Package ‘rbamtools’

February 20, 2015

Type    Package
Title   An R package for reading and writing BAM (Binary alignment) files.
Version 2.10.0
Date    2013-14-02
Author  Wolfgang Kaisers
Maintainer Wolfgang Kaisers <kaisers@med.uni-duesseldorf.de>
Description This package provides an interface to functions of the 'SAMtools' C-Library by Heng Li.
License Artistic-2.0
LazyLoad yes
Depends methods
Suggests xtable
Repository CRAN
Date/Publication 2014-09-30 16:42:11
NeedsCompilation yes

R topics documented:

rbamtools-package ......................................................... 2
alignDepth-class ........................................................ 4
as.data.frame-methods ............................................... 5
bamAlign ................................................................. 6
bamAlign-class .......................................................... 7
bamClose-methods ...................................................... 9
bamCount ................................................................. 10
bamCountAll ............................................................ 10
bamGapList-class ....................................................... 11
bamHeader-class ....................................................... 12
bamHeaderText-class .................................................. 13
bamRange ................................................................. 14
 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 alings 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)
```
alignDepth-class

`align<~getNextAlign(range)
#
`bamClose(reader)

### 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, ylim): 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])
ranges <- bamRange(reader, coords)
bamClose(reader)

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

# Prepare plotting parameter
gene <- "WASH7P"
ensg_id <- "ENSG00000227232"
enst_id <- "ENST00000538476"

# Get exon positions
start <- c(14411, 15000, 15796, 15904, 16007, 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,
               end = end, strand = "-",
               transcript = paste("Chromosome 1",
                                   "\tGene", ensg_id,
                                   "\tTranscript ", enst_id))
```

---

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

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.
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
bamAlign-class

Examples

# Create alignment object from scratch
align<-bamalign("HMUSI-0001", "ATGTACGTGC", "Qual/String", "4M0N6M",
               refid=0, position=100)

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

bamAlign-class 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

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.
bamClose-methods

Author(s)

Wolfgang Kaisers

References

The SAM Format Specification (16ede77).
github.io/hts-specs/SAMv1.pdf

Examples

# 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
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

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)

bamCountAll

bamCountAll

Description

The bamCountAll function takes a bamReader object and returns a data.frame. The data.frame contains the counts for all contained reference sequences.

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

**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 (<= 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.
\textbf{nAligns} signature(object = "bamGapList"): Returns number of aligns in specified Range.

\textbf{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)
```

---

**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<-\texttt{new("bamHeader")}.

**Slots**

header: Object of class "externalptr". Points to samtools bam\_header\_t struct.

**Methods**

\texttt{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).
sourceforge.net/SAM1.pdf

Examples

```
bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
  # Open bam file
  reader<-bamReader(bam)
  # Extract binary header structure
  header<-getHeader(reader)
  header
  # Extract textual representation
  htxt<-getHeaderText(header)
  # Extract header line segment
  hl<-headerLine(htxt)
  # Get header program
  hp<-headerProgram(htxt)
  hp
  # 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 `hl<-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

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

bamRange

bamRange(bamReader, 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(reader=NULL, coords=NULL, complex=FALSE)
**Arguments**

- **reader**: 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 ist 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)

# Eventually:
## Not run: create.index(reader,idx)
# Load BAM index file
loadIndex(reader,idx)
indexInitialized(reader)  # Should return 'TRUE'
# +++++++++++++++++++++++++++++++++++++++++++++++++++++
# 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
```
bamRange-class

Class "bamRange": Representation of genomic alignments in defined regions.

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.
# coords: refid=0, start=0, stop=249250621
### + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + #
cords<-as.integer(c(0,0,249250621))
range<-bamRange(reader,coords)
size(range)
range<-bamRange(reader,coords,complex=TRUE)
dfr<-as.data.frame(range)
size(range)
align<-getNextAlign(range)
cigarData(align)
rewind(range)

# Not run:
while(!is.null(align))
{
    print(position(align))
    align<-getNextAlign(range)
}

# End(Not run)
```
**bamReader**

bamClose(reader)

```r
## + + + + + + + + + + + + + + + + + + + + + + + + + ##
## C) Get print message and some other values
## + + + + + + + + + + + + + + + + + + + + + + + + + ##
range
getCoords(range)
getSeqLen(range)
getParams(range)
getRefName(range)
getAlignRange(range)
```

```r
## + + + + + + + + + + + + + + + + + + + + + + + + + ##
## D) Rudimentary subsetting
## + + + + + + + + + + + + + + + + + + + + + + + + + ##
range[1:5]
```

---

**bamReader**

bamReader: Connection to BAM file for reading access.

---

**Description**

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**

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

**Arguments**

- `filename`: Filename of BAM-file to be opened for reading.
- `indexname`: Optional: Name of BAM-index file
- `idx`: Logical
- `verbose`: Numeric: Quantifies the extent of textual feedback (levels: 0,1,2).

