Package 'Rsamtools'

October 8, 2014

Type Package

Title Binary alignment (BAM), variant call (BCF), or tabix file import

Version 1.16.1

Author Martin Morgan, Herv\{}'e Pag\{}`es, Valerie Obenchain

Maintainer Bioconductor Package Maintainer <maintainer@bioconductor.org>

Description This package provides an interface to the 'samtools','bcftools', and 'tabix' utilities (see 'LI-CENCE') for manipulating SAM (Sequence Alignment / Map), binary variant call (BCF) and compressed indexed tab-delimited (tabix) files.

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

License Artistic-2.0 | file LICENSE

LazyLoad yes

Depends methods, IRanges (>= 1.21.10), GenomicRanges (>= 1.15.11),XVector (>= 0.3.2), Biostrings (>= 2.31.3), GenomeInfoDb(>= 0.99.17)

Imports utils, BiocGenerics (>= 0.1.3), zlibbioc, bitops

Suggests GenomicAlignments, ShortRead (>= 1.19.10), GenomicFeatures,TxDb.Dmelanogaster.UCSC.dm3.ensGene, KEGG.db,TxDb.Hsapiens.UCSC.hg18.knownGene, RNAseqData.HNRNPC.bam.chr14,BSgenome.Hsapiens.UCSC.hg19, pasillaBamSubset, RUnit, BiocStyle

LinkingTo IRanges, XVector, Biostrings

biocViews DataImport, Sequencing

R topics documented:

Rsamtools-package   *'samtools' aligned sequence utilities interface*

Description

This package provides facilities for parsing samtools BAM (binary) files representing aligned sequences.

Details

See packageDescription(Rsamtools) for package details. A useful starting point is the scanBam manual page.

Note

This package documents the following classes for purely internal reasons, see help pages in other packages: bzfile, fifo, gzfile, pipe, unz, url.

Author(s)

Author: Martin Morgan

Maintainer: Biocore Team c/o BioC user list <bioconductor@stat.math.ethz.ch>

References

The current source code for samtools and bcftools is from <https://github.com/samtools/samtools>. Additional material is at <http://samtools.sourceforge.net/>.

Examples

```
packageDescription(Rsamtools)
```

---

applyPileups	*Create summary pile-up statistics across multiple BAM files.*

Description

applyPileups scans one or more BAM files, returning position-specific sequence and quality summaries.

Usage

```
applyPileups(files, FUN, ..., param)
```

Arguments

files	A [PileupFiles](#) instances.
FUN	A function of 1 argument, x, to be evaluated for each yield (see yieldSize, yieldBy, yieldAll). The argument x is a list, with elements describing the current pile-up. The elements of the list are determined by the argument what, and include:
	seqnames: (Always returned) A named integer() representing the seqnames corresponding to each position reported in the pile-up. This is a run-length encoding, where the names of the elements represent the seqnames, and the values the number of successive positions corresponding to that seqname.
	pos: Always returned) A integer() representing the genomic coordinate of each pile-up position.
	seq: An array of dimensions nucleotide x file x position. The 'nucleotide' dimension is length 5, corresponding to 'A', 'C', 'G', 'T', and 'N' respectively. Entries in the array represent the number of times the nucleotide occurred in reads in the file overlapping the position.
	qual: Like seq, but summarizing quality; the first dimension is the Phred-encoded quality score, ranging from '!' (0) to '~' (93).
...	Additional arguments, passed to methods.
param	An instance of the object returned by PileupParam.

Details

Regardless of `param` values, the algorithm follows samtools by excluding reads flagged as unmapped, secondary, duplicate, or failing quality control.

Value

`applyPileups` returns a `list` equal in length to the number of times FUN has been called, with each element containing the result of FUN.

`PileupParam` returns an object describing the parameters.

Author(s)

Martin Morgan

References

http://samtools.sourceforge.net/

See Also

`PileupParam`.

Examples

```
fl <- system.file("extdata", "ex1.bam", package="Rsamtools",
                  mustWork=TRUE)

fls <- PileupFiles(c(fl, fl))

calcInfo <-
    function(x)
{
    ## information at each pile-up position
    info <- apply(x[["seq"]], 2, function(y) {
        y <- y[c("A", "C", "G", "T"),,drop=FALSE]
        y <- y + 1L                      # continuity
        cvg <- colSums(y)
        p <- y / cvg[col(y)]
        h <- -colSums(p * log(p))
        ifelse(cvg == 4L, NA, h)
    })
    list(seqnames=x[["seqnames"]], pos=x[["pos"]], info=info)
}
which <- GRanges(c("seq1", "seq2"), IRanges(c(1000, 1000), 2000))
param <- PileupParam(which=which, what="seq")
res <- applyPileups(fls, calcInfo, param=param)
str(res)
head(res[[1]][["pos"]]) # positions matching param
head(res[[1]][["info"]]) # inforamtion in each file
```

```
## param as part of files
fls1 <- PileupFiles(c(fl, fl), param=param)
res1 <- applyPileups(fls1, calcInfo)
identical(res, res1)

## yield by position, across ranges
param <- PileupParam(which=which, yieldSize=500L, yieldBy="position",
                     what="seq")
res <- applyPileups(fls, calcInfo, param=param)
sapply(res, "[[", "seqnames")
```

BamFile   *Maintain and use BAM files*

Description

Use `BamFile()` to create a reference to a BAM file (and optionally its index). The reference remains open across calls to methods, avoiding costly index re-loading.

`BamFileList()` provides a convenient way of managing a list of `BamFile` instances.

Usage

```
## Constructors

BamFile(file, index=file, ..., yieldSize=NA_integer_, obeyQname=FALSE,
        asMates=FALSE)
BamFileList(..., yieldSize=NA_integer_, obeyQname=FALSE, asMates=FALSE)

## Opening / closing

## S3 method for class BamFile
open(con, ...)
## S3 method for class BamFile
close(con, ...)

## accessors; also path(), index(), yieldSize()

## S4 method for signature BamFile
isOpen(con, rw="")
## S4 method for signature BamFile
isIncomplete(con)
## S4 method for signature BamFile
obeyQname(object, ...)
obeyQname(object, ...) <- value
## S4 method for signature BamFile
```

```
asMates(object, ...)
asMates(object, ...) <- value

## actions

## S4 method for signature BamFile
scanBamHeader(files, ...)
## S4 method for signature BamFile
seqinfo(x)
## S4 method for signature BamFile
filterBam(file, destination, index=file, ...,
    filter=FilterRules(), indexDestination=TRUE,
    param=ScanBamParam(what=scanBamWhat()))
## S4 method for signature BamFile
indexBam(files, ...)
## S4 method for signature BamFile
sortBam(file, destination, ..., byQname=FALSE, maxMemory=512)
## S4 method for signature BamFileList
mergeBam(files, destination, ...)

## reading

## S4 method for signature BamFile
scanBam(file, index=file, ..., param=ScanBamParam(what=scanBamWhat()))

## counting

## S4 method for signature BamFile
countBam(file, index=file, ..., param=ScanBamParam())
## S4 method for signature BamFileList
countBam(file, index=file, ..., param=ScanBamParam())
## S4 method for signature BamFile
quickBamFlagSummary(file, ..., param=ScanBamParam(), main.groups.only=FALSE)
```

Arguments

`...`	Additional arguments.
	For `BamFileList`, this can either be a single character vector of paths to BAM files, or several instances of `BamFile` objects. When a character vector of paths, a second named argument 'index' can be a `character()` vector of length equal to the first argument specifying the paths to the index files, or `character()` to indicate that no index file is available. See `BamFile`.
`con`	An instance of `BamFile`.
`x, object, file, files`	
	A character vector of BAM file paths (for `BamFile`) or a `BamFile` instance (for other methods).
`index`	character(1); the BAM index file path (for `BamFile`); ignored for all other methods on this page.

yieldSize	Number of records to yield each time the file is read from with scanBam. See 'Fields' section for details.
asMates	Logical indicating if records should be paired as mates. See 'Fields' section for details.
obeyQname	Logical indicating if the BAM file is sorted by qname. In Bioconductor > 2.12 paired-end files do not need to be sorted by qname. Instead use asMates=TRUE for reading paired-end data. See 'Fields' section for details.
value	Logical value for setting asMates and obeyQname in a BamFile instance.
filter	A [FilterRules](#) instance. Functions in the FilterRules instance should expect a single DataFrame argument representing all information specified by param. Each function must return a logical vector, usually of length equal to the number of rows of the DataFrame. Return values are used to include (when TRUE) corresponding records in the filtered BAM file.
destination	character(1) file path to write filtered reads to.
indexDestination	
	logical(1) indicating whether the destination file should also be indexed.
byQname, maxMemory	
	See [sortBam](#).
param	An optional [ScanBamParam](#) instance to further influence scanning, counting, or filtering.
rw	Mode of file; ignored.
main.groups.only	
	See [quickBamFlagSummary](#).

Objects from the Class

Objects are created by calls of the form BamFile().

Fields

The BamFile class inherits fields from the [RsamtoolsFile](#) class and has fields:

yieldSize: Number of records to yield each time the file is read from using scanBam or, when length(bamWhich())   != 0, a threshold which yields records in complete ranges whose sum first exceeds yieldSize. Setting yieldSize on a BamFileList does not alter existing yield sizes set on the individual BamFile instances.

asMates: A logical indicating if the records should be returned as mated pairs. When TRUE scanBam attempts to mate (pair) the records and returns two additional fields groupid and mate_status. groupid is an integer vector of unique group ids; mate_status is a factor with level mated for records successfully paired by the algorithm, ambiguous for records that are possibly mates but cannot be assigned unambiguously, or unmated for reads that did not have valid mates.

Mate criteria:

- Bit 0x40 and 0x80: Segments are a pair of first/last OR neither segment is marked first/last
- Bit 0x100: Both segments are secondary OR both not secondary

- Bit 0x10 and 0x20: Segments are on opposite strands
- mpos match: segment1 mpos matches segment2 pos AND segment2 mpos matches segment1 pos
- tid match

Flags, tags and ranges may be specified in the ScanBamParam for fine tuning of results.

obeyQname: A logical(0) indicating if the file was sorted by qname. In Bioconductor > 2.12 paired-end files do not need to be sorted by qname. Instead set asMates=TRUE in the BamFile when using the readGAlignmentsListFromBam function from the **GenomicAlignments** package.

