). When no header is given, the header from the bamReader will be used. The 'SO' entry in the 'headerLine' (which gives information about the sorting status) segment will be altered. The reference sequence dictionary (RSD) of the output file will be created new inside the function. It is necessary to do that because when there are mismatches between the RSD and the contained aligns, it will be impossible to create an index for the new file (samtools will crash).
- **idxname**: (Default='filename'.bai) character. The name for the index file.
Author(s)

Wolfgang Kaisers

See Also

bamReader

Examples

```r
bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
reader<-bamReader(bam,idx=TRUE)
# Extract data for HLHL17 gene:
seqid<="chr1"
start<895967
end<901999
gene_name<="HLHL17" # (optional)
ranges<data.frame(seqid=seqid,start=start,end=end,gene_name=gene_name)
## Not run: extractRanges(reader,ranges=ranges,filename="new_file.bam")
```

Description

Reading filename from bamReader or bamWriter

Methods

```r
signature(object = "bamReader") An instance of class bamReader.
signature(object = "bamWriter") An instance of bamWriter.
```

Class "gapList": Representation of genomic alignment gaps.

Description

The gapList class represents a list of Alignment gaps (i.e. N-items in Cigar-data). For each gap, the type of left and right adjacent Cigar items is reported plus size of each. The list can be converted into a data.frame which then contains the columns: refid, position (which identify the align), left_cigar_len, left_cigar_type, left_stop (characterize the left boundary) and right_start, right_cigar_len, right_cigar_type (characterize the right boundary). The adjacent cigar-types should be 0 (i.e. M=match). 'left_stop' is the 0-based position of last exon nucleotide, right_start is the 0-based position of the first exon nucleotide.
Objects from the Class

Objects can be created by calls of the form `gapList(reader,coords)`.

Slots

- `list`: "externalptr". Point to double linked list struct.

Methods

- `size` signature(x="gapList"): Returns number of gapped-align items in list.
- `coerce` signature(from="gapList", to="data.frame"): Coercion of gapList to data.frame.
- `as.data.frame` signature(x="gapList", row.names=NULL, optional=FALSE): Coercion of gapList to data.frame.
- `show` signature(object="gapList"): Prints a short message with some summarizing data.
- `nAligns` signature(object="gapList"): Returns number of aligns in specified Range.
- `nAlignGaps` signature(object="gapList"): Returns number of align gaps in specified Range.

Author(s)

Wolfgang Kaisers

Examples

```r
# Open (indexed) BAM file
bam<-system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader<-bamReader(bam,idx=TRUE)
coords<-getRefCoords(reader,"chr1")
gl<-gapList(reader,coords)
size(gl)
nAligns(gl)
nAlignGaps(gl)
gl
dfr<-as.data.frame(gl)
# coerce
dfr<-as(gl,"data.frame")
head(df)
bamClose(reader)
```

gapSiteList-class

Class "gapSiteList": Representation of genomic alignment gap positions.

Description

The gapSiteList class represents pooled alignment-gap sites. Gap-sites are characterized by unique refid, left-end and right-start positions in each list. Numbers of aligns supporting this site, number of left-sided start positions (<= 8) and the sum of overlapping nucleotides on the left side are given.
Objects from the Class

Objects can be created by calls of the form `siteList(reader, coords)`.

Slots

list: "externalptr". Point to double linked list struct.

Methods

`size` signature(x = "gapSiteList"): Returns number of site-items in list.

`coerce` signature(from = "gapSiteList", to = "data.frame"): Coercion of `gapSiteList` to `data.frame`.

`coerce` signature("gapSiteList", "data.frame"): Coercion of `gapSiteList` to `data.frame`.

`show` signature(object = "gapSiteList"): Prints a short message with some summarizing data.

`nAligns` signature(object = "gapSiteList"): Returns number of aligns in specified Range.

`nAlignGaps` signature(object = "gapSiteList"): Returns number of align gaps in specified Range.

`refID` signature(object="gapSiteList"): Returns refID from which `gapSiteList` has been retrieved.

`refID<-` signature(object="gap", value="numeric"): Sets flag item.

Author(s)

Wolfgang Kaisers

Examples

# Open (indexed) BAM file
bam<system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader<bamReader(bam, idx=TRUE)
coords<getRefCoords(reader, "chr1")
sl<siteList(reader, coords)
size(sl)
nAligns(sl)
nAlignGaps(sl)
sl
refID(sl)
df<as.data.frame(sl)
head(df)
# Create from bamRange:
br <- bamRange(reader, coords)
sl <- siteList(br)
# bamClose(reader)
# Create empty list
sl <- siteList()
geneAlignDepth-class  

Class "geneAlignDepth": Align-depth data for single genes on multiple samples (BAM files).

Description

Objects of this class combine data from geneModel and from sampleBamFiles objects allowing to read BAM alings for gene defined regions from multiple BAM files. geneAlignDepth object contain an Align depth matrix which can be visualized using a generic plot function.

Objects from the Class

Objects can be created by calls of the form getgeneAlignDepth(gesa, gm).

Slots

- bamFiles: "character": Location of BAM files
- bamIdxFiles: "character": Location of BAM index files
- nAligns: "numeric": Total number of alignments in each BAM file.
- group: "factor": Group assignment
- label: "character": Short textual identifier for each sample.
- length: "integer": Vector length for bamFiles, bamIdxFiles, nAligns, group and label.
- ald: "matrix": Matrix containing alignment depth data.
- gapSites: "data.frame": Align gap sites data from genetic region.
- ev: "environment": Contains additional data (e.g. group table).
- gene_id: "character": Gene identifier (e.g. Ensembl or UCSC).
- gene_name: "character": Gene name.
- seq_name: "character": Reference sequence (chromosome) name.
- strand: "character": Strand orientation of gene on reference sequence (+, - or *)
- coords: "numeric": Gene coordinates (i.e. start and end position).

Methods

- show signature(object="geneAlignDepth"): Prints a short message with some summarizing data.
- geneAlignDepth signature(object=c("sampleBamFiles", "geneModel"): Constructs geneAlignDepth object and reads align depth data from BAM files.
- plot signature(object="geneAlignDepth"): Plots align depth data.

Author(s)

Wolfgang Kaisers
Examples

```r
## Construct sampleBamFiles object
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
bs <- sampleBamFiles(1)
bamFiles(bs) <- bam
sampleLabels(bs) <- "s1"
sampleGroups(bs) <- "g1"
checkBamFiles(bs)
nAligns(bs) <- bamCountAll(bs)
bs
## Construct geneModel object
library(refGenome)
ucfile <- system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc <- loadGenome(uccfile)
gt <- getGeneTable(ucc)
gene_id <- as.character(gt$gene_id[1])
gm <- geneModel(ucc, gene_id)
## Construct geneAlignDepth object
gad <- geneAlignDepth(bs, gm)
plot(gad, col="gray50")
```

Description

Objects from the Class

can be created using the genomePartition function on an opened bamReader and a data.frame containing target regions (usually gene annotation data).

Slots

ev: "environment". Environment which contains data as data.frames.
Methods

show signature(object="GenomePartition"): Prints a short message with some summarizing data.

getSeqNr signature(object="GenomePartition"): Returns number of reference sequences in refdata.

getRefData signature(object="GenomePartition"): Returns refdata data.frame.

getPos signature(object="GenomePartition"): Returns numeric position values for align depth.

countPartition signature(partition="GenomePartition", src="bamReader"): Counts BAM alignments on the contained grid from a single BAM file (represented by an opened bamReader).

countPartition signature(partition="GenomePartition", src="data.frame"): Counts BAM alignments on the contained grid from multiple BAM files (filenames and sample names given in src).

checkPartition signature(partition="GenomePartition", src="data.frame", verbose="logical"): Checks incoming arguments and present reference sequences for consistency. Intended to be used as preparation for countPartition in order to prevent unexpected routine terminations.

getFileTable signature(object="GenomePartition"): Returns the contained filetable (constituted by usage of countPartition function).

getAlignCounts signature(object="GenomePartition"): Returns counted alignments in annotated regions (exons, genes).

getGridAlignCounts signature(object="GenomePartition"): Returns counted alignments in underlying equidistant grid.