**Author(s)**

Wolfgang Kaisers

**Examples**

```r
bam<-system.file("extdata","accepted_hits.bam",package="r bamtools")
reader<-bamReader(bam)
#
align<-getNextAlign(reader)
name(align)
```
bamReader-class

Class "bamReader": Representation for file connection to BAM file for reading access.

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.

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


Examples

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)
bamsort

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))
rangen<-bamRange(reader,coords)
align<-getNextAlign(range)
#
# 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)

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

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

Arguments

object             bamReader. The reader must be opened.
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

bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
reader<-bamReader(bam)
bamSort(reader)

---

**bamWriter**

*bamWriter: Opening a file connection to a BAM file for writing access.*

**Description**

The bamWriter function takes a bamHeader object and a filename and returns an object of class 'bamWriter' which represents a writing connection to a BAM-file.
Usage

bamWriter(x, filename)

Arguments

x  
An instance of class bamHeader.

filename  
Filename of BAM-file to be opened for writing.

Author(s)

Wolfgang Kaisers

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))
rangen <- 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
pro <- new("headerProgram")
setVal(pro,"ID","1")
setVal(pro,"PN","tophat")
setVal(pro,"CL","tophat -p8 --library-type fr-unstranded hs_ucsc rna033.fastq")
setVal(pro,"VN","2.0.0")
bh <- bamHeaderText(head=headl,dict=dict,prog=pro)
## Not run: getHeaderText(bh)
header <- bamHeader(bh)

# +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# Copy aligns in range into new BAM file
## Not run:
writer <- bamWriter(header,"chr1_complex.bam")
```
bamWriter-class

```r
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".
- `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

# +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# 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=13534747)
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(head=headl,dict=dict,prog=prog)
#getHeaderText(bh)
header<-bamHeader(bh)

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

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

Arguments

object bamAlign or bamRange object.

Value

Integer (of length 4).

Author(s)

Wolfgang Kaisers

Examples

# A) For bamAlign
align<-bamAlign("HWUSI-0001","ACCGGTTTT","Qual/String","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)
ranges<-bamRange(reader, coords)
countNucs(ranges)

countTextLines

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.
createIdxBatch

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

```r
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)
```
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 <- 901099
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")
```

**filename-methods**

*Reading filename from bamReader or bamWriter*

**Description**

Reading filename from bamReader or bamWriter

**Methods**

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

**gapList-class**

*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=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 gapList 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 (<= 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 = "gapList"): Returns number of site-items in list.
coerce signature(from = "gapList", to = "data.frame"): Coercion of gapSiteList to data.frame.
coerce signature("gapList","data.frame"): Coercion of gapSiteList to data.frame.
show signature(object = "gapList"): Prints a short message with some summarizing data.

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)
bamClose(reader)

getHeaderText-methods  getHeaderText 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")
**getHeaderText-methods**

**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")
dbx<-paste(bam,"bai",sep=".")
# Open BAM file
reader<-bamReader(bam)
header<-getHeader(reader)
htxt<-getHeaderText(header)
getHeaderText(htxt)
bamClose(reader)
```
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)
indexInitialized(reader)
#
coords <- as.integer(c(0,0,249250621))
ranger <- bamRange(reader,coords)
align <- getNextAlign(range)
position(align)
#
bamClose(reader)
```
### 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))
rangex <- bamRange(reader, coords)
bamClose(reader)
# B) getQualDf
qdf <- getQualDf(range)
qdf[32:38,1:15]
```
getRefData

Retrieve reference sequence from a BAM file as data.frame

Description

The three functions: getRefCoords, getRefCount and getRefData provide reading access to data about the present reference sequences in the bamHeader.

Usage

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)
coords<-getRefCoords(reader,"chr1")
rg<-bamRange(reader,coords)
bamClose(reader)
**getVal**

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

**Description**

Retrieving value of data-member from object

**Usage**

`getVal(object, member)`

**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:

- ID  Program record identifier
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

bam<system.file("extdata","accepted_hits.bam",package="rbamtools")
reader<bamReader(bam)
header<getHeader(reader)
htxt<getHeaderText(header)
hl<headerLine(htxt)
getVal(hl,"SO")
bamClose(reader)

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 signature(x = "bamRange"): Retrieve textual representation of header
getVal signature(object = "bamRange"): Retrieving values for given item names.
setVal signature(object = "bamRange"): Setting values for given item names.
as.list signature(.Object = "bamRange"): Coercing of data into a list
headerProgram-class

Author(s)

Wolfgang Kaisers

Examples

bam<system.file("extdata","accepted_hits.bam",package="rbamtools")
reader<bamReader(bam)
header<getHeader(reader)
htxt<getHeaderText(header)
headline<getHeadLine(htxt)
getVal(headline,"VN")
getVal(headline,"SO")
setVal(headline,"SO","unsorted")
l<as.list(headline)
txt<getHeaderText(headline)
bamClose(reader)

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

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 h1<-new("headerReadGroup").

