Package: BamScale (via r-universe)

May 26, 2026

Type Package

Title Bioconductor-Friendly Multithreaded BAM Processing

Version 0.99.10

Description Multithreaded sequential BAM processing built on top of the ompBAM C++ engine. BamScale provides user-friendly BAM read and scan interfaces designed for compatibility with existing Bioconductor workflows.

License MIT + file LICENSE

biocViews Software, DataImport, Sequencing, Coverage, Alignment, QualityControl

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports BiocGenerics, BiocParallel, Biostrings, GenomicAlignments, GenomicRanges, IRanges, methods, ompBAM, Rcpp, Rsamtools, S4Vectors

LinkingTo Rcpp, ompBAM

Suggests BiocStyle, chipseqDBData, dplyr, ExperimentHub, GenomeInfoDb, ggplot2, knitr, readr, rmarkdown, scales, tibble, testthat (>= 3.0.0), tidyr,

VignetteBuilder knitr

Config/testthat/edition 3

SystemRequirements C++17, OpenMP

NeedsCompilation yes

URL https://cparsania.github.io/BamScale/

BugReports https://github.com/cparsania/BamScale/issues

Config/pak/sysreqs libbz2-dev liblzma-dev xz-utils zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-05-26 05:52:36 UTC

RemoteUrl https://github.com/bioc/BamScale

RemoteRef HEAD

RemoteSha 026ff90e052d4769cd73251ec81124cc2fb3bfd1

Contents

`bam_count` *Count BAM records with Bioconductor-compatible filtering*

Description

`bam_count()` provides a fast chromosome-level count summary, honoring key filtering fields from `ScanBamParam` (`mapqFilter`, `flag`, and `which`).

Usage

```
bam_count(
  file,
  param = NULL,
  threads = 1L,
  BPPARAM = BiocParallel::bpparam(),
  auto_threads = FALSE,
  include_unmapped = TRUE
)
```

Arguments

`file`	BAM input (`character`, `BamFile`, or `BamFileList`).
`param`	Optional `Rsamtools::ScanBamParam` (or compatible list).
`threads`	Requested number of OpenMP threads. May be capped when `auto_threads = TRUE`.
`BPPARAM`	`BiocParallel` parameter for multi-file operation. Defaults to `BiocParallel::bpparam()`. Set to `NULL` to force serial file processing.
`auto_threads`	Logical; when `TRUE` and `BPPARAM` has multiple workers, BamScale adaptively avoids oversubscription by preserving higher per-file OpenMP thread counts when possible and reducing the number of concurrently active file workers before shrinking per-file threads.

include_unmapped
: Whether to include an extra * row for unmapped records.

Details

Parallelism behavior matches bam_read(): BPPARAM distributes work across BAM files, while threads controls OpenMP work within each file. If auto_threads = TRUE and BPPARAM has multiple workers, BamScale first limits the number of concurrently active workers to preserve the requested per-file thread count within the detected core budget, then caps per-file OpenMP threads only if a single file would still oversubscribe the machine.

Value

For one file: a data.frame with columns seqname, seqlength, count. For multiple files: named list of such data.frames.

Examples

```
bam <- ompBAM::example_BAM("Unsorted")
bam_count(bam, threads = 2)
```

---

bam_read *Fast BAM reading with Bioconductor-compatible arguments*

Description

bam_read() is a multithreaded sequential BAM reader built on top of ompBAM. The interface is designed to be familiar to users of Rsamtools::scanBam(), GenomicAlignments::readGAlignments(), and GenomicAlignments::readGAlignmentPairs().

Usage

```
bam_read(
  file,
  param = NULL,
  what = NULL,
  tag = NULL,
  as = c("DataFrame", "data.frame", "GAlignments", "GAlignmentPairs", "scanBam"),
  seqqual_mode = c("compatible", "compact"),
  threads = 1L,
  BPPARAM = BiocParallel::bpparam(),
  auto_threads = FALSE,
  use.names = FALSE,
  with.which_label = FALSE,
  include_unmapped = TRUE
)
```

Arguments

file	A BAM input. Supported values are: • a single BAM path (character(1)) or multiple BAM paths, • a Rsamtools::BamFile, • a Rsamtools::BamFileList.
param	Optional Rsamtools::ScanBamParam (or a compatible list for lightweight use). The following fields are honored: mapqFilter, flag, which, what, and tag.
what	Character vector of fields to return, similar to scanBam(what=...). Supported fields are qname, flag, rname, strand, pos, qwidth, mapq, cigar, mrnm, mpos, isize, seq, qual.
tag	Character vector of 2-letter tag names to extract.
as	Output format: • "DataFrame": returns S4Vectors::DataFrame (default), • "data.frame": returns base data.frame, • "GAlignments": returns GenomicAlignments::GAlignments, • "GAlignmentPairs": returns GenomicAlignments::GAlignmentPairs, • "scanBam": returns a scanBam()-shaped list-of-lists.
seqqual_mode	Controls representation of seq/qual when those fields are requested: • "compatible" (default): return character vectors matching scanBam-style expectations, • "compact": return lower-level raw list-columns for faster/lower-overhead extraction. In compact mode, seq is returned as one raw vector per read containing BAM-native packed sequence bytes (two bases per byte), and qual is returned as one raw vector per read containing per-base Phred bytes. These are not plain character strings. qwidth is needed to decode compact seq back to base letters, and qual values of 255 correspond to missing quality values. This mode is currently supported for as = "data.frame" or as = "DataFrame".
threads	Requested number of OpenMP threads used for reading/decompression. May be capped when auto_threads = TRUE.
BPPARAM	BiocParallel parameter used when file contains more than one BAM. Defaults to BiocParallel::bpparam(). Set to NULL to force serial file processing.
auto_threads	Logical; when TRUE and BPPARAM has multiple workers, BamScale adaptively avoids oversubscription by preserving higher per-file OpenMP thread counts when possible and reducing the number of concurrently active file workers before shrinking per-file threads.
use.names	Passed to alignment object conversion. When TRUE, read names (qname) are used as object names.
with.which_label	Logical; if TRUE and param includes which, an extra which_label column is returned.
include_unmapped	Logical; whether unmapped records are retained (subject to param$flag constraints).