Functions and methods

BamFileList inherits methods from RsamtoolsFileList and SimpleList.

Opening / closing:

open.BamFile Opens the (local or remote) path and index (if bamIndex is not character(0)), files. Returns a BamFile instance.

close.BamFile Closes the BamFile con; returning (invisibly) the updated BamFile. The instance may be re-opened with open.BamFile.

isOpen Tests whether the BamFile con has been opened for reading.

isIncomplete Tests whether the BamFile con is niether closed nor at the end of the file.

Accessors:

path Returns a character(1) vector of BAM path names.

index Returns a character(0) or character(1) vector of BAM index path names.

yieldSize, yieldSize<- Return or set an integer(1) vector indicating yield size.

obeyQname, obeyQname<- Return or set a logical(0) indicating if the file was sorted by qname.

asMates, asMates<- Return or set a logical(0) indicating if the records should be returned as mated pairs.

Methods:

scanBamHeader Visit the path in path(file), returning the information contained in the file header; see scanBamHeader.

seqinfo, seqnames, seqlength Visit the path in path(file), returning a Seqinfo, character, or named integer vector containing information on the anmes and / or lengths of each sequence. Seqnames are ordered 'naturally' as, e.g., 'chr1', 'chr2', ...

scanBam Visit the path in path(file), returning the result of scanBam applied to the specified path.

countBam Visit the path(s) in path(file), returning the result of countBam applied to the specified path.

filterBam Visit the path in path(file), returning the result of filterBam applied to the specified path.

indexBam Visit the path in path(file), returning the result of indexBam applied to the specified path.

sortBam Visit the path in `path(file)`, returning the result of sortBam applied to the specified path.

mergeBam Merge several BAM files into a single BAM file. See mergeBam for details; additional arguments supported by `mergeBam,character-method` are also available for `BamFileList`.

show Compactly display the object.

Author(s)

Martin Morgan and Marc Carlson

See Also

- The readGAlignmentsFromBam, readGAlignmentPairsFromBam, and readGAlignmentsListFromBam functions defined in the **GenomicAlignments** package.

- summarizeOverlaps and findSpliceOverlaps-methods in the **GenomicAlignments** package for methods that work on a BamFile and BamFileList objects.

Examples

```
##
## BamFile options.
##

fl <- system.file("extdata", "ex1.bam", package="Rsamtools")
bf <- BamFile(fl)
bf

## When asMates=TRUE scanBam() reads the data in as
## pairs. See asMates above for details of the pairing
## algorithm.
asMates(bf) <- TRUE

## When yieldSize is set, scanBam() will iterate
## through the file in chunks.
yieldSize(bf) <- 500


##
## Reading Bam files.
##

fl <- system.file("extdata", "ex1.bam", package="Rsamtools",
                  mustWork=TRUE)
length(scanBam(fl)[[1]][[1]])  # all records

bf <- open(BamFile(fl))        # implicit index
bf
identical(scanBam(bf), scanBam(fl))
close(bf)
```

```
## Use yieldSize to iterate through a file in chunks.
bf <- open(BamFile(fl, yieldSize=1000))
while (nrec <- length(scanBam(bf)[[1]][[1]]))
    cat("records:", nrec, "\n")
close(bf)

## Repeatedly visit multiple ranges in the BamFile.
rng <- GRanges(c("seq1", "seq2"), IRanges(1, c(1575, 1584)))
bf <- open(BamFile(fl))
sapply(seq_len(length(rng)), function(i, bamFile, rng) {
    param <- ScanBamParam(which=rng[i], what="seq")
    bam <- scanBam(bamFile, param=param)[[1]]
    alphabetFrequency(bam[["seq"]], baseOnly=TRUE, collapse=TRUE)
}, bf, rng)
close(bf)
```

| BamInput | *Import, count, index, filter, sort, and merge 'BAM' (binary alignment) files.* |

Description

Import binary 'BAM' files into a list structure, with facilities for selecting what fields and which records are imported, and other operations to manipulate BAM files.

Usage

```
scanBam(file, index=file, ..., param=ScanBamParam(what=scanBamWhat()))

countBam(file, index=file, ..., param=ScanBamParam())

scanBamHeader(files, ...)
## S4 method for signature character
scanBamHeader(files, ...)

asBam(file, destination, ...)
## S4 method for signature character
asBam(file, destination, ...,
    overwrite=FALSE, indexDestination=TRUE)

asSam(file, destination, ...)
## S4 method for signature character
asSam(file, destination, ..., overwrite=FALSE)

filterBam(file, destination, index=file, ...)
## S4 method for signature character
filterBam(file, destination, index=file, ...,
```

```
    filter=FilterRules(), indexDestination=TRUE,
    param=ScanBamParam(what=scanBamWhat()))

sortBam(file, destination, ...)
## S4 method for signature character
sortBam(file, destination, ..., byQname=FALSE, maxMemory=512)

indexBam(files, ...)
## S4 method for signature character
indexBam(files, ...)

mergeBam(files, destination, ...)
## S4 method for signature character
mergeBam(files, destination, ..., region = RangedData(),
    overwrite = FALSE, header = character(), byQname = FALSE,
    addRG = FALSE, compressLevel1 = FALSE, indexDestination = FALSE)
```

Arguments

file	The character(1) file name of the 'BAM' ('SAM' for asBam) file to be processed.
files	The character() file names of the 'BAM' file to be processed. For mergeBam, must satisfy length(files) >= 2.
index	The character(1) name of the index file of the 'BAM' file being processed; this is given *without* the '.bai' extension.
destination	The character(1) file name of the location where the sorted, filtered, or merged output file will be created. For asBam asSam, and sortBam this is without the ".bam" file suffix.
region	A RangedData() instance with >= 1 rows, specifying the region of the BAM files to merged.
...	Additional arguments, passed to methods.
overwrite	A logical(1) indicating whether the destination can be over-written if it already exists.
filter	A [FilterRules](#) instance allowing users to filter BAM files based on arbitrary criteria, as described below.
indexDestination	A logical(1) indicating whether the created destination file should also be indexed.
byQname	A logical(1) indicating whether the sorted destination file should be sorted by Query-name (TRUE) or by mapping position (FALSE).
header	A character(1) file path for the header information to be used in the merged BAM file.
addRG	A logical(1) indicating whether the file name should be used as RG (read group) tag in the merged BAM file.
compressLevel1	A logical(1) indicating whether the merged BAM file should be compressed to zip level 1.

maxMemory   A numerical(1) indicating the maximal amount of memory (in MB) that the
                   function is allowed to use.

param   An instance of [ScanBamParam](). This influences what fields and which records
                   are imported.

Details

The scanBam function parses binary BAM files; text SAM files can be parsed using R's [scan]() func-
tion, especially with arguments what to control the fields that are parsed.

countBam returns a count of records consistent with param.

scanBamHeader visits the header information in a BAM file, returning for each file a list containing
elements targets and text, as described below. The SAM / BAM specification does not require
that the content of the header be consistent with the content of the file, e.g., more targets may be
present that are represented by reads in the file.

asBam converts 'SAM' files to 'BAM' files, equivalent to samtools view -Sb file > destination.
The 'BAM' file is sorted and an index created on the destination (with extension '.bai') when
indexDestination=TRUE.

asSam converts 'BAM' files to 'SAM' files, equivalent to samtools view file > destination.

filterBam parses records in file. Records satisfying the bamWhich bamFlag and bamSimpleCigar
criteria of param are accumulated to a default of yieldSize =   1000000 records (change this by
specifying yieldSize when creating a BamFile instance; see [BamFile]()-class). These records are
then parsed to a DataFrame and made available for further filtering by user-supplied FilterRules.
Functions in the FilterRules instance should expect a single DataFrame argument representing all
information specified by param. Each function must return a logical vector equal to the number
of rows of the DataFrame. Return values are used to include (when TRUE) corresponding records
in the filtered BAM file. The BAM file is created at destination. An index file is created on
the destination when indexDestination=TRUE. It is more space- and time-efficient to filter use
bamWHich, bamFlag, and bamSimpleCigar, if appropriate, than to supply FilterRules.

sortBam sorts the BAM file given as its first argument, analogous to the "samtools sort" function.

indexBam creates an index for each BAM file specified, analogous to the 'samtools index' function.

mergeBam merges 2 or more sorted BAM files. As with samtools, the RG (read group) dictionary in
the header of the BAM files is not reconstructed.

Details of the ScanBamParam class are provide on its help page; several salient points are reiterated
here. ScanBamParam can contain a field what, specifying the components of the BAM records to
be returned. Valid values of what are available with [scanBamWhat](). ScanBamParam can contain an
argument which that specifies a subset of reads to return. This requires that the BAM file be indexed,
and that the file be named following samtools convention as <bam_filename>.bai. ScanBamParam
can contain an argument tag to specify which tags will be extracted.

Value

The scanBam,character-method returns a list of lists. The outer list groups results from each
Ranges list of bamWhich(param); the outer list is of length one when bamWhich(param) has
length 0. Each inner list contains elements named after scanBamWhat(); elements omitted from
bamWhat(param) are removed. The content of non-null elements are as follows, taken from the
description in the samtools API documentation:

- qname: This is the QNAME field in SAM Spec v1.4. The query name, i.e., identifier, associated with the read.

- flag: This is the FLAG field in SAM Spec v1.4. A numeric value summarizing details of the read. See [ScanBamParam](#) and the `flag` argument, and scanBamFlag().

- rname: This is the RNAME field in SAM Spec v1.4. The name of the reference to which the read is aligned.

- strand: The strand to which the read is aligned.

- pos: This is the POS field in SAM Spec v1.4. The genomic coordinate at the start of the alignment. Coordinates are 'left-most', i.e., at the 3' end of a read on the '-' strand, and 1-based. The position *excludes* clipped nucleotides, even though soft-clipped nucleotides are included in `seq`.

- qwidth: The width of the query, as calculated from the `cigar` encoding; normally equal to the width of the query returned in `seq`.

- mapq: This is the MAPQ field in SAM Spec v1.4. The MAPping Quality.

- cigar: This is the CIGAR field in SAM Spec v1.4. The CIGAR string.

- mrnm: This is the RNEXT field in SAM Spec v1.4. The reference to which the mate (of a paired end or mate pair read) aligns.

- mpos: This is the PNEXT field in SAM Spec v1.4. The position to which the mate aligns.