Slots

nrg: integer. Number of read groups.
nrl: 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.
FO: character. Flow order.
KS: character. Array of nucleotide bases.
LB: character. Library.
PG: character. Programs used for processing.
P1: character. Predicted median insert size.
PU: character. Unique platform identifier.
SM: character. Sample name.
isOpen-methods

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

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)
getVal(readGroup,"ID")
setVal(readGroup,"ID","newID")
l<-as.list(readGroup)
txt<-getHeaderText(readGroup)
bamClose(reader)
```

```
isOpen-methods  isOpen(bamReader), isOpen(bamWriter): Checking for opened file connection to BAM file.
```

Description

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

Usage

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

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.

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,...)

Arguments

object  bamReader object. The reader must be opened and have initialized index.

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(0,0,14000))
nucStats(range)

range2fastq: Extract read information from alignments for given genomic range as fastq.

Description

The function `range2fastq` writes all (or selected) aligns from a `bamRange` into a compressed `fastq` file.

Usage

```r
range2fastq(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.

Author(s)

Wolfgang Kaisers

Examples

```r
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: range2fastq(range,"range fq.gz")
index <- sample(1:size(range),100)
## Not run: range2fastq(range,"range subset fq.gz",which=index)
```
reader2fastq  
reader2fastq: Extraction of a subset of alignments from a BAM files into fastq format.

Description

The function `reader2fastq` writes (remaining) aligns or some part of the stored aligns in a BAM file as reads into a compressed fastq file.

Usage

```r
reader2fastq(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.

Value

numeric

Author(s)

Wolfgang Kaisers

Examples

```r
bam<-system.file("extdata","accepted_hits.bam",package="rbamtools")
reader<-bamReader(bam)
## Not run: reader2fastq(reader,"out.fq.gz")
bamClose(reader)
# Reopen in order to point to first align
reader<-bamReader(bam)
index<-sample(1:100,20)
## Not run: reader2fastq(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=""))
refSeqDict-class

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")
rpbc<-readPooledBamGaps(bam)
rpdf<-readPooledBamGapDf(bam)
hist(rpdf$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 hl<-new("refSeqDict").