Author(s)

Wolfgang Kaisers

Examples

# Open (indexed) BAM file
bam<-system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader<-bamReader(bam, idx=TRUE)

# Provide exon positions
id <- 1:13
seqid <- "chr1"
gene <- "WASH7P"
ensg_id <- "ENSG00000227232"
start <- c(14411, 15000, 15796, 15904, 16607, 16748, 16858, 17233,
17602, 17915, 18268, 24737, 29534)
end <- c(14502, 15038, 15901, 15947, 16745, 16765, 17055, 17364,
17742, 18061, 18366, 24891, 29806)

ref <- data.frame(id=id, seqid=seqid, begin=start, end=end, gene=gene, ensg=ensg_id)

# Create partition (adds equidistant grid)
partition <- genomePartition(reader, ref)
getHeaderText-methods

Assembling member data into Header-Text

Description

The header section contains various data fields which provide metadata about the stored alignment results. Returns an unparsed character string containing the header section of a BAM-file. Part of the Header-Text is the RefData information.

Usage

getHeaderText(object, delim = "\n")

Arguments

object  An object of class bamHeader or bamHeaderText

delim   Character used as tag delimiter

Details

The Header section of a BAM-file consists of the header line and the Reference sequence dictionary. The header line contains information about the format version and a specification of the sorting order of alignments. The reference dictionary contains information about the name and the length of each reference sequence. The bamHeader object can be used to initialize an object of class bamWriter. The bamHeaderText can be used to inspect the content of the header.

Value

An object of class bamHeaderText (from argument bamHeader) and a string (from argument bamHeaderText)

Methods

signature(object = "bamHeader") An object of class bamHeader

Author(s)

Wolfgang Kaisers
References
The SAM Format Specification (v1.4-r985).

Examples

```r
bam<system.file("extdata","accepted_hits.bam",package="rbamtools")
idx<paste(bam,"bai",sep=".")
# Open BAM file
reader<bamReader(bam)
header<-getHeader(reader)
htxt<-getHeaderText(header)
getHeaderText(htxt)
bamClose(reader)
```

g getNextAlign-methods  getNextAlign: Retrieving next align from bamReader or bamRange

Description

Returns an object of class bamAlign from bamReader or bamRange.

Value

An object of class bamAlign or NULL.

Methods

signature(object = "bamRange") An instance of class bamRange.
signature(object = "bamReader") An instance of class bamReader

Author(s)

Wolfgang Kaisers

Examples

```r
#
bam<system.file("extdata","accepted_hits.bam",package="rbamtools")
idx<system.file("extdata","accepted_hits.bam.bai",package="rbamtools")
#
reader<bamReader(bam)
isOpen(reader)
#
align<getNextAlign(reader)
if(is.null(align))
  print("End of File!
")
#
loadIndex(reader,idx)
```
getQualDf

indexInitialized(reader)
#
coords<-as.integer(c(0,0,249250621))
range<-bamRange(reader,coords)
align<-getNextAlign(range)
position(align)
#
bamClose(reader)

---

getQualDf

Read and display Phred qualities from bamRange

Description

getQualDf takes a bamRange and returns a data.frame (128 rows, number of columns=length of the longest sequence in range). getQualDf counts occurrences for every sequence position (column) and every phred value (row). getQualQuantiles takes a bamReader and a vector of quantiles (must be between 0 and 1) and returns a data.frame. The data.frame contains one row for each quantile and also as many columns as the maximum sequence length. plotQualQuant plots the values for quantiles 0.1,0.25,0.5,0.75 and 0.9.

Usage

getQualDf(object,prob=FALSE,...)

Arguments

object bamRange.
prob logical. When TRUE each column is divided by its sum. The column sums are added as names attribute col.sum
...
(currently unused)

Details

Phred values are truncated by 127 (the maximum which can be represented by ASCII values). The function runs down each column (sequence position) and returns the row index where the quantile exceeds the cumulated column values.

Value

data.frame

Author(s)

Wolfgang Kaisers
Examples

```r
# A) Read bamRange
bam <- system.file("extdata","accepted_hits.bam",package="rbamtools")
reader <- bamReader(bam,idx=TRUE)
coords <- as.integer(c(0,0,249250621))
range <- bamRange(reader,coords)
bamClose(reader)
# B) getQualDf
qdf <- getQualDf(range)
qdf[32:38,1:15]
qdr <- getQualDf(range,prob=TRUE)
# C) getQualQuantiles
quantiles <- c(0.1,0.25,0.5,0.75,0.9)
qt <- getQualQuantiles(range,quantiles)
# D) Plot
plotQualQuant(range)
```

---

### getRefData

**Retrieve reference sequence from a BAM file as data.frame**

**Description**

The four functions: `getRefCoords`, `getRefCount`, `getRefData` and `getRefId` provide reading access to data about the present reference sequences in `bamReader` or `bamWriter`.

**Usage**

```r
getRefData(object)
```

**Arguments**

- `object` : `bamReader`. The reader must be opened (otherwise an error is thrown).

**Details**

`getRefData` returns a `data.frame` with three columns (ID, SN, LN). ID is the (0-based index which must be given when a `bamRange` is extracted. SN is the name of the sequence (e.g. chr1 for UCSC). LN is the length of the reference sequence. `getRefCount` returns the number of reference sequences. `getRefCoords` returns a vector of coordinates which can be used to extract all stored aligns for this sequence from the `bamReader` into a `bamRange` object.

**Author(s)**

Wolfgang Kaisers
Examples

bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
reader<-bamReader(bam,idx=TRUE)
getRefData(reader)
getRefCount(reader)
getRefId(reader, "chr1")
coords<-getRefCoords(reader, "chr1")
rg<-bamRange(reader,coords)
bamClose(reader)

getVal

getVal(object,members): Retrieving values for given types from an object

Description

Retrieving value of data-member from object

Usage

getVal(object,members)

Arguments

object An instance of class headerLine, headerReadGroup or headerProgram
member A (character) member identifier corresponding to the data-members of the given object (e.g. PN for headerProgram)

Details

Data members for class headerLine:

| VN | Format version |
| SO | Sorting order |

Valid values for sorting order (SO) are: unknown (default), unsorted, queryname or coordinate.

Data members for class readGroup:

| ID  | Read Group identifier |
| CN  | Name of sequencing center |
| DS  | Description |
| FO  | Flow order |
| KS  | Nucleotides corresponding to key sequence of each read |
| LB  | Library |
| PG  | Programs used for processing the Read Group |
| PI  | Predicted median insert size |
PL Sequencing Platform
SM Sample name

Valid values for Sequencing Platform (PL) are: CAPILLARY, LS454, ILLUMINA, SOLID, HELICOS, IONTORRENT or PACBIO.

Data members for class headerProgram

<table>
<thead>
<tr>
<th>ID</th>
<th>Program record identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN</td>
<td>Program name</td>
</tr>
<tr>
<td>CL</td>
<td>Command line</td>
</tr>
<tr>
<td>PP</td>
<td>Previous @PG-ID (Must match another ID in list)</td>
</tr>
<tr>
<td>VN</td>
<td>Program Version</td>
</tr>
</tbody>
</table>

Data members for refSeqDict can be accessed via data.frame generic functions (head,tail,[,<-) or by conversion (as.data.frame).

Author(s)

Wolfgang Kaisers

Examples

```r
bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
reader<-bamReader(bam)
header<-getHeader(reader)
htxt<-getHeaderText(header)
h1<-headerLine(htxt)
getVal(h1,"50")
bamClose(reader)
```

---

**groupRatio**

Calculates group-wise ratios of alignment depth (AD)

**Description**

`groupRatio` takes a `bamRange` and returns a data.frame (128 rows, number of columns=length of the longest sequence in range). `groupRatio` counts occurrences for every sequence position (column) and every phred value (row). `getQualQuantiles` takes a `bamReader` and a vector of quantiles (must be between 0 and 1) and returns a data.frame. The data.frame contains one row for each quantile and also as many columns as the maximum sequence length. `plotQualQuant` plots the values for quantiles 0.1, 0.25, 0.5, 0.75 and 0.9.