- isize: This is the TLEN field in SAM Spec v1.4. Inferred insert size for paired end alignments.

- seq: This is the SEQ field in SAM Spec v1.4. The query sequence, in the 5' to 3' orientation. If aligned to the minus strand, it is the reverse complement of the original sequence.

- qual: This is the QUAL field in SAM Spec v1.4. Phred-encoded, phred-scaled base quality score, oriented as `seq`.

- groupid: This is an integer vector of unique group ids returned when `asMates=TRUE` in a BamFile object. `groupid` values are used to create the partitioning for a GAlignmentsList object.

- mate_status: Returned (always) when `asMates=TRUE` in a BamFile object. This is a factor indicating status (`mated`, `ambiguous`, `unmated`) of each record.

scanBamHeader returns a list, with one element for each file named in `files`. The list contains two element. The `targets` element contains target (reference) sequence lengths. The `text` element is itself a list with each element a list corresponding to tags (e.g., '@SQ') found in the header, and the associated tag values.

asBam, asSam return the file name of the destination file.

sortBam returns the file name of the sorted file.

indexBam returns the file name of the index file created.

filterBam returns the file name of the destination file created.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>. Thomas Unterhiner <thomas.unterthiner@students.jku.at> (sortBam).

References

<http://samtools.sourceforge.net/>

See Also

ScanBamParam, scanBamWhat, scanBamFlag

Examples

```
fl <- system.file("extdata", "ex1.bam", package="Rsamtools",
                  mustWork=TRUE)

##
## scanBam
##

res0 <- scanBam(fl)[[1]] # always list-of-lists
names(res0)
length(res0[["qname"]])
lapply(res0, head, 3)
table(width(res0[["seq"]])) # query widths
table(res0[["qwidth"]], useNA="always") # query widths derived from cigar
table(res0[["cigar"]], useNA="always")
table(res0[["strand"]], useNA="always")
table(res0[["flag"]], useNA="always")

which <- RangesList(seq1=IRanges(1000, 2000),
                    seq2=IRanges(c(100, 1000), c(1000, 2000)))
p1 <- ScanBamParam(which=which, what=scanBamWhat())
res1 <- scanBam(fl, param=p1)
names(res1)
names(res1[[2]])

p2 <- ScanBamParam(what=c("rname", "strand", "pos", "qwidth"))
res2 <- scanBam(fl, param=p2)

p3 <- ScanBamParam(flag=scanBamFlag(isMinusStrand=FALSE))
length(scanBam(fl, param=p3)[[1]])

##
## filterBam
##

param <- ScanBamParam(
    flag=scanBamFlag(isUnmappedQuery=FALSE),
    what="seq")
dest <- filterBam(fl, tempfile(), param=param)
countBam(dest)  ## 3271 records
filt <- list(MinWidth = function(x) width(x$seq) > 35)
dest <- filterBam(fl, tempfile(), param=param, filter=FilterRules(filt))
countBam(dest)  ## 398 records
```

```
res3 <- scanBam(dest, param=ScanBamParam(what="seq"))[[1]]
table(width(res3$seq))

##
## sortBam
##

sorted <- sortBam(fl, tempfile())

## map mcols(gwhich) to output, e.g., of countBam
gwhich <- as(which, "GRanges")[c(2, 1, 3)]
mcols(gwhich)[["OriginalOrder"]] <- 1:3
cnt <- countBam(fl, param=ScanBamParam(which=gwhich))
cntVals <- unlist(split(mcols(gwhich), seqnames(gwhich)))
cbind(cnt, as.data.frame(cntVals))
```

BamSampler   *Sample from a BAM files*

Description

Use `BamSampler()` to create a reference to a BAM file (and optionally its index). Calls to `scanBam` (and many functions that use `scanBam`) draw a random sample from the BAM file.

Usage

```
## Constructors

BamSampler(file, index = file, ..., yieldSize, obeyQname = FALSE, asMates = FALSE)

## S4 method for signature BamSampler
scanBam(file, index=file, ..., param=ScanBamParam(what=scanBamWhat()))
```

Arguments

file	character(1); BAM file path for `BamSampler`, or `BamSampler` index for `scanBam` and other functions.
index	character(1); the BAM index file path (for `BamFile`); ignored for other methods.
...	Additional arguments; see [BamFile](#)-class.
yieldSize	integer(1); number of records to yield each time the file is read from using `scanBam`.
obeyQname	logical(1); indicating whether the file is sorted by qname and if so, that qnames are not split between yields.
asMates	logical(1); indicating whether the records should be returned as mated pairs.
param	An optional [ScanBamParam](#) instance to further influence scanning, counting, or filtering.

Objects from the Class

Objects are created by calls of the form BamSampler().

Fields

The BamSampler class inherits fields from the [BamFile](#) class.

Functions and methods

BamSampler inherits methods from [BamFile](#) and can be used in place of BamFile in many functions.

Author(s)

Martin Morgan

Examples

```
library(GenomicAlignments)  # for readGAlignmentsFromBam()
fl <- system.file("extdata", "ex1.bam", package="Rsamtools")
samp <- BamSampler(fl, yieldSize=1000)
## two independent samples
head(readGAlignmentsFromBam(samp))
head(readGAlignmentsFromBam(samp))
```

BamViews   *Views into a set of BAM files*

Description

Use BamViews() to reference a set of disk-based BAM files to be processed (e.g., queried using [scanBam](#)) as a single 'experiment'.

Usage

```
## Constructor
BamViews(bamPaths=character(0),
    bamIndicies=bamPaths,
    bamSamples=DataFrame(row.names=make.unique(basename(bamPaths))),
    bamRanges, bamExperiment = list(), ...)
## S4 method for signature missing
BamViews(bamPaths=character(0),
    bamIndicies=bamPaths,
    bamSamples=DataFrame(row.names=make.unique(basename(bamPaths))),
    bamRanges, bamExperiment = list(), ..., auto.range=FALSE)
## Accessors
bamPaths(x)
```

```
bamSamples(x)
bamSamples(x) <- value
bamRanges(x)
bamRanges(x) <- value
bamExperiment(x)

## S4 method for signature BamViews
names(x)
## S4 replacement method for signature BamViews
names(x) <- value
## S4 method for signature BamViews
dimnames(x)
## S4 replacement method for signature BamViews,ANY
dimnames(x) <- value

bamDirname(x, ...) <- value

## Subset
## S4 method for signature BamViews,ANY,ANY
x[i, j, ..., drop=TRUE]
## S4 method for signature BamViews,ANY,missing
x[i, j, ..., drop=TRUE]
## S4 method for signature BamViews,missing,ANY
x[i, j, ..., drop=TRUE]

## Input
## S4 method for signature BamViews
scanBam(file, index = file, ..., param = ScanBamParam(what=scanBamWhat()))
## S4 method for signature BamViews
countBam(file, index = file, ..., param = ScanBamParam())

## Show
## S4 method for signature BamViews
show(object)
```

Arguments

bamPaths	A character() vector of BAM path names.
bamIndicies	A character() vector of BAM index file path names, *without* the '.bai' extension.
bamSamples	A [DataFrame](#) instance with as many rows as length(bamPaths), containing sample information associated with each path.
bamRanges	A [GRanges](#), [RangedData](#) or missing instance with ranges defined on the spaces of the BAM files. Ranges are *not* validated against the BAM files.
bamExperiment	A list() containing additional information about the experiment.
auto.range	If TRUE and all bamPaths exist, populate the ranges with the union of ranges returned in the target element of scanBamHeader.
...	Additional arguments.

x	An instance of `BamViews`.
object	An instance of `BamViews`.
value	An object of appropriate type to replace content.
i	During subsetting, a logical or numeric index into `bamRanges`.
j	During subsetting, a logical or numeric index into `bamSamples` and `bamPaths`.
drop	A logical(1), *ignored* by all `BamViews` subsetting methods.
file	An instance of `BamViews`.
index	A character vector of indices, corresponding to the `bamPaths(file)`.
param	An optional [ScanBamParam](#) instance to further influence scanning or counting.

Objects from the Class

Objects are created by calls of the form `BamViews()`.

Slots

bamPaths A character() vector of BAM path names.

bamIndicies A character() vector of BAM index path names.

bamSamples A [DataFrame](#) instance with as many rows as `length(bamPaths)`, containing sample information associated with each path.

bamRanges A [GRanges](#) instance with ranges defined on the spaces of the BAM files. Ranges are *not* validated against the BAM files.

bamExperiment A list() containing additional information about the experiment.

Functions and methods

See 'Usage' for details on invocation.

Constructor:

BamViews: Returns a `BamViews` object.

Accessors:

bamPaths Returns a character() vector of BAM path names.

bamIndicies Returns a character() vector of BAM index path names.

bamSamples Returns a [DataFrame](#) instance with as many rows as `length(bamPaths)`, containing sample information associated with each path.

bamSamples<- Assign a [DataFrame](#) instance with as many rows as `length(bamPaths)`, containing sample information associated with each path.

bamRanges Returns a [GRanges](#) instance with ranges defined on the spaces of the BAM files. Ranges are *not* validated against the BAM files.

bamRanges<- Assign a [GRanges](#) instance with ranges defined on the spaces of the BAM files. Ranges are *not* validated against the BAM files.

bamExperiment Returns a list() containing additional information about the experiment.

names Return the column names of the `BamViews` instance; same as `names(bamSamples(x))`.

names<- Assign the column names of the `BamViews` instance.

dimnames Return the row and column names of the `BamViews` instance.

dimnames<- Assign the row and column names of the `BamViews` instance.

Methods:

"[" Subset the object by bamRanges or bamSamples.

scanBam Visit each path in `bamPaths(file)`, returning the result of `scanBam` applied to the specified path. `bamRanges(file)` takes precedence over `bamWhich(param)`.

countBam Visit each path in `bamPaths(file)`, returning the result of `countBam` applied to the specified path. `bamRanges(file)` takes precedence over `bamWhich(param)`.

show Compactly display the object.

Author(s)

Martin Morgan

Examples

```
fls <- system.file("extdata", "ex1.bam", package="Rsamtools",
                    mustWork=TRUE)
rngs <- GRanges(seqnames = Rle(c("chr1", "chr2"), c(9, 9)),
                ranges = c(IRanges(seq(10000, 90000, 10000), width=500),
                           IRanges(seq(100000, 900000, 100000), width=5000)),
                Count = seq_len(18L))
v <- BamViews(fls, bamRanges=rngs)
v
v[1:5,]
bamRanges(v[c(1:5, 11:15),])
bamDirname(v) <- getwd()
v
```

`BcfFile`   *Manipulate BCF files.*

Description

Use `BcfFile()` to create a reference to a BCF (and optionally its index). The reference remains open across calls to methods, avoiding costly index re-loading.