Slots

SN: character
LN: numeric
AS: character
MS: numeric
SP: character
UR: character
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)
```

**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:

- 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:

- ID Program record identifier
- PN Program name
- CL Command line
- PP Previous @PG-ID (Must match another ID in list)
- VN Program Version

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,"50","coordinate")
bamClose(reader)
```
## Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>Class/Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT/CG ratio</td>
<td>nucStats, 42</td>
</tr>
<tr>
<td>GC content</td>
<td>nucStats, 42</td>
</tr>
<tr>
<td>bamAlign</td>
<td>alignDepth-class, 4</td>
</tr>
<tr>
<td>bamAlign</td>
<td>bamAlign-class, 7</td>
</tr>
<tr>
<td>bamAlign</td>
<td>bamWriter, 23</td>
</tr>
<tr>
<td>bamAlign</td>
<td>countNucs, 26</td>
</tr>
<tr>
<td>bamAlign</td>
<td>gapList-class, 30</td>
</tr>
<tr>
<td>bamAlign</td>
<td>gapSiteList-class, 31</td>
</tr>
<tr>
<td>bamClose</td>
<td>bamClose-methods, 9</td>
</tr>
<tr>
<td>bamCountAll</td>
<td>bamCount, 10</td>
</tr>
<tr>
<td>bamCount</td>
<td>bamCount, 10</td>
</tr>
<tr>
<td>bamCount</td>
<td>bamCountAll, 10</td>
</tr>
<tr>
<td>bamGapList</td>
<td>bamGapList-class, 11</td>
</tr>
<tr>
<td>bamHeader</td>
<td>bamHeader-class, 12</td>
</tr>
<tr>
<td>bamHeader</td>
<td>bamHeaderText-class, 13</td>
</tr>
<tr>
<td>bamHeader</td>
<td>headerLine-class, 38</td>
</tr>
<tr>
<td>bamRange</td>
<td>as.data.frame-methods, 5</td>
</tr>
<tr>
<td>bamRange</td>
<td>bamRange-class, 14</td>
</tr>
<tr>
<td>bamRange</td>
<td>countNucs, 26</td>
</tr>
<tr>
<td>bamRange</td>
<td>getNextAlign-methods, 34</td>
</tr>
<tr>
<td>bamRange</td>
<td>getQualDf, 35</td>
</tr>
<tr>
<td>bamReader</td>
<td>bamHeader-class, 12</td>
</tr>
<tr>
<td>bamReader</td>
<td>bamHeaderText-class, 13</td>
</tr>
<tr>
<td>bamReader</td>
<td>bamReader, 19</td>
</tr>
<tr>
<td>bamReader</td>
<td>bamReader-class, 20</td>
</tr>
<tr>
<td>bamSort</td>
<td>bamSort, 22</td>
</tr>
<tr>
<td>bamSort</td>
<td>filename-methods, 30</td>
</tr>
<tr>
<td>bamSort</td>
<td>getNextAlign-methods, 34</td>
</tr>
<tr>
<td>bamSort</td>
<td>headerLine-class, 38</td>
</tr>
<tr>
<td>bamSort</td>
<td>headerProgram-class, 39</td>
</tr>
<tr>
<td>bamSort</td>
<td>headerReadGroup-class, 40</td>
</tr>
<tr>
<td>bamSort</td>
<td>nucStats, 42</td>
</tr>
<tr>
<td>bamSort</td>
<td>refSeqDict-class, 46</td>
</tr>
<tr>
<td>bamWriter</td>
<td>bamWriter, 23</td>
</tr>
<tr>
<td>bamWriter</td>
<td>filename-methods, 30</td>
</tr>
<tr>
<td>bamWriter</td>
<td>bamWriter-class, 20</td>
</tr>
<tr>
<td>bamWriter</td>
<td>bamWriter-class, 25</td>
</tr>
<tr>
<td>classes</td>
<td>bamHeader-class, 12</td>
</tr>
<tr>
<td>classes</td>
<td>bamHeaderText-class, 13</td>
</tr>
<tr>
<td>classes</td>
<td>bamRange-class, 16</td>
</tr>
<tr>
<td>classes</td>
<td>bamReader-class, 20</td>
</tr>
<tr>
<td>classes</td>
<td>bamReader-class, 25</td>
</tr>
<tr>
<td>classes</td>
<td>headerLine-class, 38</td>
</tr>
<tr>
<td>classes</td>
<td>headerProgram-class, 39</td>
</tr>
<tr>
<td>classes</td>
<td>headerReadGroup-class, 40</td>
</tr>
<tr>
<td>classes</td>
<td>refSeqDict-class, 46</td>
</tr>
<tr>
<td>countNucs</td>
<td>countNucs, 26</td>
</tr>
<tr>
<td>countTextLines</td>
<td>countTextLines, 27</td>
</tr>
<tr>
<td>createIdxBatch</td>
<td>createIdxBatch, 28</td>
</tr>
<tr>
<td>extractRanges</td>
<td>extractRanges, 29</td>
</tr>
<tr>
<td>filename</td>
<td>filename-methods, 30</td>
</tr>
<tr>
<td>gapList</td>
<td>as.data.frame-methods, 5</td>
</tr>
<tr>
<td>gapList</td>
<td>getNextAlign-methods, 34</td>
</tr>
<tr>
<td>getHeaderText</td>
<td>getHeaderText-methods, 32</td>
</tr>
<tr>
<td>getNextAlign</td>
<td>getNextAlign-methods, 34</td>
</tr>
<tr>
<td>Topic</td>
<td>Function</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>getQualDf</td>
<td>getQualDf</td>
</tr>
<tr>
<td>getRefData</td>
<td>bamCountAll</td>
</tr>
<tr>
<td>getRefData</td>
<td>getRefData</td>
</tr>
<tr>
<td>getVal</td>
<td>getVal</td>
</tr>
<tr>
<td>headerLine</td>
<td>headerLineMclass</td>
</tr>
<tr>
<td>headerProgram</td>
<td>headerProgramMclass</td>
</tr>
<tr>
<td>headerReadGroup</td>
<td>headerReadGroupMclass</td>
</tr>
<tr>
<td>index</td>
<td>bamCount</td>
</tr>
<tr>
<td>index</td>
<td>bamRange</td>
</tr>
<tr>
<td>index</td>
<td>countTextLines</td>
</tr>
<tr>
<td>isOpen</td>
<td>isOpenMmethods</td>
</tr>
<tr>
<td>methods</td>
<td>as.data.frame-methods</td>
</tr>
<tr>
<td>methods</td>
<td>bamClose-methods</td>
</tr>
<tr>
<td>methods</td>
<td>filename-methods</td>
</tr>
<tr>
<td>methods</td>
<td>getNextAlign-methods</td>
</tr>
<tr>
<td>methods</td>
<td>isopenMmethods</td>
</tr>
<tr>
<td>nucStats</td>
<td>nucStats</td>
</tr>
<tr>
<td>package</td>
<td>rbamtools-package</td>
</tr>
<tr>
<td>range2fastq</td>
<td>range2fastq</td>
</tr>
<tr>
<td>readPooledBamGapDf</td>
<td>readPooledBamGapDf</td>
</tr>
<tr>
<td>readPooledBamGaps</td>
<td>readPooledBamGaps</td>
</tr>
<tr>
<td>reader2fastq</td>
<td>reader2fastq</td>
</tr>
<tr>
<td>refSeqDict</td>
<td>refSeqDict-class</td>
</tr>
<tr>
<td>setVal</td>
<td>setVal</td>
</tr>
</tbody>
</table>