**Usage**

```r
groupRatio(object, lim=1.2, cut=0, order=NULL, f=mean)
```
Arguments

- object: exonLoessModel
- lim: numeric. Limit ratio. Must be > 1. The function returns the fraction of genetic position where AD-ratio between groups is > lim or the fraction of positions where AD-Ratio is < 1/lim (i.e. the larger ratio).
- cut: numeric. When >0, the function uses cutFlatAlignDepth for cutting out low alignment depth regions before calculating. alignment depth ratio.
- order: numeric. When given, the function reorders the sample groups. Can be used to provide ascending (or descending) group ordering, e.g. group1 < group2 < group3.
- f: function. Function for calculation of group accumulates. Defaults to mean. Alternatively median may also be used.

Details

The size of the returned value (abs(groupRatio)) indicates on which proportion of the genetic region, AD ratio between subsequent groups exceeds the given limit. For lim=1.1, group1<group2<group3, a returned value of 0.8 says that the AD ratios group2:group1 and group3:group2 are at least 1.1 (> 1) on 80 percent of the contained genomic positions. Negative values say that the relation is group1>group2>group3. This allows discrimination of up- and down-regulated genes.

Value

numeric

Author(s)

Wolfgang Kaisers

Examples

```r
## - - - - - - - - - - - - - - - - - - - - - - - - - -##
# Construct sampleBamFiles object
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
bs <- sampleBamFiles(bam)
bamFiles(bs) <- bam
sampleLabels(bs) <- "s1"
sampleGroups(bs) <- "g1"
checkBamFiles(bs)
nAligns(bs) <- bamCountAll(bs)
bs
## - - - - - - - - - - - - - - - - - - - - - - - - - -##
# Construct geneModel object
library(refGenome)
ucfile <- system.file("extdata", "hs.ucsc.small.RData", package="refGenome")
uc <- loadGenome(ucfile)
gt <- getGeneTable(uc)
gene_id <- as.character(gt$gene_id[1])
gm <- geneModel(uc, gene_id)
```
headerLine-class

Class "headerLine": Representation of header line segment of header section for BAM files.

Description

headerLine represents the header Line segment of header section.

Objects from the Class

Objects can be created by calls of the form `hl <- new("headerLine")`.

Slots

- `so`: character
- `VN`: character

Methods

- `getHeaderText`: Retrieve textual representation of header
- `getVal`: Retrieving values for given item names.
- `setVal`: Setting values for given item names.
- `as.list`: Coercing of data into a list

Author(s)

Wolfgang Kaisers

Examples

```r
bam <- system.file("extdata","accepted_hits.bam", package="rbamtools")
reader <- bamReader(bam)
header <- getHeader(reader)
htxt <- getHeaderText(header)
headline <- headerLine(htxt)
getVal(headline, "VN")
getVal(headline, "SO")
setVal(headline, "SO", "unsorted")
```
headerProgram-class

Class "headerProgram": Representation of header-program section of BAM header.

Description

headerProgram represents data contained in the header section of BAM-files.

Objects from the Class

Objects can be created by calls of the form hl<-new("headerProgram").

Slots

l: list

Methods

as.list signature(x = "headerProgram"): Converts data in object into list.
getVal signature(object = "headerProgram"): Returns value of given Segment.
setVal signature(object = "headerProgram"): Sets value of given segment.
getHeaderText signature(.Object = "headerProgram"): Returns textual representation of data as specified in SAM File Format.

Author(s)

Wolfgang Kaisers

References

The SAM Format Specification (v1.4-r985).

Examples

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader<-bamReader(bam)
isOpen(reader)
header<-getHeader(reader)
htxt<-getHeaderText(header)
headProg<-headerProgram(htxt)
getVal(headProg,"PN")
setVal(headProg,"PN","bwa")
l<-as.list(headProg)
ht<-getHeaderText(headProg)
bamClose(reader)
```
headerReadGroup-class

Class "headerReadGroup": Representation of read-group data in header section of BAM files

Description

headerReadGroup represents data contained in the header section of BAM-files.

Objects from the Class

Objects can be created by calls of the form hl<-new("headerReadGroup").

Slots

- **nrg**: integer. Number of read groups.
- **ntl**: integer. Number of different read group tags (currently = 12).
- **ID**: character. Read group identifier. Each read group must have an unique ID.
- **CN**: character. Name of sequencing center.
- **DS**: character. Description.
- **DT**: character. Date of run.
- **F0**: character. Flow order.
- **KS**: character. Array of nucleotide bases.
- **LB**: character. Library.
- **PG**: character. Programs used for processing.
- **PI**: character. Predicted median insert size.
- **PL**: character. Platform/technology for production of reads.
- **PU**: character. Unique platform identifier.
- **SM**: character. Sample name.

Methods

- **addReadGroup** signature(object="headerReadGroup", l="list"): Adds new read group to object.
- **as.list** signature(x="headerReadGroup"): Returns data stored in this object as list.
- **getVal** signature(object="headerReadGroup"): Returns value of given segment.
- **setVal** signature(object="headerReadGroup"): Sets value of given segment.
- **getHeaderText** signature(.Object="headerReadGroup"): Returns textual representation of data as specified in SAM Format.

Author(s)

Wolfgang Kaisers
isOpen-methods

References


Examples

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader <- bamReader(bam)
isOpen(reader)
header <- getHeader(reader)
htxt <- getHeaderText(header)
readGroup <- headerProgram(htxt)
getValue(readGroup, "ID")
setVal(readGroup, "ID", "newID")
l <- as.list(readGroup)
txt <- getHeaderText(readGroup)
bamClose(reader)
```

### Description

Checks whether the given bamReader or bamWriter represents an opened connection to a BAM-file.

### Usage

```r
isOpen(con, rw="")
```

### Arguments

- **con**: An instance of class bamReader or bamWriter.
- **rw**: This argument is included solely for compatibility with the function template for connections in package base. It’s not evaluated internally.

### Details

The function actually checks, if the externalptr (reader or writer) slot has nil-value.

### Methods

- signature(object = "bamReader") Returns TRUE when file connection is open.
- signature(object = "bamWriter") Returns TRUE when file connection is open.
multSeq Create combined sequences of sequential numbers.

Description
The function takes two vectors (of type integer) of equal length. For each entry pair (p,q) the returned vector contains p:q. Therefore c(a, b, c) and c(x, y, z) produces c(a:x, b:y, c:z).

Usage
multSeq(beg, end)

Arguments
beg numeric. Vector of start positions.
end numeric. Vector of end positions.

Value
Integer vector

Author(s)
Wolfgang Kaisers

Examples
# MultSeq
multSeq(c(1, 4, 7), c(2, 5, 8))

nucStats Table nucleotides in whole BAM file(s)

Description
The function counts occurrence of the Nucleotides A,C,G,T for whole BAM file from bamReader object or a list of BAM files. Letters different from A,C,G,T are subsumed in the value N. The function returns a data.frame with 8 columns. The names indicate which position contains the count value each nucleotide. The function also calculates GC-content and AT/GC ratio which make up the last two columns. The returned data.frame contains one row for each reference sequence (when used for bamReader) or one row for each given BAM file name.

Usage
nucStats(object,...)
rangeSegCount-class

Arguments

object       bamReader object. The reader must be opened and have initialized index.
...  Optional name for BAM-index files (argument name=idxInfiles).

Value

data.frame

Author(s)

Wolfgang Kaisers

Examples

bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
nucStats(bam)
idx<- system.file("extdata", "accepted_hits.bam.bai", package="rbamtools")
nucStats(bam,idxInfiles=idx)
reader<-bamReader(bam,idx=TRUE)
nucStats(reader)
range<-bamRange(reader,c(0,0,10000))
nucStats(range)

rangeSegCount-class  Class "rangeSegCount": Represents alignment counts in segments of genetic regions.

Description

rangeSegCount.

Details

Two kinds of coordinates are used: A) coords (numeric) containing seqid, start and end position (see also bamRange) which are used for reading alignments from a BAM file, and B) position (numeric) which contains an arbitrary sized vector. The position values define the borders of genomic segments in which alignments are counted. The segments usually would cover a whole chromosome and separates genetic and intergenic regions. The result is then the number of alignments in genetic and intergenic regions which then can be used as gene expression estimates. The segments are defined as right open intervals. When the position is given by c(1,10,20) for example, the first segment is defined be nucleotide positions 1 to 9 and the second segment is 10 to 19. The counting mechanism only takes the alignment position into account. A possible overhang of the alignment over a segment boundary is not represented here (because this would complicate the model of 'counting' whole alignments.)