`BcfFileList()` provides a convenient way of managing a list of `BcfFile` instances.

Usage

```
## Constructors

BcfFile(file, index = file,
        mode=ifelse(grepl("\\.bcf$", file), "rb", "r"))
BcfFileList(...)

## Opening / closing

## S3 method for class BcfFile
open(con, ...)
## S3 method for class BcfFile
close(con, ...)

## accessors; also path(), index()

## S4 method for signature BcfFile
isOpen(con, rw="")
bcfMode(object)

## actions

## S4 method for signature BcfFile
scanBcfHeader(file, ...)
## S4 method for signature BcfFile
scanBcf(file, ..., param=ScanBcfParam())
## S4 method for signature BcfFile
indexBcf(file, ...)
```

Arguments

con, object	An instance of BcfFile.
file	A character(1) vector of the BCF file path or, (for indexBcf) an instance of BcfFile point to a BCF file.
index	A character(1) vector of the BCF index.
mode	A character(1) vector; mode="rb" indicates a binary (BCF) file, mode="r" a text (VCF) file.
param	An optional [ScanBcfParam](#) instance to further influence scanning.
...	Additional arguments. For BcfFileList, this can either be a single character vector of paths to BCF files, or several instances of BcfFile objects.
rw	Mode of file; ignored.

Objects from the Class

Objects are created by calls of the form BcfFile().

Fields

The `BcfFile` class inherits fields from the `RsamtoolsFile` class.

Functions and methods

`BcfFileList` inherits methods from `RsamtoolsFileList` and `SimpleList`.

Opening / closing:

open.BcfFile Opens the (local or remote) `path` and index (if bamIndex is not `character(0)`), files. Returns a `BcfFile` instance.

close.BcfFile Closes the `BcfFile` con; returning (invisibly) the updated `BcfFile`. The instance may be re-opened with `open.BcfFile`.

Accessors:

path Returns a character(1) vector of the BCF path name.

index Returns a character(1) vector of BCF index name.

bcfMode Returns a character(1) vector BCF mode.

Methods:

scanBcf Visit the path in `path(file)`, returning the result of `scanBcf` applied to the specified path.

show Compactly display the object.

Author(s)

Martin Morgan

Examples

```
fl <- system.file("extdata", "ex1.bcf", package="Rsamtools",
                  mustWork=TRUE)
bf <- BcfFile(fl)        # implicit index
bf
identical(scanBcf(bf), scanBcf(fl))

rng <- GRanges(c("seq1", "seq2"), IRanges(1, c(1575, 1584)))
param <- ScanBcfParam(which=rng)
bcf <- scanBcf(bf, param=param)  ## all ranges

## ranges one at a time bf
open(bf)
sapply(seq_len(length(rng)), function(i, bcfFile, rng) {
    param <- ScanBcfParam(which=rng)
    bcf <- scanBcf(bcfFile, param=param)[[1]]
    ## do extensive work with bcf
    isOpen(bf)  ## file remains open
}, bf, rng)
```

BcfInput   *Operations on 'BCF' files.*

Description

Import, coerce, or index variant call files in text or binary format.

Usage

```
scanBcfHeader(file, ...)
## S4 method for signature character
scanBcfHeader(file, ...)

scanBcf(file, ...)
## S4 method for signature character
scanBcf(file, index = file, ..., param=ScanBcfParam())

asBcf(file, dictionary, destination, ...,
      overwrite=FALSE, indexDestination=TRUE)
## S4 method for signature character
asBcf(file, dictionary, destination, ...,
      overwrite=FALSE, indexDestination=TRUE)

indexBcf(file, ...)
## S4 method for signature character
indexBcf(file, ...)
```

Arguments

file	For scanBcf and scanBcfHeader, the character() file name of the 'BCF' file to be processed, or an instance of class [BcfFile]().
index	The character() file name(s) of the 'BCF' index to be processed.
dictionary	a character vector of the unique "CHROM" names in the VCF file.
destination	The character(1) file name of the location where the BCF output file will be created. For asBcf this is without the ".bcf" file suffix.
param	A instance of [ScanBcfParam]() influencing which records are parsed and the 'INFO' and 'GENO' information returned.
...	Additional arguments, e.g., for scanBcfHeader,character-method, mode of [BcfFile]().
overwrite	A logical(1) indicating whether the destination can be over-written if it already exists.
indexDestination	
	A logical(1) indicating whether the created destination file should also be indexed.

Details

`bcf*` functions are restricted to the GENO fields supported by 'bcftools' (see documentation at the url below). The argument `param` allows portions of the file to be input, but requires that the file be BCF or bgzip'd and indexed as a `TabixFile`. For similar functions operating on VCF files see `?scanVcf` in the `VariantAnnotation` package.

Value

`scanBcfHeader` returns a list, with one element for each file named in `file`. Each element of the list is itself a list containing three elements. The `reference` element is a character() vector with names of reference sequences. The `sample` element is a character() vector of names of samples. The `header` element is a character() vector of the header lines (preceeded by "##") present in the VCF file.

`scanBcf` returns a list, with one element per file. Each list has 9 elements, corresponding to the columns of the VCF specification: CHROM, POS, ID, REF, ALTQUAL, FILTER, INFO, FORMAT, GENO.

The GENO element is itself a list, with elements corresponding to fields supported by 'bcftools' (see documentation at the url below).

`asBcf` creates a binary BCF file from a text VCF file.

`indexBcf` creates an index into the BCF file.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>.

References

<http://vcftools.sourceforge.net/specs.html> outlines the VCF specification.

<http://samtools.sourceforge.net/mpileup.shtml> contains information on the portion of the specification implemented by `bcftools`.

<http://samtools.sourceforge.net/> provides information on `samtools`.

See Also

`BcfFile`, `TabixFile`

Examples

```
fl <- system.file("extdata", "ex1.bcf", package="Rsamtools",
                  mustWork=TRUE)
scanBcfHeader(fl)
bcf <- scanBcf(fl)
## value: list-of-lists
str(bcf[1:8])
names(bcf[["GENO"]])
str(head(bcf[["GENO"]][["PL"]]))
example(BcfFile)
```

Compression   *File compression for tabix (bgzip) and fasta (razip) files.*

Description

These functions compress files for use in other parts of **Rsamtools**: bgzip for tabix files, razip for random-access fasta files.

Usage

```
bgzip(file, dest=sprintf("%s.bgz", sub("\\.gz$", "", file)),
      overwrite = FALSE)
razip(file, dest=sprintf("%s.rz", sub("\\.gz$", "", file)),
      overwrite = FALSE)
```

Arguments

file   A character(1) path to an existing uncompressed or gz-compressed file. This file will be compressed.

dest   A character(1) path to a file. This will be the compressed file. If dest exists, then it is only over-written when overwrite=TRUE.

overwrite   A logical(1) indicating whether dest should be over-written, if it already exists.

Value

The full path to dest.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>

References

<http://samtools.sourceforge.net/>

See Also

TabixFile, FaFile.

Examples

```
from <- system.file("extdata", "ex1.sam", package="Rsamtools",
                     mustWork=TRUE)
to <- tempfile()
zipped <- bgzip(from, to)
```

deprecated	*Deprecated functions*

Description

Functions listed on this page are no longer supported.

Details

For `yieldTabix`, use the `yieldSize` argument of `TabixFiles`.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>.

FaFile	*Manipulate indexed fasta files.*

Description

Use `FaFile()` to create a reference to an indexed fasta file. The reference remains open across calls to methods, avoiding costly index re-loading.

`FaFileList()` provides a convenient way of managing a list of `FaFile` instances.

Usage

```
## Constructors

FaFile(file, ...)
FaFileList(...)

## Opening / closing

## S3 method for class FaFile
open(con, ...)
## S3 method for class FaFile
close(con, ...)

## accessors; also path(), index()

## S4 method for signature FaFile
isOpen(con, rw="")

## actions
```

```
## S4 method for signature FaFile
indexFa(file, ...)

## S4 method for signature FaFile
scanFaIndex(file, ...)
## S4 method for signature FaFileList
scanFaIndex(file, ..., as=c("GRangesList", "GRanges"))

## S4 method for signature FaFile
seqinfo(x)

## S4 method for signature FaFile
countFa(file, ...)

## S4 method for signature FaFile,GRanges
scanFa(file, param, ...,
    as=c("DNAStringSet", "RNAStringSet", "AAStringSet"))
## S4 method for signature FaFile,RangesList
scanFa(file, param, ...,
    as=c("DNAStringSet", "RNAStringSet", "AAStringSet"))
## S4 method for signature FaFile,RangedData
scanFa(file, param, ...,
    as=c("DNAStringSet", "RNAStringSet", "AAStringSet"))
## S4 method for signature FaFile,missing
scanFa(file, param, ...,
    as=c("DNAStringSet", "RNAStringSet", "AAStringSet"))

## S4 method for signature FaFile
getSeq(x, param, ...)
## S4 method for signature FaFileList
getSeq(x, param, ...)
```

Arguments

con, x	An instance of FaFile or (for getSeq) FaFileList.
file	A character(1) vector of the fasta file path (for FaFile), or an instance of class FaFile or FaFileList (for scanFaIndex, getSeq).
param	An optional [GRanges](), [RangesList](), or [RangedData]() instance to select reads (and sub-sequences) for input. See Methods, below.
...	Additional arguments.
	• For FaFileList, this can either be a single character vector of paths to BAM files, or several instances of FaFile objects.
	• For scanFa,FaFile,missing-method this can include arguemnts to readDNAStringSet / readRNAStringSet / readAAStringSet when param is 'missing'.
rw	Mode of file; ignored.
as	A character(1) vector indicating the type of object to return.

- For scanFaIndex, default GRangesList, with index information from each file is returned as an element of the list.
- For scanFa, default DNAStringSet.

GRangesList, index information is collapsed across files into the unique index elements.

Objects from the Class

Objects are created by calls of the form FaFile().

Fields

The FaFile class inherits fields from the [RsamtoolsFile](#) class.

