Rsamtools

October 5, 2010

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=new("DataFrame", nrows=length(bamPaths),
                         rownames=make.unique(basename(bamPaths))),
         bamRanges, bamExperiment = list(), ...)
## S4 method for signature 'missing':
BamViews(bamPaths=character(0),
         bamIndicies=bamPaths,
         bamSamples=new("DataFrame", nrows=length(bamPaths),
                         rownames=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 method for signature 'BamViews':
names(x) <- value
## S4 method for signature 'BamViews':
dimnames(x)
## S4 method for signature 'BamViews':
```

1

```
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)
```

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

Description

This package provides facilities for parsing samtools BAM (binary) files representing aligned se-
quences.

Details

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

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')
```

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 unmapped (TRUE), mapped (FALSE), 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"]])
```

---

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) files.*

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, ...)

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

filterBam(file, destination, index=file, ...)
## S4 method for signature 'character':
filterBam(file, destination, index=file, ...,
    indexDestination=TRUE, param=ScanBamParam())
```

Arguments

file	The character(1) file name of the 'BAM' file to be processed.
files	The character() file names of the 'BAM' file to be indexed.
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 filtered output file will be created.
...	Additional arguments, passed to methods.
indexDestination	
	A logical(1) indicating whether the created destination file should also be indexed.
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.

The `countBam` function returns a count of records consistent with `param`.

The `scanBamHeader` function 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.

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

The `filterBam` function 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`.

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.

`indexBam` returns the file name of the index file created.

`filterBam` returns the file name of the destination file created.

Author(s)

Martin Morgan

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]])
```

Index