Objects from the Class

Objects can be created by calls of the form range<-rangeSegCount(object, coords, segments, complex).
Slots

position: Integer. Vector of chromosomal positions which define the segments in which alignments are counted.

count: Integer. Number of alignments which are located in the position segments

refname: Character. Name of the reference sequence as given in the header section of the BAM file (e.g. 'chr1').

LN: Integer. Total size of reference sequence (chromosome).

coords: Numeric. A numeric vector giving the genomic coordinates from which alignments were counted.

complex: Logical.

Methods

as.data.frame signature(x="rangeSegCount") : Returns data.frame representation of aligns.

coerce signature(from="rangeSegCount", to="data.frame") : Coercion of rangeSegCount to data.frame.

meltDownSegments signature(object="rangeSegCount", factor="numeric"):

Defines the factor by which the number of counting segments is shrunk. factor=2 means that counts for two adjacent segments are accumulated.

Author(s)

Wolfgang Kaisers

Examples

# A) Open reader
#--------------------------------------------------
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
idx<-paste(bam,".bai",sep=".")
# Open BAM file
reader<-bamReader(bam, idx=TRUE)

# B) Count range segment
#--------------------------------------------------
coords <- c(0, 0, 2e4)
segments <- seq(14000, 20000, 20)
segcount<-rangeSegCount(reader, coords, segments)
segcount
dfr<as.data.frame(segcount)
## Not run:
plot(count~position, dfr, type="l", las=1, bty="n", lwd=1.5)

## End(Not run)
rangeToFastq

rangeToFastq: Extract read information from alignments for given genomic range as fastq.

Description

The function rangeToFastq writes all (or selected) aligns from a bamRange into a compressed fastq file.

Usage

rangeToFastq(object, filename, which, append=FALSE)

Arguments

object  bamReader. Must be opened.
filename Name of output 'fastq' file.
which (Optional) Logical vector. When given, the routine checks for each of the given values the value of which. When TRUE the corresponding align will be written to the output file, otherwise will be skipped.
append (Optional) When TRUE, the routine will append to an existing file. Otherwise existing files will be overwritten.

Details

range2fastq ist deprecated and will be replaced by rangeToFastq soon.

Author(s)

Wolfgang Kaisers

Examples

bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader <- bamReader(bam, idx=TRUE)
coords <- as.integer(c(0, 0, 249250621))
range <- bamRange(reader, coords)
## Not run: rangeToFastq(range, "range.fq.gz")
index <- sample(1:size(range), 100)
## Not run: rangeToFastq(range, "range_subset.fq.gz", which=index)
**readerToFastq**

*readerToFastq: Extraction of a subset of alignments from a BAM files into fastq format.*

**Description**

The function `readerToFastq` writes (remaining) aligns or some part of the stored aligns in a BAM file as reads into a compressed fastq file.

**Usage**

```r
readerToFastq(object, filename, which, append=FALSE)
```

**Arguments**

- **object**: bamReader. Must be opened.
- **filename**: Name of output 'fastq' file.
- **which**: (Optional) Logical vector. When given, the routine checks for each of the given values the value of which. When TRUE the next retrieved align will be written to the output file, otherwise will be skipped.
- **append**: (Optional) When TRUE, the routine will append to an existing file. Otherwise existing files will be overwritten.

**Details**

The function 'reader2fastq' is deprecated and will be replaced by 'readerToFastq' soon.

**Value**

numeric

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader <- bamReader(bam)
## Not run: readerToFastq(reader, "out.fq.gz")
bamClose(reader)
# Reopen in order to point to first align
reader <- bamReader(bam)
index <- sample(1:100, 20)
## Not run: readerToFastq(reader, "out_subset.fq.gz", which=index)
```
readPooledBamGapDf

Description

The function takes vectors of BAM-file names (plus accompanying names of BAM-index-files) and returns extracted data as data.frame. The data frame contains coordinates of align gaps plus a gap-quality-score (gqs) which quantifies information amount for detection of splice sites.

Usage

readPooledBamGapDf(infiles, idxInfiles=paste(infiles, ".bai", sep=""))

Arguments

infiles Filenames of BAM-files to be opened for data extraction.
idxInfiles Optional: Name of BAM-index files

Author(s)

Wolfgang Kaisers

Examples

bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
rpb <- readPooledBamGaps(bam)
rpdf <- readPooledBamGapDf(bam)
hist(rpdf$gqs, las=1, xlab="", main="Gap quality score")

readPooledBamGaps

Description

The function takes vectors of BAM-file names (plus accompanying names of BAM-index-files) and return a bamGapList object. The bamGapList object can be merged with other bamGapList objects and data can be extracted with the 'as.data.frame' method.

Usage

readPooledBamGaps(infiles,idxInfiles=paste(infiles,".bai",sep=""))
Arguments

infiles  Filenames of BAM-files to be opened for data extraction.
idxInfiles  Optional: Name of BAM-index files

Value

bamGapList

Author(s)

Wolfgang Kaisers

Examples

bam<--system.file("extdata","accepted_hits.bam",package="rbamtools")
rpb<--readPooledBamGaps(bam)
rdpdf<--readPooledBamGapDf(bam)
hist(rdpdf$gqs)

refSeqDict-class  Class "refSeqDict": Representation of data from reference sequence dictionary in BAM file header.

Description

refSeqDict represents Data contained in the header section of BAM-files.

Objects from the Class

Objects can be created by calls of the form h1<-new("refSeqDict").

Slots

SN: character
LN: numeric
AS: character
MS: numeric
SP: character
UR: character
sampleBamFiles-class

Methods

`dim` signature(x="refSeqDict"): Returns dimension of data: number of rows and 6 columns.

`as.data.frame` signature(object="refSeqDict"): Combines data of Slots to data.frame.

`coerce` signature(from="bamRange", to="data.frame"): Coercion of refSeqDice to data.frame.

`removeSeqs` signature(object="refSeqDict", rows="numeric"): Removes reference sequence (i.e. one row from data.frame) entry.

`addSeq` signature(.Object="refSeqDict", SN, LN, AS, MS, SP, UR): Adds reference sequence (i.e. one row in data.frame) entry.

`head` signature(.Object="refSeqDict", n): Returns head of data.frame representation.

`tail` signature(.Object="refSeqDict", n): Returns tail of data.frame representation.

`getHeaderText` signature(.Object="refSeqDict"): Returns textual representation of data stored in this class as described in SAM Format Specification.

Author(s)

Wolfgang Kaisers

References

The SAM Format Specification (v1.4-r985).

Examples

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader <- bamReader(bam)
isOpen(reader)
header <- getHeader(reader)
htxt <- getHeaderText(header)
refSeqs <- refSeqDict(htxt)
dim(refSeqs)
ht <- getHeaderText(refSeqs)
df <- as.data.frame(refSeqs)
addSeq(refSeqs, SN="nextSeqName", LN=1000)
removeSeqs(refSeqs, 3)
bamClose(reader)
```

---

**sampleBamFiles-class**  
Class "sampleBamFiles": Data on multiple BAM files from an experimental setting.

Description

The object contains data on location of BAM files, experimental group assignment and number of aligns per BAM file. The align-numbers are used for normalisation when alignment depth is plotted for single genes.
**Objects from the Class**

Objects can be created by calls of the form `bs <- sampleBamFiles(object)` where object is a numeric vector of length 1.

**Slots**

- `bamFiles`: "character": Location of BAM files
- `bamIdxFiles`: "character": Location of BAM index files
- `nAligns`: "numeric": Total number of alignments in each BAM file.
- `group`: "factor": Group assignment
- `label`: "character": Short textual identifier for each sample.
- `length`: "integer": Vector length for bamFiles, bamIdxFiles, nAligns, group and label.
- `env`: "environment": Contains additional data (e.g. group table).

**Methods**

- `show` signature(object="sampleBamFiles"): Prints a short message with some summarizing data.
- `bamFiles` signature(object="sampleBamFiles"): Returns names of BAM files.
- `bamIdxFiles` signature(object="sampleBamFiles"): Returns names of BAM index files.
- `length` signature(object="sampleBamFiles"): Returns number of BAM files.
- `nAligns` signature(object="sampleBamFiles"): Returns total number of aligns in BAM files.
- `sampleLabels` signature(object="sampleBamFiles"): Returns sample labels for BAM files.
- `sampleGroups` signature(object="sampleBamFiles"): Returns group assignment for BAM files.
- `groupTable` signature(object="sampleBamFiles"): Returns group table (if present).