*Methods:*

- `<=,refSeqDict-method`: [refSeqDict-class], 46
- `addReadGroup`: [headerReadGroup-class], 40
- `addReadGroup`: [headerReadGroup-method] (headerReadGroup-class), 40
- `addReadGroup-methods`: (headerReadGroup-class), 40
- `addSeq`: [refSeqDict-class], 46
- `addSeq`: [refSeqDict-method] (refSeqDict-class), 46
- `alignDepth`: [alignDepth-class], 4
- `alignDepth`: [alignDepth-class], 4
- `alignDepth-methods`: [alignDepth-class], 4
- `alignQual`: [bamAlign-class], 7
- `alignQual`: [bamAlign-class], 7
- `alignQual`: [bamAlign-class], 7
- `alignQual`: [bamAlign-class], 7
- `alignQual`: [bamAlign-class], 7
- `alignQual`: [bamAlign-class], 7
- `alignSeq`: [bamAlign-class], 7
- `alignSeq`: [bamAlign-class], 7
- `alignSeq`: [bamAlign-class], 7
- `alignSeq`: [bamAlign-class], 7
- `as.character`: [bamHeader-class], 12
- `as.character`: [bamHeader-method] (bamHeader-class), 12
- `as.data.frame`: [refSeqDict-method] (refSeqDict-class), 46
- `as.data.frame`: [refSeqDict-method] (refSeqDict-class), 46
- `as.data.frame`: [bamGapList-class], 11
- `as.data.frame`: [bamRange-class], 16
- `as.data.frame`: [gapList-class], 30
- `as.data.frame`: [gapSiteList-class], 31
- `as.data.frame`: [refSeqDict] (refSeqDict-class), 46
- `as.list`: [headerLine-method] (headerLine-class), 38
- `as.list`: [headerProgram-method] (headerProgram-class), 39