Functions and methods

FaFileList inherits methods from [RsamtoolsFileList](#) and [SimpleList](#).

Opening / closing:

open.FaFile Opens the (local or remote) path and index files. Returns a FaFile instance.

close.FaFile Closes the FaFile con; returning (invisibly) the updated FaFile. The instance may be re-opened with open.FaFile.

Accessors:

path Returns a character(1) vector of the fasta path name.

index Returns a character(1) vector of fasta index name (minus the '.fai' extension).

Methods:

indexFa Visit the path in path(file) and create an index file (with the extension '.fai').

scanFaIndex Read the sequence names and and widths of recorded in an indexed fasta file, returning the information as a [GRanges](#) object.

seqinfo Consult the index file for defined sequences (seqlevels()) and lengths (seqlengths()).

countFa Return the number of records in the fasta file.

scanFa Return the sequences indicated by param as a [DNAStringSet](#) instance. seqnames(param) selects the sequences to return; start(param) and end{param} define the (1-based) region of the sequence to return. Values of end(param) greater than the width of the sequence are set to the width of the sequence. When param is missing, all records are selected. When length(param)==0 no records are selected.

getSeq Returns the sequences indicated by param from the indexed fasta file(s) of file.

For the FaFile method, the return type is a DNAStringSet. The getSeq,FaFile and scanFa,FaFile,GRanges methods differ in that getSeq will reverse complement sequences selected from the minus strand.

For the FaFileList method, the param argument must be a GRangesList of the same length as file, creating a one-to-one mapping between the ith element of file and the ith element of param; the return type is a SimpleList of DNAStringSet instances, with elements of the list in the same order as the input elements.

show Compactly display the object.

Author(s)

Martin Morgan

Examples

```
fl <- system.file("extdata", "ce2dict1.fa", package="Rsamtools",
                  mustWork=TRUE)
fa <- open(FaFile(fl))                  # open
countFa(fa)
(idx <- scanFaIndex(fa))
(dna <- scanFa(fa, param=idx[1:2]))
ranges(idx) <- narrow(ranges(idx), -10)  # last 10 nucleotides
(dna <- scanFa(fa, param=idx[1:2]))
```

FaInput   *Operations on indexed 'fasta' files.*

Description

Scan indexed fasta (or compressed fasta) files and their indicies.

Usage

```
indexFa(file, ...)
## S4 method for signature character
indexFa(file, ...)

scanFaIndex(file, ...)
## S4 method for signature character
scanFaIndex(file, ...)

countFa(file, ...)
## S4 method for signature character
countFa(file, ...)

scanFa(file, param, ...,
    as=c("DNAStringSet", "RNAStringSet", "AAStringSet"))
## S4 method for signature character,GRanges
scanFa(file, param, ...,
    as=c("DNAStringSet", "RNAStringSet", "AAStringSet"))
## S4 method for signature character,RangesList
scanFa(file, param, ...,
    as=c("DNAStringSet", "RNAStringSet", "AAStringSet"))
## S4 method for signature character,RangedData
```

```
scanFa(file, param, ...,
    as=c("DNAStringSet", "RNAStringSet", "AAStringSet"))
## S4 method for signature character,missing
scanFa(file, param, ...,
    as=c("DNAStringSet", "RNAStringSet", "AAStringSet"))
```

Arguments

`file`	A character(1) vector containing the fasta file path.
`param`	An optional [GRanges](#), [RangesList](#), or [RangedData](#) instance to select reads (and sub-sequences) for input.
`as`	A character(1) vector indicating the type of object to return; default `DNAStringSet`.
`...`	Additional arguments, passed to readDNAStringSet / readRNAStringSet / readAAStringSet when param is 'missing'.

Value

`indexFa` visits the path in `file` and create an index file at the same location but with extension '.fai').

`scanFaIndex` reads the sequence names and and widths of recorded in an indexed fasta file, returning the information as a [GRanges](#) object.

`countFa` returns the number of records in the fasta file.

`scanFa` return the sequences indicated by `param` as a [DNAStringSet](#), [RNAStringSet](#), [AAStringSet](#) instance. `seqnames(param)` selects the sequences to return; `start(param)` and `end{param}` define the (1-based) region of the sequence to return. Values of `end(param)` greater than the width of the sequence are set to the width of the sequence. When `param` is missing, all records are selected. When `param` is GRanges(), no records are selected.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>.

References

<http://samtools.sourceforge.net/> provides information on `samtools`.

Examples

```
fa <- system.file("extdata", "ce2dict1.fa", package="Rsamtools",
                  mustWork=TRUE)
countFa(fa)
(idx <- scanFaIndex(fa))
(dna <- scanFa(fa, idx[1:2]))
ranges(idx) <- narrow(ranges(idx), -10)  # last 10 nucleotides
(dna <- scanFa(fa, idx[1:2]))
```

---

headerTabix   *Retrieve sequence names defined in a tabix file.*

Description

This function queries a tabix file, returning the names of the 'sequences' used as a key when creating the file.

Usage

```
headerTabix(file, ...)
## S4 method for signature character
headerTabix(file, ...)
```

Arguments

file	A character(1) file path or [TabixFile] instance pointing to a 'tabix' file.
...	Additional arguments, currently ignored.

Value

A list(4) of the sequence names, column indicies used to sort the file, the number of lines skipped while indexing, and the comment character used while indexing.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>.

Examples

```
fl <- system.file("extdata", "example.gtf.gz", package="Rsamtools",
                   mustWork=TRUE)
headerTabix(fl)
```

---

indexTabix   *Compress and index tabix-compatible files.*

Description

Index (with indexTabix) files that have been sorted into ascending sequence, start and end position ordering.

Usage

```
indexTabix(file,
           format=c("gff", "bed", "sam", "vcf", "vcf4", "psltbl"),
           seq=integer(), start=integer(), end=integer(),
           skip=0L, comment="#", zeroBased=FALSE, ...)
```

Arguments

file	A characater(1) path to a sorted, bgzip-compressed file.
format	The format of the data in the compressed file. A characater(1) matching one of the types named in the function signature.
seq	If format is missing, then seq indicates the column in which the 'sequence' identifier (e.g., chrq) is to be found.
start	If format is missing, start indicates the column containing the start coordinate of the feature to be indexed.
end	If format is missing, end indicates the column containing the ending coordinate of the feature to be indexed.
skip	The number of lines to be skipped at the beginning of the file.
comment	A single character which, when present as the first character in a line, indicates that the line is to be omitted. from indexing.
zeroBased	A logical(1) indicating whether coordinats in the file are zero-based.
...	Additional arguments.

Value

The return value of indexTabix is an updated instance of file reflecting the newly-created index file.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>.

References

<http://samtools.sourceforge.net/tabix.shtml>

Examples

```
from <- system.file("extdata", "ex1.sam", package="Rsamtools",
                    mustWork=TRUE)
to <- tempfile()
zipped <- bgzip(from, to)
idx <- indexTabix(zipped, "sam")

tab <- TabixFile(zipped, idx)
```

PileupFiles   *Represent BAM files for pileup summaries.*

Description

Use `PileupFiles()` to create a reference to a BAM files (and their indicies), to be used for calculating pile-up summaries.

Usage

```
## Constructors
PileupFiles(files, ..., param=PileupParam())
## S4 method for signature character
PileupFiles(files, ..., param=PileupParam())
## S4 method for signature list
PileupFiles(files, ..., param=PileupParam())

## opening / closing
## S3 method for class PileupFiles
open(con, ...)
## S3 method for class PileupFiles
close(con, ...)

## accessors; also path()
## S4 method for signature PileupFiles
isOpen(con, rw="")
plpFiles(object)
plpParam(object)

## actions
## S4 method for signature PileupFiles,missing
applyPileups(files, FUN, ..., param)
## S4 method for signature PileupFiles,PileupParam
applyPileups(files, FUN, ..., param)

## display
## S4 method for signature PileupFiles
show(object)
```

Arguments

files   For `PileupFiles`, a `character()` or `list` of `BamFile` instances representing
                files to be included in the pileup. Using a `list` of `BamFile` allows indicies to be
                specified when these are in non-standard format. All elements of ... must be
                the same type.

	For applyPileups,PileupFiles-method, a PileupFiles instance.
...	Additional arguments, currently ignored.
con, object	An instance of PileupFiles.
FUN	A function of one argument; see applyPileups.
param	An instance of PileupParam, to select which records to include in the pileup, and which summary information to return.
rw	character() indicating mode of file; not used for TabixFile.

Objects from the Class

Objects are created by calls of the form PileupFiles().

Fields

The PileupFiles class is implemented as an S4 reference class. It has the following fields:

files A list of BamFile instances.

param An instance of PileupParam.

Functions and methods

Opening / closing:

open.PileupFiles Opens the (local or remote) path and index of each file in the PileupFiles instance. Returns a PileupFiles instance.

close.PileupFiles Closes each file in the PileupFiles instance; returning (invisibly) the updated PileupFiles. The instance may be re-opened with open.PileupFiles.

Accessors:

plpFiles Returns the list of the files in the PileupFiles instance.

plpParam Returns the PileupParam content of the PileupFiles instance.

Methods:

applyPileups Calculate the pileup across all files in files according to criteria in param (or plpParam(files) if param is missing), invoking FUN on each range or collection of positions. See applyPileups.

show Compactly display the object.

Author(s)

Martin Morgan

Examples

```
example(applyPileups)
```

PileupParam   *Parameters for creating pileups from BAM files*

Description

Use `PileupParam()` to create a parameter object influencing what fields and which records are used to calculate pile-ups, and to influence the values returned.