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
bs <- sampleBamFiles(bam)
sampleLabels(bs) <- "s1"
sampleGroups(bs) <- "g1"
checkBamFiles(bs)
nAligns(bs) <- bamCountAll(bs)
bs
```
**Description**

The function `saveAldData` reads alignment data for all genes in `geneList` sequentially from all Bam files listed in `sampleBamFiles`. The function cuts out intronic regions using the `exonAlignDepth` class and calculates a smoothed groupwise estimate using the `exonLoessModel` function. In a second region cutting step, remaining regions with low alignment depth are cut out using `cutFlatAlignDepth`. Finally, the function calculates group-wise ratios of alignment depth using the `groupRatio` function.

**Usage**

```r
saveAldData(bs, gl, path, order=NULL, lim=1.1, startId=1, f=mean)
```

**Arguments**

- `bs` sampleBamFiles. Contains BAM file names and group assignment.
- `gl` geneList. Contains a list of geneModel objects. Only gene-name, gene-id and global genetic region is used. Object construction can be abridged using `interior=FALSE`.
- `path` character. Base directory where output files are written to.
- `order` numeric. Defines order in which group-levels of `sampleBamFiles` object are used for `groupRatio`.
- `lim` numeric. Limit used for `groupRatio` function. The percentage of genomic position where this limit is exceeded between all (ordered) group levels is given as `gr` (and `cgr`) value in `aldrat` table.
- `startId` numeric. Index used for first gene. Starting value for subsequent numbering of genes.
- `f` function. Function for calculation of group accumulates. Defaults to `mean`. Alternatively `median` may also be used.

**Author(s)**

Wolfgang Kaisers

**See Also**

`bamReader`
Examples

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader <- bamReader(bam, idx=TRUE)
# Extract data for HLHL17 gene:
seqid <- "chr1"
start <- 895967
end <- 901099
gene_name <- "HLHL17" # (optional)
ranges <- data.frame(seqid=seqid, start=start, end=end, gene_name=gene_name)
## Not run: saveAldData(reader, ranges=ranges, filename="new_file.bam", f=mean)
```

---

**segmentize**

*Segmentation of vector using segment coordinates*

**Description**

The function returns extracts of a data vector using two vector defining begin and end coordinates. An offset can be used.

**Usage**

```r
segmentize(x, begin, end, offset=1, margin=1, invert=FALSE)
```

**Arguments**

- `x` numeric object. Data vector from which segments are extracted.
- `begin` numeric object. Indices first segment elements.
- `end` numeric object. Indices of last segment elements. Length must be equal to `length(begin)`. `segmentize` requires all(end>=begin).
- `offset` numeric object. The indices are shifted so that the first entry is indexed by value indicated by offset. Standard value is 1 (default behaviour in R).
- `margin` numeric object. Direction of segmentation for `matrix` and `data.frame` objects (1=rows, 2=columns).
- `invert` logical object. When TRUE, the complement of the selected segments is returned. Here, the ordering of the returned values cannot be changed.

**Details**

The inversion option internally operates using logical values. As a consequence, when `begin` and `end` define overlapping regions, in effect the union of these regions is removed, because regions are only retained when they are not covered by any segment defined by `begin` and `end`.

**Value**

Integer vector
setVal

Author(s)

Wolfgang Kaisers

Examples

# Create data vector
x <- rep(8, 11)
x[3:5] <- 1:3
x[7:9] <- 4:6
names(x) <- 10:20
# Define extracted segments
sgb <- c(3, 7)
sge <- c(5, 9)
sgm <- segmentize(x, sgb, sge)
names(sgm) <- segmentize(names(x), sgb, sge)
# Use offset
offset <- 10
sgb <- c(12, 16)
sge <- c(14, 18)
segmentize(x, sgb, sge, offset)
# Matrix
m <- matrix(0L, nrow=11, ncol=5)
rownames(m) <- 10:20
colnames(m) <- letters[1:5]

for(i in 1:5)
  m[c(3:5, 7:9), i] <- c(1:3, 4:6) * i
ms <- segmentize(m, sgb, sge, offset)

setVal

setVal(object, members, values): Setting values for given data items

Description

Setting values of data-members for object

Usage

setVal(object, members, values)

Arguments

object An instance of class headerLine, headerReadGroup or headerProgram
members A (character) vector of member identifiers corresponding to data-members of
  the given object (e.g. PN for headerProgram)
values Values that are written into data members
Details

The members and values (vectors) must have the same length.

Data members for class headerLine:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>VN</td>
<td>Format version</td>
</tr>
<tr>
<td>SO</td>
<td>Sorting order</td>
</tr>
</tbody>
</table>

Valid values for sorting order (SO) are: unknown (default), unsorted, queryname or coordinate.

Data members for class readGroup:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Read Group identifier</td>
</tr>
<tr>
<td>CN</td>
<td>Name of sequencing center</td>
</tr>
<tr>
<td>DS</td>
<td>Description</td>
</tr>
<tr>
<td>FO</td>
<td>Flow order</td>
</tr>
<tr>
<td>KS</td>
<td>Nucleotides corresponding to key sequence of each read</td>
</tr>
<tr>
<td>LB</td>
<td>Library</td>
</tr>
<tr>
<td>PG</td>
<td>Programs used for processing the Read Group</td>
</tr>
<tr>
<td>PI</td>
<td>Predicted median insert size</td>
</tr>
<tr>
<td>PL</td>
<td>Sequencing Platform</td>
</tr>
<tr>
<td>SM</td>
<td>Sample name</td>
</tr>
</tbody>
</table>

Valid values for Sequencing Platform (PL) are: CAPILLARY, LS454, ILLUMINA, SOLID, HELLICOS, IONTORRENT or PACBIO.

Data members for class headerProgram

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Program record identifier</td>
</tr>
<tr>
<td>PN</td>
<td>Program name</td>
</tr>
<tr>
<td>CL</td>
<td>Command line</td>
</tr>
<tr>
<td>PP</td>
<td>Previous @PG-ID (Must match another ID in list)</td>
</tr>
<tr>
<td>VN</td>
<td>Program Version</td>
</tr>
</tbody>
</table>

Data members for refSeqDict can be accessed via data.frame generic functions (head,tail[,]<-) or by conversion (as.data.frame).

Author(s)

Wolfgang Kaisers

Examples

```r
bam <- system.file("extdata", "accepted_hits.bam", package="rbamtools")
reader<-bamReader(bam)
header<-getHeader(reader)
htxt<-getHeaderText(header)
```
hl<-'headerLine(htxt)
setVal(hl,"SO","coordinate")
bamClose(reader)
Index

*Topic **AT/CG ratio**  
nucStats, 54

*Topic **GC content**  
nucStats, 54

*Topic **bamAlign**  
alignDepth-class, 4  
as.data.frame-methods, 6  
bamAlign, 6  
bamAlign-class, 7  
bamRange-class, 16  
bamWriter, 25  
countNucs, 28  
exonAlignDepth-class, 30  
exonLoessModel-class, 32  
gapList-class, 37  
gapSiteList-class, 38  
geneAlignDepth-class, 40  
GenomePartition-class, 41

*Topic **bamClose**  
bamClose-methods, 10

*Topic **bamCountAll**  
bamCount, 10  
bamCountAll, 11

*Topic **bamGapList**  
bamGapList-class, 11

*Topic **bamHeader**  
bamHeader-class, 12  
bamHeaderText-class, 13  
bamRange-class, 16  
bamReader-class, 20  
bamSave, 23  
bamSort, 24  
filename-methods, 37  
getNextAlign-methods, 44  
headerLine-class, 50  
headerProgram-class, 51  
headerReadGroup-class, 52  
nucStats, 54  
rangeSegCount-class, 55  
refSeqDict-class, 60

*Topic **bamSave**  
bamSave, 23

*Topic **bamSort**  
bamSort, 24

*Topic **bamWriter**  
bamWriter, 25  
filename-methods, 37

*Topic **classes**  
bamHeader-class, 12  
bamHeaderText-class, 13  
bamRange-class, 16  
bamReader-class, 20  
bamWriter-class, 26  
headerLine-class, 50  
headerProgram-class, 51  
headerReadGroup-class, 52  
rangeSegCount-class, 55  
refSeqDict-class, 60

