Rsamtools

October 25, 2011

`BamFile` *Maintain SAM and 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.

Usage

```
## Opening / closing

BamFile(file, index=file)
## S3 method for class 'BamFile'
open(con, ...)
## S3 method for class 'BamFile'
close(con, ...)

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

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

## actions

## S4 method for signature 'BamFile'
scanBamHeader(files, ...)
## S4 method for signature 'BamFile'
scanBam(file, index=file, ..., param=ScanBamParam())
## S4 method for signature 'BamFile'
countBam(file, index=file, ..., param=ScanBamParam())
## S4 method for signature 'BamFile'
filterBam(file, destination, index=file, ...,
    indexDestination=TRUE, param=ScanBamParam())
## S4 method for signature 'BamFile'
indexBam(files, ...)
## S4 method for signature 'BamFile'
```

```
sortBam(file, destination, ..., byQname=FALSE, maxMemory=512)
## S4 method for signature 'BamFile'
readBamGappedAlignments(file, index, ..., which)
```

Arguments

`...`	Additional arguments.
`con`	An instance of `BamFile`.
`file, files`	A character vector of BAM file paths (for `BamFile`) or a `BamFile` instance (for other methdods).
`index`	A character vector of indices (for `BamFile`); ignored for all other methods on this page.
`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.
`which`	An optional `RangesList` instance to further subset `file`.
`rw`	Mode of file; ignored.

Objects from the Class

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

Fields

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

Functions and methods

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`.

Accessors:

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

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

Methods:

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

countBam Visit the path 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.

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

show Compactly display the object.

Author(s)

Martin Morgan

Examples

```
fl <- system.file("extdata", "ex1.bam", package="Rsamtools")
bf <- open(BamFile(fl))        # implicit index
bf
identical(scanBam(bf), scanBam(fl))

rng <- GRanges(c("seq1", "seq2"), IRanges(1, c(1575, 1584)))

## repeatedly visit 'bf'
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)
```

`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())
## S4 method for signature 'BamViews'
countBam(file, index = file, ..., param = ScanBamParam())
## S4 method for signature 'BamViews'
readBamGappedAlignments(file, index, ..., which)

## 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.
which	An optional RangesList instance to further subset file.

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)`.

readBamGappedAlignments Visit each path in `bamPaths(file)`, returning the result of `readBamGappedAlign` applied to the specified path. When `index` is missing, it is set equal to `bamIndicies(file)`. When `which` is missing, only reads in `bamRanges(file)` are returned. When `which` is present, reads matching `bamRanges(file[which])` are returned. The return value is a `SimpleList`, with elements of the list corresponding to each path. `bamSamples(file)` is available as `elementMetadata` of the returned `SimpleList`.

show Compactly display the object.

Author(s)

Martin Morgan

See Also

`readBamGappedAlignments`

Examples

```
fls <- list.files(system.file("extdata", package="Rsamtools"),
                  "\\.bam$", full=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

bv <- BamViews(fls,
               bamSamples=DataFrame(info="test", row.names="ex1"),
               auto.range=TRUE)
aln <- readBamGappedAlignments(bv)
aln
aln[[1]]
aln[colnames(bv)]
elementMetadata(aln)
```

BcfFile *Manipulate BCF or VCF files.*

Description

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

Usage

```
## Opening / closing

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

Arguments

`con, object`	An instance of `BcfFile`.
`file`	A character(1) vector of the VCF or BCF file path.
`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, currently unused.
`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

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")
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)
```

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.

Usage

```
## Opening / closing

FaFile(file, ...)
## 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 'FaFile'
countFa(file, ...)
## S4 method for signature 'FaFile,GRanges'
scanFa(file, param=GRanges(), ...)
## S4 method for signature 'FaFile,missing'
scanFa(file, param=GRanges(), ...)
```

Arguments

con	An instance of FaFile.
file	A character(1) vector of the fasta file path (for FaFile), or an instance of class FaFile.
param	An optional GRanges instance to select reads (and sub-sequences) for input.
...	Additional arguments, currently unused.
rw	Mode of file; ignored.

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

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.

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 param is GRanges(), no records are selected.

show Compactly display the object.

Author(s)

Martin Morgan

Examples

```
fl <- system.file("extdata", "ce2dict1.fa", package="Rsamtools")
fa <- open(FaFile(fl))                    # open
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]))
```

`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

http://samtools.sourceforge.net/

Examples

```
packageDescription('Rsamtools')
```

`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
path(object)
index(object)
## S4 method for signature 'RsamtoolsFile'
isOpen(con, rw="")
## S4 method for signature 'RsamtoolsFile'
show(object)
```

Arguments

con, object	An instance of a class derived from RsamtoolsFile.
rw	Mode of file; ignored.

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.

Functions and methods

Accessors:

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

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

Methods:

isOpen Report whether the file is currently open.

show Compactly display the object.

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 = scanBamWhat(), which)

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

# Accessors
bamFlag(object)
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)
```

Arguments

`flag`	An integer(2) vector used to filter reads based on their 'flag' entry. This is most easily created with the `scanBamFlag()` helper function.
`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) vector which, when 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.
`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`, 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).