Details

`bam_read()` is intentionally column-compatible with common BAM fields used by Bioconductor workflows and can be used as a fast drop-in reader before conversion to downstream classes.

Parallelism model:

- `BPPARAM` parallelizes across files (one file per BiocParallel worker).
- `threads` parallelizes within each file via OpenMP.
- Effective total concurrency is approximately `min(length(file), BiocParallel::bpnworkers(BPPARAM)) * threads`.
- If `auto_threads = TRUE` and `BPPARAM` has multiple workers, BamScale first limits the number of concurrently active workers to preserve the requested per-file thread count within the detected core budget, then caps per-file OpenMP threads only if a single file would still oversubscribe the machine.

Compatibility notes:

- Region filtering via `param$which` is supported as a sequential filter (not index-jump random access).
- Flag filtering uses `ScanBamFlag` semantics by converting logical flag requirements into required-set and required-unset bit masks.
- Tag values are returned as character columns. Scalar tags are scalar strings; `B` tags are comma-separated vectors.
- `seqqual_mode = "compact"` is optimized for throughput-oriented benchmarking and returns raw list-columns for `seq`/`qual`, not ordinary sequence or quality strings. In this representation, `seq` contains BAM-packed nucleotide bytes and `qual` contains raw Phred bytes. Compact output is intended for users who want to defer or avoid full string-materialization costs; use `decode_compact_seq()`, `decode_compact_qual()`, or `decode_seqqual_compact()` to decode compact output back to standard string form when needed.
- `"GAlignments"` and `"GAlignmentPairs"` output exclude unmapped records.
- `as = "scanBam"` returns a strict scan-like list-of-lists: without `param$which`, it returns one unnamed batch; with `param$which`, it returns one batch per range label (including empty ranges), with requested `what` fields and `tag` values under `$tag`. In this output mode, `seq` and `qual` are returned as `Biostrings::DNAStringSet` and `Biostrings::PhredQuality` for closer `scanBam()` compatibility.

Value

If `file` is length 1: one object in the format specified by `as`. If `file` has length > 1 (or is a `BamFileList`): a named list of outputs, one per BAM file.

Examples

```
bam <- ompBAM::example_BAM("Unsorted")

# Familiar scanBam-like field selection
x <- bam_read(bam, what = c("qname", "flag", "rname", "pos", "cigar"))
```

```
# Include sequence + quality
y <- bam_read(bam, what = c("qname", "seq", "qual"), threads = 2)

# scanBam-shaped output
z <- bam_read(bam, what = c("qname", "flag"), tag = "NM", as = "scanBam")
```

`decode_compact_qual` *Decode compact BamScale quality output*

Description

Decodes `qual` values returned by `bam_read(..., seqqual_mode = "compact")` back to ASCII Phred-quality strings.

Usage

```
decode_compact_qual(qual)
```

Arguments

`qual`	A list (or list-column) of `raw` vectors produced by compact BamScale quality extraction.

Value

A character vector containing decoded quality strings. Entries with all-missing quality bytes are returned as `"*"`, matching BamScale's compatibility mode.

See Also

`decode_compact_seq()`, `decode_seqqual_compact()`, `bam_read()`

Examples

```
decode_compact_qual(
  qual = list(as.raw(c(0L, 1L, 2L, 3L)))
)
```

decode_compact_seq *Decode compact BamScale sequence output*

Description

Decodes `seq` values returned by `bam_read(..., seqqual_mode = "compact")` back to ordinary character strings.

Usage

```
decode_compact_seq(seq, qwidth)
```

Arguments

`seq`	A list (or list-column) of `raw` vectors produced by compact BamScale sequence extraction.
`qwidth`	Integer vector of read widths. This is required because compact sequence bytes use BAM's 4-bit packed encoding (two bases per byte).

Value

A character vector containing decoded sequence strings.

See Also

`decode_compact_qual()`, `decode_seqqual_compact()`, `bam_read()`

Examples

```
decode_compact_seq(
  seq = list(as.raw(c(0x12, 0x48))),
  qwidth = 4L
)
```

decode_seqqual_compact *Decode compact* `seq` *and* `qual` *columns in BamScale output*

Description

Convenience wrapper for converting a compact `bam_read()` result back to ordinary sequence and quality strings.

Usage

```
decode_seqqual_compact(
  x,
  seq_col = "seq",
  qual_col = "qual",
  qwidth_col = "qwidth"
)
```

Arguments

x	A data.frame, S4Vectors::DataFrame, or list-like object containing compact BamScale seq and/or qual columns.
seq_col	Name of the compact sequence column.
qual_col	Name of the compact quality column.
qwidth_col	Name of the read-width column used to decode compact sequence bytes.

Value

x with compact seq and/or qual columns replaced by decoded character vectors. The input class is preserved.

See Also

decode_compact_seq(), decode_compact_qual(), bam_read()

Examples

```
x <- data.frame(qwidth = 4L)
x$seq <- I(list(as.raw(c(0x12, 0x48))))
x$qual <- I(list(as.raw(c(0L, 1L, 2L, 3L))))
decode_seqqual_compact(x)
```

Index