*Topic **countNucs**  
countNucs, 28

*Topic **countTextLines**  
countTextLines, 29

*Topic **createIdxBatch**  
createIdxBatch, 30
INDEX

*Topic extractBamRegions
extractBamRegions, 34

*Topic extractGeneRegions
extractGeneRegions, 35

*Topic extractRanges
extractRanges, 36

*Topic filename
filename-methods, 37

*Topic gapList
  as.data.frame-methods, 6

*Topic getHeaderText
getHeaderText-methods, 43

*Topic getNextAlign
getNextAlign-methods, 44

*Topic getQualDf
getQualDf, 45

*Topic getRefData
bamCountAll, 11
getRefData, 46

*Topic getVal
getVal, 47

*Topic groupRatio
groupRatio, 48

*Topic headerLine
headerLine-class, 50

*Topic headerProgram
headerProgram-class, 51

*Topic headerReadGroup
headerReadGroup-class, 52

*Topic bamCount
bamCount, 10
bamRange, 15
countTextLines, 29

*Topic isOpen
isOpen-methods, 53

*Topic methods
  as.data.frame-methods, 6
  bamClose-methods, 10
  filename-methods, 37
  getHeaderText-methods, 43
  getNextAlign-methods, 44
  isOpen-methods, 53

*Topic multiSeq
multiSeq, 54

*Topic nucStats
nucStats, 54

*Topic package
rbamtools-package, 3

*Topic rangeToFastq
rangeToFastq, 57

*Topic readPooledBamGapDf
readPooledBamGapDf, 59
readPooledBamGaps, 59

*Topic readPooledBamGaps
readPooledBamGapDf, 59
readPooledBamGaps, 59

*Topic readerToFastq
readerToFastq, 58

*Topic refSeqDict
refSeqDict-class, 60

*Topic sampleBamFiles
sampleBamFiles-class, 61

*Topic saveAldData
saveAldData, 63

*Topic segmentize
  multiSeq, 54
  segmentize, 64

*Topic setVal
setVal, 65

[<-,refSeqDict-method
  (refSeqDict-class), 60

addReadGroup (headerReadGroup-class), 52
addReadGroup, headerReadGroup-method
  (headerReadGroup-class), 52
addReadGroup-methods
  (headerReadGroup-class), 52
addSeq (refSeqDict-class), 60
addSeq, refSeqDict-method
  (refSeqDict-class), 60
aldRatio (exonAlignDepth-class), 30
aldRatio, exonAlignDepth-method
  (exonAlignDepth-class), 30
bamIdxFiles<-., sampleBamFiles, character-method
(bamIdxFiles-class), 61
bamIdxFiles<--methods
(sampleBamFiles-class), 61
bamRange, 15
bamRange-class, 16
bamReader, 19
bamReader-class, 20
bamSave, 23
bamSave, bamReader, ANY, ANY-method
(bamSave), 23
bamSave, bamReader, bamWriter, missing-method
(bamSave), 23
bamSave, bamWriter, bamAlign, numeric-method
(bamSave), 23
bamSave, bamWriter, bamRange, character-method
(bamSave), 23
bamSave, bamWriter, bamRange, missing-method
(bamSave), 23
bamSave, bamWriter, bamRange, numeric-method
(bamSave), 23
bamSave, bamWriter-method
(bamWriter-class), 26
bamSave-methods (bamSave), 23
bamSort, 24
bamSort, bamReader, ANY, ANY, ANY-method
(bamSort), 24
bamSort, bamReader-method
(bamReader-class), 20
bamSort, sampleBamFiles, ANY, ANY, ANY-method
(bamSort), 24
bamSort, sampleBamFiles-method
(bamSort), 24
bamSort-methods (bamSort), 24
bamWriter, 25
bamWriter, bamHeader-method
(bamHeader-class), 12
bamWriter-class, 26
checkBamFiles (sampleBamFiles-class), 61
checkBamFiles, sampleBamFiles-method
(sampleBamFiles-class), 61
checkBamFiles-methods
(sampleBamFiles-class), 61
checkPartition (GenomePartition-class), 41
cigarData (bamAlign-class), 7
cigarData, bamAlign-method
(bamAlign-class), 7
cigarData-methods (bamAlign-class), 7
coerce, rangeSegCount,
data.frame-method
(rangeSegCount-class), 55
coerce, bamGapList, data.frame-method
(bamGapList-class), 11
coerce, bamRange, data.frame-method
(bamRange-class), 16
coerce, gapList, data.frame-method
gapList-class), 37
coerce, gapSiteList, data.frame-method
gapSiteList-class), 38
coerce, refSeqDict, data.frame-method
(refSeqDict-class), 60
countNucs, 28
countNucs, bamAlign-method (countNucs),
countNucs, bamRange-method (countNucs),
countNucs-methods (countNucs), 28
countPartition (GenomePartition-class),
countPartition, GenomePartition, bamReader-method
genomePartition-class), 41
countPartition, GenomePartition, data.frame-method
genomePartition-class), 41
countPartition-methods
genomePartition-class), 41
countTextLines, 29
create.idx.batch (createIdxBatch), 30
create.index (bamReader-class), 20
create.index (bamReader-class), 20
create.index-methods (bamReader-class), 20
createIdxBatch, 30
createIndex (bamReader-class), 20
createIndex, bamReader, character-method
(bamReader-class), 20
countPartition, GenomePartition, data.frame-method
(sampleBamFiles-class), 61
createIndex(sampleBamFiles, missing_method) (sampleBamFiles-class), 61
createIndex-methods (bamReader-class), 20
cutFlatAlignDepth (exonAlignDepth-class), 30
cutFlatAlignDepth, exonAlignDepth-method (exonAlignDepth-class), 30
cutFlatAlignDepth, exonLoessModel-method (exonLoessModel-class), 32
cutFlatAlignDepth-methods (exonAlignDepth-class), 30
dim, refSeqDict-method (refSeqDict-class), 60
exonAlignDepth (exonAlignDepth-class), 30
exonAlignDepth, geneAlignDepth-method (exonAlignDepth-class), 30
exonAlignDepth-class, 30
exonAlignDepth-methods (exonAlignDepth-class), 30
exonLoessModel (exonLoessModel-class), 32
exonLoessModel, exonAlignDepth-method (exonLoessModel-class), 32
exonLoessModel-class, 32
exonLoessModel-methods (exonLoessModel-class), 32
extractBamRegions, 34
extractGeneRegions, 35
extractGeneRegions, bamReader, bamWriter, geneAlignDepth-method (extractGeneRegions), 35
extractGeneRegions, bamReader, character, genelist-method (extractGeneRegions), 35
extractGeneRegions, sampleBamFiles, sampleBamFiles, geneAlignDepth-method (extractGeneRegions), 35
extractGeneRegions-methods (extractGeneRegions), 35
extractRanges, 36
extractRanges, bamReader-method (extractRanges), 36
extractRanges-methods (extractRanges), 36
failedQC (bamAlign-class), 7
failedQC, bamAlign-method (bamAlign-class), 7
failedQC-methods (bamAlign-class), 7
failedQC<- (bamAlign-class), 7
failedQC<<-, bamAlign-method (bamAlign-class), 7
failedQC<<<--methods (bamAlign-class), 7
filename (filename-methods), 37
filename, bamReader-method (bamReader-class), 20
filename, bamWriter-method (bamWriter-class), 26
filename-methods, 37
firstInPair (bamAlign-class), 7
firstInPair, bamAlign-method (bamAlign-class), 7
firstInPair-methods (bamAlign-class), 7
firstInPair<-- (bamAlign-class), 7
firstInPair<---, bamAlign-method (bamAlign-class), 7
firstInPair<<<--methods (bamAlign-class), 7
flag (bamAlign-class), 7
flag, bamAlign-method (bamAlign-class), 7
flag-methods (bamAlign-class), 7
flag<- (bamAlign-class), 7
flag<-, bamAlign-method (bamAlign-class), 7
flag<<<--methods (bamAlign-class), 7
gapList (gapList-class), 37
gapList, bamReader, numeric-method (gapList-class), 37
gapList-class, 37
gapList-methods (gapList-class), 37
geneAlignDepth (geneAlignDepth-class), 40
geneAlignDepth, sampleBamFiles, geneModel-method (geneAlignDepth-class), 40
geneAlignDepth-class, 40
geneAlignDepth-methods (geneAlignDepth-class), 40
genomePartition (GenomePartition-class), 41
genomePartition, bamReader, data.frame-method (GenomePartition-class), 41
GenomePartition-class, 41
genomePartition-methods (GenomePartition-class), 41
getAlignCounts (GenomePartition-class), 41
getIndex 