*Methods:*

- `[,bamRange-method`: [bamRange-class], 16
- `[,refSeqDict-method`: [refSeqDict-class], 46
- `[-methods`: [bamRange-class], 16
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>as.list</td>
<td></td>
</tr>
<tr>
<td>headerReadGroup-method</td>
<td>(headerReadGroup-class), 40</td>
</tr>
<tr>
<td>bamAlign</td>
<td>6</td>
</tr>
<tr>
<td>bamAlign-class</td>
<td>7</td>
</tr>
<tr>
<td>bamClose (bamClose-methods)</td>
<td>9</td>
</tr>
<tr>
<td>bamClose, bamReader-method</td>
<td>(bamReader-class), 20</td>
</tr>
<tr>
<td>bamClose, bamWriter-method</td>
<td>(bamWriter-class), 25</td>
</tr>
<tr>
<td>bamClose-methods</td>
<td>9</td>
</tr>
<tr>
<td>bamCopy (bamReader-class)</td>
<td>20</td>
</tr>
<tr>
<td>bamCopy, bamReader-method</td>
<td>(bamReader-class), 20</td>
</tr>
<tr>
<td>bamCopy-methods (bamReader-class)</td>
<td>20</td>
</tr>
<tr>
<td>bamCount</td>
<td>10</td>
</tr>
<tr>
<td>bamCount, bamReader-method (bamCount)</td>
<td>10</td>
</tr>
<tr>
<td>bamCount-methods (bamCount)</td>
<td>10</td>
</tr>
<tr>
<td>bamCountAll</td>
<td>10</td>
</tr>
<tr>
<td>bamCountAll, bamReader-method</td>
<td>(bamCountAll), 10</td>
</tr>
<tr>
<td>bamCountAll-methods (bamCountAll)</td>
<td>10</td>
</tr>
<tr>
<td>bamGapList</td>
<td>(bamGapList-class), 11</td>
</tr>
<tr>
<td>bamGapList, bamReader-method</td>
<td>(bamGapList-class), 11</td>
</tr>
<tr>
<td>bamGapList-class</td>
<td>11</td>
</tr>
<tr>
<td>bamGapList-methods (bamGapList-class)</td>
<td>11</td>
</tr>
<tr>
<td>bamHeader</td>
<td>(bamHeader-class), 12</td>
</tr>
<tr>
<td>bamHeader, bamHeaderText-method</td>
<td>(bamHeaderText-class), 13</td>
</tr>
<tr>
<td>bamHeader-class</td>
<td>12</td>
</tr>
<tr>
<td>bamHeaderText</td>
<td>(bamHeaderText-class), 13</td>
</tr>
<tr>
<td>bamHeaderText-class</td>
<td>13</td>
</tr>
<tr>
<td>bamRange</td>
<td>14</td>
</tr>
<tr>
<td>bamRange-class</td>
<td>16</td>
</tr>
<tr>
<td>bamReader</td>
<td>19</td>
</tr>
<tr>
<td>bamReader-class</td>
<td>20</td>
</tr>
<tr>
<td>bamSave (bamReader-class)</td>
<td>25</td>
</tr>
<tr>
<td>bamSave, bamReader-method</td>
<td>(bamReader-class), 20</td>
</tr>
<tr>
<td>bamSave, bamWriter-method</td>
<td>(bamWriter-class), 25</td>
</tr>
<tr>
<td>bamSave-methods (bamWriter-class)</td>
<td>25</td>
</tr>
<tr>
<td>bamSort</td>
<td>22</td>
</tr>
<tr>
<td>bamSort, bamReader-method</td>
<td>(bamReader-class), 20</td>
</tr>
<tr>
<td>bamWriter</td>
<td>23</td>
</tr>
<tr>
<td>bamWriter, bamHeader-method</td>
<td>(bamHeader-class), 12</td>
</tr>
<tr>
<td>bamWriter-class</td>
<td>25</td>
</tr>
<tr>
<td>cigarData (bamAlign-class)</td>
<td>7</td>
</tr>
<tr>
<td>cigarData, bamAlign-method</td>
<td>(bamAlign-class), 7</td>
</tr>
<tr>
<td>cigarData-methods (bamAlign-class)</td>
<td>7</td>
</tr>
<tr>
<td>coerce, bamGapList, data.frame-method</td>
<td>(bamGapList-class), 11</td>
</tr>
<tr>
<td>coerce, bamRange, data.frame-method</td>
<td>(bamRange-class), 16</td>
</tr>
<tr>
<td>coerce, gapList, data.frame-method</td>
<td>(gapList-class), 30</td>
</tr>
<tr>
<td>coerce, gapSiteList, data.frame-method</td>
<td>(gapSiteList-class), 31</td>
</tr>
<tr>
<td>coerce, refSeqDict, data.frame-method</td>
<td>(refSeqDict-class), 46</td>
</tr>
<tr>
<td>countNucs</td>
<td>26</td>
</tr>
<tr>
<td>countNucs, bamAlign-method (countNucs), 26</td>
<td></td>
</tr>
<tr>
<td>countNucs, bamRange-method (countNucs), 26</td>
<td></td>
</tr>
<tr>
<td>countNucs-methods (countNucs)</td>
<td>26</td>
</tr>
<tr>
<td>countTextLines</td>
<td>27</td>
</tr>
<tr>
<td>create.idx.batch (createIdxBatch), 28</td>
<td></td>
</tr>
<tr>
<td>create.index (bamReader-class), 20</td>
<td></td>
</tr>
<tr>
<td>create.index, bamReader-method</td>
<td>(bamReader-class), 20</td>
</tr>
<tr>
<td>create.index-methods (bamReader-class), 20</td>
<td></td>
</tr>
<tr>
<td>createIdxBatch, 28</td>
<td></td>
</tr>
<tr>
<td>createIndex, bamReader-method</td>
<td>(bamReader-class), 20</td>
</tr>
<tr>
<td>createIndex-methods (bamReader-class), 20</td>
<td></td>
</tr>
<tr>
<td>dim, refSeqDict-method</td>
<td>(refSeqDict-class), 46</td>
</tr>
<tr>
<td>extractRanges</td>
<td>29</td>
</tr>
<tr>
<td>extractRanges, bamReader-method</td>
<td>(extractRanges), 29</td>
</tr>
<tr>
<td>extractRanges-methods (extractRanges), 29</td>
<td></td>
</tr>
<tr>
<td>failedQC (bamAlign-class), 7</td>
<td></td>
</tr>
<tr>
<td>failedQC, bamAlign-method</td>
<td>(bamAlign-class), 7</td>
</tr>
</tbody>
</table>
getRefCoords (getRefData), 36
getRefCoords, bamReader-method
  (getRefData), 36
getRefCoords-methods (getRefData), 36
getRefCount (getRefData), 36
getRefCount, bamReader-method
  (getRefData), 36
getRefCount-methods (getRefData), 36
getRefData, 36
getRefData, bamReader-method
  (getRefData), 36
getRefData-methods (getRefData), 36
getRefName (bamRange-class), 16
getRefName, bamRange-method
  (bamRange-class), 16
getRefName-methods (bamRange-class), 16
getSeqlen (bamRange-class), 16
getSeqlen, bamRange-method
  (bamRange-class), 16
getSeqlen-methods (bamRange-class), 16
getVal, 37
getVal, headerLine-method
  (headerLine-class), 38
getVal, headerProgram-method
  (headerProgram-class), 39
getVal, headerReadGroup-method
  (headerReadGroup-class), 40