Usage

```
# Constructor
PileupParam(flag = scanBamFlag(),
    minBaseQuality = 13L, minMapQuality = 0L,
    minDepth = 0L, maxDepth = 250L,
    yieldSize = 1L, yieldBy = c("range", "position"), yieldAll = FALSE,
    which = GRanges(), what = c("seq", "qual"))

# Accessors
plpFlag(object)
plpFlag(object) <- value
plpMaxDepth(object)
plpMaxDepth(object) <- value
plpMinBaseQuality(object)
plpMinBaseQuality(object) <- value
plpMinDepth(object)
plpMinDepth(object) <- value
plpMinMapQuality(object)
plpMinMapQuality(object) <- value
plpWhat(object)
plpWhat(object) <- value
plpWhich(object)
plpWhich(object) <- value
plpYieldAll(object)
plpYieldAll(object) <- value
plpYieldBy(object)
plpYieldBy(object) <- value
plpYieldSize(object)
plpYieldSize(object) <- value

## S4 method for signature PileupParam
show(object)
```

Arguments

flag   An instance of the object returned by scanBamFlag, restricting various aspects
                  of reads to be included or excluded.

minBaseQuality	The minimum read base quality below which the base is ignored when summarizing pileup information.
minMapQuality	The minimum mapping quality below which the entire read is ignored.
minDepth	The minimum depth of the pile-up below which the position is ignored.
maxDepth	The maximum depth of reads considered at any position; this can be used to limit memory consumption.
yieldSize	The number of records to include in each call to FUN.
yieldBy	How records are to be counted. By range (in which case yieldSize must equal 1) means that FUN is invoked once for each range in which. By position means that FUN is invoked whenever pile-ups have been accumulated for yieldSize positions, regardless of ranges in which.
yieldAll	Whether to report all positions (yieldAll=TRUE), or just those passing the filtering criteria of flag, minBaseQuality, etc. When yieldAll=TRUE, positions not passing filter criteria have '0' entries in seq or qual.
which	A GRanges or RangesList instance restricting pileup calculations to the corresponding genomic locations.
what	A character() instance indicating what values are to be returned. One or more of c("seq", "qual").
object	An instace of class PileupParam.
value	An instance to be assigned to the corresponding slot of the PileupParam instance.

Objects from the Class

Objects are created by calls of the form PileupParam().

Slots

Slot interpretation is as described in the 'Arguments' section.

flag Object of class integer encoding flags to be kept when they have their '0' (keep0) or '1' (keep1) bit set.

minBaseQuality An integer(1).

minMapQuality An integer(1).

minDepth An integer(1).

maxDepth An integer(1).

yieldSize An integer(1).

yieldBy An character(1).

yieldAll A logical(1).

which A GRanges or RangesList instance.

what A character().

Functions and methods

See 'Usage' for details on invocation.

Constructor:

PileupParam: Returns a `PileupParam` object.

Accessors: get or set corresponding slot values; for setters, `value` is coerced to the type of the corresponding slot.

plpFlag, plpFlag<- Returns or sets the named `integer` vector of flags; see `scanBamFlag`.

plpMinBaseQuality, plpMinBaseQuality<- Returns or sets an `integer(1)` vector of miminum base qualities.

plpMinMapQuality, plpMinMapQuality<- Returns or sets an `integer(1)` vector of miminum map qualities.

plpMinDepth, plpMinDepth<- Returns or sets an `integer(1)` vector of miminum pileup depth.

plpMaxDepth, plpMaxDepth<- Returns or sets an `integer(1)` vector of the maximum depth to which pileups are calculated.

plpYieldSize, plpYieldSize<- Returns or sets an `integer(1)` vector of yield size.

plpYieldBy, plpYieldBy<- Returns or sets an `character(1)` vector determining how pileups will be returned.

plpYieldAll, plpYieldAll<- Returns or sets an `logical(1)` vector indicating whether all positions, or just those satisfying pileup positions, are to be returned.

plpWhich, plpWhich<- Returns or sets the object influencing which locations pileups are calculated over.

plpWhat, plpWhat<- Returns or sets the `character` vector describing what summaries are returned by pileup.

Methods:

show Compactly display the object.

Author(s)

Martin Morgan

See Also

applyPileups.

Examples

```
example(applyPileups)
```

quickBamFlagSummary   *Group the records of a BAM file based on their flag bits and count the number of records in each group*

Description

`quickBamFlagSummary` groups the records of a BAM file based on their flag bits and counts the number of records in each group.

Usage

```
quickBamFlagSummary(file, ..., param=ScanBamParam(), main.groups.only=FALSE)

## S4 method for signature character
quickBamFlagSummary(file, index=file, ..., param=ScanBamParam(),
    main.groups.only=FALSE)

## S4 method for signature list
quickBamFlagSummary(file, ..., param=ScanBamParam(), main.groups.only=FALSE)
```

Arguments

`file, index`	For the character method, the path to the BAM file to read, and to the index file of the BAM file to read, respectively.
	For the list() method, `file` is a named list with elements "qname" and "flag" with content as from [scanBam](#).
`...`	Additional arguments, perhaps used by methods.
`param`	An instance of [ScanBamParam](#). This determines which records are considered in the counting.
`main.groups.only`	
	If `TRUE`, then the counting is performed for the main groups only.

Value

Nothing is returned. A summary of the counts is printed to the console unless redirected by [sink](#).

Author(s)

H. Pages

References

<http://samtools.sourceforge.net/>

See Also

[scanBam](#), [ScanBamParam](#).

[BamFile](#) for a method for that class.

Examples

```
bamfile <- system.file("extdata", "ex1.bam", package="Rsamtools",
                        mustWork=TRUE)
quickBamFlagSummary(bamfile)
```

---

readPileup   *Import samtools 'pileup' files.*

Description

Import files created by evaluation of samtools' pileup -cv command.

Usage

```
readPileup(file, ...)
## S4 method for signature connection
readPileup(file, ..., variant=c("SNP", "indel", "all"))
```

Arguments

file	The file name, or [connection](#), of the pileup output file to be parsed.
...	Additional arguments, passed to methods. For instance, specify variant for the readPileup,character-method.
variant	Type of variant to parse; select one.

Value

readPileup returns a [GRanges](#) object.

The value returned by variant="SNP" or variant="all" contains:

space: The chromosome names (fastq ids) of the reference sequence

position: The nucleotide position (base 1) of the variant.

referenceBase: The nucleotide in the reference sequence.

consensusBase; The consensus nucleotide, as determined by samtools pileup.

consensusQuality: The phred-scaled consensus quality.

snpQuality: The phred-scaled SNP quality (probability of the consensus being identical to the reference).

maxMappingQuality: The root mean square mapping quality of reads overlapping the site.

coverage: The number of reads covering the site.

The value returned by variant="indel" contains space, position, reference, consensus, consensusQuality, snpQuality, maxMappingQuality, and coverage fields, and:

alleleOne, alleleTwo The first (typically, in the reference sequence) and second allelic variants.

alleleOneSupport, alleleTwoSupport The number of reads supporting each allele.

additionalIndels The number of additional indels present.

Author(s)

Sean Davis

References

http://samtools.sourceforge.net/

Examples

```
fl <- system.file("extdata", "pileup.txt", package="Rsamtools",
                  mustWork=TRUE)
(res <- readPileup(fl))
xtabs(~referenceBase + consensusBase, mcols(res))[DNA_BASES,]

## Not run: ## uses a pipe, and arguments passed to read.table
## three successive piles of 100 records each
cmd <- "samtools pileup -cvf human_b36_female.fa.gz na19240_3M.bam"
p <- pipe(cmd, "r")
snp <- readPileup(p, nrow=100)  # variant="SNP"
indel <- readPileup(p, nrow=100, variant="indel")
all <- readPileup(p, nrow=100, variant="all")

## End(Not run)
```

RsamtoolsFile   *A base class for managing file references in Rsamtools*

Description

RsamtoolsFile is a base class for managing file references in **Rsamtools**; it is not intended for direct use by users – see, e.g., `BamFile`.

Usage

```
## accessors
index(object)
## S4 method for signature RsamtoolsFile
```

```
path(object, ...)
## S4 method for signature RsamtoolsFile
isOpen(con, rw="")
## S4 method for signature RsamtoolsFile
yieldSize(object, ...)
yieldSize(object, ...) <- value
## S4 method for signature RsamtoolsFile
show(object)
```

Arguments

`con, object`	An instance of a class derived from `RsamtoolsFile`.
`rw`	Mode of file; ignored.
`...`	Additional arguments, unused.
`value`	Replacement value.

Objects from the Class

Users do not directly create instances of this class; see, e.g., [`BamFile`](#)-class.

Fields

The `RsamtoolsFile` class is implemented as an S4 reference class. It has the following fields:

.extptr An `externalptr` initialized to an internal structure with opened bam file and bam index pointers.

path A character(1) vector of the file name.

index A character(1) vector of the index file name.

yieldSize An integer(1) vector of the number of records to yield.

Functions and methods

Accessors:

path Returns a character(1) vector of path names.

index Returns a character(1) vector of index path names.

yieldSize, yieldSize<- Return or set an integer(1) vector indicating yield size.

Methods:

isOpen Report whether the file is currently open.

show Compactly display the object.

Author(s)

Martin Morgan

RsamtoolsFileList   *A base class for managing lists of Rsamtools file references*

Description

RsamtoolsFileList is a base class for managing lists of file references in **Rsamtools**; it is not intended for direct use – see, e.g., BamFileList.

Usage

```
## S4 method for signature RsamtoolsFileList
path(object, ...)
## S4 method for signature RsamtoolsFileList
isOpen(con, rw="")
## S3 method for class RsamtoolsFileList
open(con, ...)
## S3 method for class RsamtoolsFileList
close(con, ...)
## S4 method for signature RsamtoolsFileList
names(x)
## S4 method for signature RsamtoolsFileList
yieldSize(object, ...)
```

Arguments

con, object, x   An instance of a class derived from RsamtoolsFileList.

rw   Mode of file; ignored.

...   Additional arguments.

Objects from the Class

Users do not directly create instances of this class; see, e.g., BamFileList-class.

Functions and methods

This class inherits functions and methods for subseting, updating, and display from the SimpleList class.

Methods:

isOpen: Report whether each file in the list is currently open.

open: Attempt to open each file in the list.

close: Attempt to close each file in the list.

names: Names of each element of the list or, if names are NULL, the basename of the path of each element.

Author(s)

Martin Morgan

ScanBamParam   *Parameters for scanning BAM files*

Description

Use ScanBamParam() to create a parameter object influencing what fields and which records are imported from a (binary) BAM file. Use of which requires that a BAM index file (<filename>.bai) exists.