getAlignCounts, GenomePartition-method 
  (GenomePartition-class), 41
getAlignCounts-methods 
  (GenomePartition-class), 41
getAlignRange (bamRange-class), 16
getAlignRange, bamRange-method 
  (bamRange-class), 16
getAlignRange-methods (bamRange-class), 16
getCoords (bamRange-class), 16
getCoords, bamRange-method 
  (bamRange-class), 16
getCoords-methods (bamRange-class), 16
getDepth (alignDepth-class), 4
getDepth, alignDepth-method 
  (alignDepth-class), 4
getDepth-methods (alignDepth-class), 4
getFileSizeTable (GenomePartition-class), 41
getFileSizeTable, GenomePartition-method 
  (GenomePartition-class), 41
getFileSizeTable-methods 
  (GenomePartition-class), 41
getGridAlignCounts 
  (GenomePartition-class), 41
getGridAlignCounts, GenomePartition-method 
  (GenomePartition-class), 41
getGridAlignCounts-methods 
  (GenomePartition-class), 41
getHeader (bamHeaderText-class), 13
getHeader, bamHeaderText-method 
  (bamHeaderText-class), 13
getHeader, bamReader-method 
  (bamReader-class), 20
getHeaderText (getTextHeaderText-methods), 43
ggetTextHeaderText, bamHeader-method 
  (bamHeader-class), 12
ggetTextHeaderText, bamHeaderText-method 
  (bamHeaderText-class), 13
getTextHeaderText, bamReader-method 
  (bamReader-class), 20
ggetTextHeaderText, headerLine-method 
  (headerLine-class), 50
ggetTextHeaderText, headerProgram-method 
  (headerProgram-class), 51
getTextHeaderText, headerReadGroup-method 
  (headerReadGroup-class), 52
ggetTextHeaderText, refSeqDict-method 
  (refSeqDict-class), 60
ggetTextHeaderText-methods, 43
ggetNextAlign (getNextAlign-methods), 44
ggetNextAlign, bamRange-method 
  (bamRange-class), 16
ggetNextAlign, bamReader-method 
  (bamReader-class), 20
ggetNextAlign-methods, 44
ggetNormFactor (exonAlignDepth-class), 30
ggetNormFactor, exonAlignDepth-method 
  (exonAlignDepth-class), 30
ggetNormFactor-methods 
  (exonAlignDepth-class), 30
ggetParams (bamRange-class), 16
ggetParams, alignDepth-method 
  (alignDepth-class), 4
ggetParams, bamRange-method 
  (bamRange-class), 16
ggetParams-methods (alignDepth-class), 4
ggetParams-methods (bamRange-class), 16
ggetPos (alignDepth-class), 4
ggetPos, alignDepth-method 
  (alignDepth-class), 4
ggetPos-methods (alignDepth-class), 4
ggetPos-methods (bamRange-class), 16
ggetPrevAlign (bamRange-class), 16
ggetPrevAlign, bamRange-method 
  (bamRange-class), 16
ggetPrevAlign-methods (bamRange-class), 16
ggetQualDf, 45
ggetQualDf, bamRange, logical-method 
  (getTextHeaderText), 45
ggetQualDf, bamRange, missing-method 
  (getTextHeaderText), 45
ggetQualDf-methods (getTextHeaderText), 45
ggetQualQuantiles (getTextHeaderText), 45
ggetQualQuantiles, bamRange, numeric-method 
  (getTextHeaderText), 45
ggetQualQuantiles-methods (getTextHeaderText), 45
ggetTextHeaderText, refData-methods (getTextHeaderText), 45
ggetRefCoords (getTextHeaderText), 46
ggetTextHeaderText, refData-methods (getTextHeaderText), 45
ggetRefCount (getTextHeaderText), 46
ggetTextHeaderText, refData-methods (getTextHeaderText), 45
ggetRefCount-methods (getTextHeaderText), 46
ggetTextHeaderText, refData-methods (getTextHeaderText), 45
index.initialized-methods
(indexReader-class), 20
indexInitialized-methods
(indexReader-class), 20
indexinitialized-methods
(indexReader-class), 20
initialize, exonAlignDepth
(exonAlignDepth-class), 30
initialize, exonLoessModel
(exonLoessModel-class), 32
initialize, geneAlignDepth
(geneAlignDepth-class), 40
initialize, rangeSegCount-method
(rangeSegCount-class), 55
initialize, sampleBamFiles
(sampleBamFiles-class), 61
initialize, bamAlign-method
(bamAlign-class), 7
initialize, bamGapList-method
(bamGapList-class), 11
initialize, bamHeader-method
(bamHeader-class), 12
initialize, bamHeaderText-method
(bamHeaderText-class), 13
initialize, bamRange-method
(bamRange-class), 16
initialize, bamWriter-method
(bamWriter-class), 26
initialize, gapList-method
(gapList-class), 37
initialize, gapSiteList-method
(gapSiteList-class), 38
initialize, headerLine-method
(headerLine-class), 50
initialize, headerProgram-method
(headerProgram-class), 51
initialize, headerReadGroup-method
(headerReadGroup-class), 52
initialize, refSeqDict-method
(refSeqDict-class), 60
insertPastCurrent-methods
(bamRange-class), 16
load-index-methods
(bamReader-class), 20
loadindex-methods
(bamReader-class), 20
mapQuality-methods
(bamAlign-class), 7
matePosition-methods
(bamAlign-class), 7
mateRefID-methods
(bamAlign-class), 7
insertPastCurrent-methods
(bamRange-class), 16
index-initialized-methods
(bamReader-class), 20
insertSize-methods
(bamAlign-class), 7
isOpen-methods
(bamReader-class), 20
isOpen-methods
(bamWriter-class), 26
junctionSites-methods
(exonAlignDepth-class), 30
length-methods
(exonAlignDepth-class), 30
load-index-methods
(bamReader-class), 20
loadIndex-methods
(bamReader-class), 20
mapQuality-methods
(bamAlign-class), 7
matePosition-methods
(bamAlign-class), 7
mateRefID-methods
(bamAlign-class), 7
insertPastCurrent-methods
(bamRange-class), 16
materesversestrand (bamAlign-class), 7
materesversestrand,bamAlign-method
(bamAlign-class), 7
materesversestrand-methods
(bamAlign-class), 7
materesversestrand<-(bamAlign-class), 7
materesversestrand<-,bamAlign-method
(bamAlign-class), 7
materesversestrand<--methods
(bamAlign-class), 7
mateunmapped (bamAlign-class), 7
mateunmapped,bamAlign-method
(bamAlign-class), 7
mateunmapped-methods (bamAlign-class), 7
mateunmapped<-(bamAlign-class), 7
mateunmapped<-,bamAlign-method
(bamAlign-class), 7
mateunmapped<--methods
(bamAlign-class), 7
meltDownSegments (rangeSegCount-class), 55
meltDownSegments,rangeSegCount-method
(rangeSegCount-class), 55
meltDownSegments-methods
(rangeSegCount-class), 55
merge.bamGapList (bamGapList-class), 11
merge.gapSiteList (gapSiteList-class), 38
moveCurrentAlign (bamRange-class), 16
moveCurrentAlign,bamRange-method
(bamRange-class), 16
moveCurrentAlign-methods
(bamRange-class), 16
multSeq, 54
nAlignGaps (gapList-class), 37
nAlignGaps,bamGapList-method
(bamGapList-class), 11
nAlignGaps,gapList-method
(gapList-class), 37
nAlignGaps,gapSiteList-method
(gapSiteList-class), 38
nAlignGaps-methods (gapList-class), 37
nAligns (gapList-class), 37
nAligns,bamGapList-method
(bamGapList-class), 11
nAligns,gapList-method (gapList-class), 37
nAligns,gapSiteList-method (gapSiteList-class), 38
nAligns sampleBamFiles-method
(sampleBamFiles-class), 61
nAligns-methods (gapList-class), 37
nAligns<-(sampleBamFiles-class), 61
nAligns<-,sampleBamFiles,character,numeric-method
(sampleBamFiles-class), 61