`isPrimaryRead`
A logical(1) indicating whether reads that are not primary (FALSE), are primary (TRUE) or whose primary status does not matter (NA) should be returned. A non-primary read might result when portions of a read aligns to multiple locations, e.g., when spanning splice junctions).

`isValidVendorRead`
A logical(1) indicating whether invalid (FALSE), valid (TRUE), or any (NA) read should be returned. A 'valid' read is one flagged by the vendor as passing quality control criteria.

`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`.

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")
res <- scanBam(fl, param=p2)[[1]]
lapply(res, head)

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

ScanBcfParam-class *Parameters for scanning VCF / BCF files*

Description

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

Usage

```
ScanBcfParam(info=character(), geno=character(), trimEmpty=TRUE,
             which)
## S4 method for signature 'missing'
ScanBcfParam(info=character(), geno=character(), trimEmpty=TRUE,
             which)
## S4 method for signature 'RangesList'
ScanBcfParam(info=character(), geno=character(), trimEmpty=TRUE,
             which)
## S4 method for signature 'RangedData'
ScanBcfParam(info=character(), geno=character(), trimEmpty=TRUE,
             which)
## S4 method for signature 'GRanges'
ScanBcfParam(info=character(), geno=character(), trimEmpty=TRUE,
             which)
```

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

Arguments

info	A character() vector of 'INFO' fields (see scanBcfHeader) to be returned. Not currently implemented.
geno	A character() vector of 'GENO' fields (see scanBcfHeader) 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.

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.

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

Functionas 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

scanBcf

Examples

```
p0 <- ScanBcfParam()

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

## return only specified GENO field(s)
p2 <- ScanBcfParam(geno="GT")
```

---

readBamGappedAlignments

GappedAlignments objects

Description

Read a BAM file as a GappedAlignments object.

Usage

```
readBamGappedAlignments(file, index, ..., which)
```

Arguments

file	The character(1) file name of the 'BAM' file to be processed.
index	The character(1) name of the index file of the 'BAM' file being processed; this is given *without* the '.bai' extension.
...	Additional arguments, currently unused.
which	An object passed to which in ScanBamParam to specify ranges from which alignments will be retrieved. Valid types are described on the ScanBamParam help page.

Details

See ?GappedAlignments-class for a description of GappedAlignments objects.

See ?scanBam for a description of the arguments. Unlike SAM/BAM records, we don't support unaligned queries so we discard those records.

Author(s)

H. Pages

See Also

GappedAlignments-class, scanBam

Examples

```
aln1_file <- system.file("extdata", "ex1.bam", package="Rsamtools")
aln1 <- readBamGappedAlignments(aln1_file)
aln1
```

---

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")
(res <- readPileup(fl))
xtabs(~referenceBase + consensusBase, elementMetadata(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)
```

BamInput — *Import, count, index, and other operations on 'BAM' (binary alignment)*

Description

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

Usage

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

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)

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

```
filterBam(file, destination, index=file, ...,
    indexDestination=TRUE, param=ScanBamParam())

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

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

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.
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 or filtered output file will be created. For asBam and sortBam this is without the ".bam" file suffix.
...	Additional arguments, passed to methods.
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.
byQname	A logical(1) indicating whether the sorted destination file should be sorted by Query-name (TRUE) or by mapping position (FALSE).
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 function, 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 the samtools view -Sb file > destination. The 'BAM' file is sorted and an index created on the destination (with extension '.bai') when indexDestination=TRUE.

filterBam parses records in file satisfying the bamWhich of param, writing each record satisfying the bamFlag and bamSimpleCigar criteria of param to file destination. An index file is created on the destination when indexDestination=TRUE.

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.

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	The query name, i.e., identifier, associated with the read.
flag	A numeric value summarizing details of the read. See ScanBamParam and the flag argument, and scanBamFlag().
rname	The name of the reference to which the read is aligned.
strand	The strand to which the read is aligned.
pos	The genomic coordinate at the start of the alignment. Coordinates are 'leftmost', 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.
mrnm	The reference to which the mate (of a paired end or mate pair read) aligns.
mpos	The position to which the mate aligns.
isize	Inferred insert size for paired end alignments.
seq	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	Phred-encoded, phred-scaled base quality score, oriented as seq.

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 returns the file name of the BAM 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")

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)
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]])

sorted <- sortBam(fl, tempfile())

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

BcfInput *Operations on 'VCF' or 'BCF' (variant call) files.*

Description

Scan headers of 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())
```

Arguments

file	The character() file name(s) of the 'VCF' or 'BCF' file to be processed, or an instance of class BcfFile.
index	The character() file name(s) of the 'BCF' index to be processed.
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.

Value

scanBcfHeader returns a list, with one element for each file named in files. Each element of the list is itself a list containing three element. 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 those supported by 'bcftools' (see documentation at the url below).

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.

Examples

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

```
str(head(bcf[["GENO"]][["PL"]]))
example(BcfFile)
```

`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=GRanges(), ...)
## S4 method for signature 'character,GRanges'
scanFa(file, param=GRanges(), ...)
## S4 method for signature 'character,missing'
scanFa(file, param=GRanges(), ...)
```

Arguments

`file`	A character(1) vector containing the fasta file path.
`param`	An optional `GRanges` instance to select reads (and sub-sequences) for input.
`...`	Additional arguments, currently unused.

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` 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")
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]))
```

Index