head (refSeqDict-class), 46
head, refSeqDict-method
  (refSeqDict-class), 46
headerLine (bamHeaderText-class), 13
headerLine, bamHeaderText-method
  (bamHeaderText-class), 13
headerLine-class, 38
headerLine<- (bamHeaderText-class), 13
headerLine<-, bamHeaderText-method
  (bamHeaderText-class), 13
headerProgram (bamHeaderText-class), 13
headerProgram, bamHeaderText-method
  (bamHeaderText-class), 13
headerProgram-class, 39
headerProgram<- (bamHeaderText-class), 13
headerProgram<-, bamHeaderText-method
  (bamHeaderText-class), 13
headerReadGroup (bamHeaderText-class), 13
headerReadGroup-methods (bamHeaderText-class), 13
insertPreCurrent, bamRange-method
(bamRange-class), 16
insertPreCurrent-methods
(bamRange-class), 16
insertSize (bamAlign-class), 7
insertSize, bamAlign-method
(bamAlign-class), 7
insertSize-methods (bamAlign-class), 7
isOpen (isOpen-methods), 41
isOpen, bamReader-method
(bamReader-class), 20
isOpen, bamWriter-method
(bamWriter-class), 25
isOpen-methods, 41
load.index (bamReader-class), 20
load.index, bamReader-method
(bamReader-class), 20
load.index-methods (bamReader-class), 20
loadIndex (bamReader-class), 20
loadIndex, bamReader-method
(bamReader-class), 20
loadIndex-methods (bamReader-class), 20
mapQuality (bamAlign-class), 7
mapQuality, bamAlign-method
(bamAlign-class), 7
mapQuality-methods (bamAlign-class), 7
matePosition (bamAlign-class), 7
matePosition, bamAlign-method
(bamAlign-class), 7
matePosition-methods (bamAlign-class), 7
mateRefID (bamAlign-class), 7
mateRefID, bamAlign-method
(bamAlign-class), 7
mateRefID-methods (bamAlign-class), 7
mateReverseStrand (bamAlign-class), 7
mateReverseStrand, bamAlign-method
(bamAlign-class), 7
mateReverseStrand-methods
(bamAlign-class), 7
mateReverseStrand< (bamAlign-class), 7
mateReverseStrand<, bamAlign-method
(bamAlign-class), 7
mateReverseStrand<--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
merge, bamGapList-method (bamGapList-class), 11
merge, gapSiteList-method (gapSiteList-class), 31
moveCurrentAlign (bamRange-class), 16
moveCurrentAlign, bamRange-method
(bamRange-class), 16
moveCurrentAlign-methods
(bamRange-class), 16
nAlignGaps (gapList-class), 30
nAlignGaps, bamGapList-method
(bamGapList-class), 11
nAlignGaps, gapList-method
(gapList-class), 30
nAlignGaps, gapSiteList-method
(gapSiteList-class), 31
nAlignGaps-methods (gapList-class), 30
nAligns (gapList-class), 30
nAligns, bamGapList-method
(bamGapList-class), 11
nAligns, gapList-method (gapList-class), 30
nAligns, gapSiteList-method
(gapSiteList-class), 31
nAligns-methods (gapList-class), 30
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
nucStats, 42
nucStats, bamRange-method (nucStats), 42
nucStats, bamReader-method (nucStats), 42
nucStats, character-method (nucStats), 42
nucStats-methods (nucStats), 42
paired (bamAlign-class), 7
paired, bamAlign-method
(bamAlign-class), 7
paired-methods (bamAlign-class), 7
paired<= (bamAlign-class), 7
mateUnmapped-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
insertPreCurrent-lbamrangeM-method
(lbamrangeM-class), 16
insertPreCurrent-methods
(lbamrangeM-class), 16
insertSize (bamAlign-class), 7
insertSize, bamAlign-method
(bamAlign-class), 7
insertSize-methods (bamAlign-class), 7
isOpen (isOpen-methods), 41
isOpen, bamReader-method
(bamReader-class), 20
isOpen, bamWriter-method
(bamWriter-class), 25
isOpen-methods, 41
load.index (bamReader-class), 20
load.index, bamReader-method
(bamReader-class), 20
load.index-methods (bamReader-class), 20
loadIndex (bamReader-class), 20
loadIndex, bamReader-method
(bamReader-class), 20
loadIndex-methods (bamReader-class), 20
mapQuality (bamAlign-class), 7
mapQuality, bamAlign-method
(bamAlign-class), 7
mapQuality-methods (bamAlign-class), 7
matePosition (bamAlign-class), 7
matePosition, bamAlign-method
(bamAlign-class), 7
matePosition-methods (bamAlign-class), 7
mateRefID (bamAlign-class), 7
mateRefID, bamAlign-method
(bamAlign-class), 7
mateRefID-methods (bamAlign-class), 7
mateReverseStrand (bamAlign-class), 7
mateReverseStrand, bamAlign-method
(bamAlign-class), 7
mateReverseStrand-methods
(bamAlign-class), 7
mateReverseStrand< (bamAlign-class), 7
mateReverseStrand<, bamAlign-method
(bamAlign-class), 7
mateReverseStrand<--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
merge, bamGapList-method (bamGapList-class), 11
merge, gapSiteList-method (gapSiteList-class), 31
moveCurrentAlign (bamRange-class), 16
moveCurrentAlign, bamRange-method
(bamRange-class), 16
moveCurrentAlign-methods
(bamRange-class), 16
nAlignGaps (gapList-class), 30
nAlignGaps, bamGapList-method
(bamGapList-class), 11
nAlignGaps, gapList-method
(gapList-class), 30
nAlignGaps, gapSiteList-method
(gapSiteList-class), 31
nAlignGaps-methods (gapList-class), 30
nAligns (gapList-class), 30
nAligns, bamGapList-method
(bamGapList-class), 11
nAligns, gapList-method (gapList-class), 30
nAligns, gapSiteList-method
(gapSiteList-class), 31
nAligns-methods (gapList-class), 30
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
nucStats, 42
nucStats, bamRange-method (nucStats), 42
nucStats, bamReader-method (nucStats), 42
nucStats, character-method (nucStats), 42
nucStats-methods (nucStats), 42
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
push_front-methods (bamRange-class), 16
range2fastq, 43
range2fastq, bamRange-method
  (range2fastq), 43
range2fastq-methods (range2fastq), 43
rbamtools (rbamtools-package), 2
rbamtools-package, 2
reader2fastq, 44
reader2fastq-methods (reader2fastq), 44
dataPooledBamGapDf, 45
dataPooledBamGaps, 45
refID (bamAlign-class), 7
refID, bamAlign-method (bamAlign-class), 7
refID, gapSiteList-method
  (gapSiteList-class), 31
refID-methods (bamAlign-class), 7
refSeqDict (bamHeaderText-class), 13
refSeqDict, bamHeaderText-method
  (bamHeaderText-class), 13
refSeqDict-class, 46
refSeqDict<-, (bamHeaderText-class), 13
refSeqDict<-, bamHeaderText-method
  (bamHeaderText-class), 13
removeSeqs (refSeqDict-class), 46
removeSeqs, refSeqDict-method
  (refSeqDict-class), 46
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
rewind (bamReader-class), 20
rewind, bamRange-method
  (bamRange-class), 16
rewind, bamReader-method
  (bamReader-class), 20
rewind-methods (bamReader-class), 20
secondaryAlign (bamAlign-class), 7
INDEX