Usage

```
# Constructor
ScanBamParam(flag = scanBamFlag(), simpleCigar = FALSE,
    reverseComplement = FALSE, tag = character(0),
    what = character(0), which)

# Constructor helpers
scanBamFlag(isPaired = NA, isProperPair = NA, isUnmappedQuery = NA,
    hasUnmappedMate = NA, isMinusStrand = NA, isMateMinusStrand = NA,
    isFirstMateRead = NA, isSecondMateRead = NA, isNotPrimaryRead = NA,
    isNotPassingQualityControls = NA, isDuplicate = NA,
    isValidVendorRead = NA)

scanBamWhat()

# Accessors
bamFlag(object, asInteger=FALSE)
bamFlag(object) <- value
bamReverseComplement(object)
bamReverseComplement(object) <- value
bamSimpleCigar(object)
bamSimpleCigar(object) <- value
bamTag(object)
bamTag(object) <- value
bamWhat(object)
bamWhat(object) <- value
bamWhich(object)
bamWhich(object) <- value

## S4 method for signature ScanBamParam
show(object)
```

```
# Flag utils
bamFlagAsBitMatrix(flag, bitnames=FLAG_BITNAMES)
bamFlagAND(flag1, flag2)
bamFlagTest(flag, value)
```

Arguments

flag
: For ScanBamParam, an integer(2) vector used to filter reads based on their 'flag' entry. This is most easily created with the scanBamFlag() helper function.

    For bamFlagAsBitMatrix, bamFlagTest an integer vector where each element represents a 'flag' entry.

simpleCigar
: A logical(1) vector which, when TRUE, returns only those reads for which the cigar (run-length encoded representation of the alignment) is missing or contains only matches / mismatches (M).

reverseComplement
: A logical(1) vectors. BAM files store reads mapping to the minus strand as though they are on the plus strand. Rsamtools obeys this convention by default (reverseComplement=FALSE), but when this value is set to TRUE returns the sequence and quality scores of reads mapped to the minus strand in the reverse complement (sequence) and reverse (quality) of the read as stored in the BAM file. This might be useful if wishing to recover read and quality scores as represented in fastq files, but is NOT appropriate for variant calling or other alignment-based operations.

tag
: A character vector naming tags to be extracted. A tag is an optional field, with arbitrary information, stored with each record. Tags are identified by two-letter codes, so all elements of tag must have exactly 2 characters.

what
: A character vector naming the fields to return scanBamWhat() returns a vector of available fields. Fields are described on the scanBam help page.

which
: A GRanges, RangesList, RangedData, any object that can be coerced to a RangesList, or missing object, from which a IRangesList instance will be constructed. Names of the IRangesList correspond to reference sequences, and ranges to the regions on that reference sequence for which matches are desired. Because data types are coerced to IRangesList, which does *not* include strand information (use the flag argument instead). Only records with a read overlapping the specified ranges are returned. All ranges must have ends less than or equal to 536870912.

isPaired
: A logical(1) indicating whether unpaired (FALSE), paired (TRUE), or any (NA) read should be returned.

isProperPair
: A logical(1) indicating whether improperly paired (FALSE), properly paired (TRUE), or any (NA) read should be returned. A properly paired read is defined by the alignment algorithm and might, e.g., represent reads aligning to identical reference sequences and with a specified distance.

isUnmappedQuery
: A logical(1) indicating whether unmapped (TRUE), mapped (FALSE), or any (NA) read should be returned.

hasUnmappedMate

    A logical(1) indicating whether reads with mapped (FALSE), unmapped (TRUE), or any (NA) mate should be returned.

isMinusStrand   A logical(1) indicating whether reads aligned to the plus (FALSE), minus (TRUE), or any (NA) strand should be returned.

isMateMinusStrand

    A logical(1) indicating whether mate reads aligned to the plus (FALSE), minus (TRUE), or any (NA) strand should be returned.

isFirstMateRead

    A logical(1) indicating whether the first mate read should be returned (TRUE) or not (FALSE), or whether mate read number should be ignored (NA).

isSecondMateRead

    A logical(1) indicating whether the second mate read should be returned (TRUE) or not (FALSE), or whether mate read number should be ignored (NA).

isNotPrimaryRead

    A logical(1) indicating whether alignments that are primary (FALSE), are not primary (TRUE) or whose primary status does not matter (NA) should be returned. A non-primary alignment ("secondary alignment" in the SAM specification) might result when a read aligns to multiple locations. One alignment is designated as primary and has this flag set to FALSE; the remainder, for which this flag is TRUE, are designated by the aligner as secondary.

isNotPassingQualityControls

    A logical(1) indicating whether reads passing quality controls (FALSE), reads not passing quality controls (TRUE), or any (NA) read should be returned.

isValidVendorRead

    Deprecated; use isNotPassingQualityControls.

isDuplicate   A logical(1) indicating that un-duplicated (FALSE), duplicated (TRUE), or any (NA) reads should be returned. 'Duplicated' reads may represent PCR or optical duplicates.

object   An instance of class ScanBamParam.

value   An instance of the corresponding slot, to be assigned to object or, for bamFlagTest, a character(1) name of the flag to test, e.g., "isUnmappedQuery", from the arguments to scanBamFlag.

asInteger   logical(1) indicating whether 'flag' should be returned as an encoded integer vector (TRUE) or human-readable form (FALSE).

bitnames   Names of the flag bits to extract. Will be the colnames of the returned matrix.

flag1, flag2   Integer vectors containing 'flag' entries.

Objects from the Class

Objects are created by calls of the form ScanBamParam().

Slots

flag Object of class integer encoding flags to be kept when they have their '0' (keep0) or '1' (keep1) bit set.

simpleCigar Object of class `logical` indicating, when TRUE, that only 'simple' cigars (empty or 'M') are returned.

reverseComplement Object of class `logical` indicating, when TRUE, that reads on the minus strand are to be reverse complemented (sequence) and reversed (quality).

tag Object of class `character` indicating what tags are to be returned.

what Object of class `character` indicating what fields are to be returned.

which Object of class `RangesList` indicating which reference sequence and coordinate reads must overlap.

Functions and methods

See 'Usage' for details on invocation.

Constructor:

ScanBamParam: Returns a `ScanBamParam` object. The `which` argument to the constructor can be one of several different types, as documented above.

Accessors:

bamTag, bamTag<- Returns or sets a `character` vector of tags to be extracted.

bamWhat, bamWhat<- Returns or sets a `character` vector of fields to be extracted.

bamWhich, bamWhich<- Returns or sets a `RangesList` of bounds on reads to be extracted. A length 0 `RangesList` represents all reads.

bamFlag, bamFlag<- Returns or sets an `integer(2)` representation of reads flagged to be kept or excluded.

bamSimpleCigar, bamSimpleCigar<- Returns or sets a `logical(1)` vector indicating whether reads without indels or clipping be kept.

bamReverseComplement, bamReverseComplement<- Returns or sets a `logical(1)` vector indicating whether reads on the minus strand will be returned with sequence reverse complemented and quality reversed.

Methods:

show Compactly display the object.

Author(s)

Martin Morgan

See Also

scanBam

Examples

```
## defaults
p0 <- ScanBamParam()

## subset of reads based on genomic coordinates
which <- RangesList(seq1=IRanges(1000, 2000),
                    seq2=IRanges(c(100, 1000), c(1000, 2000)))
p1 <- ScanBamParam(which=which)

## subset of reads based on flag value
p2 <- ScanBamParam(flag=scanBamFlag(isMinusStrand=FALSE))

## subset of fields
p3 <- ScanBamParam(what=c("rname", "strand", "pos", "qwidth"))

## use
fl <- system.file("extdata", "ex1.bam", package="Rsamtools",
                  mustWork=TRUE)
res <- scanBam(fl, param=p2)[[1]]
lapply(res, head)

## tags; NM: edit distance; H1: 1-difference hits
p4 <- ScanBamParam(tag=c("NM", "H1"), what="flag")
bam4 <- scanBam(fl, param=p4)
str(bam4[[1]][["tag"]])

## flag utils
flag <- scanBamFlag(isUnmappedQuery=FALSE, isMinusStrand=TRUE)
flag
bamFlagAsBitMatrix(flag)
flag4 <- bam4[[1]][["flag"]]
bamFlagAsBitMatrix(flag4[1:9], bitnames=c("isUnmappedQuery", "isMinusStrand"))
```

ScanBcfParam-class   *Parameters for scanning BCF files*

Description

Use ScanBcfParam() to create a parameter object influencing the 'INFO' and 'GENO' fields parsed, and which sample records are imported from a BCF file. Use of which requires that a BCF index file (<filename>.bci) exists.

Usage

```
ScanBcfParam(fixed=character(), info=character(), geno=character(),
             samples=character(), trimEmpty=TRUE, which, ...)

## S4 method for signature missing
```

```
ScanBcfParam(fixed=character(), info=character(), geno=character(),
             samples=character(), trimEmpty=TRUE, which, ...)
## S4 method for signature RangesList
ScanBcfParam(fixed=character(), info=character(), geno=character(),
             samples=character(), trimEmpty=TRUE, which, ...)
## S4 method for signature RangedData
ScanBcfParam(fixed=character(), info=character(), geno=character(),
             samples=character(), trimEmpty=TRUE, which, ...)
## S4 method for signature GRanges
ScanBcfParam(fixed=character(), info=character(), geno=character(),
             samples=character(), trimEmpty=TRUE, which, ...)
## S4 method for signature GRangesList
ScanBcfParam(fixed=character(), info=character(), geno=character(),
             samples=character(), trimEmpty=TRUE, which, ...)

## Accessors
bcfFixed(object)
bcfInfo(object)
bcfGeno(object)
bcfSamples(object)
bcfTrimEmpty(object)
bcfWhich(object)
```

Arguments

`fixed`	A logical(1) for use with `ScanVcfParam` only.
`info`	A character() vector of 'INFO' fields (see [scanVcfHeader](#)) to be returned.
`geno`	A character() vector of 'GENO' fields (see [scanVcfHeader](#)) to be returned. `character(0)` returns all fields, `NA_character_` returns none.
`samples`	A character() vector of sample names (see [scanVcfHeader](#)) to be returned. `character(0)` returns all fields, `NA_character_` returns none.
`trimEmpty`	A logical(1) indicating whether 'GENO' fields with no values should be returned.
`which`	An object, for which a method is defined (see usage, above), describing the sequences and ranges to be queried. Variants whose `POS` lies in the interval(s) `[start, end)` are returned.
`object`	An instance of class `ScanBcfParam`.
`...`	Arguments used internally.