nAligns<-,sampleBamFiles,integer-method
(sampleBamFiles-class), 61
nAligns<--methods
(sampleBamFiles-class), 61
name (bamAlign-class), 7
name,bamAlign-method (bamAlign-class), 7
name-methods (bamAlign-class), 7
nCigar (bamAlign-class), 7
nCigar,bamAlign-method
(bamAlign-class), 7
nCigar-methods (bamAlign-class), 7
nCigar-methods (bamAlign-class), 7
nucStats, 54
nucStats,bamRange-method (nucStats), 54
nucStats,bamReader-method (nucStats), 54
nucStats,character-method (nucStats), 54
nucStats-methods (nucStats), 54
paired (bamAlign-class), 7
paired,bamAlign-method
(bamAlign-class), 7
paired-methods (bamAlign-class), 7
paired<-(bamAlign-class), 7
paired<-,bamAlign-method
(bamAlign-class), 7
paired<--methods (bamAlign-class), 7
pcrORopt_duplicate (bamAlign-class), 7
pcrORopt_duplicate,bamAlign-method
(bamAlign-class), 7
pcrORopt_duplicate-methods
(bamAlign-class), 7
pcrORopt_duplicate<-(bamAlign-class), 7
pcrORopt_duplicate<-,bamAlign-method
(bamAlign-class), 7
pcrORopt_duplicate<--methods
(bamAlign-class), 7
plotAlignDepth (alignDepth-class), 4
plotAlignDepth,alignDepth-method
(alignDepth-class), 4
plotAlignDepth-methods
  (alignDepth-class), 4
plotQualQuant (getQualDF), 45
plotQualQuant, bamRange-method
  (getQualDF), 45
plotQualQuant-methods (getQualDF), 45
pop_back (bamRange-class), 16
pop_back, bamRange-method
  (bamRange-class), 16
pop_back-methods (bamRange-class), 16
pop_front (bamRange-class), 16
pop_front, bamRange-method
  (bamRange-class), 16
pop_front-methods (bamRange-class), 16
position (bamAlign-class), 7
position, bamAlign-method
  (bamAlign-class), 7
position-methods (bamAlign-class), 7
properPair (bamAlign-class), 7
properPair, bamAlign-method
  (bamAlign-class), 7
properPair-methods (bamAlign-class), 7
properPair<- (bamAlign-class), 7
properPair<-, bamAlign-method
  (bamAlign-class), 7
properPair--methods (bamAlign-class), 7
push_back (bamRange-class), 16
push_back, bamRange-method
  (bamRange-class), 16
push_back-methods (bamRange-class), 16
push_front (bamRange-class), 16
push_front, bamRange-method
  (bamRange-class), 16
push_front-methods (bamRange-class), 16
range2fastq (rangeToFastq), 57
range2fastq, bamRange-method
  (rangeToFastq), 57
range2fastq-methods (rangeToFastq), 57
rangeSegCount (rangeSegCount-class), 55
rangeSegCount, bamReader-method
  (bamReader-class), 20
rangeSegCount-class, 55
rangeSegCount-methods
  (bamReader-class), 20
rangeToFastq, 57
rangeToFastq, bamRange, character-method
  (rangeToFastq), 57
rangeToFastq-methods (rangeToFastq), 57
rbamtools (rbamtools-package), 3
rbamtools-package, 3
read2fastq (readerToFastq), 58
reader2fastq, bamReader-method
  (readerToFastq), 58
reader2fastq-methods (readerToFastq), 58
readerToFastq, 58
readerToFastq, bamReader-method
  (readerToFastq), 58
readerToFastq-methods (readerToFastq), 58
readPooledBamGapDF, 59
readPooledBamGaps, 59
readRange (bamRange-class), 16
readRange, bamReader, character, numeric, logical-method
  (bamRange-class), 16
readRange, bamReader, character, numeric, missing-method
  (bamRange-class), 16
readRange, bamReader, factor, numeric, logical-method
  (bamRange-class), 16
readRange, bamReader, factor, numeric, missing-method
  (bamRange-class), 16
readRange-methods (bamRange-class), 16
refID (bamAlign-class), 7
refID, bamAlign-method (bamAlign-class), 7
refID, gapSiteList-method
  (gapSiteList-class), 38
refID-methods (bamAlign-class), 7
refID<- (gapSiteList-class), 38
refID<-, gapSiteList, numeric-method
  (gapSiteList-class), 38
refID--methods (gapSiteList-class), 38
refSeqDict (bamHeaderText-class), 13
refSeqDict, bamHeaderText-method
  (bamHeaderText-class), 13
refSeqDict-class, 60
refSeqDict<- (bamHeaderText-class), 13
refSeqDict<-, bamHeaderText-method
  (bamHeaderText-class), 13
removeSeqs (refSeqDict-class), 60
removeSeqs, refSeqDict-method
  (refSeqDict-class), 60
reverseStrand (bamAlign-class), 7
reverseStrand, bamAlign-method
  (bamAlign-class), 7
reverseStrand-methods (bamAlign-class), 7
reverseStrand<-(bamAlign-class), 7
reverseStrand<-(bamAlign-method
(bamAlign-class), 7
reverseStrand<--methods
(bamAlign-class), 7
rewind (bamReader-class), 20
rewind, bamRange-method
(bamRange-class), 16
rewind, bamReader-method
(bamReader-class), 20
rewind-methods (bamReader-class), 20
sampleBamFiles (sampleBamFiles-class), 61
sampleBamFiles, character-method
(sampleBamFiles-class), 61
sampleBamFiles, factor-method
(sampleBamFiles-class), 61
sampleBamFiles, integer-method
(sampleBamFiles-class), 61
sampleBamFiles, numeric-method
(sampleBamFiles-class), 61
sampleBamFiles-class, 61
sampleBamFiles-methods
(sampleBamFiles-class), 61
sampleGroups (sampleBamFiles-class), 61
sampleGroups, sampleBamFiles-method
(sampleBamFiles-class), 61
sampleGroups-methods
(sampleBamFiles-class), 61
sampleGroups<-(sampleBamFiles-class), 61
sampleGroups<-, sampleBamFiles, character-method
(sampleBamFiles-class), 61
sampleGroups<-, sampleBamFiles, factor-method
(sampleBamFiles-class), 61
sampleGroups<--methods
(sampleBamFiles-class), 61
sampleLabels (sampleBamFiles-class), 61
sampleLabels, sampleBamFiles-method
(sampleBamFiles-class), 61
sampleLabels-methods
(sampleBamFiles-class), 61
sampleLabels<-(sampleBamFiles-class), 61
sampleLabels<-, sampleBamFiles, integer-method
(sampleBamFiles-class), 61
sampleLabels<-, methods
(sampleBamFiles-class), 61
saveAldData, 63
saveAldData, sampleBamFiles, geneList-method
(saveAldData), 63
saveAldData-methods (saveAldData), 63
secondaryAlign (bamAlign-class), 7
secondaryAlign, bamAlign-method
(bamAlign-class), 7
secondaryAlign-methods
(bamAlign-class), 7
secondaryAlign<-(bamAlign-class), 7
secondaryAlign<-, bamAlign-method
(bamAlign-class), 7
secondaryAlign<--methods
(bamAlign-class), 7
secondaryAlignPair (bamAlign-class), 7
secondaryAlignPair, bamAlign-method
(bamAlign-class), 7
secondaryAlignPair-methods
(bamAlign-class), 7
secondaryAlignPair<-(bamAlign-class), 7
secondaryAlignPair<-, bamAlign-method
(bamAlign-class), 7
secondaryAlignPair<--methods
(bamAlign-class), 7
segmentize, 64
segmentize, ANY-method (segmentize), 64
segmentize, data.frame-method
(segmentize), 64
segmentize, matrix-method (segmentize), 64
segmentize-methods (segmentize), 64
setVal, 65
setVal, headerLine-method
(headerLine-class), 50
setVal, headerProgram-method
(headerProgram-class), 51
setVal, headerReadGroup-method
(headerReadGroup-class), 52
show, alignDepth-method
(alignDepth-class), 4
show, bamGapList-method
(bamGapList-class), 11
show, bamHeader-method
(bamHeader-class), 12
show, exonAlignDepth-method