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
secondInPair (bamAlign-class), 7
secondInPair, bamAlign-method (bamAlign-class), 7
secondInPair-methods (bamAlign-class), 7
secondInPair<- (bamAlign-class), 7
secondInPair<-, bamAlign-method (bamAlign-class), 7
secondInPair<--methods (bamAlign-class), 7
setVal, 47
setVal, headerLine-method (headerLine-class), 38
setVal, headerProgram-method (headerProgram-class), 39
setVal, headerReadGroup-method (headerReadGroup-class), 40
show, alignDepth-method (alignDepth-class), 4
show, bamGapList-method (bamGapList-class), 11
show, bamHeader-method (bamHeader-class), 12
show, gapList-method (gapList-class), 30
show, gapSiteList-method (gapSiteList-class), 31
siteList (gapSiteList-class), 31
siteList, bamReader-method (gapSiteList-class), 31
size (bamRange-class), 16
size, bamGapList-method (bamGapList-class), 11
size, bamGapList-method (bamRange-class), 16
size, gapList-method (gapList-class), 30
size, gapSiteList-method (gapSiteList-class), 31
size-methods (bamRange-class), 16
stepNextAlign (bamRange-class), 16
stepNextAlign-methods (bamRange-class), 16
stepPrevAlign (bamRange-class), 16
stepPrevAlign, bamRange-method (bamRange-class), 16
stepPrevAlign-methods (bamRange-class), 16
summary, bamGapList (bamGapList-class), 11
suppAlign (bamAlign-class), 7
suppAlign, bamAlign-method (bamAlign-class), 7
suppAlign-methods (bamAlign-class), 7
suppAlign<- (bamAlign-class), 7
suppAlign<-, bamAlign-method (bamAlign-class), 7
suppAlign<--methods (bamAlign-class), 7
tail (refSeqDict-class), 46
tail, refSeqDict-method (refSeqDict-class), 46
unmapped (bamAlign-class), 7
unmapped, bamAlign-method (bamAlign-class), 7
unmapped-methods (bamAlign-class), 7
unmapped<- (bamAlign-class), 7
unmapped<-, bamAlign-method (bamAlign-class), 7
unmapped<--methods (bamAlign-class), 7
writeCurrentAlign (bamRange-class), 16
writeCurrentAlign, bamRange-method (bamRange-class), 16
writeCurrentAlign-methods (bamRange-class), 16