Objects from the Class

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

Slots

`which`: Object of class `"RangesList"` indicating which reference sequence and coordinate variants must overlap.

info: Object of class "character" indicating portions of 'INFO' to be returned.

geno: Object of class "character" indicating portions of 'GENO' to be returned.

samples: Object of class "character" indicating the samples to be returned.

trimEmpty: Object of class "logical" indicating whether empty 'GENO' fields are to be returned.

fixed: Object of class "character". For use with ScanVcfParam only.

Functions and methods

See 'Usage' for details on invocation.

Constructor:

ScanBcfParam: Returns a `ScanBcfParam` object. The `which` argument to the constructor can be one of several types, as documented above.

Accessors:

bcfInfo, bcfGeno, bcfTrimEmpty, bcfWhich: Return the corresponding field from `object`.

Methods:

show Compactly display the object.

Author(s)

Martin Morgan mtmorgan@fhcrc.org

See Also

scanVcf ScanVcfParam

Examples

```
## see ?ScanVcfParam examples
```

---

seqnamesTabix   *Retrieve sequence names defined in a tabix file.*

Description

This function queries a tabix file, returning the names of the 'sequences' used as a key when creating the file.

Usage

```
seqnamesTabix(file, ...)
## S4 method for signature character
seqnamesTabix(file, ...)
```

Arguments

file	A character(1) file path or [TabixFile](#) instance pointing to a 'tabix' file.
...	Additional arguments, currently ignored.

Value

A character() vector of sequence names present in the file.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>.

Examples

```
fl <- system.file("extdata", "example.gtf.gz", package="Rsamtools",
                   mustWork=TRUE)
seqnamesTabix(fl)
```

TabixFile *Manipulate tabix indexed tab-delimited files.*

Description

Use TabixFile() to create a reference to a Tabix file (and its index). Once opened, the reference remains open across calls to methods, avoiding costly index re-loading.

TabixFileList() provides a convenient way of managing a list of TabixFile instances.

Usage

```
## Constructors

TabixFile(file, index = paste(file, "tbi", sep="."), ...,
         yieldSize=NA_integer_)
TabixFileList(..., yieldSize=NA_integer_)

## Opening / closing

## S3 method for class TabixFile
open(con, ...)
## S3 method for class TabixFile
close(con, ...)

## accessors; also path(), index(), yieldSize()

## S4 method for signature TabixFile
isOpen(con, rw="")
```

```
## actions

## S4 method for signature TabixFile
seqnamesTabix(file, ...)
## S4 method for signature TabixFile
headerTabix(file, ...)
## S4 method for signature TabixFile,GRanges
scanTabix(file, ..., param)
## S4 method for signature TabixFile,RangesList
scanTabix(file, ..., param)
## S4 method for signature TabixFile,RangedData
scanTabix(file, ..., param)
## S4 method for signature TabixFile,missing
scanTabix(file, ..., param)
## S4 method for signature character,ANY
scanTabix(file, ..., param)
## S4 method for signature character,missing
scanTabix(file, ..., param)

countTabix(file, ...)
```

Arguments

con	An instance of `TabixFile`.
file	For TabixFile(), A character(1) vector to the tabix file path; can be remote (http://, ftp://). For `countTabix`, a character(1) or `TabixFile` instance. For others, a `TabixFile` instance.
index	A character(1) vector of the tabix file index.
yieldSize	Number of records to yield each time the file is read from using scanTabix. Only valid when `param` is unspecified. `yieldSize` does not alter existing yield sizes, include NA, when creating a `TabixFileList` from `TabixFile` instances.
param	An instance of GRanges, IRangedData, or RangesList, used to select which records to scan.
...	Additional arguments. For `TabixFileList`, this can either be a single character vector of paths to tabix files, or several instances of `TabixFile` objects.
rw	character() indicating mode of file; not used for `TabixFile`.

Objects from the Class

Objects are created by calls of the form TabixFile().

Fields

The `TabixFile` class inherits fields from the [`RsamtoolsFile`](#) class.

Functions and methods

TabixFileList inherits methods from [RsamtoolsFileList](#) and [SimpleList](#).

Opening / closing:

open.TabixFile Opens the (local or remote) path and index. Returns a TabixFile instance. yieldSize determines the number of records parsed during each call to scanTabix; NA indicates that all records are to be parsed.

close.TabixFile Closes the TabixFile con; returning (invisibly) the updated TabixFile. The instance may be re-opened with open.TabixFile.

Accessors:

path Returns a character(1) vector of the tabix path name.

index Returns a character(1) vector of tabix index name.

yieldSize, yieldSize<- Return or set an integer(1) vector indicating yield size.

Methods:

seqnamesTabix Visit the path in path(file), returning the sequence names present in the file.

headerTabix Visit the path in path(file), returning the sequence names, column indicies used to sort the file, the number of lines skipped while indexing, the comment character used while indexing, and the header (preceeded by comment character, at start of file) lines.

countTabix Return the number of records in each range of param, or the count of all records in the file (when param is missing).

scanTabix For signature(file="TabixFile"), Visit the path in path(file), returning the result of [scanTabix](#) applied to the specified path. For signature(file="character"), call the corresponding method after coercing file to TabixFile.

indexTabix This method operates on file paths, rather than TabixFile objects, to index tab-separated files. See [indexTabix](#).

show Compactly display the object.

Author(s)

Martin Morgan

Examples

```
fl <- system.file("extdata", "example.gtf.gz", package="Rsamtools",
                   mustWork=TRUE)
tbx <- TabixFile(fl)

param <- GRanges(c("chr1", "chr2"), IRanges(c(1, 1), width=100000))
countTabix(tbx)
countTabix(tbx, param=param)
res <- scanTabix(tbx, param=param)
sapply(res, length)
res[["chr1:1-100000"]][1:2]
```

```
## parse to list of data.frames
dff <- Map(function(elt) {
    read.csv(textConnection(elt), sep="\t", header=FALSE)
}, res)
dff[["chr1:1-100000"]][1:5,1:8]

## parse 100 records at a time
length(scanTabix(tbx)[[1]]) # total number of records
tbx <- open(TabixFile(fl, yieldSize=100))
while(length(res <- scanTabix(tbx)[[1]]))
   cat("records read:", length(res), "\n")
close(tbx)
```

TabixInput   *Operations on 'tabix' (indexed, tab-delimited) files.*

Description

Scan compressed, sorted, tabix-indexed, tab-delimited files.

Usage

```
scanTabix(file, ..., param)
## S4 method for signature character,RangesList
scanTabix(file, ..., param)
## S4 method for signature character,RangedData
scanTabix(file, ..., param)
## S4 method for signature character,GRanges
scanTabix(file, ..., param)
```

Arguments

file	The character() file name(s) of the tabix file be processed, or more flexibly an instance of class `TabixFile`.
param	A instance of GRanges, RangedData, or RangesList provide the sequence names and regions to be parsed.
...	Additional arguments, currently ignored.

Value

scanTabix returns a list, with one element per region. Each element of the list is a character vector representing records in the region.

Error

scanTabix signals errors using `signalCondition`. The following errors are signaled:

scanTabix_param `yieldSize(file)` must be NA when more than one range is specified.

scanTabix_io A read error occured while inputing the tabix file. This might be because the file is corrupt, or of incorrect format (e.g., when `path` points to a plain text file but `index` is present, implying that `path` should be a bgziped file.

Author(s)

Martin Morgan <mtmorgan@fhcrc.org>.

References

<http://samtools.sourceforge.net/tabix.shtml>

Examples

```
example(TabixFile)
```

---

yieldReduce	*Iterate through a BAM (or other) file, reducing output to a single result.*

Description

Rsamtools files can be created with a 'yieldSize' argument that influences the number of records (chunk size) input at one time (see, e.g,. [BamFile](#)). yieldReduce iterates through the file, processing each chunk and reducing it with previously input chunks. This is a memory efficient way to process large data files, especially when the final result fits in memory.

Usage

```
yieldReduce(X, MAP, REDUCE, DONE, ..., init, ITERATE = TRUE)
```

Arguments

X	A [BamFile](#) instance (or other class for which `isOpen`, `open`, `close` methods are defined, and which support input of sequential chunks).
MAP	A function of one or more arguments, `X`, `...`, returning a VALUE passed to DONE and REDUCE.
REDUCE	A function of one (ITERATE=FALSE or two (ITERATE=TRUE) arguments, returning the reduction (e.g., addition) of the argument(s). If missing, REDUCE is c (when ITERATE=TRUE) or `identity` when (when ITERATE=FALSE).
DONE	A function of one argument, the VALUE of the most recent call to `MAP(X, ...)`. If missing, DONE is `function(VALUE) length(VALUE) == 0`.

...	Additional arguments, passed to MAP.
init	(Optional) Initial value used for REDUCE when ITERATE=TRUE.
ITERATE	logical(1) determining whether the call to REDUCE is iterative (ITERATE=TRUE) or cumulative (ITERATE=FALSE).

Details

When ITERATE=TRUE, REDUCE is initially invoked with either the init value and the value of the first call to MAP or, if init is missing, the values of the first two calls to MAP.

When ITERATE=FALSE, REDUCE is invoked with a list containing a list with as many elements as there were calls to MAP. Each element the result of an invocation of MAP.

Value

The return value is the value returned by the final invocation of REDUCE, or init if provided and no data were yield'ed, or list() if init is missing and no data were yield'ed.

Author(s)

Martin Morgan mtmorgan@fhcrc.org

See Also

[BamFile](), [TabixFile](), [RsamtoolsFile]().

Examples

```
fl <- system.file(package="Rsamtools", "extdata", "ex1.bam")

## nucleotide frequency of mapped reads
bf <- BamFile(fl, yieldSize=500) ## typically, yieldSize=1e6
param <- ScanBamParam(
    flag=scanBamFlag(isUnmappedQuery=FALSE),
    what="seq")
MAP <- function(X, param) {
    value <- scanBam(X, param=param)[[1]][["seq"]]
    if (length(value))
        alphabetFrequency(value, collapse=TRUE)
    else value      # will be integer(0)
}
REDUCE <- +        # add successive alphabetFrequency matrices
yieldReduce(bf, MAP, REDUCE, param=param)

## coverage
if (require(GenomicAlignments)) {
    MAP <- function(X)
        coverage(readGAlignments(X))
    REDUCE <- +
    DONE <- function(VALUE)
        ## coverage() on zero GAlignments returns an RleList,
        ## each element of  which has 0 coverage
```

```
        sum(sum(VALUE)) == 0L
    yieldReduce(bf, MAP, REDUCE, DONE)
}
```

Index